Individual and group dynamic behaviour patterns in bound spaces

Pawel Gasiorowski

School of Computing and Digital Media
London Metropolitan University

This dissertation is submitted for the degree of

Doctor of Philosophy

December 2017

I would like to dedicate this thesis to my loving parents …

Acknowledgements

After nearly four years of intense work, today I am writing the final section of my dissertation: a special note of thanks. It was a long journey during which I have had developed in so many ways that I believe my thesis would not had been complete if I did not reflect on people who have helped me through this period.

I would first like to thank my supervisors, Dr. Vassil Vassilev and Prof. Karim Ouazzane for their guidance, support and patience thorough the duration of this programme. You were always willing to help me by pointing me in the right direction when I encountered obstacles in my path. You have given me motivation to move forward and taught me to never give up even in moments of doubt. In addition, I would like to give my thanks to Dr. Romas Mikusauskas for sharing his knowledge from the area of 3D technologies and whole academic board and staff at London Metropolitan University. This programme was the biggest challenge of my life so far and it was an honour to take it on with such brilliant companions like you.

To my fellow PhD students, especially Majid Afzal, with whom I spent countless hours on exchanging thoughts, brainstorming and experimenting with innovating technologies so that I could gain inspiration for finding a solution to that little problem that was preventing me from progressing further.

I would like to give special thanks to CEO's of The Vinyl Factory Limited, Mark Wadhwa and Tim Robinson, for their continuing faith in me and funding of this programme.

I would also like to thank my wonderful parents for their commitment in supporting me in difficult times and always being there for me when I need it most. Finally, I would like to thank all my friends, who were understanding enough to forgive my absences from various events due to my works on research. We were able to have some great times that allowed me to think and talk about things other than just academic papers for a change.

Thank you very much, everyone!

Pawel Gasiorowski

London, December 11, 2017.

Abstract

The behaviour analysis of individual and group dynamics in closed spaces is a subject of extensive research in both academia and industry. However, despite recent technological advancements the problem of implementing the existing methods for visual behaviour data analysis in production systems remains difficult and the applications are available only in special cases in which the resourcing is not a problem. Most of the approaches concentrate on direct extraction and classification of the visual features from the video footage for recognising the dynamic behaviour directly from the source. The adoption of such an approach allows recognising directly the elementary actions of moving objects, which is a difficult task on its own. The major factor that impacts the performance of the methods for video analytics is the necessity to combine processing of enormous volume of video data with complex analysis of this data using and computationally resource-demanding analytical algorithms. This is not feasible for many applications, which must work in real time. In this research, an alternative simulation-based approach for behaviour analysis has been adopted. It can potentially reduce the requirements for extracting information from real video footage for the purpose of the analysis of the dynamic behaviour. This can be achieved by combining only limited data extracted from the original video footage with a symbolic data about the events registered on the scene, which is generated by 3D simulation synchronized with the original footage. Additionally, through incorporating some physical laws and the logics of dynamic behaviour directly in the 3D model of the visual scene, this framework allows to capture the behavioural patterns using simple syntactic pattern recognition methods. The extensive experiments with the prototype implementation prove in a convincing manner that the 3D simulation generates sufficiently rich data to allow analysing the dynamic behaviour in real-time with sufficient adequacy without the need to use precise physical data, using only a limited data about the objects on the scene, their location and dynamic characteristics. This research can have a wide applicability in different areas where the video analytics is necessary, ranging from public safety and video surveillance to marketing research to computer games and animation. Its limitations are linked to the dependence on some preliminary processing of the video footage which is still less detailed and computationally demanding than the methods which use directly the video frames of the original footage.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans have developed a natural way of perceiving and interpreting our surroundings based on the contextual knowledge that they possess. Our brains can easily recognise objects, their relationships and behaviour on the basis of constant delivery of imagery data. We can process huge amounts of visual information almost instantly and make conclusions about its meaning. To build a visual system that could mimic the natural perception system of humans is a challenging task if one considers all the complexities that come with it.

1.1. Research problem

The key research question which is addressed in this research is to determine whether it is possible to construct efficient algorithms for online recognition, classification and characterisation of the agent behaviour and to make inference about the behavioural patterns using limited information about the location and the position of the body and its movements through the space of a visual scene, observed by virtual surveillance cameras. This is necessary for the purpose of subsequent personalisation, statistical and demographic analysis, planning of actions and navigation through the space in various software applications, which incorporate 3D micro-world models. The performance bottleneck which limits the possibility to address this problem in video analytics applications comes from the fact that the information which is contained in live video feeds coming from video camera is too rich and contextually redundant, which makes the data analysis extremely computationally demanding. By combining limited physical information that could potentially be extracted from video signals with simulated data coming from a simulator of the visual scene one can reduce the amount of physical information needed for the analysis of the dynamic behaviour. It is believed that using model-driven approach based on 3D simulation the information generated by the simulation will be adequate for the purpose of the analysis despite the fact that it will not be extracted from the original video. The successful answer of this question will bring significant advantages to diverse areas where the dynamic behaviour is important, such as e-business, safety and security monitoring, assistive technologies and entertainment industry.

1.2. Aim and objectives of the research

The main aim of this research is to develop an object-centric and event-driven framework capable of analysing the individual and group behaviour of human agents in real time. The analysis will be based on the structural and spatio-temporal properties of recognised objects and their interaction with physical environment. In order to reach this goal, a set of the following objectives has been established prior to the beginning of the research:

1. To investigate the current state of research in the area of visual analytics and to identify the main problems, existing approaches and available methods for reporting typical behaviour within enclosed spaces over limited period of time.
2. To study the underlying logic of 3D programming in order to specify a 3D model of a micro world consisting of active physical agents surrounded with passive and active artefacts, capable of presenting and influencing static and dynamic aspects of the individual and group behaviour.
3. To simulate the dynamics of the micro world in real time for a chosen application domain (i.e. evacuation of buildings or large vessels in the case of safety alerts, preventing or recovering from incidents during public gatherings, supermarket shopping sessions, computer games etc.).
4. To analyse the historical data from the simulation, derive patterns of dynamic behaviour of the agents in the micro-world and develop a model of individual and group behaviour for the purpose of subsequent analysis and control.
5. To develop classification method for behavioural patterns and pattern-matching algorithms using event-driven approach. The main focus here will be on the development of a language processor capable of analysing the events defining behavioural patterns.
6. To validate the developed framework using a test application in a chosen application domain and compare it to other relevant research developments.

1.3. Research methodology

In order to build an appropriate level of awareness in the area of visual analytics as well as to construct a computational model, which is based on a sound theory, we have adopted two complementary approaches, which we applied at different stages of the research. The first, analytical approach, is focused on critical review of the recent achievements in the field of visual analysis of human behaviour. It was carried out with particular attention to the *methods which can accommodate the conceptual solutions in constructive models*. During this stage we developed ideas, which allowed us to establish a logical link between the real world environment and the virtual space of the modelled micro world. The constructive approach adopted in this research is based on the *theory of 3D visual simulation* as well as on the *techniques used in computer games*. The reason for this choice was the understanding that an adequate simulation of the visual scene for analytical purposes can be achieved only by having a full control of the entire simulated world, which is the case in game programming. The second reason for the choice of 3D simulation was the need to incorporate means for analytical pre-processing of the simulated data for the purpose of the analysis, which is possible only if the model of the visual scene has rich semantics, close enough to the ontology of the real physical world.

1.3.1. 3D modelling of the visual scene

The limited micro-world of the agents has been modelled using concepts and programming techniques derived from the area of 3D modelling and computer graphics. This approach closely resembles the physical world observed by surveillance cameras. It allows to attribute additional information about spatial location and motion, necessary for supporting possible dynamic changes in the graphical representation under the circumstances in a most adequate way in comparison to 2D approaches which perform the visual analytics directly using the visual frames of the video feed. In addition, the 3D approach allows to account the physical laws and logical dependencies between the objects which govern the dynamics of the visual scene, which allows to incorporate efficient method for preliminary processing of the visual information directly in the graphic simulation, which leads to significant simplifications and analytical gains.

1.3.2. Event-driven simulation of the agent behaviour

The agent behaviour simulation is based on the classical computer games approach for modelling interactive 3D environments. Each agent is active within the bound space and its reactions to events are governed by the input taken from the input data stream, the laws of physics and the logics of spatial dependencies on the visual scene. For the purpose of this research it is assumed that the input data stream is externally generated using separate modules, which are outside of the scope of this work.

1.3.3. Pattern recognition for classification of the agent behaviour

This is the main analytical part of the research which will be based on capturing the events occurring during simulation, analysing them and inferring the corresponding behaviour based on recognition of the event patterns. The simulator will log the events occurring at runtime in symbolic format, which will be then analysed using syntactic methods for pattern recognition suitable for symbolic data logs.

1.3.4. Experimental prototyping

As an end result of the research, it is planned to produce an intelligent controller for dynamic behaviour analysis in a form of a self-contained software application capable of constructing and representation of behavioural patterns in suitable format for further use by other software applications which are part of the overall video analytics framework developed at the Cyber Security Research centre of London Metropolitan University. For the implementation of the prototype an open-source game engines have been selected. The developed prototype has modular architecture organised around several design patterns. It inherits all advantages of the component approach and object-oriented methodology for software engineering in general. This allows an easy integration of the application with other applications, which require spatial modelling and agent behaviour analysis and makes the video analytics more widely possible.

1.4. Scope of the research

The automatic recognition and evaluation of human behaviour is a challenging problem that requires studying wide range of scientific topics. In our approach we will replace the live video signal feed with a combination of a basic data extracted from it and simulated data approximating the missing data and sufficient for subsequent analysis of the visually observed individual and group dynamic behaviour. Since it is important to be able to use all relevant information from the images taken from video footage or similar medium for precise analysis, the image-processing techniques are also reviewed but their detailed analysis is left out of the main scope of this research. Instead, the focus is on developing algorithms capable of simulating objects that represent real subjects in virtual scene and extracting, collecting and evaluating the relevant information.

The behavioural patterns have been defined purely linguistically, using a context-free grammar to make this method more universal. Consequently, the output of the analyser is the classification of a pattern according to the pattern syntax built directly into the grammar of the pattern language. This is approach is more flexible and powerful solution as opposed to other pattern recognition approaches which typically match the live data against static database of patterns defined statistically or graphically. However, the further contextual analysis of the behaviour patterns may require additional processing of the simulation output log, which is beyond the scope of this research.

1.5. Structure of this report

The entire material presented in this report has been organised in a systematic way that reflects the constructive bottom-up development approach mentioned earlier. Each chapter is a foundation for the next one and progressively describes the methods for behaviour's modelling, simulation and analysis.

Chapter 2: Methods for visual modelling and analysis of human dynamic behaviour

In this chapter, different computational models of human behaviour have been reviewed and studied in the context of the problem domain.

The nature of the existing research, the adopted methodologies and proposed solutions to the problems have been presented in a cohesive manner so that the literature review is naturally structured from constructive point of view.

Chapter 3: Conceptual Framework for model-driven dynamic behaviour analysis

The focus of this chapter is based on conceptualisation of the architecture of the entire framework for simulated video analytics. It presents the model of a micro-world, which is used for the analysis of human behaviour and the simulation-based event-driven approach adopted in this research in a systematic manner.

Chapter 4: 3D Simulator of individual and group dynamics in visual scenes

This chapter describes the implementation process of the software simulator, which is based on 3D graphics mathematical theory and common software engineering practices in graphic programming.

Chapter 5: Experimental validation of the simulator

This chapter is dedicated to presenting the functionality of the simulator of dynamic behaviour patterns, its evaluation and validation with a set of experiments.

Chapter 6: Experimental analysis and evaluation of the patterns of individual and group dynamic behaviour

This chapter is dedicated to presenting the results of the pattern analysis and experimental evaluation of the entire simulation-based event-driven framework developed in this research.

Chapter 7: Conclusions and recommendation for future work

This chapter summarises the entire research, claiming some original contributions and makes recommendations for further exploration of the developed framework.

Chapter 2

Methods for visual modelling and analysis of human dynamic behaviour

With the growth of computer technologies, recent years have seen an increasing interest in developing computational models for analysing human behaviour. The model typically requires accounting of the set of activities that can establish a pattern of dynamic behaviour and can be constructed through observation of the visual scene. The Oxford Dictionary defines behaviour as *"the way in which one acts or conducts oneself, especially towards others"* (Oxford Dictionary, 2015). In that sense, the behaviour may be considered as a set of motions and actions exhibited by people that can be clearly visible. In addition, the Oxford Dictionary defines a pattern as *"a regular and intelligible form or sequence discernible in the way in which something happens or is done"* (Oxford Dictionary, 2015). Therefore, one may define a behaviour pattern as a *regular and intelligible sequence discernible in the way in which one acts or conducts oneself or towards others.*

Visual modelling, simulation and analysis of dynamic behaviour are currently becoming an important direction of research in video surveillance due to its importance for safety management, security and disasters recovery. Many researchers try to address issues related to the range of complexities in derivation of behaviour patterns from live video footage. The research in this area includes extraction of image features, identification of shapes, detecting movements and modelling human behaviour for further analysis. The review focuses on the following tasks:

- To learn the latest advancements of image processing and video analysis methods used specifically for dynamic behaviour analysis. The main focus here was made on determining what kind of information can be captured from the observed visual scene.

- To gain insight into the classification methods used for recognition of movements and actions related to human behaviour. The main focus here was on learning what kind of actions and behaviours can be recognised using the information extracted from visual scene. The classification of the movements allows identifying the actions, which are the building blocks of the patterns to be recognized.

- To study modelling of human behaviour and associated formal theories for the purpose of simulation and analysis. The investigation was oriented towards exploring methods that are used for modelling human behaviour in simulated environments. The third review area was selected given the importance of the models in the simulation.

- To explore frameworks, which utilise combination of methods for extraction, classification and simulation of information, related to human behaviour. The focus here is made on existing software systems that can evaluate human behaviour with the use of multiple different techniques. The comparative analysis of the frameworks was chosen to obtain the knowledge of the current technological achievement in video analytics software development.

2.1. Visual Feature Extraction for Dynamic Behaviour Analysis

The observation of the people in front of the camera is a first step in defining the dynamics of individuals and groups within the boundaries of the visual scene. It usually deals with building descriptors of the visual features based on the data extracted from the video footage. The methods of extracting information as well as the data being extracted may vary, depending on the requirements of the analysis.

The first step is to account the properties of objects, then the activities carried out by those objects under different circumstances and constraints. Numerous approaches found in the literature make an attempt to address this complex problem (Gong and Xiang, 2011). They are classified into four categories: *object-based, part-based, pixel-based* and *event based.*

The *object-based* approach exploits the idea that any given object can be explicitly described using several attributes that can be extracted directly from observing the visual scene. They define objects' appearance in a certain location that can change over time (Gong and Xiang, 2011) and further divide this approach into *trajectory-based* and *template-based.*

The *trajectory-based* approaches aim at reconstructing the movement history of an object or its constituent parts based on the changes registered during the monitoring process (Abdul-Azim and Hemayed, 2015; Jiang et al., 2012; Oikonomopoulos et al., 2007). Afzal et al. (2017) make an attempt of reconstructing the trajectories in an incremental manner, which is suitable for processing in real time. The continuous video stream is first being pre-processed and then each frame is separated in order to extract the location of an object at a given time. In this case, the features of an object are narrowed down to spatial location, rotation, the direction of movement and velocity.

Figure 2-1. The reconstructed trajectory of a moving subject; the moving subject's volume is approximated with a sphere (Afzal et al., 2017)

As observed by the authors, shape also plays an important role in tracking a moving object, since one cannot simply assume that it will not change over time. The object may bend and twist during the movement and therefore the physical structure needs to be accounted for. The solution to this problem lies in the introduction of a bounding volume that either matches its silhouette or approximates it as shown in Figure 2-1. However, the precise detection of the silhouette of a 3D object in a 2D imagery environment is not suitable for real-time applications due to the complex computations that need to be carried out. As noted by Mulayim, Yilmaz and Atalay (2003) it may also require a well-controlled environment where each property can be clearly defined which is rarely available. A simpler shape in the form of a primitive geometry such as a sphere, which wraps the identified object is more suitable from performance point of view.

Once the object shape is established, one needs to consider the constituent parts of that object. Afzal et al. (2017) continues with an explanation on how the representation of the body of an object can be reduced to interconnected spherical shapes, where each sphere corresponds to a body part (Figure 2-2). It is a valid assumption because each part would then occupy certain space in a close proximity from the median point of the body so that when the body moves its part will move in a coordinated manner.

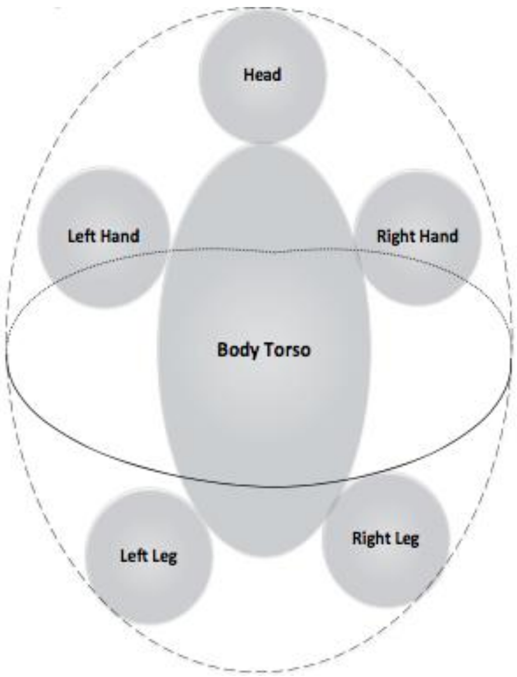


Figure 2-2. The seven-sphere model adopted by Afzal et al. (2017). The model gives an approximation of the volume occupied by the individual limbs of each agent

Similar models have already been developed and proven to work in trajectory tracking of individual objects, with Microsoft Kinect line-based system being the most popular one (Biswas and Basu, 2011; Kar, 2010; Ren et al. 2013). An important characteristic of these systems is that once a skeleton consisting of connected nodes is initially captured in the visual scene, it becomes easier to track each individual object even in heavily occluded areas. The positions of unobstructed parts of the body may provide a foundation for approximating the location of the remaining parts with the assumption that they are within a reasonable proximity.

Many researchers have adopted the trajectory-based approach as a starting point in their behaviour analysis (Anjum and Cavallaro, 2007; Shen et al., 2015; Azorín-López et al. 2013; Azorin-López et al. 2014; Chen, 2012), but as noted by Gong and Xiang (2011), the reconstruction of trajectory alone may not be enough to successfully capture the behaviour. The major flaw of trajectory-based representation of human behaviour is the fact that it is usually built on data that is too noisy and may come from cluttered environments where tracking individual objects may not be feasible at all times. The authors outlined three main drawbacks from using this method:

a) Computations of trajectories are being made on the assumption that shapes of the objects provide high quality of details. Should the observed environment consists of great number of agents, the algorithms for tracking may not extract sufficient information about the features of interest and therefore provide inaccurate data as a result.

b) It may not be possible to continuously monitor each object's trajectory. This is true if one realises that there will always be some obstacles in the visual scene. For instance, in the visual space of the shop there might be a shelf that will cause occlusions.

c) The whole analysis of behaviour cannot be based solely on trajectories because it might not provide sufficient contextual information in order to support more detailed analysis.

An alternative method of measuring and analysing the movements of objects is to use pre-defined templates. In the *template-based* approach, an action is described by a sequence of historical images that is then matched against template images of known movements. That is, a sequence of images comprises a single-descriptor of a well-defined action. The most recent recorded motion in that sequence is indicated by more intense values of the pixels in a corresponding region of an image. At the recognition stage, each pixel value taken from a recorded historical image is being compared with the corresponding value stored within the template image to determine whether a given action took place.

An exhaustive description of this method has been presented by Ahad et al. (2009). In his paper the author highlights research of Bobick and Davis (2001) as a prime example of application of this method. Bobick and Davis (2001) use two types of images: the ones that are used to determine in which visual region a motion has occurred (motion-energy images, MEI) and the ones that record information on how the motion appeared at a given pixel (motion-history images, MHI) as shown in Figure 2-3.

Their idea is then to compute and combine the occupancy of the pixels of those two historical images to generate a vector-based image that can be used for segmentation and recognition. Bobick and Davis (2001) indicate several issues with this approach. Firstly, there is a need of recording historical images in a well-controlled environment. The presented experiments were mostly conducted in well-lit spaces with cameras positioned orthogonally in front or at a specific angle relative to the actor. Secondly, the method may fail in a scenario where two or more actors are being visible in the camera view due to the fact that separating their contour may be very difficult. Thirdly, as previously noted by Akita (1984), it may cause a situation where movements within the silhouette of a moving object may never be accounted for.

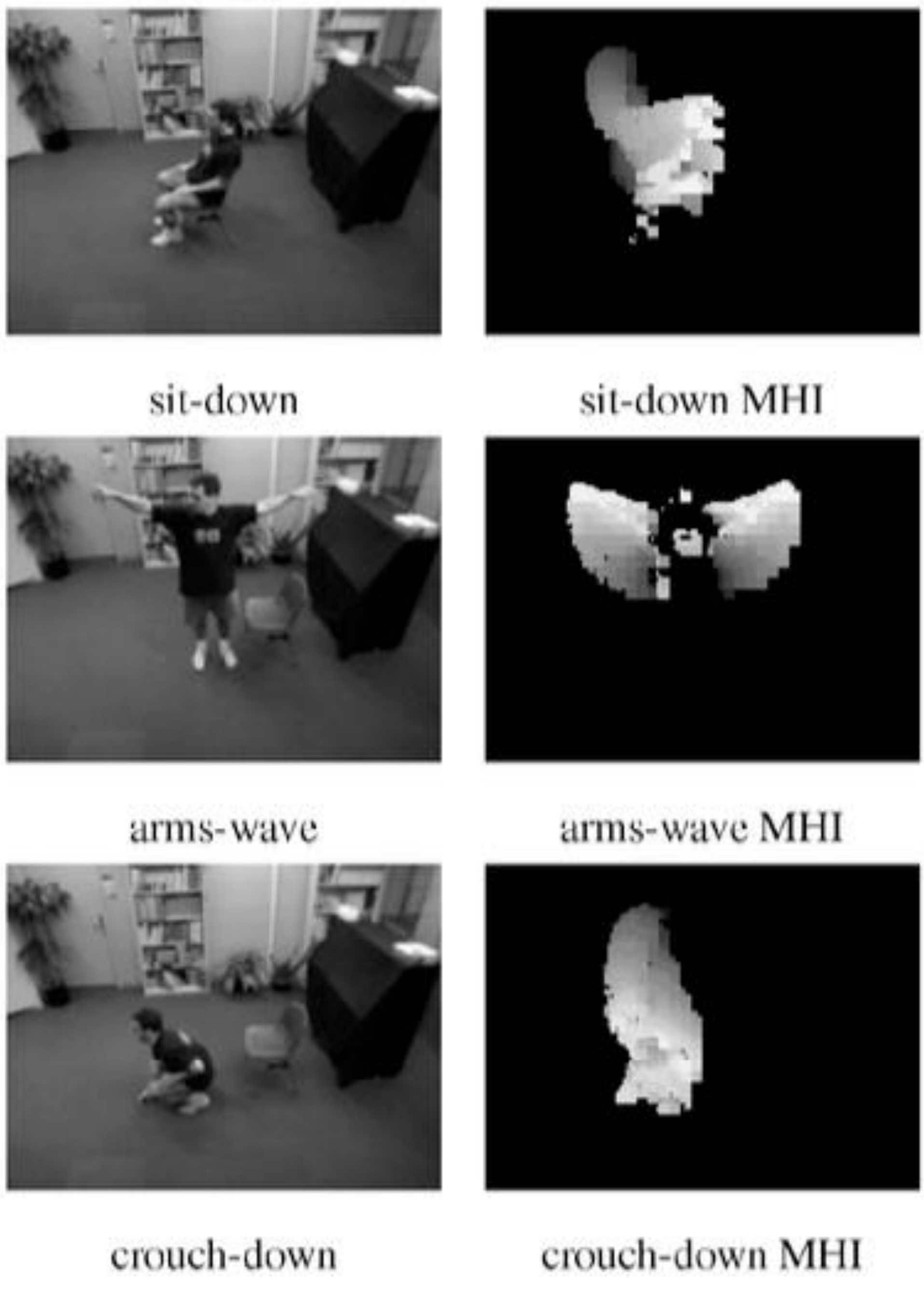


Figure 2-3. Examples of simple movements along with their motion history images (MHI) (Bobick and Davis, 2001)

An interesting expansion of this approach was presented by Weinland, Ronfard and Boyer (2006). They made an attempt to introduce a third dimension in the historical images. It allowed them to store movements in a form of a so-called motion-history volumes, or MHV (Figure 2-4). As stated in their paper, this enhanced method carries multiple benefits, because in principle, more information about the motion carried out by the observed object can be registered. However, it also imposes a major constraint on the setup where the world has to be monitored by more than one camera simultaneously to generate MHV at specific times.

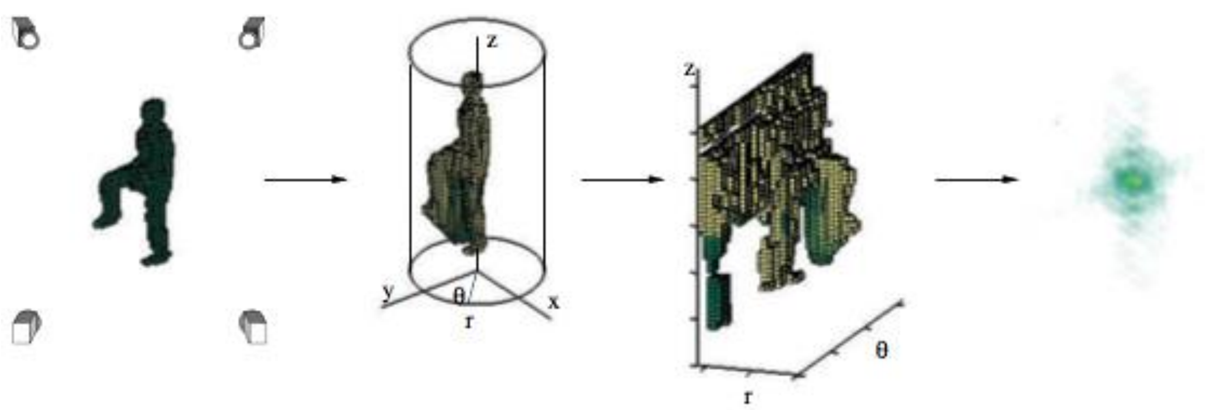


Figure 2-4. The expanded version of method developed by Bobick and Davis (2001) where motion-history volumes (MHV) are constructed from images captured by four cameras (Weinland, Ronfard and Boyer, 2006)

Similar methods following the template-based approach can be found in works of many scientists attempting to extract and reconstruct objects shapes (Demisse, Aouada and Ottersten, 2015), motions (Efros et al., 2003; Hariyono and Jo, 2015; Lu and Little, 2006; Yhang, 2015), gestures (Shan et al., 2004) and even facial expressions under occlusion (Rao et al., 2009). One of the biggest difficulties in all of them remains the requirement to be able to implement an efficient method for extracting accurate data from cluttered environments.

When the segmentation of an object becomes unreliable due to poor quality of the video footage and severe occlusions and unwanted artefacts occur, a solution may be to limit it to a partial extraction of features only. This can be still useful for capturing data on the object's individual moving parts. There are two variations of this method; a *constellation model* and *non-constellation model.* In the former, the object is represented by a set of parts where each part contains some visual information and it is assumed that there exist some geometric constraints linking the parts. In the latter, a spatial location of each part is disregarded to represent a more relaxed structure of the object. Both variations share a common strategy - detecting local features and their spatio-temporal changes rather than the whole objects in relation to the global world.

The algorithm presented by Chen and Hauptmann (2009) is based on the idea that every object is treated as a collection of points of interest from its distinctive parts. The authors name three important steps in the development of part-based representation: detecting points of interest carrying enough information for successful recognition of an action, constructing a valid description of the detected points of interest and building of a classifier for the actions. What is worth noting is that the part-based description of an action can be done with the use of "bag-of-words" concept, where points of interest of a given image (or video frame) can be formed as a cluster surrounding the contour of a silhouette (Onofri and Soda, 2012; Mukherjee, Biswas and Mukherjee, 2011). Such a cluster can be then treated as a visual word that describes a local portion of an image. This process in turn can result in the generation of a visual-word vocabulary, providing a description of repeated patterns found in the image (Yang et al., 2015). In other words, if the clustering occurs in every frame of the video footage in a sequence then it is possible to determine movement of a specific part of an object.

Part-based approach can become quite useful for extracting certain features of an object from a cluttered environment, but sometimes even then it may not be feasible. When surveillance scenario does not allow accurate object segmentation and reliable tracking, it becomes more logical to disregard the identities of the individual objects and perform the processing directly on the pixels. In its simplest form, this method concentrates on the image analysis in a way that allows distinguishing between the foreground pixels and the background pixels (Zhu, Mastorakis and Zhuang, 2015). This in turn allows creating a history of moving foreground pixels over time without the need of detecting and tracking specific objects in the scene (Gong and Xiang, 2003; Gong and Xiang 2006). It implies that movement of a group of pixels in the video sequence can represent an object on the move, but it does not necessarily mean that an accurate labelling of an object consisting of several moving parts can be made. This method has been proven to be more suitable for tracking spatio-temporal changes of a single object motion in traffic environment using map of registered vectors, called optical flow (Bhaskar, Yong and Jung, 2015; Xiang et al., 2016).

The representations of movements as an optical flow have also been useful for analysing crowds in high-density areas where cluttering and occlusions are likely to appear. In the method presented by Rao et al. (2013) there are five stages of the analysis (Figure 2-5). At the pre-processing stage, the frame images are being processed from RGB to Grayscale. 2D Gaussian filter is then applied to reduce the overall noise while preserving crucial information such as edges of shapes contours (Cabello et al., 2015; Nguyen and Blumenstein, 2011; Khorbotly and Hassan, 2011). Similar procedure was carried out by the detection module of Afsar, Cortez and Santos (2015) where background subtraction has been based on Gaussian mixture filtering to determine pixel affiliation to either part of the background or object currently in motion. At the next stage, the optical flow between every fifth frame obtained during this procedure is calculated. The process finishes with application of spatial and temporal filtering to acquire spatio-temporal information about the movements in a matrix form that can be then classified.

The experimental results are promising because they indicate the possibility to approximate reliably objects locations, their directions of movement and velocity even in a highly congested environment. However, since the authors use Horn-Shunck method that assumes the maintenance of certain level of brightness and low object displacement between frames, the video frames need to be pre-processed additionally (Horn and Schunck, 1981). There is a danger that this might quickly become a bottleneck in applications where performance is of utmost importance.

Additionally to this, Hospedales, Gong and Xiang (2009) propose to divide the camera view into uniform size cells and calculate the cardinal direction of optical flow in a given cell. This can be done by matching the actual magnitude of the flow to a preset threshold for enhancing the robustness of the method. Similar approaches utilising the concept of segmentation can be found in the literature on temporal (Mashtalir and

Mashtalir, 2016; Cernekova, Nikolaidis and Pitas, 2006) and spatial (Yu et al., 2015; Jia-Wen, 2015; Dorea, Pardas and Marques, 2005) partitioning.

Contrary to the methods concentrating on extracting motion information about objects, there are attempts to reconstruct the object-independent paths of events occurring in the visual scene (Tran and Yuan, 2011; Wang and Ji, 2016; Nevatia, Zhao and Hongeng, 2003). In this approach, the main goal is to find reliable descriptors of the events occurring in the visual scene rather than to reconstruct the objects themselves or their trajectory. The relationship between the identified events can be computed and evaluated later on by performing a clustering, for example. An example of this method can be found in the research of Xiang and Gong (2006).

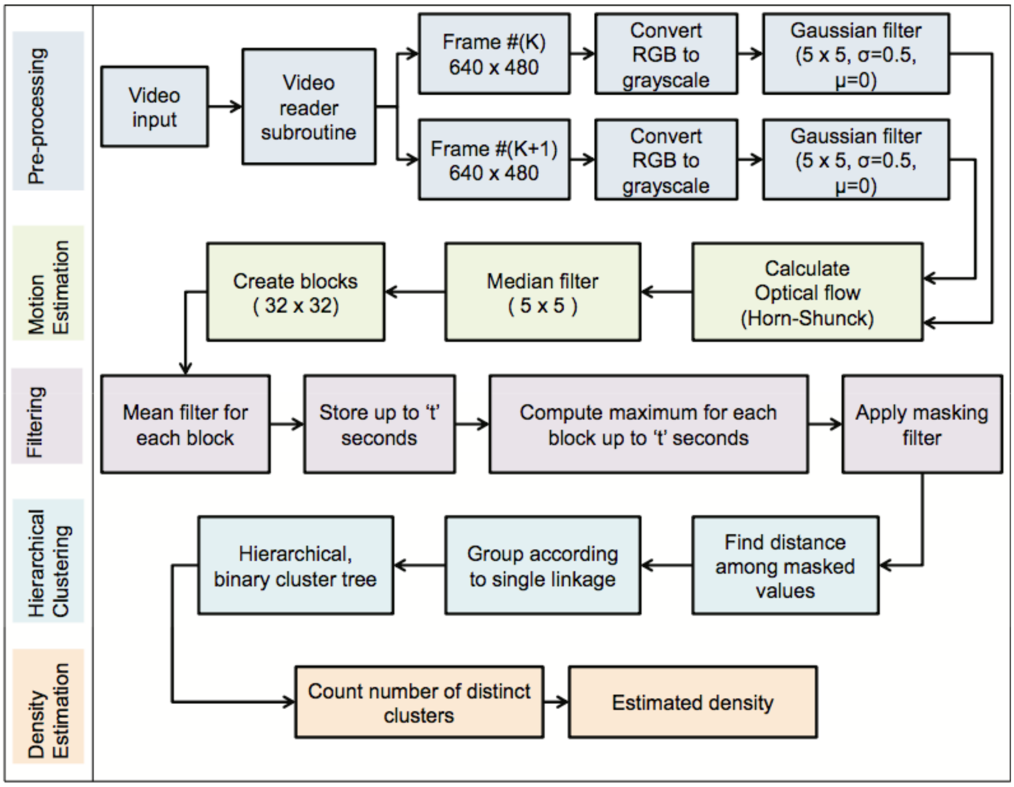


Figure 2-5. More detailed flowchart of crowd density estimation (Rao et al., 2013)

There are also methods that try to recover a 3D geometry from either 2D static images (Bao and Iwamoto, 2016; Sattler, Leibe and Kobbelt, 2011; Kim et al., 2015) or directly from the video frames (Zhao et al. 2010; Chen et al., 2011; Zimmermann, Kim and Shahabi; Fitzgibbon and Zisserman, 1998). The framework presented by Sattler, Leibe and Kobbelt (2011) consists of three modules (Figure 2-6). The first one is responsible for associating each visible 2D feature with the visual word. The second module that performs linear search of the potential matching features verifies the correspondences. The last module then utilises 2D-to-3D mapping to estimate the location of each point using the n-point estimation.

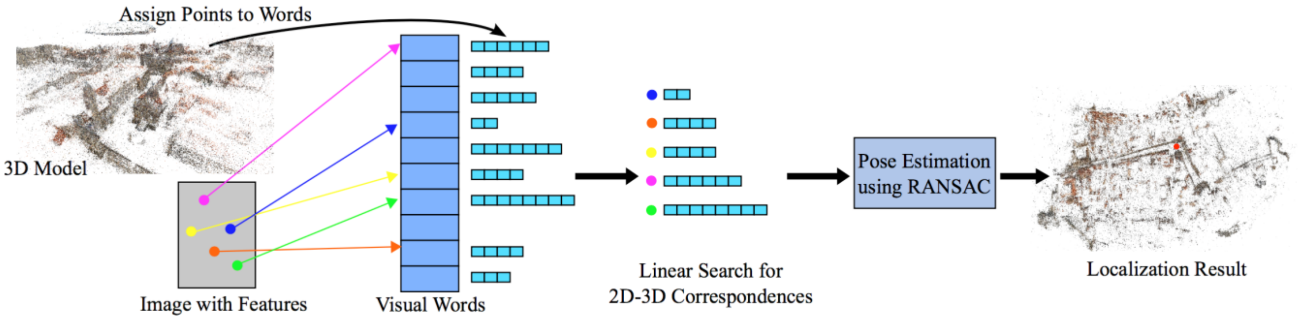


Figure 2-6. A 2D-to-3D matching framework for reconstructing 3D body from a 2D image (Sattler, Leibe and Kobbelt, 2011)

The results show that fast and accurate reconstruction can be done by reducing the localisation of the points of interest to a mean point. The mean point is calculated as an average Euclidean distance. The experimental results of applying the 2D-to-3D framework confirm that 2D images of some bodies potentially can be recreated automatically and represented in a Cartesian coordinate space as a 3D model with satisfactory precision. The bottom-right corner of the picture on Figure 2-7 presents an original query image. The middle part presents the reconstructed model and the top-left corner – the same reconstructed model from a different angle.

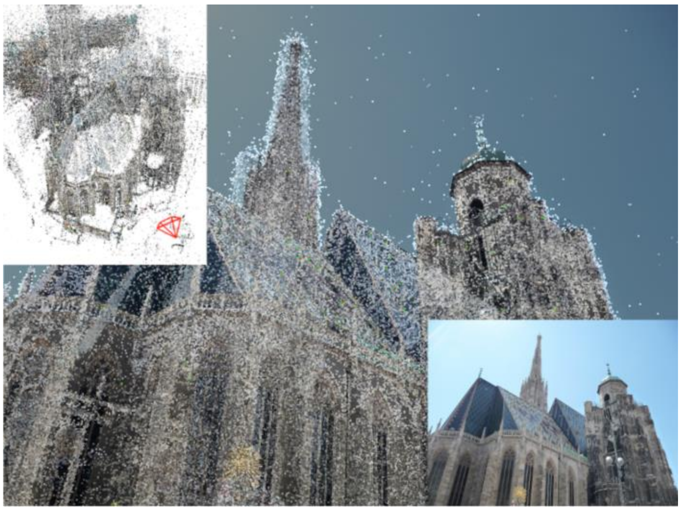


Figure 2-7. The framework proposed by Sattler, Leibe and Kobbelt (2011) allows reconstructing entire locations from recorded 2D images with high accuracy

There are more sophisticated methods for extracting and analysing imagery features related to motion in video footage since their use is not limited to the area of visual analytics. Although a vast literature in Computer Vision focuses on tracking spatio-temporal changes in a video feed for the purpose of analysing the individual behaviour and crowd control, in Computer Graphics there can be also found attempts to model dynamic behaviour and then to simulate them in various environments under specific conditions for other purpose, like computer games or intelligent robots, to mention a few.

2.2. Classification of Activities in Individual and Group Dynamics

Classification in the most general case is a process of organising a flat set of data into a structure based on finding common features. Classification in visual analytics targets the individuals and groups observed in the visual scene. The main focus of this part of the literature review is made on recognition of their actions because they are a fundamental construct of dynamic behaviour patterns. The classification process can help one doing it.

Some researchers have developed a range of methods for human posture and action recognition through classification (Maik et al.,2010; Cucchiara, Prati and Vezzani, 2005; Aloysius et al.,2004). In the application presented by Hu and Wo (2010) one can find a two-step procedure for detecting and recognition of human activity. Initially, the human features are extracted from a video with a *template match* method based on estimating the edge gradient orientation (Figure 2-8). *Hidden Markov model* (HMM) is then used to do the behaviour classification on the base of extracted features. The authors concentrate on extracting the positions of the limbs such as knees and elbows because they provide critical information about human pose. They claim that individual body parts can be detected according to the gradient orientation and their connections can present human posture contour. The variation in limbs positions can therefore result in the recognition of a different behaviour.

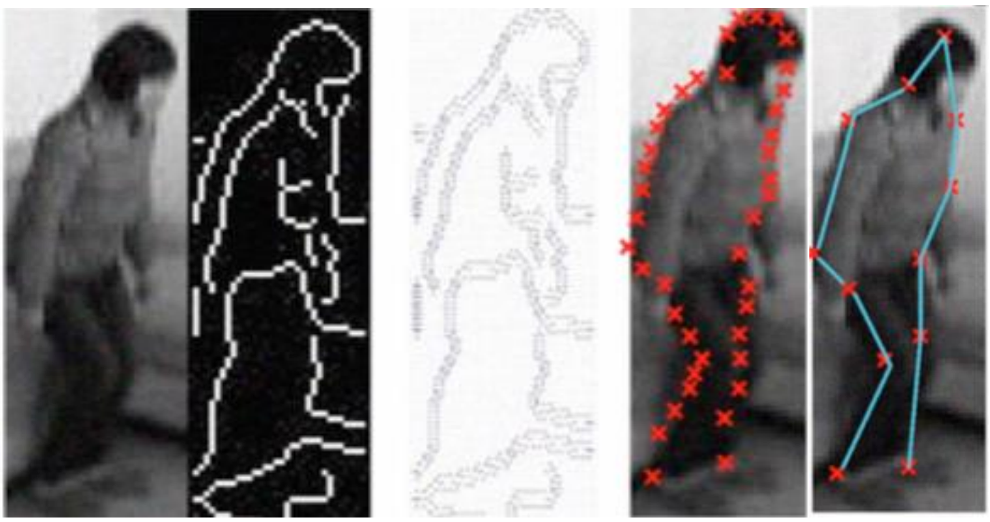


Figure 2-8. Different stages of image processing that result in the extraction of human posture contour (Hu and Wo 2010)

The method has been proved to be efficient, but it is only capable of reporting simple activities carried out by individuals, such as sitting or walking. This is satisfactory for simple visual analytics, where the work is concentrated more on the process of extraction of visual features and establishing their meaning rather than for analysis of their participation in more complex behaviour patterns.

Yordanova (2011) presented a better method that utilizes atomic action templates. In this approach, a typical set of activities carried out by humans recorded in predefined templates is used to match the events occurring in the visual scene. This allows determining what kind of situation is exhibited during particular dynamic scenario. For example by recording the actions and their order of execution as templates it is possible to reconstruct a totally ordered action plans. The extension of this approach adds classification hierarchy consisting of three layers (Gong and Xiang, 2011). Each layer represents a bigger building block upon which activities can be formed. At the bottom of this hierarchy are the so-called 'atomic' actions that correspond to the smallest actions that human can carry out, such as 'moving a leg'. On the next level, there are actions formed by a series of atomic actions that result in fulfilling particular purpose, such as 'walking'. At the top level, a sequence of actions define 'activities' that in essence describe what a person is doing over a period of time in particular space. For instance, moving upper limbs in a particular manner can constitute waving, which in turn can result in a situation where someone greets another person.

Afsar, Cortez and Santos (2015) propose a two-step approach for recognising actions. Their algorithm uses a HMM to recognise postures from previously detected human silhouettes. The method then proceeds with transformation of human postures gathered in sequential frames into feature vectors that are finally quantised into symbols. A database of posture symbols is being created as a result of this process, where each symbol represents different action. The recognition occurs when the extracted feature vector is matched with a symbol residing in the database. Although the experimental results have shown high accuracy in recognition of sitting and walking under difficult weather conditions, certain limitations are clearly visible. First, the method does not account other aspects of the actions such as their relationships with other detected objects, e.g. sitting on the bench, walking down the stairs, etc. Secondly, there is no way of detecting a movement or the viewing directions of the participants. Thirdly, the recognition algorithm takes posture silhouette as an input, which implies that individual joints actions, such as waving a hand or kicking cannot be classified.

A different approach was used in the work of Maik et al. (2010). With the assumption that spatial information of body joints can be estimated, the proposed method classifies the human poses by measuring the angles between individual joints as shown in Figure 2-9. The body of a human is described as a collection of 23 joints where each joint is described on three axes and constrained according to its anatomical degree of mobility. The entire set of joints is further divided into three layers according to their level of contribution to pose formation, which is purely logical since only limited number of limbs joints are used in the execution of most human activities, like for example legs being used for walking.

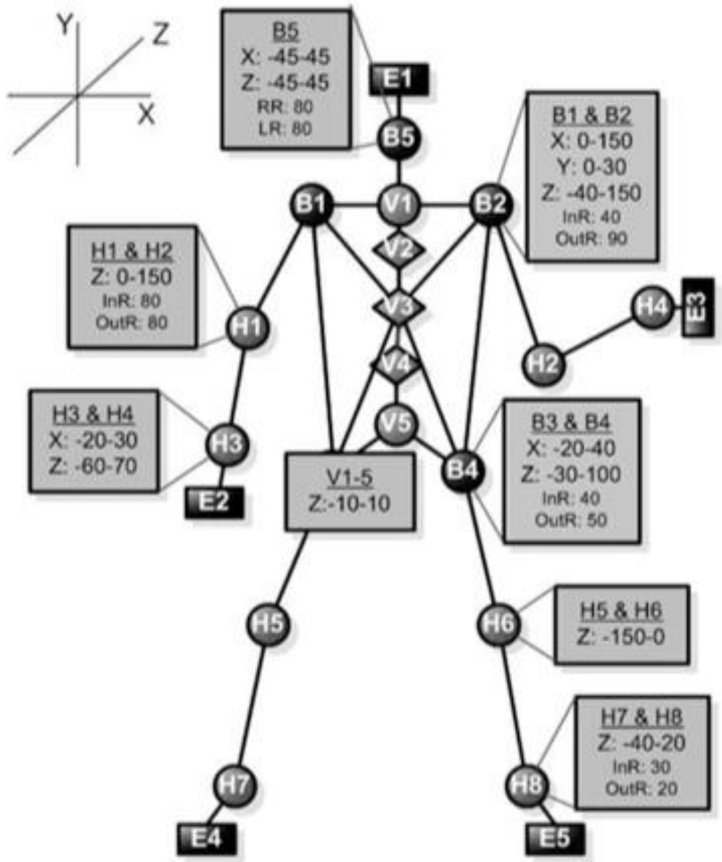


Figure 2-9. Body joints representation along with their minimum and maximum angular variations (Maik et al., 2010).

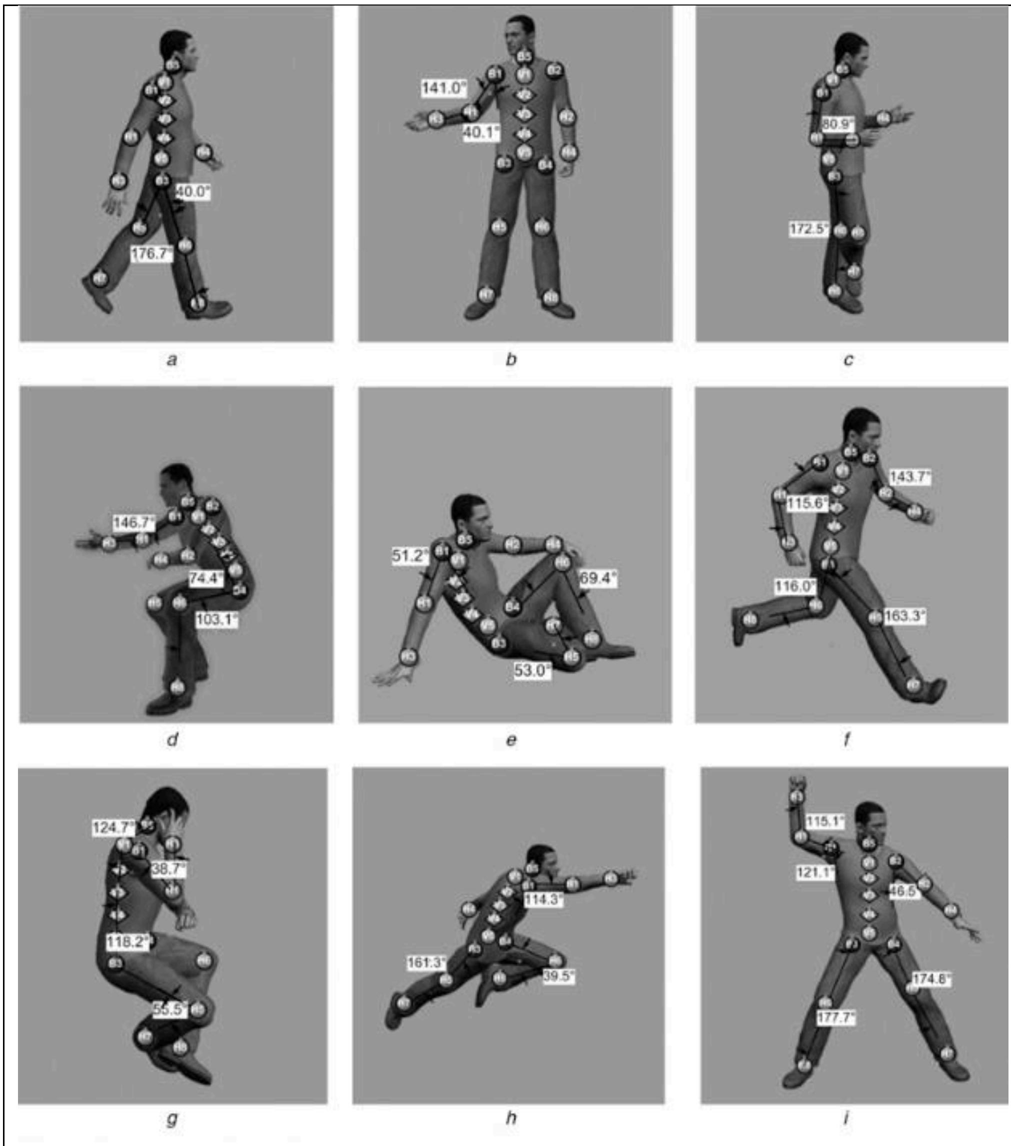
There are two types of joints in this model: major joints and minor joints. Major joints are the ones participating in the genesis of poses, usually related to locomotion, such as arms and legs. The minor joints, such as neck and spinal joints, typically contribute to the emergence of fine-grained interactive actions, but do not influence the development of the poses.

Every joint being part of the body possesses minimum and maximum angular variations, so that a table of their ranges can be created and used as a base for classification (Table 2-1). The pose is then classified according to the variations of angular measurements of the joints in the three-level hierarchy. The poses formed by major joints are being classified to the highest level of the hierarchy in the beginning of the classification process. The interactive actions are being categorised at lower levels at a later stage. The reason for this ordering is the fact that small angular changes of major joints may have a great impact on the classification process because they may lead to erroneous results, e.g. walking might be confused with jogging.

Table 2-1. Table presenting the angular variation range for given poses on three axes (Maik et al., 2010).

Body joints	Pose examples	Joint angular range		
		X	Y	Z
$(E_2{-}E_5)$	grip/hold	0	0	75
B_5	head shake	0	$B(-80, 80)$	0
	nod	0	0	(120,180)
	bow	0	0	(90,180)
$(H_1{-}H_4)$	lift object	0	0	(30,150)
$(V_1{-}V_5)$	bend	0	0	(170,180)
$(E_2{-}E_5)$, $(H_1{-}H_4)$	beckoning	0	0	$E(0,75)$, $H(60,120)$
$(B_1{-}B_2)$, $(H_1{-}H_4)$	pick, lift	$B(0,90)$	0	$B(-30,90)$, $H(0,150)$
	kick	$B(0,45)$	0	$B(-30,90)$, $H(-100,0)$
	walk	0	0	$B(-15,30)$, $H(-30,45)$
$(B_1{-}B_2)$, $(V_1{-}V_5)$	bend down	0	0	$B(30,210)$, $V(-10,10)$
	bend side	$B(0,45)$, $V(-10,10)$	0	0
	turn around	0	B$(-80,80)$ and $V(-75,75)$	0
$(E_2{-}E_5)$, $(B_1{-}B_2)$, $(H_1{-}H_4)$	throw, push, pull	$B(0,150)$	0	$E(0,75)$, $B(-30,120)$, $H(0,120)$
$(B_1{-}B_2)$, $(H_1{-}H_4)$, $(V_1{-}V_5)$	bend and lift	0	0	$B(30,180)$, $H(30,180)$ and $V(170,190)$
$(B_1{-}B_2)$, $(H_1{-}H_4)$, $(V_1{-}V_5)$, $(E_2{-}E_4)$	throw, push, pull	$B(0,150)$, $V(-10,10)$	0	$E(0,75)$, $B(-30,120)$, $H(0,120)$ and $V(-10,10)$

The use of classification in recognition of human poses and their association with potential activities shows promising results in 2D setting. In one of their experiments the authors successfully proved also the applicability of this approach in the case of 3D as shown in Figure 2-10.



(a) Walking

(b) Shaking hand

(c) Chatting

(d) Combined sitting and shaking hand

(e) Sitting

(f) Running

(g) Interacting

(h) Sword fighting

(i) Threatening

Figure 2-10. The angular variations of joints and their impact on classification of different poses (Maik et al., 2010).

2.3. Simulation Models for Visual Scene Analysis

Computer simulations are used in many different areas of science and industry to help the design, construction and evaluation of systems that can be too complex to prototype. Simulations are executed in the form of computer programs that are based on mathematical description or model of a real system (Encyclopedia Britannica, 2017). The execution of a program produces data containing description of mathematical dynamics representing the behaviour of the real system that can also be graphically visualized in a form of computer-generated image frames recorded in animation sequence (Encyclopedia Britannica, 2017). Analysts test different case scenarios so that the most efficient systems fulfilling the original requirements can be found using the data that is accumulated directly from the simulation. The simulation approach is natural in computer games and animation but in visual analytics it is still in its infancy. Saboia and Goldenstein (2011) indicated the importance of understanding the movements of individual participants in crowds within closed spaces for the purpose of safety planning, evacuation coordination and potential improvement of the layout of public spaces. Sung, Gleicher and Chenney (2004) also stated that high quality behaviour simulations might be a core feature of educational and training applications. A comprehensive evaluation of simulation accuracy of crowd models has been presented by Zhao, Cai and Turner (2015).

Depending on the requirements, there are broadly three different types of computer simulations: *real-time simulation* (i.e. where the data is generated and simulated on the fly) (Bélanger, Venne and Paquin, 2010), *pre-processing* (i.e. where the data is first processed by the program and next simulated, usually for visualisation) (Hassan et al., 2011) and *post-processing* (i.e. where the data is processed after the simulation) (Alexandre, David and Innocenti). In the context of dynamic behaviour analysis, scientists tend to focus on constructing online simulations, where the actions and events are determined in real-time basing on some well-defined criteria for emerging circumstances (Joselli, Silva and Clue, 2014; Musse and Thalmann, 2002; Wang et al., 2011). An alternative approach was developed by Iuppa and Borst (2009). They suggest analysing the information processes occurring within the simulator itself. Based on the simulation outputs, one can establish what is required by a system that is being designed. By providing different inputs, it is possible to define different models manifesting situations that can occur in real life. This approach is constructive and efficient but it does not allow to control the adequacy of the simulation model to the external world and because of this is less useful for analysing behaviour.

In order to build a computer simulation of the visual scene, one has to understand both the logical semantics of the world, which is modelled, and the operational semantics of its changes, which is executed during simulation. Sun and Wang (2008) present a method for developing a simulation. The simulator was first cognised on three levels, namely philosophical, theoretical and technical levels, and then the appropriate models were created. As a result, the simulator integrated them seamlessly into a world simulator, which accounts all three aspects. In essence, the world needs to be investigated at least in terms of the structural and dynamic features it exhibits so that the simulation can faithfully resemble its dynamics.

One could outline three broad categories of human behaviour models: *force-based, agent-based* and *event-based*. Although the underlying theory behind each of them may be different, in case of 3D simulations it revolves around vector calculus, matrix transforms and quaternions (Tebjan, 2011; Lengyel, 2016; Kowalczyk, Niedzialomski and Obczynski, 2011; DeLoura, 2000; Lengyel, 2003; Sellers, Wright and Haemel, 2013). They are absolutely sufficient to allow defining the location, translation, rotation and scaling of any spatial entity within the coordinate system of the scene and dynamically modifying of its physical appearance and behaviour in the viewport.

Navigation through the space plays a special role in visual simulations of dynamic behaviour due to the need to control the locations analytically without breaking the physical laws. The main difficulty is to accurately plan and generate the most probable trajectory for an entity residing in the virtual world while avoiding collisions. Formulas for calculating the potential trajectory within the boundaries of the physical world are a hot topic for many researchers in various areas of computer science, ranging from robotics (Bogdanovych, Bauer and

Simoff, 2009; Mlynek and Martinec, 2014; Zhang and Zhou, 2007) to computer games (Sasiadek and Duleba, 2000; Milam, Mushambi and Murray, 2000; Pelechano, Allbeck and Badler, 2007). Depending on the implementation domain, there are different factors to be accounted, but what most of those models have in common is that they are all based on the use of Cartesian coordinate or equivalent regardless the dimensions of the virtual space which can be 1D, 2D or 3D.

In computer games it is also common to use path-finding algorithms, which are responsible for finding shortest routes to the target (Software Developer's Journal, 2014; Lester, 2005; Wang and Lu, 2012; He, Wang and Cao, 2012; Terzimehic et al., 2011). The famous algorithm of Dijkstra and A* are the most popular amongst them. The idea of this approach is that the layout of virtual space is represented in the form of global navigational mesh of location cells. Each cell of the mesh has an assigned weight value, which is used in the calculation of the cost of movement to a neighbouring cell. The lower cost means that there is a higher chance an agent will move to that cell in the next step. This method has been successfully implemented in many projects, but it does not consider other variables that might have an impact on motion of dynamic entity, such as mental or physical state of the agent. It can, however be used as part of the trajectory execution algorithm.

Simulations that do not use navigational system are possible. They are typically based solely on controlling quantitative parameters, such as timing errors or virtual time (Sasiadek and Duleba, 2000). Zhang and Zhou (2007) use matrix transformations to describe the position of a robot in 3D space, using the modified version of D-H notation. However, the trajectories contain very useful information that can be used for reconstruction and recognition of agent’s behaviour or intentions (Terzimehic et al., 2011). For this reason, one should consider developing an agent model that would be complex enough to allow controlling realistic movement patterns.

Simulations of human behaviour are typically very different for individuals and for crowds but what is common in all of them is the presence of real world motions and activities carried out by agents. It is natural to assume that whenever an individual is part of a bigger crowd, its motions will be more constrained. Depending on the level of relationship between people, they will attempt to maintain an appropriate distance between others (Kendra, 2016). On the other hand, when an open space is shared by a low number of individuals, their movements will be more liberated. In such a situation, people will focus more on avoiding any physical obstacles rather than circumvent others. As a consequence, the trajectories of individuals within the crowd are easier to track. The abnormal behaviour that might emerge from, for instance, sudden changes in velocities of movements can be easier to recognise. Regardless of the density of the crowd, when a three-dimensional space is considered it is important to incorporate the laws of physics into it. This way the simulated micro-world cannot only be physically more realistic, but also potentially more informative deliver for the analysis. The individuals on the scene are considered being potentially grouped and special attention is made on their participation in this research.

2.3.1. Force-based models

Several researchers dedicated their efforts to applying dynamic forces to simulate movements of individual agents as part of a crowd (Ali and Shah, 2007; Ali and Shah, 2008; Mehran, Oyama and Shah, 2009; Wu, Moore and Shah, 2010; Lin, Grimson and Fisher, 2010). In this approach, each agent is perceived as a particle that is subjected to forces, such as physical, psychological (Helbing, Farkas and Viscek, 2000) or social which force them to execute motions accordingly (Zhong et al., 2014). There are also models, which account the dynamics of the density and velocity of the crowd itself. In such cases the modelling requires more complex partial differential equations (Hughes, 2002; Huang, et al. 2009; AlGadhi and Mahmassani; Treuille, Cooper and Popović, 2006).

Each of these models considers various aspects of crowds, but the main issue remains whether the crowd is of homogeneous or heterogeneous nature. The homogenous crowd exhibits aggregated behaviour, where single individual is part of the majority and moves coordinated with the flow. In heterogeneous crowd, each agent acts independently, taking into consideration its current state and requirements for satisfying its personal needs.

This leads to another question, namely if the simulation needs to be carried out either on microscopic or macroscopic level. If one considers the crowd on microscopic level, most of the attention should be paid to the responses of individual entities to the surrounding obstacles and events developed as a result of a kind of chain reaction. Opposite to that, on macroscopic level there is a need to find a complementary relationship between common goal of individuals who participate in the crowd and the overall features that control the flow of the crowd, such as density and speed, for example.

The major flaw of force-based models is that in simulations of crowds of low density occupying large areas the single agents may exhibit erratic movements, which is counterintuitive, i.e. a pedestrian moving in opposite direction to his target destination location of the crowd (Saboia and Goldenstein, 2011). As a result, the observed movements resemble more the movements of the sub-atomic particles rather than the humans. Although there are ways to minimise the unusual movements of the agents by incorporating additional concepts, such as cellular automata, which are able to obtain information about close-range neighbours (Guo and Huang, 2008; Varas et al., 2007; Perez et al., 2002), or stochastic automata with movements based on probabilistic theory (Muramatsu, Irie and Nagatani, 1999; Tajima and Nagatani, 2001), this approach is still more focused on construction of the trajectories of the agents rather than on their activities and interaction with the surrounding environment. Furthermore, it needs to be accounted that each agent possesses some kind of emotional state and cognitive abilities to learn about the surroundings through which it can develop knowledge and make decisions on its own.

2.3.2. Agent-based models

Several researchers made an attempt to sort out these problems by taking a completely different view of the problem through accounting the full autonomy of the agents in the crowd (Hluchy et al., 2011; Rossmann, Hempe and Tietjen, 2009; Ben et al., 2013; Qin and Wei, 2010; Sharma and Lohgaonkar, 2010; Li, Tang and Simpson, 2004). In the agent-based simulation approach, each agent has an internal structure and state, steering its behaviour. This introduces additional complexity in the overall model of the simulation, but at the same time it enhances the realism of the exhibited behaviour, making the crowd more heterogeneous and the agents more autonomous.

Some of the simulators, which follow this approach, have layered architecture, where the different layers reflect the level of collaboration and allow an agent to make its own decisions within certain level of autonomy. The highest level decisions are responsible for real-time active planning, i.e. capturing and processing the available information and passing the outcomes of it to the middle level layer, which in turn calculates the trajectory and/or action. The execution of the plan is usually done at the bottom layer.

This approach allows accounting higher level cognitive skills and is advantageous where there is a need to consider emotional and social state as well. Urban and Schmidt (2001) propose the so-called PECS architecture (Physis, Emotion, Cognition, Social Status), which can be used as a reference in designing of an agent model best resembling humans appraisal and decision making processes. This framework was originally developed for the purpose of simulating and analysing the behaviour of soldiers during military operations. The agents are described with a set of personality attributes and internal processes that generate behaviour based on their psychological and emotional states.

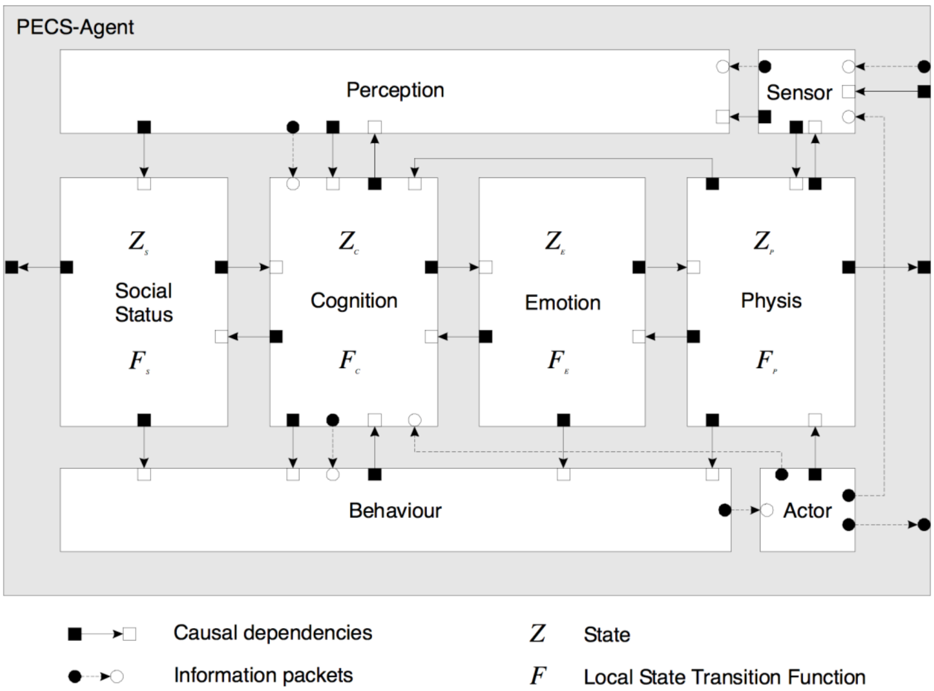


Figure 2-11. An agent-based PECS architecture (Urban and Schmidt, 2001)

The PECS architecture decomposes the overall complex model of the agent into several smaller, interconnected component models (Figure 2-11). Each component possesses its own internal state that is influenced by the inputs and the outputs of other components, depending on the requirements. The **Perception** and **Sensor** components gather the information about agent surroundings. The collected data is then passed onto the internal middle layer where it is processed according to the set of predefined rules by the PECS components. The **Physis** component is concerned about agent appearance and physical changes that might be a result of an action or passive events. The **Cognition** component is responsible for storing agent's information about the environment and its internal state on a basis of delivered data. The authors also suggest introducing the concept of an agent losing part of the possessed knowledge due to natural human factors such as memory loss. The **Social Status** component provides a set of attributes defining an agent social affiliation within groups. This component influences each agent behaviour according to its social needs and, as the name implies, social status. Since PECS model's assumption is that emotion play key role in the emergence of agent behaviour, the **Emotion** component is responsible for generating emotional states and processes that have an impact on the final action. The generated emotion is a result of cognitive appraisal and information processing by other modules. An adequate action is selected by **Behaviour** and **Actor** components on the basis of the outputs of middle layer's components.

The structure of PECS architecture was designed primarily from the perspective of testing various cognitive theories by introducing different extensions to the basic model. One of the suggested extensions is to incorporate additional planning component that generates an agent's personal goal. The goal may be represented by a certain state that needs to be reached in a series of actions in order for it to be fulfilled. The planned strategy for achieving a given goal is dictated by accumulated data processed by the components in the middle layer.

The behaviour of an agent with this structure is always determined using purely statistical data calculated by its individual components which can lead conclusions which are purely formal but do not have proper interpretation from psychological and social viewpoint. An example of this can be found in one of the experiments of Urban and Schmidt (2001). In the simulation of students who want to prepare for an examinations and revise, there is a rule dictating how much each of them has to learn in a group rather than in solitude. But when studying in isolation the level of knowledge would increase or decrease according to the personal capability, while the social satisfaction would diminish over time. The authors describe this phenomenon with a differential equation (eq. 2-1).

```
SocAct ´: = -b * SocCap * SocNormal * SocMakeUp/100 * SocAct
```
(eq. 2-1)

A differential equation describing the social satisfaction deterioration during student's study in isolation (Urban and Schmidt, 2001)

The authors also presented the decrease in social satisfaction of an agent studying alone over time in graphic format (Figure 2-12).

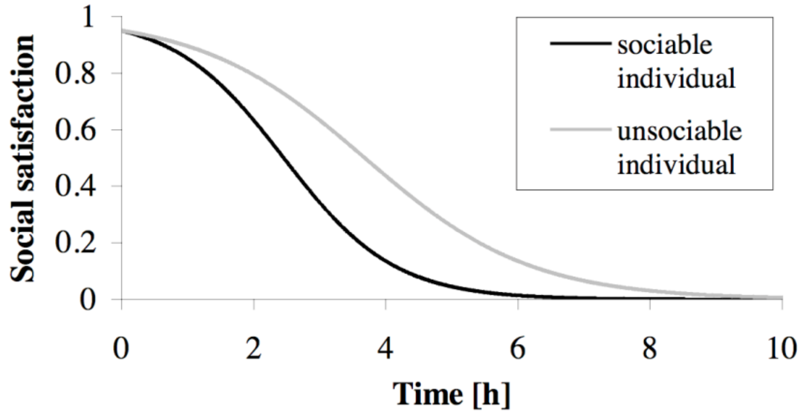


Figure 2-12. Graphics of the decrease in social satisfaction of an agent studying alone over time (Urban and Schmidt, 2001)

PECS was used as a template for development of a generic agent model described by Kvassay et al. (2012) as shown in Figure 2-13. Each action is dependent on motives and internal state of an agent. The behaviour patterns are defined as personal aims developed over time in a process of planning. The plans for achieving goals are devised on the basis of the internal state and knowledge about the environment. At specific intervals, the simulation algorithm selects the most suitable behaviour pattern according to the agent's strongest motive.

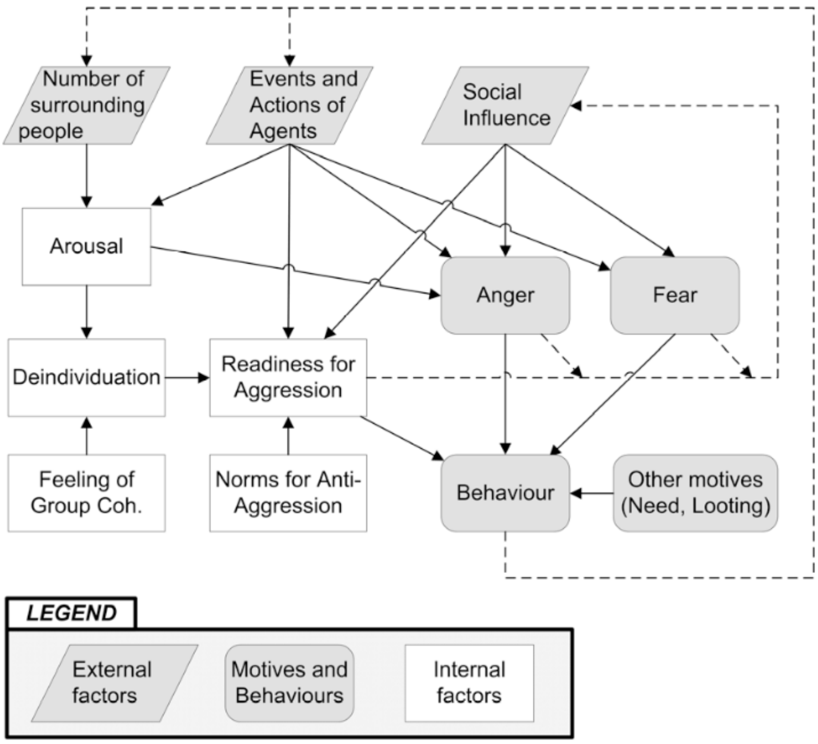


Figure 2-13. The internal state of an agent, its actions and motives are driven by the number of people within a range with their actions, the events occurring and the social influences (Kvassay et al., 2012)

Kvassay et al. (2012) made an attempt of measuring the impact of collective aggressive behaviour of one group of agents towards another (i.e. civilians *vs*. soldiers) that may result in escalation of dangerous situations. The idea here is to determine the level of aggression of each agent based on the following:

- The influence of individual agent's needs and emotions on the group behaviour
- The process of diffusing of an already escalated situation with agent's self-preservation instincts
- The social influence of agents exhibiting leadership characteristics towards the rest of the agents in the same group, i.e. one role model agent may persuade the others to fulfil specific group task

The internal attributes are adjusted according to the external factors (labelled as Arousal on Figure 2-14). Actions of other agents and events occurring in the scene may influence the agent's motives that consequently are used to build a behaviour pattern. The author gives an example of an increased fear of an agent caused by its observation of other agents under high stress and fear. Although the simulation model adopts principles from psychology, the dynamics are built by probabilistic schemes, e.g. the "Fear" dynamics is based on a formula that do not consider real world components such as current location of detected human shape in a video. It is driven by existing "motives" and "states" that can be treated as artificially calculated from pre-set data based on the empirical estimation of purely quantified psychological values.

The authors never considered the possibility of collecting the data from visual scenes observed by camera. Although PECS architecture can be used as a guideline in conceptualisation of any agent-based simulation frameworks, its drawbacks must be considered carefully before employing the model in visual analytics context.

A different approach was taken by Rai and Hu (2013). A modular simulation framework consisting of an agent-based model is driven by data collected from sensors in real-time. The two-layered architecture of the system defines a clear separation of the responsibilities (Figure 2-14). Whereas the simulation model captures the low-level dynamics of the real world, the behaviour detection layer is concerned with the recognition of a high-level behaviour pattern.

The behaviour patterns are recognised using real-time data by a pattern detection module, which utilises the Hidden Markov Model (HMM) method. In order for HMM to recognise the patterns, it is trained with a data accumulated during the observation stage. At this stage the maximum likelihood of each state is estimated and normalised so that the probability of particular real-world state occurrence can be estimated during the simulation. The HMM parameters are calculated using the Baum-Welch learning algorithm (Baum et al.,1970).

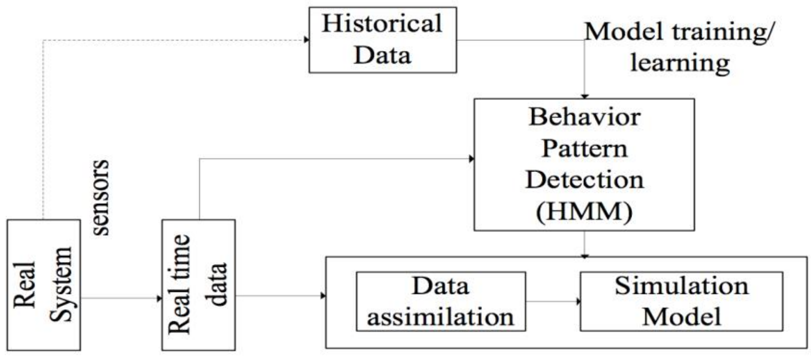


Figure 2-14. Framework for simulation driven by real-time data with pattern recognition (Rai and Hu, 2013)

The data assimilation module is then used to combine the real time data with the recognised pattern to update the state of the simulation. Various data assimilation methods are used to account the observed results in order to improve the state estimation (Bouttier and Courtier, 1999). One such method is proposed by Kawamoto (2010). A pedestrian motion is estimated on the basis of accounting the social forces. In this case the micro-world is bound by the building. The state of the simulation model at any given time is defined by the location, velocity and current destination of every agent within the building. The navigation of each individual agent is projected onto the map of the building floor represented as a waypoint graph. Since the vertices of the graph reside only at the intersections and corners of the corridors on the same floor, the movements of the agents are limited to straight lines and do not cross the walls. The distance controls the collision avoidance between agents. The optimal route is rescheduled each time an agent finds itself within a certain proximity to another.

The standard process of data assimilation starts with the input of all states available at time step t-1into simulation model at step t as presented in Figure 2-15. This process results in a generation of a set of new states that are then used to create a set of new "observations". The generated observations are then compared with "real" observation set. The results are normalised to estimate their significance and all states are assigned correct "weights" for the use by the resampling algorithm. The output is the set of states used as an input for the next time step t+1.

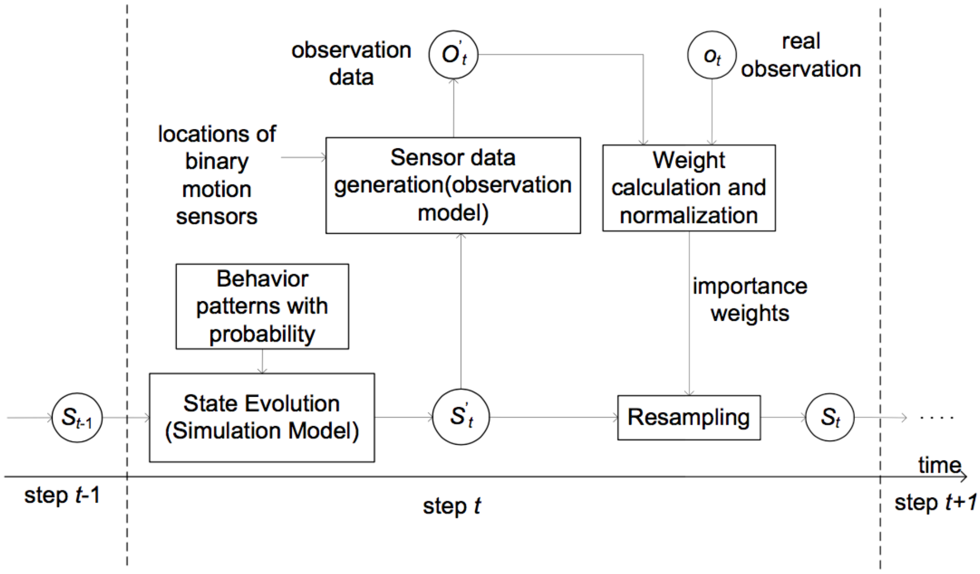


Figure 2-15. Data assimilation process is reinforced with behaviour patterns information output (Rai and Hu, 2013)

Rai and Hu (2013) enhanced this model by introducing the behaviour pattern detection module into the process that provides an additional input about the patterns. At each time step, the HMM first identifies the behaviour patterns of the system and provides them in the form of a set of states to the simulation module. The simulation model then uses this information to model the selected behaviour pattern set. This in turn is used in the sampling process that outputs more accurate results than the simulation without input from the real-time data.

The method of Rai and Hu (2013) was applied to an office environment with a set of sensors deployed in strategic locations such as doors. The sensors were able to register the occupancy of individuals in the covered areas. However, due to technical limitations they were unable to identify individuals passing by certain locations. Additionally, the sensors were prone to errors caused by highly congested parts of the office, and the recognition of group movements was practically impossible. Because of these limitations, the recognised behaviour patterns possess characteristics of a simple binary system with two actions only, like “entering the conference room” and “leaving the conference room” as shown in the diagram in Figure 2-16. The physical state and appearance of the individual agents in the space is therefore lost. However, if the binary motion detecting sensors were replaced with cameras capturing the visual scene in more details, more information would immediately become available. This would allow defining more complex behaviour patterns exhibited by an agent such as "agent A is walking along a shelf" or "agent B picks up a bag from the floor". High-level information like this is very difficult to obtain directly from the camera feed as commented by the author.

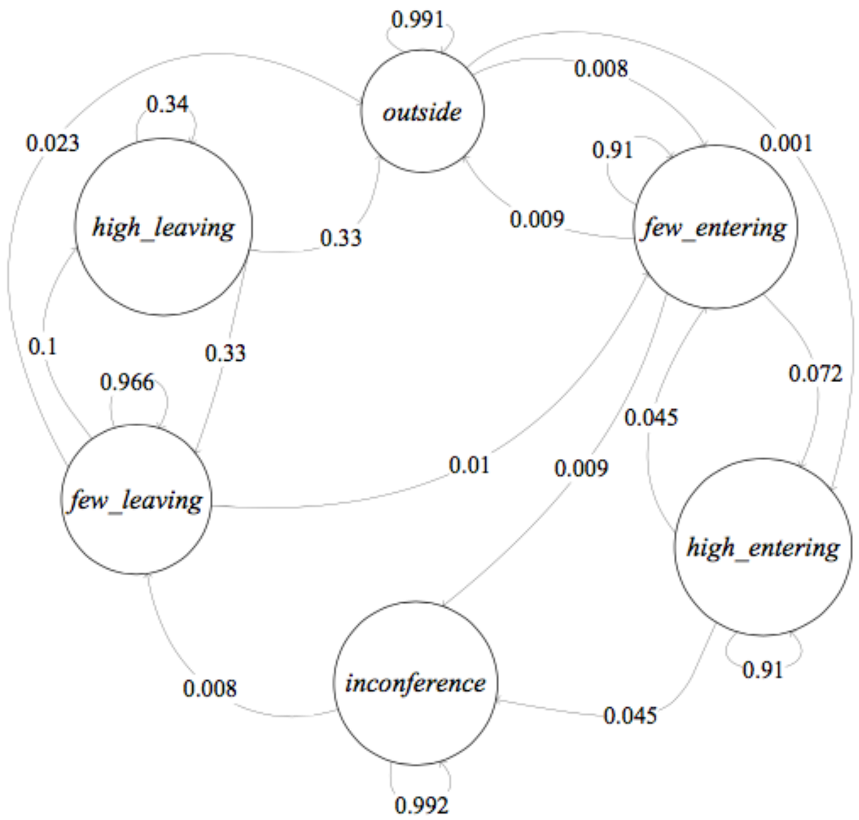


Figure 2-16. The state transitions and their probabilities for Hidden Markov Model pattern recognition in an office scenario (Rai and Hu, 2013)

2.3.3. Event-based models

In the event-based models, the actions are triggered by events and situations occurring in the world at given moments of time (Xiang et al., 2016; Rao et al., 2013; Cabello et al., 2015). The event-oriented characterisation of the actions can also be found in models that are used to formulate a theory of moving objects (such as vehicles in simulations of car-accident scenarios), not only of people (Nguyen and Blumenstein, 2011).

From conceptual and logical perspective, this approach complements agent-based methodology by adding contextual and semantic information about the environment. Typically this is implemented in the form of a separate layer in the overall architecture or at least an additional module, responsible for broadcasting the updates when the situation changes.

The situation-based strategy was adopted by Sung, Gleicher and Chenney (2004) to address the issue of controlling multiple agents in a crowd simultaneously. In this approach, the behaviour of each agent is driven by events occurring in his local environment as well as by its internal state. The high level layer gives an agent details about how it should act adequately to the situation it found it in. At a low-level layer the authors use a scheme that calculates the probability of the possible state transitions. When the process finishes, the result is sampled to move the simulation further along the most probable transition. The authors concentrate more on the anonymous nature of the crowds and are not interested in tracking the individual actions of the separate agents. Instead, in their solution the crowd is controlled by the events that occur in their environment. The events have an impact on how the crowd moves, its direction of flow and so on.

The behaviour is perceived holistically as behaviour of the crowd rather than as behaviour of the individual entities so it is impossible to infer more complex behaviours at a higher level of granularity. Each agent's contribution to the overall behaviour of the crowd is short-term and therefore its identity is not being considered in a long-term evolution for exhibiting more complex behaviours.

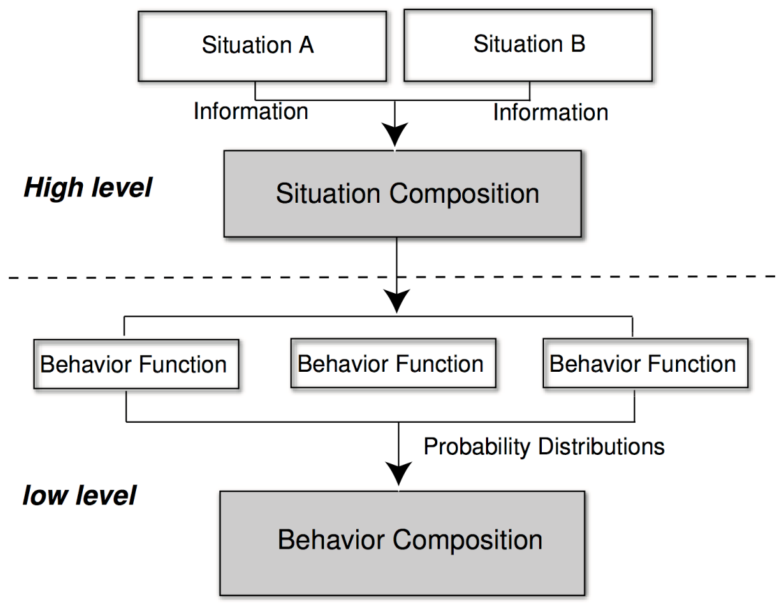


Figure 2-17. The architecture of situation-based framework for behaviour analysis (Sung, Gleicher and Chenney, 2004)

The situation-based approach determines which behaviour function is going to influence the action of an agent at any given time as shown in Figure 2-17. The determinants may vary, but they can be narrowed down to location of the individual agent, what an agent can sense within its local area or the density of the crowd. When an agent enters the area covered by a certain "situation", its model is extended with a behaviour function that allows executing an adequate action. Analogically, when an agent leaves the "situation" area the ability to execute an action adequately to that situation is removed.

In this approach there are two types of situations: *spatial* and *non-spatial*. Once defined, spatial situations remain static and cannot be moved, but they can be removed at run-time. The non-spatial situations are not bound to any specific predefined areas, but rather to dynamic objects such as moving groups. This implies that agents can be either added or removed from non-spatial situations by fulfilling certain conditions only at run-time. There is also a difference in terms of how the two types of situations operate. A spatial situation always adds behaviour components whenever an agent is within a close distance to it and removes them when agent leaves the area as indicated in Figure 2-18. On the other hand, non-spatial situations affect agent's behaviour by assigning itself to its model. By doing so, the agent is under constant influence of the situation that may move along with him. The authors claim that dynamically adding and removing behaviours at run time can result in a more scalable framework.

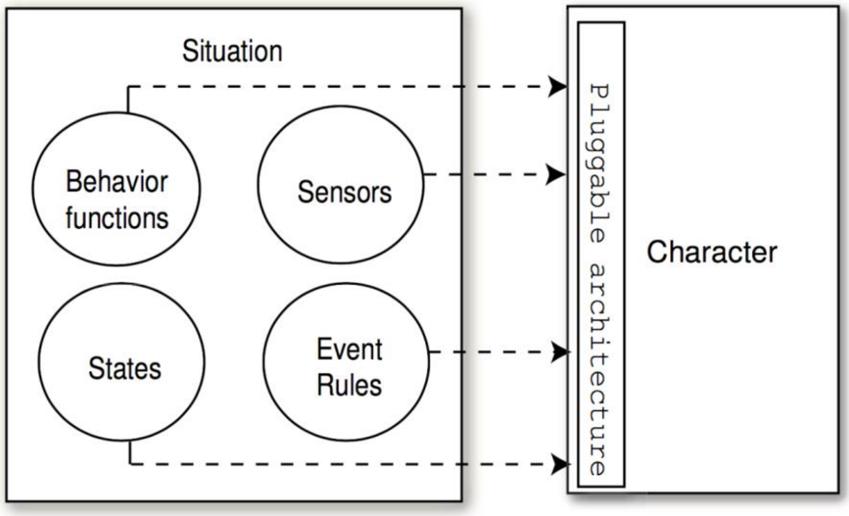


Figure 2-18. The agent receives the information from the environment about events, states, sensors and behaviour functions to select adequate actions (Sung, Gleicher and Chenney, 2004)

Virtual sensors attached to an agent upon entering "situation" area capture the events occurring in the scene. The collected information is then stored in agent's internal data structure for further computations and evaluation of behaviour functions. In the framework of Sung, Gleicher and Chenney (2004) there are four types of sensors:

- Empty sensor responsible for detecting presence of agents in particular area of the scene.
- Proximity sensor, which is used for maintaining certain distance from a location in the scene.
- Signal sensor for checking particular signal status in the scene such as signalling lights at pedestrian crossing.
- Agent sensor, which is used for checking the state of an agent behaviour including position and orientation.

Since every agent is involved in several situations at the same time, its behaviour, especially movements, might become erratic. For this reason, the authors introduced a rule-based system that cancels certain behaviours under specific circumstances. For instance, in a situation where an agent is supposed to sit on a bench, it must not suddenly turn around and walk away from it.

The authors also specified several default behaviour functions when the agent is outside every possible situation to allow its free movement around the scene. These functions prevent collisions with obstacles in agent’s way such as static objects or other dynamic agents and potentially plan its future destination target in the form of a waypoint graph. The behaviour function evaluates the probability of the next state from a given set of available states at the given time. This operation is performed for each behaviour function available to the agent at a simulation step.

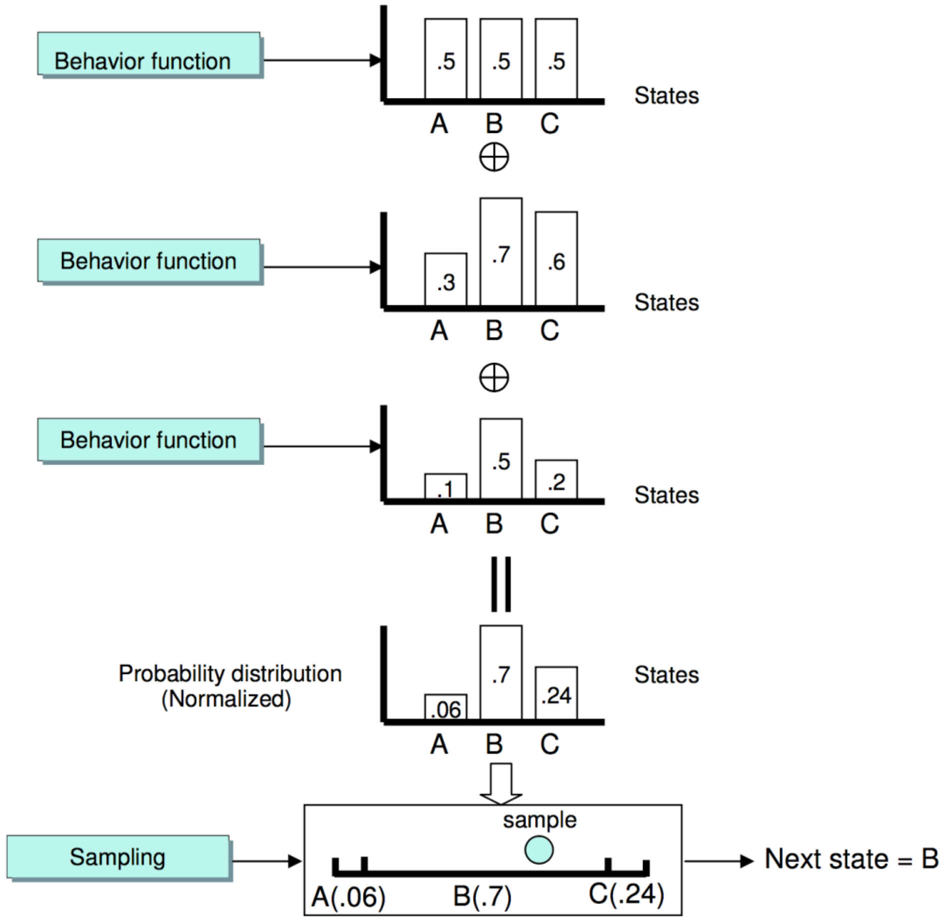


Figure 2-19. The output of behaviour functions are composed and final result is sampled in order to select the next state (Sung, Gleicher and Chenney, 2004)

The result is then composed and normalised using sigmoid function, commonly found in Neural Networks (Callan, 1998). The output value of such function is within the range [0,1]. The authors mention that multiplication of the outputs allows the possibility of one state to cancel another. The sampling of the final result (Figure 2-19) enables an agent to select an action with both high probability and low probability and consequently to generate more random flow of the crowd.

Since the agent's behaviour is dependent on the location of the situations, one of the major requirements is to have a possibility to specify their regions in the scene. The authors developed a tool that allows drawing the desired area on top of the physical architecture of the scene (Figure 2-20). It has multi-layered structure where each layer defines specific situation. The map of regions is then stored as a bitmap file that can be read during initialisation of the simulation.

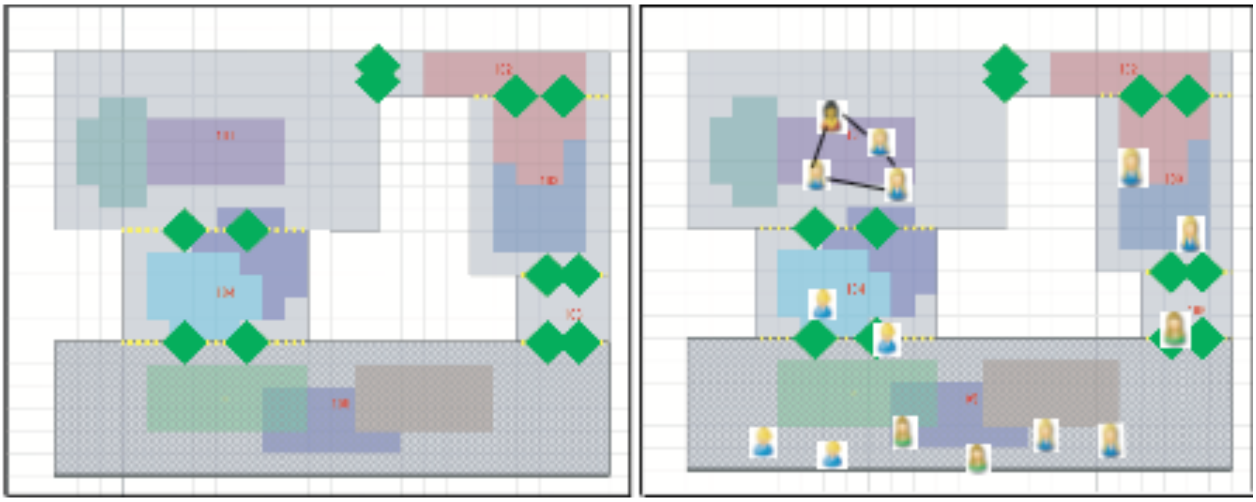


Figure 2-20. A visual tool for setting situation regions within the environment is shown on the left. On the right figure the formation of non-spatial situations out of the group at run-time can be seen (Sung, Gleicher and Chenney, 2004)

To simplify the description of the situations in the scene, one could also attach situations with appropriate radius to successfully recognised static objects as simulator input. The non-spatial situations can also be attached to the median point of the dynamic events, e.g. when a group of agents meet, their average distance to each other can be calculated and treated as a centre point of the situation location. By doing so, the group can be tracked as a singular entity while preserving the identity of each participant.

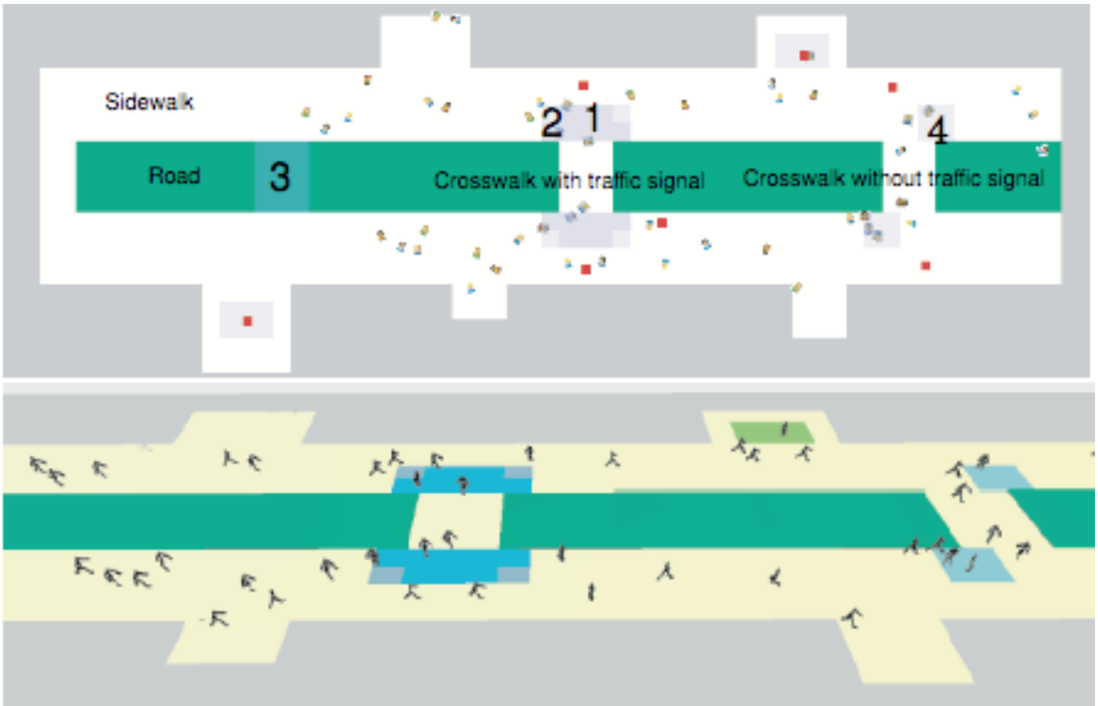


Figure 2-21. Situation areas in a street environment; each situation is labelled with a number for easy monitoring and evaluation of case scenarios (Sung, Gleicher and Chenney, 2004)

One of the simulated scenarios using this framework is a street environment as shown in Figure 2-21. In the visual scene, most of the situations occur at pedestrian crossings. There is a "crossing street" situation defined at both unsigned (4) and signed (1) pedestrian crossings. The "traffic sign" situation (2) can occur at signed pedestrian crossing because of the presence of traffic lights. One additional situation present in the scene is "in-a-hurry" situation (3) that defines behaviour of crossing the street without reaching pedestrian crossing first. At the beginning of the simulation, the agents are walking on pavements on both sides of the street. When any of them reaches the "situation" area, it immediately responds to it. At pedestrian crossing with lights the agents receive information about the light state through the sensor provided by the situation and stand in a stationary position until the light changes to green. On the pedestrian crossing without lights the agents simply cross the street without paying attention to any additional environmental conditions. When agents found themselves in a "in-a-hurry" situation, a set of running activities are added to the set of possible behaviour states so that an agent can cross the street in a hurry.

The authors identify several advantages of using their strategy. First, the design of an agent is broken down to the design of local activities, which means there is no need of constructing complex internal model of the agent. Second, once the actions are assigned to specific situations, the situations can be reused in any other scenarios. Thirdly, the solution provides an efficient way of controlling agent's behaviour because it is based on delivering only partial information to the agent about the world at any given time.

2.4. Analysis of the patterns of human dynamic behaviour

The automatic detection of human behaviour patterns is a hot topic among researchers due to its applicability in a wide range of domains. There exist examples of works whose main focus is to develop an efficient method for analysis and recognition of behavioural patterns. Fraud Detection in online banking, abnormal behaviour after user authentication, detecting anomalies in online traffic to predict user intention or derivation of individual movement behaviour patterns on a basis of spatio-temporal data obtained from GPS sensors or similar are only few research directions. The system developed by BioCatch *"distinguishes between a real user and an impostor by recognizing normal user behaviour and fraudster behaviours, even when no profile exists"* (BioCatch, 2017). The solution concentrates on a range of problems related to the analysis of information that is being entered by online users at different stages of sensitive private webpages browsing or device interaction. The difference between the solution offered by BioCatch and common security practices (e.g. entering passwords) is that the platform continuously analyses user's behaviour and his interactions with the device to detect vulnerabilities at every point during the session. A biometric profile of individual user is built as a result of this process that is subsequently used for recognizing cybersecurity threats such as malware, remote access Trojans (RATs) or online bots activities. BioCatch technology conducts identification of suspicious human behaviour patterns relying on data that is input by users directly to devices. Similar methods can be found in models for identification and analysis of emerging user behavioural patterns during interaction with multimedia social networks (Zhang et al., 2017), mobile phone use (Zhou, Xu and Huang, 2010) or general web browsing (Gao, 2010). The mutual characteristic of these models is that they depend on the data that is explicitly generated by users during their interaction with the system. Such data is not adequate for performing visual human dynamic behavioural patterns recognition since no spatio-temporal information such as location or orientation is enclosed.

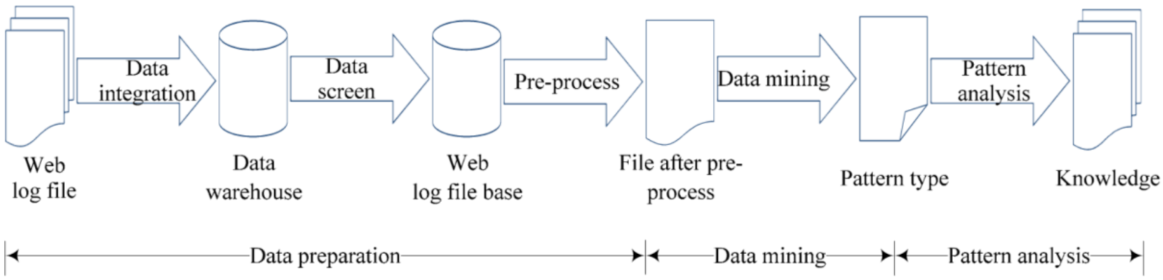


Figure 2-22. The method for behaviour pattern recognition of online users depends on data collected during browsing sessions (Gao, 2010)

The study conducted by Kelly, Smyth and Caufield (2013) focuses on evaluating the hypothesis that demographic and social information of an individual can be inferred from his location data. To that end a global position system (GPS) technology embedded into mobile phones was used to examine the relation between human location behaviour patterns and certain characteristics about the individual. The aim is to build a dataset from location data signals received by sensors and subsequently use them to analyse the individual movements between different geographic locations. Identification of geographical areas that an individual visits on a daily basis can help understanding his routines. Kelly, Smyth and Caufield (2013) have formulated their problem as a clustering problem where they built the map location clusters from individual's location data (Figure 2-23).

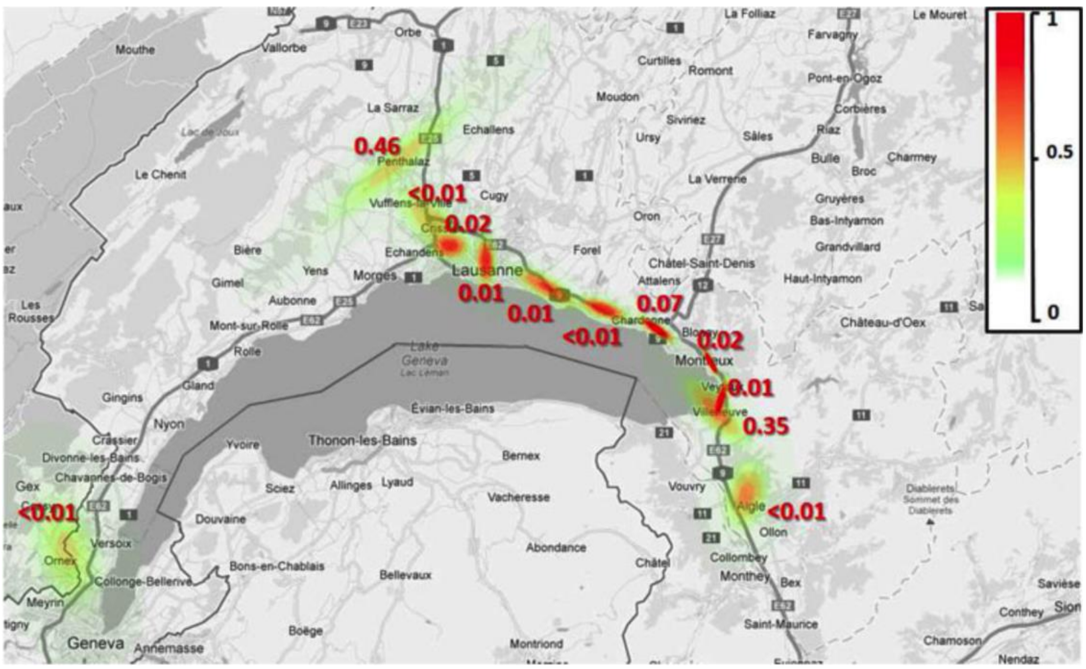


Figure 2-23. The location clusters built on a basis of individual’s location data collected by GPS sensors used in the movement behaviour pattern recognition method proposed by Kelly, Smyth and Caufield (2013)

The Hidden Markov Model (HMM) was then configured to represent the most usual movement behavioural patterns of an individual. The most visited geographical areas are represented as a sequence of hidden states while the dataset of location points is defined as a sequence of observations. The utilization of HMM results in the computation of temporal entropy vectors of an individual taking into account his transitions from one global location to another in order to describe his overall daily behavioural pattern. The results of conducted experiments show that 17 different characteristics of individual can be successfully predicted with an average accuracy of up to 85.5% with the proposed method. Similar models were developed for discovering higher level behavioural patterns from individual and vehicle GPS trajectories (Qiu and Bandara, 2015; Wuang, Huang and Shan, 2015; Vieira et al., 2015; Naji et al., 2017). Since the solutions depend on the data obtained with GPS technology rather than visual information derived from video feed, they may not be suitable for utilization in indoor spaces. GPS signals are carried at a frequency that do not easily penetrate solid objects such as walls or roofs and require clear line of sight between the satellite and the sensor-equipped device to pinpoint its accurate global position (Fredrick). However, GPS is not the only technology found in today's mobile phones that is being used by researchers to derive dynamic behavioural patterns. For instance, Shila et al. (2016) propose user authentication method for mobile devices using human movement and location patterns derived by machine learning techniques on the data collected from device sensors such as accelerometer, gyroscope, magnetometer or Wi-Fi. The approach requires one to adopt a model that depends on constant monitoring of the devices that individuals are using (e.g. with a bespoke application pre-installed on the device) and reliability of the signals received by their sensors, i.e. the availability of the location data.

There have been attempts at recognition of dynamic behavioural patterns on a basis of visual data obtained from the camera feed. As indicated by Saitou et al. (2006), many researchers propose various methods for recognition of the scene and human activities separately. The authors suggest that activities carried out by humans are closely related to objects in a context of a visual scene. This consideration has led them to the development of a framework where complementary modules are capable of recognizing relationship between actions and objects to formulate a social dynamic pattern. To this end, first the movements of human head and hands are monitored by stereo vision. The extracted information on the position and orientation of limbs is subsequently supplied to Dynamic Bayesian Network (DBN) in order to perform the classification of the actions. In the final phase of the process novel conceptual models are used to refine the obtained data on the relationship between dynamic activities and objects. The hierarchical model consists of functionality labels of objects rather than their appearance. By doing so, the relationship between functional attributes and potential human actions can be easily established on a given model.

The framework presented by Saitou et al. (2006) use two cameras simultaneously to obtain images and depth maps in parallel. The depth map is used for tracking the skin contour and object silhouette from the video. The visual features are extracted from the video frame images maps. The output of this operation consists of 3D positions of human limbs so that adequate label makers can be placed on captured images. This results in extraction of critical visual features that are next supplied to DBN in order to classify actions that are next used in the hierarchical model to report a dynamic pattern. The pattern is reported on a basis of defined relationships between dynamic actions and objects. The flow of the framework process is presented in Figure 2-24.

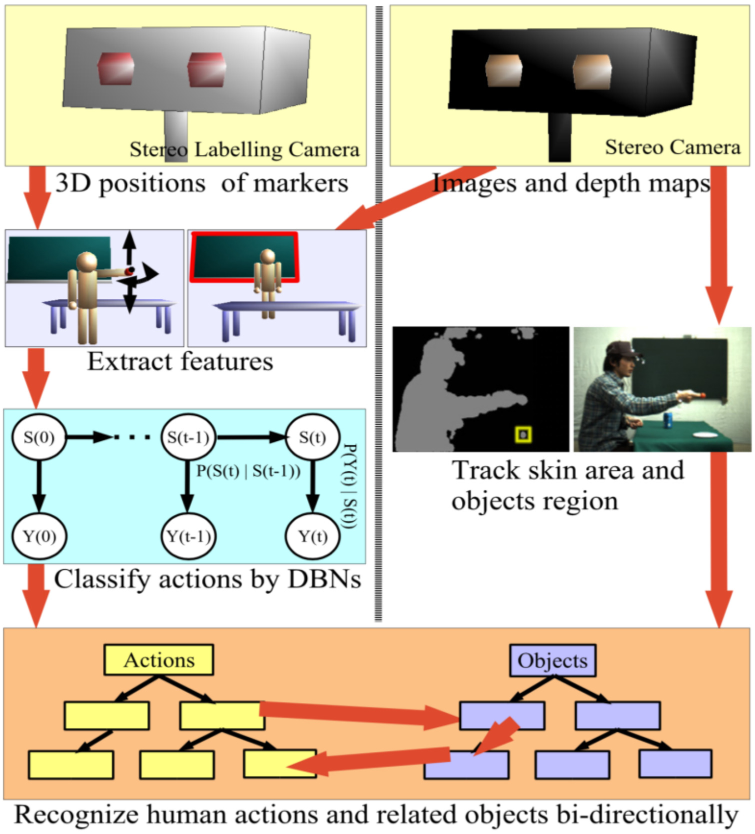


Figure 2-24. The framework developed by Saitou et al. (2006) is capable of establishing a relationship of the action executed by a human and object and derive a dynamic behavioural pattern on that basis

The results of the conducted experiments are promising (Table 2-2), but the authors are considering head and hand movements as the most crucial in the analysis of human action. The adopted model would have to be expanded in order to recognize dynamic patterns related to locomotion around the visual scene such as "walking along an object".

Table 2-2. Results of the analytical evaluation of the framework proposed by Saitou et al. (2006).

	PUT DOWN	PICK UP	EAT	DRINK	Total
Recall	89.7%	90.5%	66.7%	87.5%	85.7%
Precision	81.3%	90.5%	66.7%	87.5%	82.2%

2.5. Review of the available 3D modelling tools and technologies

In order to successfully implement any simulation-based dynamic behaviour analytics approach the software tools are of key importance. There currently exist multiple tools that can support the development of application capable of simulating dynamic human behaviours simultaneously allowing their analysis on a basis of spatial properties of objects. A review of the available 3D modelling tools suitable for this purpose is presented in the next sub sections.

2.5.1. OpenGL

The Open Graphics Library (OpenGL) is an universal graphics library that is widely used in the industry for development of 2D and 3D graphics in a wide range of application games and simulations. The API provides a programmable rendering pipeline that is executed directly on Graphical Processing Unit (GPU). The library is a “window-system and operating-system independent as well as network transparent” (Khronos Group, 2017). The model developed by Zhang et al. (2016) is a prime example of OpenGL utilization for development of three-dimensional human representation and virtual agent's behaviour characteristics. Their adopted geometric model of human body was developed using 3Ds Max modelling software (see section 2.5.7). The OpenGL graphics library was subsequently used to reproduce this model in a dynamic three-dimensional environment. The API that is provided by the library can be used to access low-level features of the underlying graphics hardware for rendering 3D graphics in real-time.

2.5.2. Second Life

Second Life is a free to use online application simulating a 3D-based virtual world in which one has the ability to create his own character called avatar (or *resident*) and participate in its development (Rymaszewski et al., 2006). The developer of Second Life, Linden Labs, provides a client application (*viewer)* for each of the major operating systems, including Windows, Mac OS and Linux that allows connecting to one of its servers where the virtual world is being hosted on. Among a variety of tools, the platform comprises of 3D modelling tool that allows constructing own objects (*primitives*) and a sophisticated, state and event driven scripting language called Linden Scripting Language (LSL). The LSL can be used for building custom functionalities for manipulating spatial properties and dynamics of objects, controlling avatar behaviours and registering and responding to events that may occur in the virtual world. The framework allows developing 3D simulations of specific case scenarios that may help in the visualisation of dynamic behaviour of agents in a given environment. Second Life is a great tool for creating immersive 3D worlds in which people can interact with each other: talk, play, lead conferences, educational purposes or even conduct businesses. It is a sophisticated 3D simulation tool but not good analytical tool. The LSL interface enables one to attach various primitives to constituent parts of the avatar body (Rymaszewski et al., 2006), but the adopted agent model does not consider spatial information of the limbs such as world location and orientation that could be used for analysis of the patterns. To circumvent this problem, one could load poses animations with BVH files and use them as a source for collecting data on location and rotation of various (Second Life Wiki). There are two problems with such approach: the streamlined files would have to be pre-recorded with special hardware such as previously mentioned Microsoft Kinect (Microsoft, 2015) that may not be available and the bottleneck that may arise from parsing the complex structure of these files in relatively short amount of time. The application does not incorporate a mechanism that would automatically perform the analysis of dynamic behaviour patterns exhibited by agents. Such functionality would have to be programmed manually. Several technical limitations of the framework have also been identified:

- A memory cap of total size 64 KB (including bytecode, stack and heap size) imposed on every LSL script greatly limits the capabilities of the software. In this research, individual and group dynamic patterns of agents are considered to be analysed simultaneously, which presumably will require more resources for intense computations of data stored in the memory.
- LSL is greatly limited in terms of data structures support which are of great importance when it comes to accumulating valuable dynamic information on agents participating in the simulation required for performing subsequent pattern analysis
- Inability to include own libraries means that a separate pattern analyser module would have to be implemented separately. The events registered during the simulation would have to be sent to a separate server dealing with this process.
- As indicated by García-Zubia et al. (2010), Second Life platform was not developed with real software development in mind and therefore, provides only limited interfaces that a programmer can utilize to process input data from the user, namely a dialog window with a selection of answer-buttons (llDialog) or text box where user can write characters (llTextBox). In order to streamline data about visual scene coming from a different application, such as trajectory reconstruction module, one has to implement a special interface that utilizes embedded browser provided by Second Life in an unconventional way (García-Zubia et al., 2010).
- The configuration needs to be made on Linden Lab servers, which may significantly reduce control over the implementation setup

2.5.3. JmonkeyEngine

JMonkeyEngine is an open source game engine written in Java based on OpenGL (Kusterer, 2013). It can be used for the development of 3D applications that can be universally run on any platform equipped with Java Virtual Machine (JVM). As of time of this writing, JVM is supported by major operating systems up to date including Windows, Mac OS and Linux (jMonkeyEngine, 2016). Apart from Java native libraries, the engine incorporates a wide array of libraries that provide the out-of-the-box capabilities for implementing a variety of 3D functionalities: physics (jBullet), networking (SpiderMonkey) and graphical user interface (Nifty GUI) to name a few. In addition, the engine is distributed with BSD license and fully-fledged Integrated Development Environment (IDE) (Figure 2-23) that has been built on top of NetBeans platform. Such configuration ensures ease for handling code (e.g. debugging), manipulation of graphical and non-graphical assets (e.g. 3D meshes, textures and XML files) within application folder structure or shaders creation (special low-level programs for controlling stages of graphics rendering process executed on GPU (Khronos Group, 2017)).

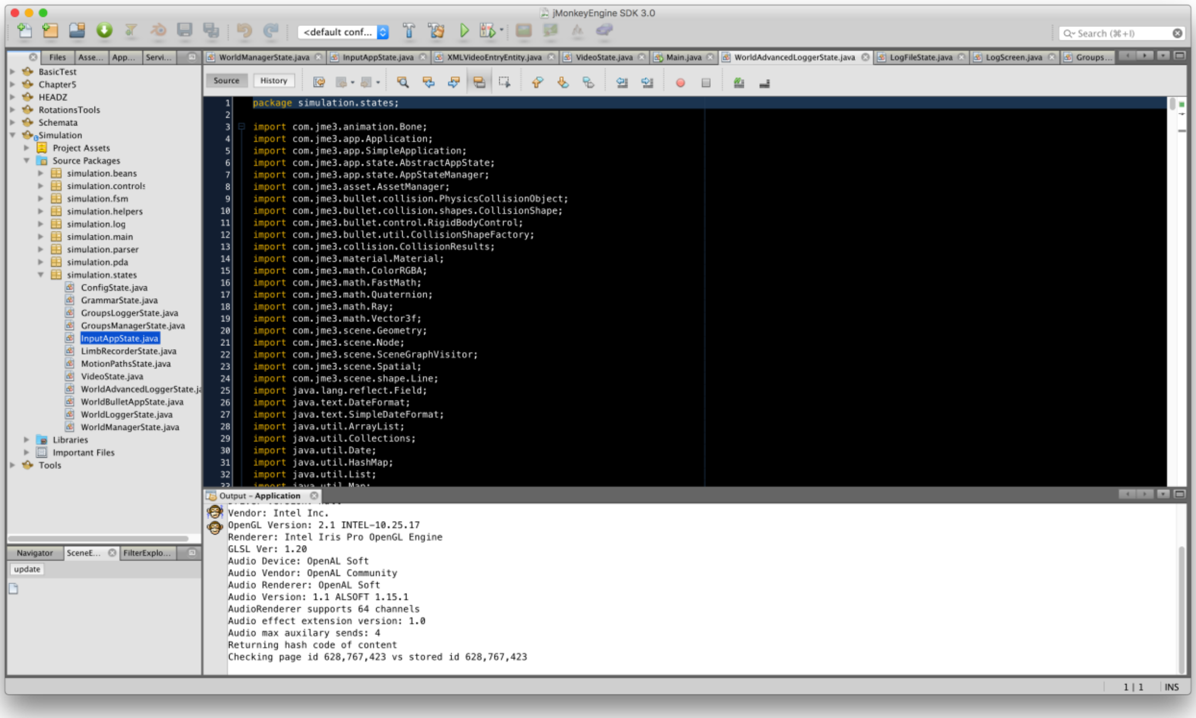


Figure 2-25. jMonkeyEngine IDE iOS interface (jMonkeyEngine, 2016).

2.5.4. Ogre

Ogre is an open source, 3D graphics toolkit entirely written in C++ (Ogre3D, 2016). Rather than a fully capable game engine, Ogre delivers means for developing one according to the requirements of the project. With specially designed internal set of libraries, it removes problems associated with utilization of OpenGL and Microsoft Direct3D functionalities (Navarro, Pradilla and Rios, 2012). The toolkit is distributed with MIT license and has been used to develop applications for wide range of popular platforms such as Windows, Mac OS and Linux.

2.5.5. Unity

Unity is a sophisticated, cross-platform game engine, which has been used for development of applications with complex graphical requirements (Navarro, Pradilla and Rios, 2012). It is delivered with a set of tools among whom one can find an visual editor for designing a 3D scene and IDE for development of the application code. The engine supports three programming languages for development: C#, JavaScript and a language for .Net called Boo (Boo-Lang). All of the languages can be used in parallel for the construction of application architecture. Unity supports development of 3D games for several major platforms: Web, iOS, Android, Windows, MacOS and Wii, Xbox360 and Playstation consoles (Unity Game Engine, 2017). Unity engine has been built on top of state-of-art technologies in mind for an efficient and complex 3D projects development. The render engine reduces workload for the graphic drivers by combining different geometries into parallelizable units and provides great support for shaders creation. The engine fully supports the OpenGL library, which means all applications can be optimized for mobile devices (Navarro, Pradilla and Rios, 2012). The physics engine is based on Nvidia's PhysX (PhysX Technology, 2017) that allows one to introduce the laws of physics into the virtual scene such as gravity, friction and so on.

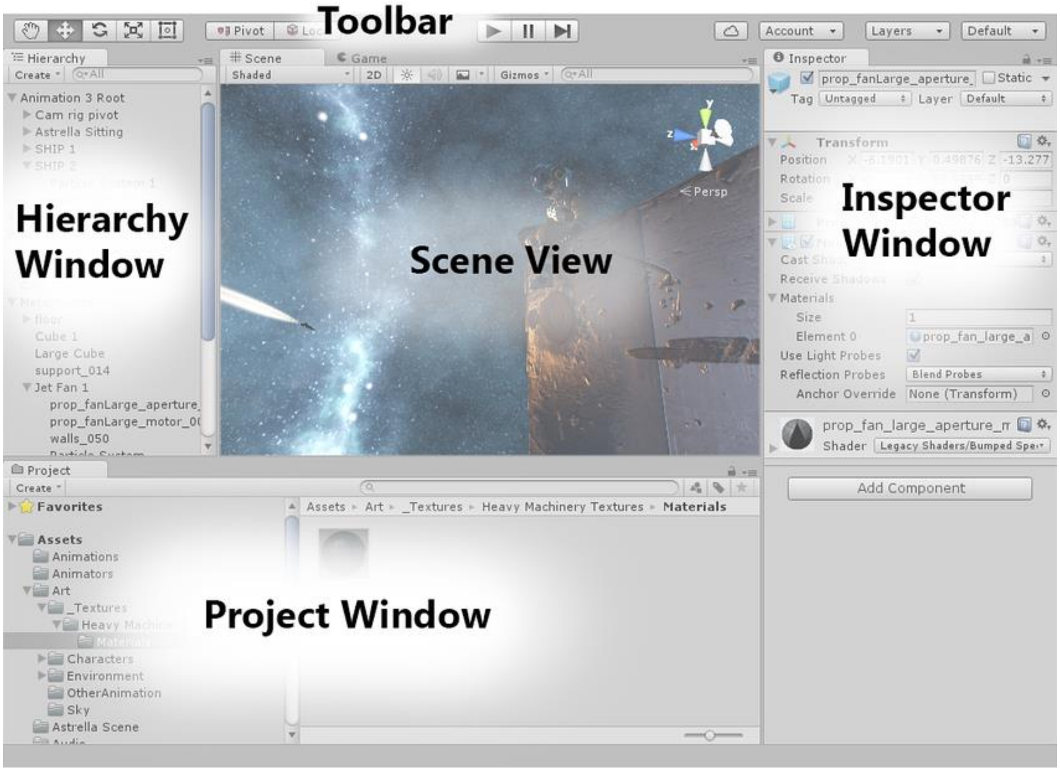


Figure 2-26. Unity engine interface with labels of the most crucial panels (Unity Game Engine, 2017).

2.5.6. Unreal Development Kit (UDK)

The Unreal Development Kit (UDK) is one of the most advanced game engines available on the market that is distributed under a non-commercial free user license and commercial profit share based license (Savage, 2015). UDK combines several tools for development of high quality 3D applications for a wide range of major platforms used today: XboxOne, Playstation 4, Nintendo Switch, MacOS, Windows, Linux and iOS to name a few. According to developer, the engine has been entirely designed and based on C++ for creating highly optimized 3D applications (Epic Games Inc., 2017). Similar to Unity, the physics engine is based on the latest implementations of Nvidia's PhysX (PhysX Technology, 2017) that can be used to introduce physics into virtual scenes.

The Sequencer, a tool for creating cinematics within games, can be used for visually directing various case scenarios, which are a matter of interest for subsequent agent behaviour analysis. An interesting feature of the engine is a unique visual scripting system called Blueprints that allows one to create a complete logic of the application without manually writing code. Blueprints can be utilized to quickly prototype behaviours of objects, mechanisms for controlling user input and visualizing the entire flow of the application at run-time (Epic Games Inc., 2017). The other features of the engine include VFX and particle system, terrain editor, multiplayer framework, AI module, material editor and recently developed solution for creating virtual reality (VR) and augmented reality (AR) applications (Epic Games Inc., 2017).

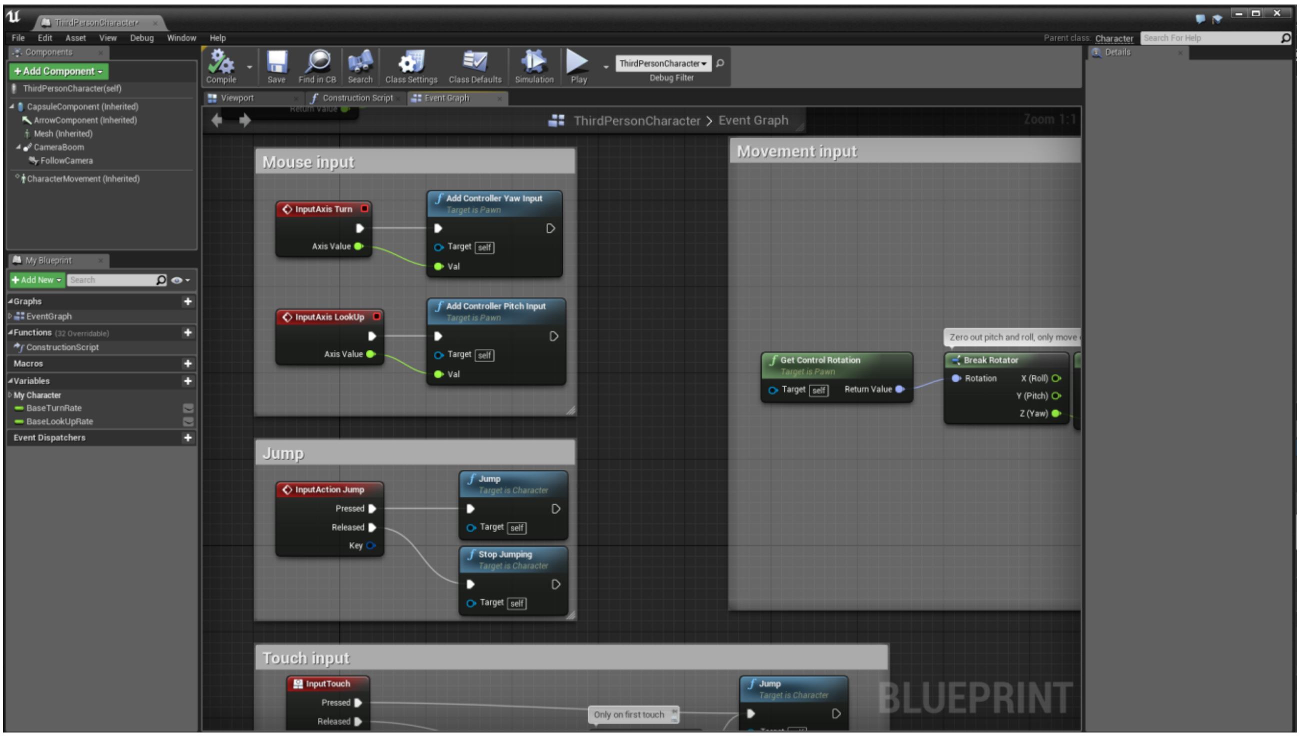


Figure 2-27. Blueprints system allows one to visually script application logic without writing a code (Epic Games Inc., 2017)

2.5.7. Cry Engine

The Cry Engine was developed and is maintained by Crytek Inc. (Crytek Inc., 2017). Next to UDK, Cry Engine is a so-called next generation games engine providing a sophisticated IDE for development of high quality 3D applications. It is distributed under “100% royalty free” license. The engine supports DirectX12, an array of programming interfaces (API’s) developed by Microsoft to “handle tasks related to rendering 2D and 3D vector graphics, rendering video and playing audio on the Windows Platform” (Parrish, 2016). The current version supports several platforms including Windows, Xbox One, Playstation 4 and even VR hardware such as Oculus Rift. The engine has several interesting features for specifying dynamic navigational data structures that update according to spatio-temporal changes of the visual scene (Multi-Layer Navigation Mesh module), controlling skeletal animations of agents and their movements (Parametric Skeletal Animation module) and caching system that allows recording a complex 3D simulation and reducing its original file size up to 10% (Crytek Inc., 2017). Cry Engine provides a Flowgraph visual scripting system that is similar in operation to UDK Blueprints. Instead of developing application logic by writing a code, it allows one to develop it using an intuitive visual interface as shown in Figure 2-28.

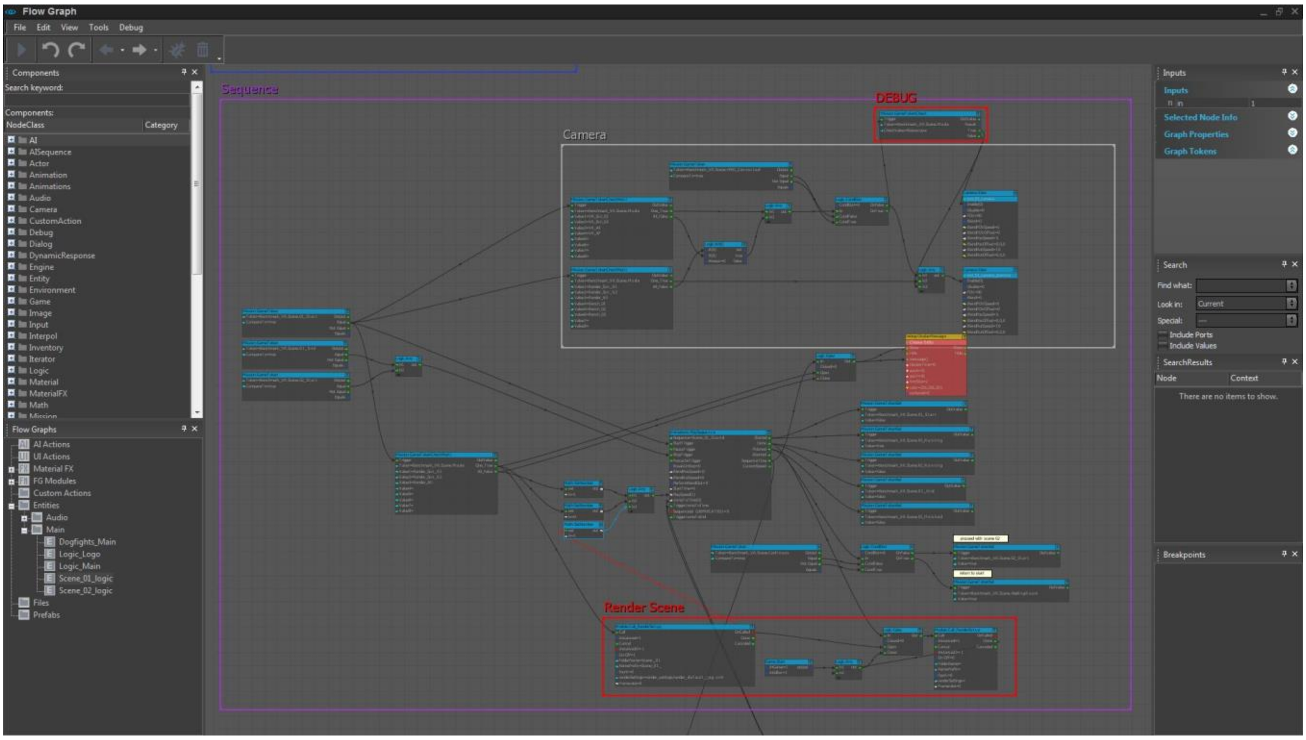


Figure 2-28. Flowgraph system in many ways is a Cry Engine counterpart of UDK Blueprints (Crytek Inc., 2017)

2.5.8. 3DS Max

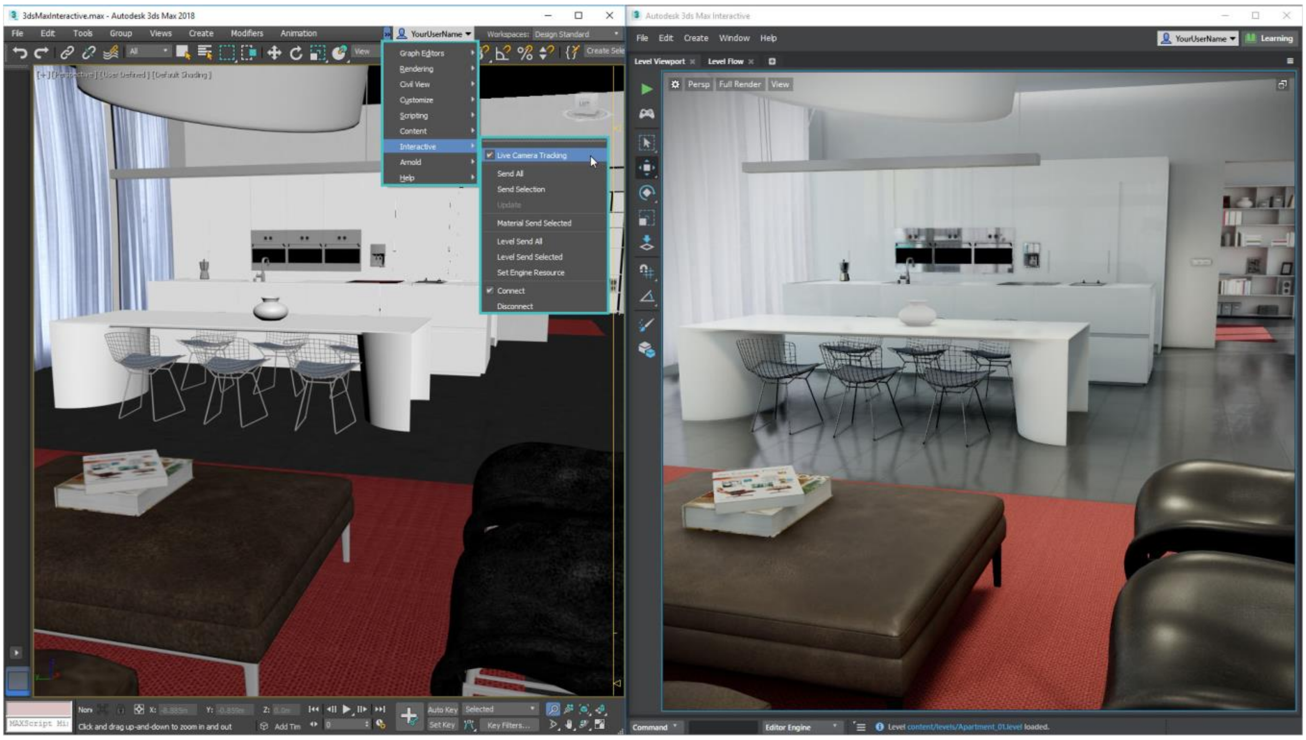


Figure 2-29. The interface of 3DS Max allows one to preview and manipulate visual scenes in real-time (Autodesk Inc., 2017)

The 3DS Max is a software for 3D design, modelling and animation developed and maintained by Autodesk Inc. (Autodesk Inc., 2017). With a set of comprehensive tools provided by the application one can create 3D representational models of world objects, define their physical properties such as shapes, materials and textures and generate procedural and key-framed animations. The interface allows to visually manipulate objects and their properties in real-time. The 3DS Max was built with development of 3D assets in mind and although it contains some interesting features such as crowd generation (Autodesk Inc., 2017), it does not contain sufficient programming IDE that is essential for implementation of pattern recognition model. Autodesk Inc. distributes software with a commercial or educational license only.

2.5.9. Maya

Maya is another 3D modelling software developed and maintained by Autodesk Inc. (Autodesk Inc., 2017). According to its developer, “Maya 3D animation, modelling, simulation and rendering software provides an integrated, powerful toolset that can be used for animation, environments, motion graphic, virtual reality and character creation” (Autodesk Inc., 2017). In many ways it is similar to 3DS Max, but with a stronger emphasis on more precise character modelling and animation (Autodesk Inc., 2017). Similarly to 3DS Max, Maya does not provide a satisfactory IDE for the implementation of pattern recognition model and it is distributed with a commercial or educational license only.

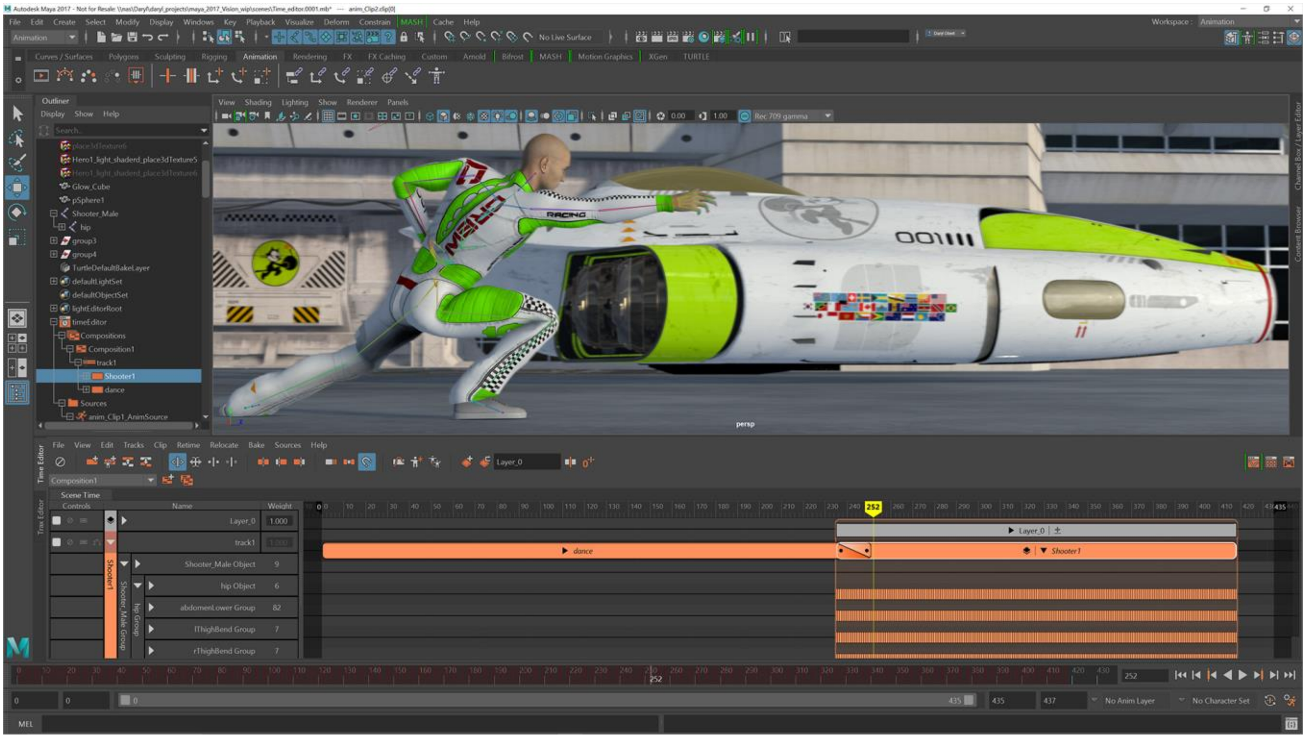


Figure 2-30. Maya software interface (Autodesk Inc., 2017)

2.5.10. Blender

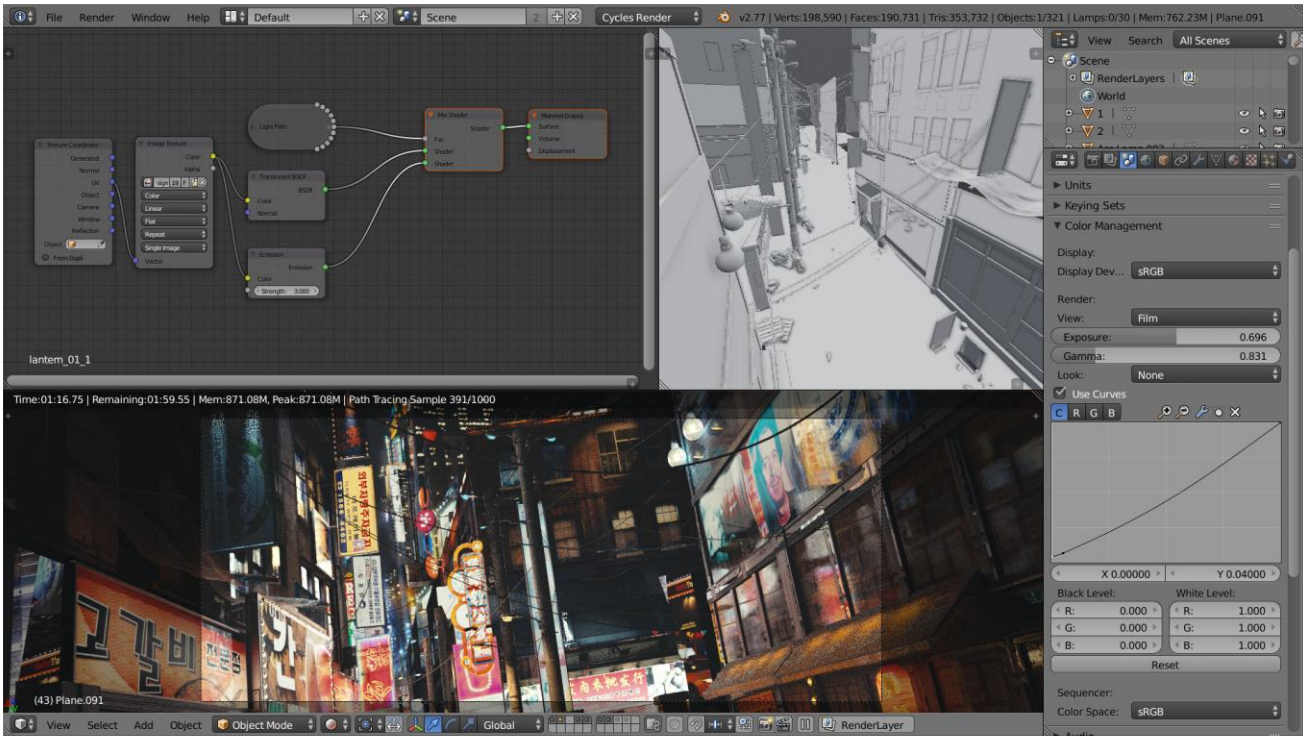


Figure 2-31. Blender software interface (Blender Foundation, 2015)

Blender is the advanced, open-source 3D modelling and animation software that combines sophisticated tools for development of visual scenes and objects. The software allows one to create 3D visual scenes, objects and define their physical properties using an intuitive visual interface in real time (Figure 2-31). As with 3DS Max and Maya, Blender contains essential tools for creating complex animations that may involve armature development for agents. An interesting feature of Blender is that it has a built-in game engine, which not only fully supports OpenGL features such as lighting or materials but also provides one with Python scripting API interface for developing game logic and custom scripts. Unlike its counterparts, Blender is distributed with GNU General Public License (GPL), which makes it free software.

2.6. Potential use of software engines for simulator implementation

In order to implement successfully any simulation-based dynamic behaviour analytics approach the software tools are of key importance. After careful examination of the available options, a well-known game engine **jMonkeyEngine** was selected (Kusterer, 2013; Reese and Johnson, 2015; Edén, 2014). Both the situation-based strategy used in the architecture of Sung, Gleicher and Chenney (2004) and the agent-based approach developed by Urban and Schmidt (2001) can be easily implemented using this engine. There are similarities between conceptual ideas incorporated in the architecture of Sung, Gleicher and Chenney (2004), the modular approach adopted in PECS system (Urban and Schmidt, 2001) and the internal core architecture of the jMonkeyEngine framework (Kusterer, 2013). The functionalities provided by the libraries of

jMonkeyEngine are modular and allow coordinating the behaviour of the scene on local and global levels as shown in Figure 2-32. The “app states” are used to maintain the global state of the simulation world whereas “controls” are used to enhance agent’s abilities with actions necessary for building a specific behaviour.

The visual scene in **jMonkeyEngine** is described in a hierarchical data structure called *scene graph*, where each element of the graph corresponds to an entity in the visual scene. In this framework the entities can be also grouped in a *parent-child hierarchy*. Two types of classes control the behaviour. The so-called "app states" are responsible for maintaining global state of the scene whereas "controls" enable one to encapsulate specific dynamic behaviour of an entity in a class. Both can be added or removed at run-time. Each "control" possesses its own update loop that updates the internal state of an agent according to its implemented logic and control rules. The “app state” role is to update the main update loop and maintain the global state of the scene, e.g. physics state that regulates the gravitational forces.

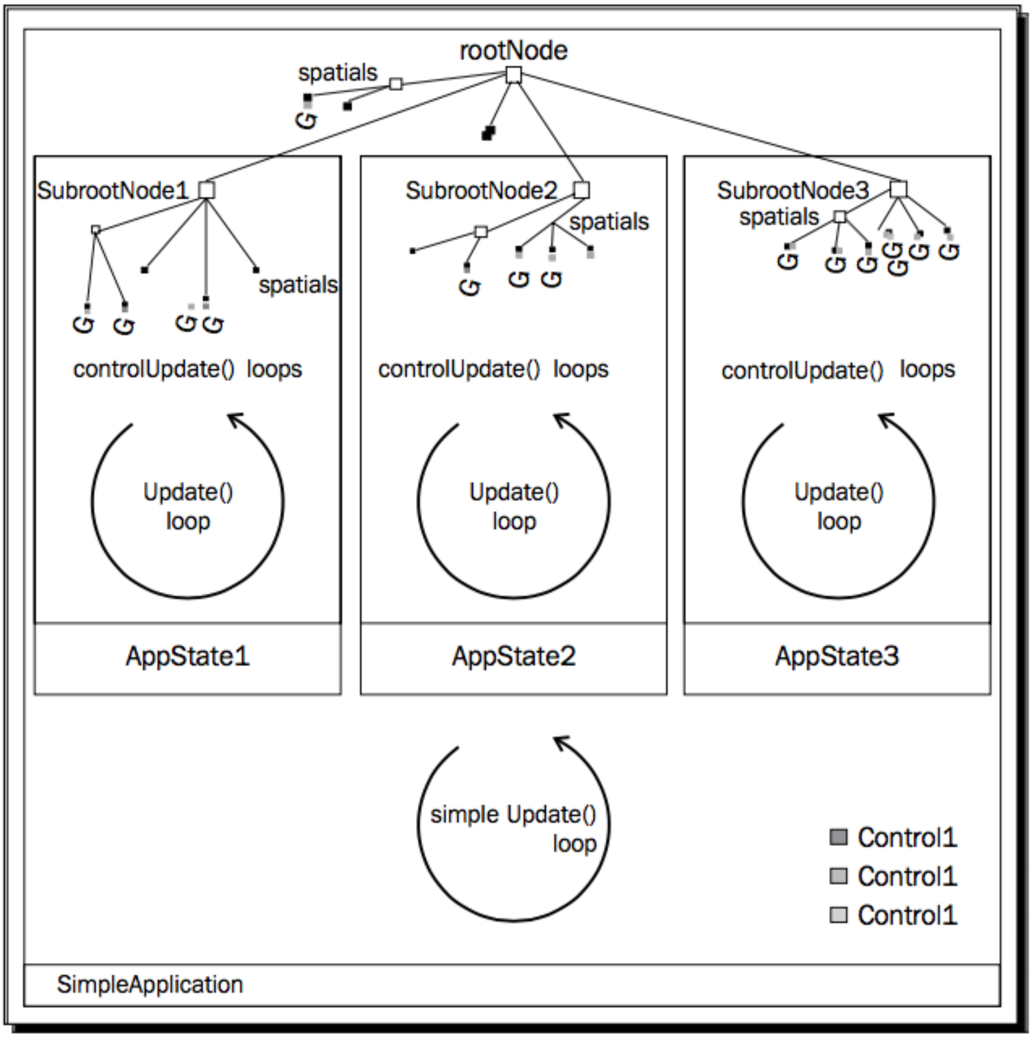


Figure 2-32. jMonkeyEngine framework provides means of adding and removing behaviour on both local and global levels (Kusterer, 2013)

2.7. Research summary

The research conducted in several areas related to human behaviour analysis from visual data has raised the awareness not only of the available methods with their advantages, but also of their limitations. This is extremely important for the success of the entire research program for dynamic behaviour analysis from live video footage because of its complexity and it requires well-informed choice. The aim of the extraction methods is to successfully identify static and moving objects in the video footage taken by a physical camera, which monitors different environments. The classification methods make an attempt to classify the identified objects and movements in order to establish their meaning. The simulation models are needed for evaluating different scenarios of human behaviour under specific conditions and according to pre-defined set of rules. The literature contains some examples of combination of the above methods - either to make the analysis of human behaviour more sufficient, more accurate, or both. However, none of the reviewed frameworks considers incorporating of a full 3D simulation model driven by real world data that would be used for inferring behaviour patterns of both individuals and groups in real time. The reviewed literature mostly concerns about the methods for extraction and classification based on direct processing of the image frames obtained from the video feed. Very rarely if ever they consider the logical relationships between static and dynamic entities of the world, which is of paramount importance for any real-world application. It would be difficult to recognise, for instance, a pattern of a "man walking along a shelf" in a shop scenario based on the methods discussed in this chapter, because such a pattern would involve processing of too much information that needs to be done in real-time: identification of the physical boundaries of the shelf, recognition of the walking activity, estimation of the distances between different entities, involved, establishing the direction of movement for a limited period of time, etc. This might be possible by visual data processing methods alone but this would require huge computational power because of the volume of information which needs to be processed in real time and the complexity of the images which contain many layers of information. The original problem can be simplified by splitting the analysis into three steps – first selective extraction of the physical information which needs to be analysed by focusing on the essential features which define the dynamic behaviour, then restructuring of the extracted information and enriching it by organising it into a dynamic model of the video feed by 3D simulation and finally recognition and classification of the dynamic patterns through algorithmic analysis of the observed simulation. The model-driven analysis could simplify the analysis of the behaviour by replacing the analysis of the original footage with analysis of the simulated dynamics because it can incorporate into the model physical laws and logical dependencies which are not explicitly present in the original video footage and thus, to provide essential information for supporting the analysis. In the next chapter, we will develop a conceptualisation of a model-basic event-driven framework which adopts this model-based event-driven approach.

Chapter 3

Conceptual framework for model-driven dynamic behaviour analysis

Humans can interpret dynamic behaviour on the basis of visual observation and individual level of knowledge. With the recent advances in technology it is possible to delegate the process of analysis to machines. The benefit of this is that one can automate the understanding of human behaviour by the machines and, this way, improve security and safety management, gain better business customer insight, further advance video games programming, etc. However, performing such analysis in real time is a challenging task due to the necessity to combine purely visual data processing with complex analytical data processing. Whereas the processing of visual data can be done to a satisfactory level thanks to recent technological development, the second one requires model-driven behaviour pattern analysis which makes the task much more difficult.

Human behaviour can be analysed from different perspectives:

- ***Dynamic behaviour*** is perceived directly from visual observation of physical movements and gestures of individuals.
- ***Cognitive behaviour*** is inferred by analysing individual's goals, intentions and actions in pursuit of reaching them.
- ***Psychological behaviour*** is assessed by monitoring the emotional state of individuals.
- ***Social behaviour*** is analysed on a basis of interrelations among individuals and their actions towards each other.

The possibility of performing other types of analysis, namely cognitive, psychological and social, may be considered in potential future work, but the focus of this research is the analysis of dynamic behaviour of humans in restricted spaces.

Dynamic behaviour can be also scrutinised within the context of different research paradigms. Few label often met in various areas are:

- Visual analysis of the individual movements as perceived by the observer
- Logical analysis of the events as recognised by the listener
- Statistical analysis of the group location and individual appearance as measured by the measurer, etc.

In this research, the aim is to combine visual observation of individual and group dynamics through the eye of a virtual video camera with logical analysis of the events on the visual scene as registered by various sensors, excluding any statistical analysis of the group behaviour.

3.1. Overview of the conceptual framework

Dynamic human behaviour is manifested in the form of physical activities that allow the individuals to fulfil their specific goals in their daily life. On a closer examination they can be fragmented into smaller actions, executed in a specific and orderly manner. Let us consider a typical shopping mall scenario where an individual purchases a product. The relevant information includes the *topology* of the building, the *location* of various objects and individuals within the physical space at any moment of time and the dynamic *movements* of the individuals through the space. This can be considered *input data* of our analysis. The activities may include navigating through the layout of the mall, acquiring the product from inside a specific shop and leaving the building through a designated exit. On a lower level of details it can be observed that the navigation consists of simple movements, such as walking along a designated pathway, climbing stairs inside the building, entering a shop through doors, walking towards a shelf, reaching out for a product, and so on. These movements are the building blocks of the *behaviour patterns* through which the observer analyses human behaviour. When an individual is walking towards a static object on the scene with minimal deviation from this direction for a certain amount of time, this particular behaviour can be labelled as “an individual walking towards a static object”. When a specific angular configuration of the limbs indicates that the individual’s arm is stretching out, the pattern can be labelled as "reaching out for an object". In other words, the physical movements of the individuals and the parts of their bodies can be used as a basis for formulating various dynamic patterns of individual’s behaviour. So the *output of our analysis* is the patterns of behaviour.

Dynamic patterns always possess algorithmic nature. Following the order of actions and the events which happen over the time it is possible to realise that they appear one after another in a strict *sequence*, that they have been *repeated* a number of times, that some of them *occur* only in some occasion while some other happen *unconditionally*, and so on. This in turn provides information about the emergence of new situations in observed scene which can be described in terms of an *abstract algorithm*.

Another important characteristic of the dynamic patterns of behaviour is that they are relatively *robust* and do not depend significantly on the precision of the data. For example, walking towards a door does not depend on the speed of walking, neither on the precise distance to the door and the direction of movement can be estimated with an relaxed interval of acceptable deviation. This means that behaviour patterns can be derived from approximate perception of the world rather than from accurate footage. Sufficiently rich data suitable for the purpose of the analysis can be generated by 3D simulation.

3.2 Simulation-based analysis of dynamic behaviour

In the simulator-based approach we adopt for the purpose of simulated video analytics the information can be split into two parts - static information about the visual scene, which is a result of visual reconnaissance of the scene obtained priory to the analysis, and dynamic information about the changes in the simulated scene. The static information is a result of purely visual information processing and is considered external input. It is loaded periodically as initialization data for the scene while the dynamic information can combine both physical data, such as data about locations, positions and directions, and abstract data, such as synchronous data about the trajectories of movements on the scene as well as asynchronous events, which may occur on the scene at any time. Because the generated data is limited to only a few dynamic characteristics – absolute and relative positions of whole bodies and parts in the 3D space, absolute and relative directions of individual objects and groups of objects, it can be processed efficiently. In this paradigm the simulator uses minimal physical data and combines it with data generated through simulation, which significantly reduces the requirements in comparison with the use of physical data directly from video footage for analysing the behaviour.

But the role of the simulator is not limited to approximating physical information with generated data. The simulator can also enhance the information needed for analysis by establishing non-visual relationships between static and dynamic objects, which depend on the physical laws and the dynamic events, occurring during the simulation. Each event can be considered to have representation in a certain symbolic language, which the simulator understands, so that it can pass it to the event analyser for further analysis. The framework for analysis of the behaviour can be constructed as a linear workflow of several modules, responsible for annotating the video signal with additional data at each stage of the analysis. The first module will receive as an input the raw video signal from the physical camera observing the scene. The last module will generate as an output the information, which can be used for decision-making. The framework as sketched here is depicted on Figure 3-1.

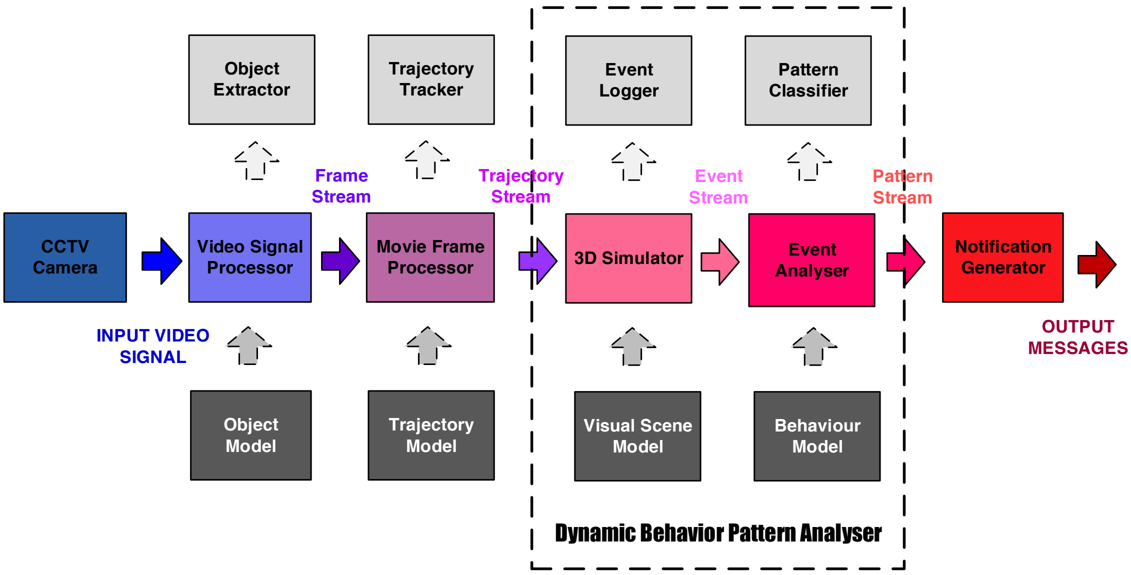


Figure 3-1. General workflow of the framework

The first module, **Video Signal Processor** extracts and processes data about the individual objects on the scene according to a model suitable for the process of recognition. At this stage, the dimensions of the space are estimated and the location, shape and texture of the objects are identified and measured. This information forms the input to the second module, **Movie Frame Processor**, which is responsible for reconstructing the trajectory of movements of the dynamic objects on the scene. It must take into account the object location, viewing and walking directions of the moving objects, as well as their velocity and speed of movement. The third module, **3D Simulator**, maintains the state of the simulated world, constantly updating it using the stream of reconstructed data about object trajectories and generates an event log which mixes the physical events at its input with the logical events at its output. The **Event Analyser** performs the proper behaviour pattern analysis, which is then fed into the last module of the framework, the **Notification Generator.** Its output will be used for decision making outside of the framework in various area of applicability – safety management, security control, customer insight, etc. This framework and the simulator as its central component were first presented by Gasiorowski, Vassilev and Ouazzane (2016). Later on it was expanded with the trajectory reconstruction (Afzal et al., 2016). Currently, several other research projects are dedicated to the other components of the framework.

Critical feature of our approach is the replacement of the physical video information used for visual data processing with logical symbolic information generated through simulation. The graphical data usually contains an enormous amount of information that is often noisy and therefore unsuitable for real-time processing since it requires a lot of filtering and additional pre-processing. The symbolic information can be processed much quicker and with lot less demanding requirements. Furthermore, it is much easier to establish a behaviour pattern from symbolic representation because the structure of the descriptors is much more logical.

Second distinctive feature of the framework is the use of a pattern classifier, which would allow generating meaningful messages depending on the situation their meaning in a situation. In most typical contexts the behaviour patterns can be classified as dangerous, suspicious, or just normal. For instance, the behaviour of a monitored individual can be classified as suspicious in different scenarios when he passes through the same location several times without stopping, while turning head in the direction of one and the same static object:

- In a shopping case scenario, an individual might be scouting area to safely shoplift without being noticed.
- In a bank scenario, an individual might be probing security measures put into place before potential robbery.
- In a meeting scenario, an individual might be scanning for the appearance of another individual in a specific place.

Suspicious behaviour typically requires specific actions which may involve some kind of probing, while a pattern of behaviour perceived as potentially *dangerous* may require more drastic measures. For example, when an individual who has been walking straight for a certain amount of time suddenly collapses on the floor this might be a signal for the need of immediate assistance. In general, the classification of a pattern contributes to the understanding of the situation within the scope of the visual scene, which is reported by the notification generator.

The conceptual architecture of **Dynamic Behaviour Pattern Analyser** component of the framework is depicted in Figure 3-2.

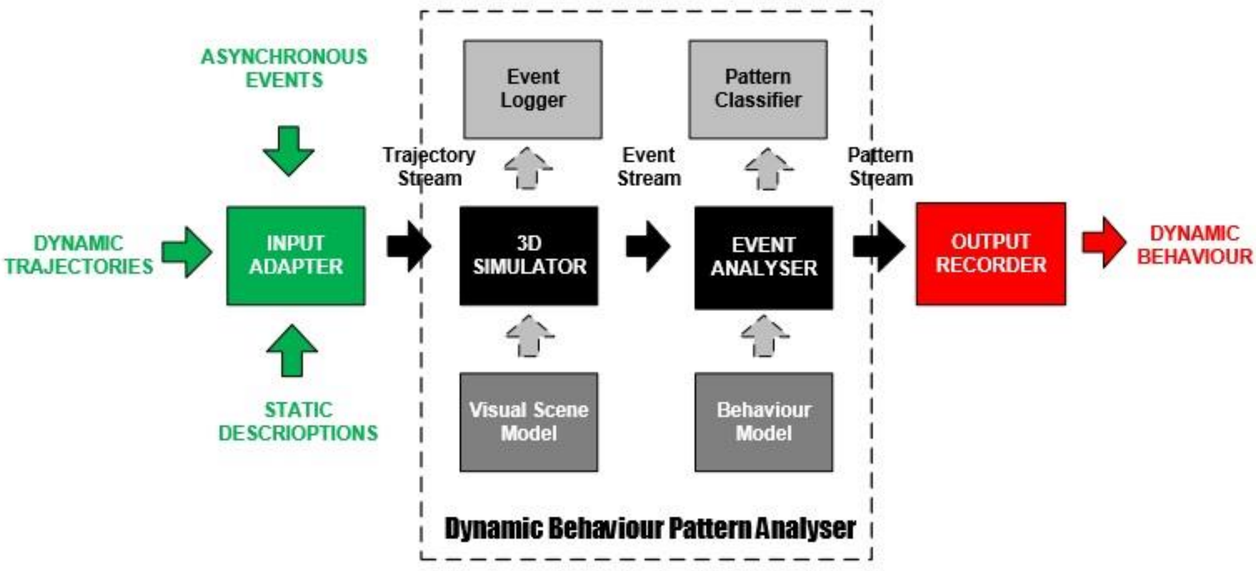


Figure 3-2. Conceptual structure of **Dynamic Behaviour Pattern Analyser**

The input of the simulator mixes three types of data: the *static information* about the visual scene, describing the identified objects and their physical properties, the *trajectories of dynamic objects* as reconstructed at the previous stage of video processing and *asynchronous events* that may be delivered along with the video signal (i.e., data readings, warnings and alerts coming from various sensors monitoring the physical space of the visual scene).

The asynchronous input events may be caused by physical changes in the visual scene as observed by the camera, such as the appearance of new objects within its scope, disappearance of the tracked entities from the observed space, changing the orientation of the camera, etc. The global state of the simulator is periodically updated using the information received from other modules of the framework during video processing but between the changes the simulated world is fully steered by the reconstructed trajectories of object movements. Since the three-dimensional space is defined in Cartesian coordinate system, in our framework the trajectories are assumed to provide information on location, velocity and rotation in standard 3D vector notation. To represent more complex movements of dynamic objects, such as twisting and variations in object viewing, or changes of the walking and viewing directions, the trajectories must also include description of the angular variations which can be based on the quaternion theory. The simulator may also add to this input internal data, generated as a result of the simulation itself to comply with the physical laws of the world.

While the trajectories provide sufficient basis for analysing the behaviour of single dynamic objects, additional means has to be used for identifying, tracking and analysing the group dynamic behaviour of the same individuals as part of the crowd. In our approach the individual objects will be grouped on the base of purely logical links to each other, not statistically, which will allow to analyse the group behaviour logically exactly as the analysis of the individual behaviour. This is another critical feature of our approach because the simulator can perform analysis of crowded scenes on group (macro) and individual (micro) levels simultaneously, unlike the approaches based on "crowd management" theory that focuses only on the groups but never on the separate individuals in the groups.

3.3. Ontology of the visual scene

The purpose of the ontology of visual scenes is to provide an abstract representation of the information, which can be used in the logical analysis of behaviour patterns. The ontology of the bound worlds is not a new thing - it has been used in Computer Games (Zagal et al., 2007) and Robotics (IEEE Standard Ontologies for Robotics and Automation, 2015). Both areas share certain commonalities considering the fact that in both worlds the visual scene is observed from the point of view of an eye – be it the eye of the robot, or the eye of the gamer. In Computer Games, the objects are part of the game world that can be managed or interacted with by the players and their location is only logical. In Robotics, objects occupy a real physical location in space that can be manipulated by robots.

The ontologies in principle define key concepts and relationships in a given domain. At the bottom of the ontology are the individuals, who are objects residing in the world. The objects recognised in a video can be described explicitly by their physical attributes (location, velocity, orientation etc.), by the way these attributes can be altered and by the abilities to execute some form of dynamic actions they possess (Zagal et al., 2007). There are static entities, which do not possess the ability to execute any action on their own and usually are just part of the game world, without changing their physicality. On other side of the spectrum, there are dynamic entities possessing the ability to perform an action in order to manipulate the properties of other entities.

In Robotics, the ontologies contain also the "autonomous robot" object that is capable of adapting to changing environmental and operational conditions and executing actions on their own without human intervention (IEEE Standard Ontologies for Robotics and Automation, 2015). The autonomous individual captured in the video footage may be considered as an individual capable of controlling its own movements and interaction with other objects on its own, without the need of intervention from any other objects. Individuals naturally form social groups in order to collaborate on achieving mutual goal. This is closely related to the definition of "robot group" found in IEEE Standard Ontologies for Robotics and Automation (2015) where the term is defined as "*a group of robots organized to achieve at least one common goal*".

The Game Ontology Project (GOP) introduced by Zagal et al. (2007) for describing and analysing computer games was built on ideas of Rosch et al. (2004), where the objects and relationships between them are

identified on the basis of visual perception and analysis of videogames. In this approach the ontology is built without an insight of game designers knowledge, intentions or plans. Following this approach, from the review of a video footage one can define the ontology of visual scenes that matches those that can be found in Computer Games and Robotics. The main deviation of the approach adopted in this research is when one starts considering the group formation. In the special case of a *pair*, when the group is made out of two individuals only, the above approach is not entirely adequate because it allows representing only binary relations in the groups (IEEE Standard Ontologies for Robotics and Automation, 2015). However, in many applications an "observer" plays a vital role and ternary relationships are vital. For example, two individuals who are talking to each other are also listening to each other while talking, but the third who is in the same group maybe only listening to them without talking to any of them. This is especially important if we want to formalize the concept of an "observer", which is a natural candidate for accounting the presence of a camera in the scene. Because of this in our ontology we will assume that the groups consist of three or more members and we will consider the pairs as a separate entity on the scene.
The following concepts play a vital role in modelling and building our framework.

- ***Scene*****:** delimits the boundaries of the space where all objects are situated. It provides the basis for the coordinate system of the restricted world monitored by video camera.
- ***Object*****:** an identified object that has physical location in space and time. There are three types of objects that can be identified: Static Objects, Dynamic Objects and Individuals.
- ***Static Object*****:** object that do not possess ability to execute any action and whose physical attributes can be altered by dynamic objects or individuals only. This type of object remains static for most of the time. *Example:* doors, shelves, stairs.
- ***Dynamic Object*****:** object that possess the ability to change physical properties of objects due to external factors or intervention or interaction of other objects at particular time. *Example:* shopping trolley, product, envelope.
- ***Individual***: an autonomous dynamic object that has some degree of control over its movements. Individuals are capable of executing actions on their own without the need of intervention of other objects that may lead to interaction with other individuals or objects. *Example:* human, animal, robot.
- ***Pair***: two individuals that formed a relationship in which a certain degree of collaborative activities and interrelation can be observed. The activities in such a relationship can only be perceived as symmetric, anti-symmetric or general, without any further complexity. *Example:* Individual and another individual walk together (symmetric relation); Individual talks to another individual (antisymmetric relation); Individual likes the other individual (general type of relation)
- ***Group*****:** an identified collection of three or more individuals exhibiting similar motions and potentially some level of collaborative activities in order to achieve mutual goal. A group can be treated as a single entity by aggregating all its participants' activities. *Example:* All participants in a group are moving in the same directions (**"group moves towards stairs")**; A participant in a group is throwing an object (**"group vandalises an objects")**

The above concepts provide some hints about what kind of activities and events can be expected to take place in the observed scene. However, all of them can be static. The really vital clues about the behaviour are the actions and events, which are essentially dynamic concepts.

3.4. Ontology of the dynamic behaviour

The patterns of behaviour are derived from observation and analysis of the dynamics of objects identified in the visual scene. Assuming that we know only the global location of each individual and the position of their limbs relative to the body at any moment of time we can define a number of actions, which can be executed by that individual. Therefore the model of actions is a cornerstone in the theory of simulation.

The correlation between individual actions of the individuals and the events, which occur at the visual scene, can be modelled using three alternative approaches:

1. *The actions are considered as changing the world and the events are only triggering them*. In this approach, changes may or may not occur with the time because the world remains in the same state if no activities are taking place. The changes are always caused by activities, while the events are relative to the time but independent from the actions. This approach is suitable for modelling actions that are instantaneous and triggered by events; the processes unlike them they have duration. It is commonly adopted in object-oriented modelling paradigm because the objects remain in the same state if no external activities are affecting them. This is the oldest approach widely employed in the early research in intelligent robots (IEEE Standard Ontologies for Robotics and Automation, 2015). Similarity can be found also in the "Interface" conceptual element of the game ontology (Zagal et al., 2007). The input device provides the players means of sending signals to the game interface so that they can be turned into suitable actions. Whenever player causes an event in the form of pressing a button, a corresponding action is executed on the screen. It may or may not change the state of the game world (change attributes of the entities of the game world). Time in this case can be completely disregarded as it does not influence the way events and actions occur. However, this approach leads to representational issues related to the so called "frame problem" in AI (Shanahan and Witkowski, 2000).

2. The events are considered as changing the world; the actions are just collecting them. In this approach the events are happening all the time, so the time is attributed to them. The state of the world in this case is defined in terms of the history of events. The world in such a case may or may not change depending on the events, not on the actions. The time measures the delay between events (frame update) but it does not initiate the changes. To that end, the actions would have to be defined through events as well. This approach is relatively new in Computer Science. It leads to more complex logics (Kowalski and Sergot, 1986). But the effect of the events happening in the world according to this approach coincides with the effect of the actions, which change it in accordance with the previous approach if there is only one observer in the world.

3. *The world changes constantly with the time, the events and actions are just happening along the line.* In this approach the changes are caused by the time while the actions are no longer instantaneous and have physical duration. It has been successfully used in AI planning (Allen, 1983). This approach would allow proper treatment of parallel activities but may require additional synchronisation of the video signals and can lead to very complicated implementation of the multi-threaded services, which typically run on a central server.

The approach that we will adopt to model our world follows closely the first approach as outlined above because our working assumption is that we have only one camera and all information collected from it is processed in a centralised manner. More complex approaches to the dynamic ontology may need to be introduced later when multiple cameras are considered to monitor the same scene. In case of multiple cameras, for example, the video signals need to be synchronised in order to be analysed. This could involve synchronisation of frame rates, elimination of overlapping signals, reducing the delay of frame updates, etc. If, for instance, the movements of one object are identified in one camera output but not in the other because of the differences in frame rate, the discrepancies may occur between data coming from the two signals. This in turn may result in erroneous analytical output. A good candidate for this is the ontology of actions and time based on event structures (Kowalski and Sergot, 1986).

Any activity has a beginning and can be considered to be a one-off activity (instantaneous) or on-going activity (process executing for a certain amount of time). The high-level compound activities are assembled out of primitive ones in an event of receiving new information about the surroundings in a so-called “sense-plan-act” cycle in robot control systems (Shanahan and Witkowski, 2000). The “frame problem” under this assumption will be resolved naturally by the algorithm of the simulator.

During simulation the events may be registered with the simulator in two different ways – either externally, upon arrival of new data from other modules of the framework, or internally, generated directly by the virtual sensors attached to recognised objects in the scene itself. Our simulation follows the concept of “interface” found in the ontology of computer games (Zagal et al., 2007), but in this case the player is substituted with methods responsible for generation of the events within the simulated world. The asynchronous events immediately cause change of the simulated scene e.g. the alarm goes off, somebody steps in and appears on the scene, somebody leaves the scene, etc. The synchronous events are occurring as effects of some actions - e.g. the door becomes open when the agent opens it; the flow of people leaves the room when the last participant in the group leaves the room, etc. Once the individuals and groups are identified and placed in different locations within the boundary of the virtual world the analysis can begin. In the remaining of this section the main concepts of the dynamic ontology, which will be the basis of the analysis are introduced.

3.4.1. Actions

a) ***Action*:** identified action executed by an object that may lead to an interaction with another object, potentially altering its physical attributes.

There are several different types of actions:

b) ***Dynamic Action*:** action executed by a dynamic object that may lead to alteration of physical attributes of another objects.

Examples:

- Object **falls down** from the shelf (the 'location' attribute is changed)
- Object **hits** the wall (the 'location' and 'velocity' attributes are changed)

c) ***Active Action*:** action executed by an individual exhibiting some degree of interaction with another object that may lead to alteration of its physical attributes.

Examples:

- Individual **picks up** a package from the floor
- Individual **touches** the door
- Individual **kicks** another individual

d) ***Passive Action*:** an action executed by an object under specific circumstances that may or may not lead to alteration of physical attributes of other objects.

Examples:

- Object **pushes** a trolley **forward**
- Object **pulls out** a package from the bag

e) ***Group Action*:** action executed by a group of individuals that is a result of a joint effort, i.e. aggregated action of the group participants. The group action may or may not lead to alteration of physical attributes of another object.

Examples:

- Group **moves towards** the door
- Group **climbs up** the stairs
- Group **moves along** the corridor

f) ***Follow Up Action*:** action executed by a dynamic or static object following an event or another action that may or may not lead to alteration of physical attributes of another object.

Examples:

- Individual **pushes** the trolley **forward** until the trolley **collides with** the shelf
- Individual **pushes** the button and the door **opens**

g) ***Reaction*:** an action executed by an individual in response to external cause such as an event, direct or indirect action of another individual. Reaction may or may not lead to alteration of physical attributes of another object.

Examples:

- Individual **punches** another individual who **sustains** an injury
- Individual **falls down** on the floor but quickly **stands up**
- Alarm **goes off** and an individual **looks for** the source of the sound

3.4.2. Events

a) ***Event*:** an identified condition occurring within visual scenes at a specified time that may or may not naturally follow an action. The event primarily describes changes in the state of an object at particular point in time. It may be classified as follows:

b) ***Asynchronous Event*:** event that occurs due to external stimuli that may or may not lead to alteration of physical attributes of an object.

Examples:

- An alarm **goes off**
- The ceiling **falls down** on the floor
- A smoke **appears** in the air

c) ***Change Event*:** event that occurs when an object physical attributes are altered at particular moment in time.

Examples:

- The door **was shut** by the warden
- The light **went off**
- The trolley **has stopped**

d) ***Cause Event*:** event that may potentially cause another event.

Examples:

- A customer **entered through** the door, he **is inside** the shop
- A trolley **has appeared** ahead, the path **is** blocked

e) ***Result Event*:** event occurring within a scene in the aftermath of an action that may or may not lead to alteration of physical attributes of an object.

Examples:

- Individual **punched** another individual, who **sustained** an injury and **fell down** to the floor
- Individual **picks up** a package from another individual and the package **changes** its location
- Individual **dropped down** a bag which **fell on** the floor
- Individual **kicked** the door which **has opened**

f) ***Result Group Event*:** event occurring in the visual scene in the aftermath of an action that may or may not lead to alteration of physical attributes of a group.

Examples:

- Individual **walks away** from the crowd, he **is far away** from it and he **has left** it
- Individual **moves towards** another individual, he **is in close proximity** to him, they finally **meet**
- The Blues **moved towards** the Gunners, The Blues **arrived in close proximity** to the Gunners, The Blues **clashed out** with the Gunners

3.4.3. States

a) ***State*:** a description of an object contained using a collection of physical attributes related to its spatial-temporal appearance on the scene.

Examples:

- Individual **is moving along** the shelf
- Individual **remains on** the floor
- The door **is** jammed

b) ***Configuration*:** a partial, spatial-temporal description of the scene that may include multiple object states. The configuration is a result of the analysis of object's behaviour captured during simulation over a certain period of time.

Examples:

- One participant in the group is **moving alongside** the wall
- For the past 8 minutes object **was laying** on the floor
- From 14:23 to 14:43 the detective **has been visiting** the crime scene 3 times
- Currently 3 groups are **involved in fights** on the street

c) ***Situation*:** a global spatio-temporal description of the visual scene, which is a result of the analysis of different configurations over certain period of time.

Examples:

- At present time the situation **is normal**
- During the last 5 minutes the situation has been **looking suspicious**
- Between 12:00 and 12:16 the situation **became dangerous**

More formal ontology model can be specified using formal languages such as DL (Description Logic). Figure 3-3 shows the top-level class view of the ontology modelled using **Protégé**.

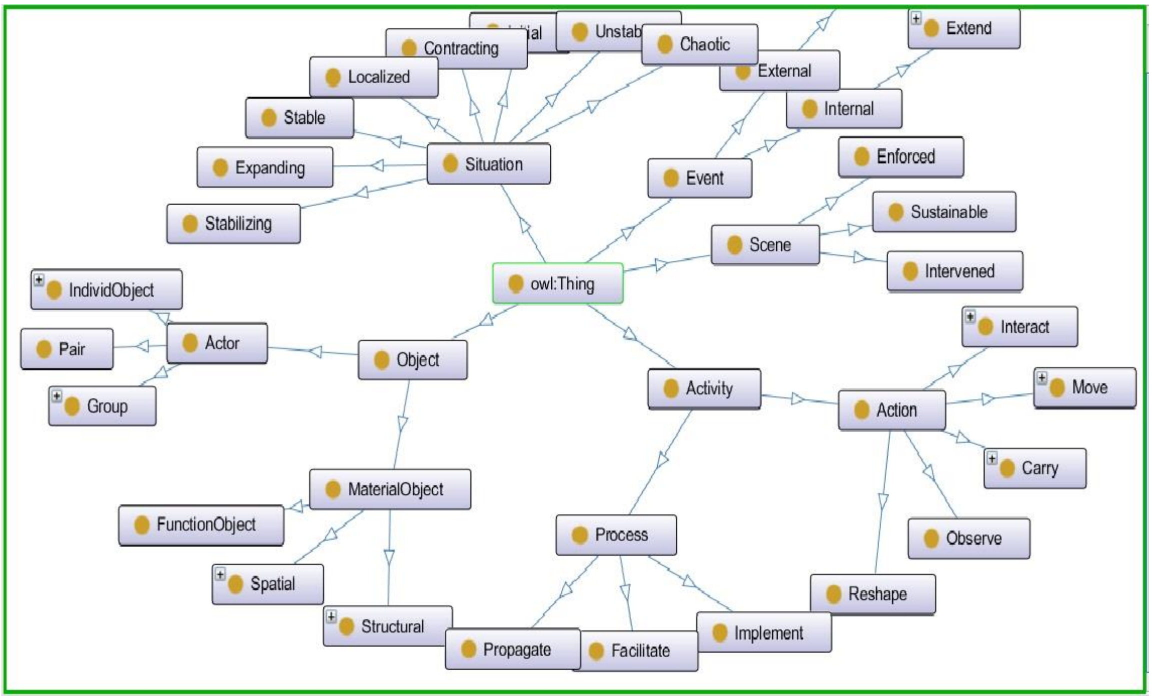


Figure 3-3. Top-level view at the ontology of visual scene combined with the ontology of dynamic behaviour

3.5. Language of Dynamic Behaviour

For the purpose of analysing the dynamic behaviour a linguistic approach has been adopted in which the patterns are described using symbolic expressions, generated by the simulator in the form of event logs. This allows performing the analysis by means of purely syntactic data processing, which is very efficient. In principle, all structural and dynamic patterns of individual and group behaviour can be described using a context-free language. This implies that the patterns are independent on both the nature of the observed world and the particular scenario of using them, which makes our approach generic. More thorough analysis of the patterns can be performed after contextualisation of their appearance in specific situations, e.g. joining hands by two people walking down a dark corridor have different meaning than in the situation when two people meet on the street and reach out to shake hands.

Our language of dynamic behaviour describes what is happening in the visual scene as modelled in the ontology but in purely linguistic terms. The language grammar has been constructed in such a way so that all recognised objects, their states and actions as well as all registered events with their conditions and effects can be described using simple production rules. The terminal symbols of the language have been chosen to drive the parser further into the analysis whereas the non-terminal symbols have been chosen to correspond directly to the classification categories. This allows using the parser as a classifier which significantly simplifies the pattern analysis since the linguistic parsing directly performs the semantic classification.

The main categories defined in the language grammar correspond directly to the ontology as specified above and include the following classifications:

- ***Object***: expression, which describes an object found within the boundaries of the visual scene observed by the camera.
- ***Action***: expression describing physical movements of objects or their constituent parts.
- ***Event***: expression describing an event observed on the scene as a result of an action or other event.
- ***State***: expression for describing of the physical appearance of objects on the scene
- ***Configuration***: expression describing partially the observed scene within the focus of the camera
- ***Situation***: expression describing the global state of matter on the scene in high-level terms

To describe various specific use case scenarios non-terminal descriptors have also been added to the language grammar. They are parts of the service-level descriptions, which target specific use-case scenarios, and the overall maintainability of the framework.

a) ***Motion Action Descriptor*:** a more detailed description of an action executed by an individual or a group of individuals that is directly related to the physics of movement.

Examples:

- Individual **walks towards** the fence
- Group **moves along** the building on the left
- Individual **runs away from** the van

b) ***Interactive Action Descriptor*:** a more detailed description of an action executed by individual or group of individuals that is directly related to its interactive nature that may change the State of an Object. These descriptors are usually part of a description of a more general activity and include terms such as 'touches', 'picks up', 'kicks', 'punches' etc.

Examples:

- Individual **touches** the door handle
- Group **forces** a policemen to move aside
- Individual **kicks** another individual he is involved in fight with

The principle driving the construction of the language grammar must be to capture the reality on sufficient level of granularity which reflect the appearance of all relevant elements at given time and a given space. The control over the granularity is achieved entirely using classification of terminal and non-terminal symbols. The so-called non-terminal descriptors are used only to give the expressions more specific meaning, preserving the context-free nature of the language. The complete specification of the language grammar in Backus-Naur form is provided in Appendix A.1.

3.6. Research hypothesis

The inference of behaviour directly from a video camera feed is a difficult task due to the complex nature of visual data that needs to be processed. Format, resolution, frame rate of the received video, physical capabilities and limitations of hardware, or ever changing environmental conditions such as illumination variations or dynamic cluttering have an impact on the visual output that has to be analysed in real-time. The main goal of this research is to construct an efficient framework for real-time analysis of dynamic behaviour of individuals and groups of individuals moving in enclosed public spaces at walking speeds using limited information extracted from live video feeds as a combination of minimal physical data and sufficient simulated data generated in real-time alongside the video feed. The key to the success of this task is the implementation of efficient algorithms for recognition of the patterns of individual and group dynamic behaviour using limited data about the objects on the scene, their location within the 3D space of the scene with estimation of the dimensions of their constituent parts and information about the direction of possible movement. The approach, which is adopted in this research, is based on replacing the necessity for image processing of the entire video stream with symbolic analysis of partial and approximate information generated by simulation. This approach was originally presented in Gasiorowski, Vassilev and Ouazzane (2016) and is elaborated in two subsequent articles prepared for publication.

In our hypothesis it is assumed that the object identification, the recognition of their physical properties and the reconstruction of their trajectories of movement can be established prior to the analysis to propel the simulation (Fig. 3-1). Two types of input data are required by the simulator to support the simulation cycle. The first type is needed to perform the initial setup of the bound world when the simulation starts, e.g. sets the physical properties of the scene and objects being part of it. This situation will be detected by the simulator which will receive all its input data in a streamlined XML format. This happens relatively rarely when new objects appear on the scene as a result of entering the focus of the video camera and it leads to re-initialisation of the simulator loop. The other type of data is concerned with the incremental changes in the visual scene that drive the continuous simulation. It may involve alteration of attributes of existing objects or inclusion of new ones but will be executed within the current setup of the simulator loop.

Since the simulator is constructed using 3D programming techniques, it is anticipated that data will be delivered to the simulator in a standard vector notation to describe the location, velocity, dimensions and orientation of every object identified in the video stream. In order to represent more complex motions, the input data can be extended to include twisting, bending, rotation, and so on. Such data can be also represented in XML format. The theoretical foundations for such an extension will be discussed in details in the next chapter.

The adopted approach also poses few uncertainties. There is a possibility that object's representation discrepancies in video and simulation may become too great to reliably project the reality in three-dimensional space. If, for instance, new objects are discovered in the visual scene, but their physical properties are misinterpreted, this may lead to their appearance in wrong locations within the simulated world. That may lead to the generation of inaccurate data which when analysed will produce wrong analytical results. The contingency plan for situations like this contemplates minimising the errors by adjusting the parameters, which govern the simulation.

Chapter 4

3D simulator of individual and group dynamics in visual scenes

With the use of latest technological developments in the area of 3D graphics, an approximated appearance of the visual scene and its dynamics has been constructed. It is assumed that the data originating in frames captured by surveillance cameras can deliver sufficiently rich information to represent objects and their movements within the boundaries of the monitored physical space. In order to achieve this goal, one can assume that other modules of the framework can streamline the input data in standard mathematical notation (see Appendix A1). The implementation of the simulator uses a combination of open-source software packages.

- The Java-based **jMonkeyEngine** provides an adequate high-level tool for managing the simulator and analyser. The engine is utilised for visualising the dynamic movements. It also incorporates a logger, which is suitable for implementing the event capturing mechanisms.
- The low-level 3D graphics support is provided by **OpenGL** graphics library, on top of which **jMonkeyEngine** is built**.**
- **Blender** software is used for offline modelling of the visual appearance and animation of various objects and their dynamic behaviour. In this chapter, the development of the simulator application with attention to demonstrating where it fits in the recognition of dynamic behaviours workflow is presented.

4.1. Overview of the software implementation

The requirements for the simulator were set initially to meet the ontological model of visual scene presented in Chapter 3. The focus was on the construction and implementation of a graphical model capable of providing the operational semantics for the pattern language. Straight from the beginning a bottom-up approach has been adopted, i.e. more sophisticated features were constructed out of critical, but often simplified methods. The technological principles behind **OpenCV** are based on the 3D modelling which uses standard mathematical theory so all graphical descriptions were implemented in relation to Cartesian coordinate system. Brief summary of the vector calculus and analytical geometry on which 3D modelling is based are presented in Appendix A1.

To avoid facing the complexity of modelling too early, few reasonable limitations were put into place:

a) The visual scene needs to fit within the boundaries of the area in which objects can be positioned.

b) The appearance of various objects on the scene is limited to abstract representations using simple 3D geometries such as dots and spheres.

c) The *location* and *direction* vectors describe the position and orientation of an object within the scene relative to the beginning of the coordinate system.

Initially, graphic models were constructed directly using the available graphics libraries in **OpenCV**. This allowed building and testing the theory behind movements of objects without the need of elaborate modelling. The process involved only simple geometric transformations for simulating movements, executed by the built-in engine and employed only collision detection methods, controlled by **jMonkeyEngine**.

Once the first cut-off simulator was produced **Blender** software was introduced to further elaborate the 3D models to achieve higher degree of resemblance to the real world. A specific advantage of this software is that it allows visual manipulation of every part of the model in real-time (Blender Foundation, 2015). The final outcome of the process of modelling using **Blender** was a 3D model in a universal file format (e.g., XML), which can be read directly by the interfaces of graphics engines, including **jMonkeyEngine**.

Further enhancement of the simulation was implemented through considering the body parts and attaching *ghost controls* to them. The *ghost* is a non-physical invisible geometry attached to physical one (Reese and Johnson, 2015; Edén, 2014). It can be used to probe the space and report geometries that come into contact with it. Ghost controls have been assigned to the overall shape of individual as well as to its constituent body parts. The built-in ghost controls created the possibility to monitor the individuals at a lower level of granularity while preserving the holistic approach to locomotion tracking. Following this strategy, additional information, which can be used to analyse the group behaviour, has been produced.

4.2. Software architecture

The prototype is implemented after an object-oriented approach and consists of several subsystems as depicted on the component diagram shown in Figure 4-1. The software architecture closely matches the conceptual structure and workflow presented in Chapter 3 (Figure 3-2).

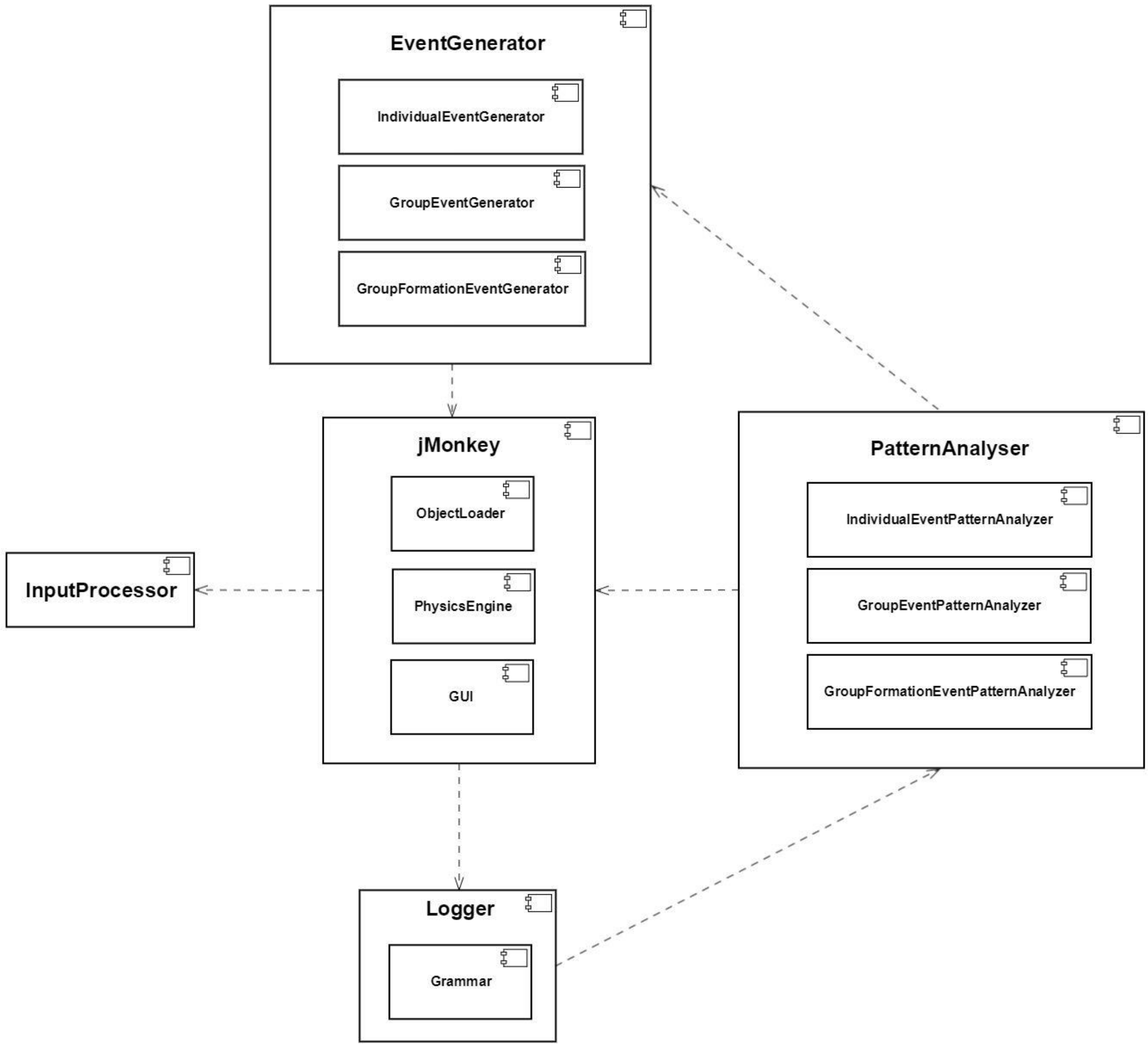


Figure 4-1. Component Diagram of the Simulator. The directed dashed lines indicate the direct data dependencies between the components

The input data is processed by the **Input Processor** component, which configures all parameters of the simulation, such as number of recognised objects, their locations and orientation, etc. The **Object Loader** then initialises the visual scene and loads all required graphical models into the three-dimensional space of the scene. The **jMonkeyEngine** component lies at the core of the simulator, as it is responsible for updating the state of the visual scene from time to time and control the rendering. The changes are invoked by *external events*, such as arrivals of new data and *internal events*, caused by the simulation itself. In addition, it integrates the built-in engine for simulating physics of three-dimensional scene, synchronises the data flow within the application and provides **Graphical User Interface (GUI)** for real time manipulation of parameters and information display. It can be switched off if the simulator is used with other modules of the framework.

The task of the **Event Generator** is to evaluate the state of the simulation and to capture the events, which may arise during the simulation. The logics of this component are largely based on the collision detection techniques, such as *ray cast*ing and *ghost controls*. The events are processed by the corresponding components, e.g. the **Individual Event Generator** produces events related to dynamic movement of an individual that are next analysed by **Individual Event Pattern Analyser** being an integral part of **Pattern Analyser** subsystem. The parsing results in classification of the pattern that corresponds to the observed dynamic behaviour. The **Logger** component is responsible for formulating sentences following the grammar rules of the pattern language, which is supported by the **Grammar** subcomponent.

The modular nature of the software architecture of the simulator itself makes it also possible to expand further its functionality by simply introducing additional types of events without breaking the existing functionality.

4.3. Design considerations

The **Simulator** is a self-contained software application capable of simulating dynamic movements of individuals within the three-dimensional space in real-time as illustrated in Figure 4-2. The simulator possesses several critical features that are important for the subsequent analysis of behaviour. It allows

- To observe the changes occurring in a 3D scene visually.
- To adjust the simulation parameters interactively at runtime.
- To trace the events arising during unfolding of different case scenarios in a console panel.
- To save and load the simulation setup in configuration files.
- To generate log files in two distinct file formats (text and XML) which can be used for further analysis through contextualisation or decision-making.

Figure 4-2 shows the three panes of the simulator – the visual output produced during simulation, the event log generated by the Logger and the configuration parameters used by the simulator.

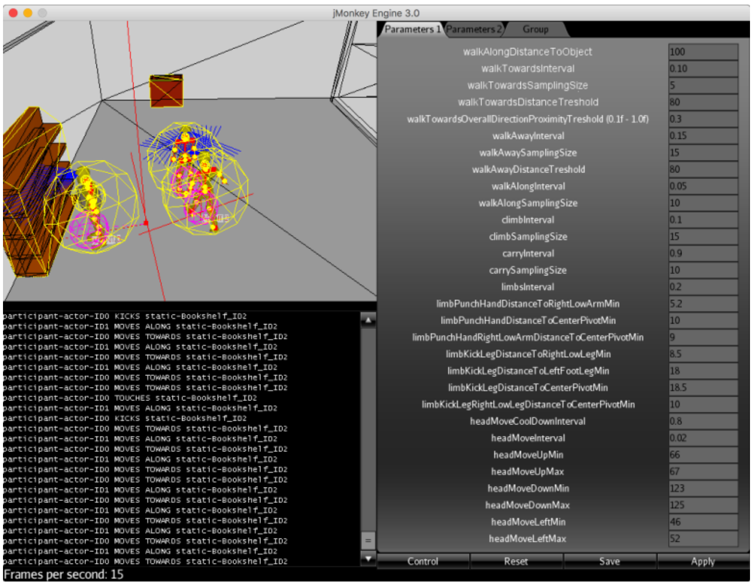


Figure 4-2. The simulator interface

The simulator was designed to be a central component of the model-driven dynamic behaviour analysis framework (Figure 3-1) and as such it interacts with other modules of the framework. However, since it is possible to generate input data interactively under the control of the keyboard, it can be used as a standalone product as well. This is particularly useful for preparation of configuration files, since the tuning of the parameters can be done in offline mode without impacting the production environment and the configuration file can be automatically generated.

4.3.1. Simulation data

In order to function properly the application must be initialised using data, which sets the visual scene (*world data*). The world data (a sample is shown on Listing 4-1) is parsed at runtime in the beginning of the simulation loop and at its next review. The minimal data needed to setup the visual scene includes the following:

a) *Total number* of recognised individuals in monitored area in a form of a real number.
b) *Identifiers* of any individual being under surveillance.
c) *Physical location* of the individuals in vector notation.
d) *Viewing direction* of individual in a form of a normalized vector.


```
<streamInput>
	<parameterSet class="WorldManagerState">
		<parameter>
			<name>numberOfAgents</name>
			<value>
				<actors number="3">
					<actor id="actor-ID0">
						<location>
							<vectorX>161.19113</vectorX>
							<vectorY>4.3999987</vectorY>
							<vectorZ>-0.8794838</vectorZ>
						</location>
						<viewDirection>
							<vectorX>-0.87773572</vectorX>
							<vectorY>0.0</vectorY>
							<vectorZ>0.48221964</vectorZ>
						</viewDirection>
					</actor>
					<actor id="actor-ID1">
						<location>
							<vectorX>121.19113</vectorX>
							<vectorY>4.3999987</vectorY>
							<vectorZ>21.8794838</vectorZ>
						</location>
						<viewDirection>
							<vectorX>-0.87773572</vectorX>
		…
```


Listing 4-1. An excerpt of streamlined XML file describing spatial properties of visual scene captured from camera

During simulation the simulator constantly updates the current state of the visual scene using the incoming data, which is streamlined from the keyboard generator or another module at a constant rate. To retain the independence all input data of the simulator is in XML format. When simulator receives an input directly from the keyboard, the physical properties of the scene and objects are being updated according to the implemented rules at the next frame update. When an external module supplies data, the simulator is required to first parse the received XML file and store the extracted tree-structured data in the memory before it can update the visual scene. The simulator then updates the physical properties of the visual scene accounting the current frame rate of the machine and elapsed timestamp (in seconds) extracted from the received data. The frequency of the input coming from another module may vary and is subject to the performance of all modules participating in the operation. It is assumed that at the very minimum the received XML file contains data on the reconstructed trajectory for each recognized agent (Listing 4-2) defined in a file responsible for setting the visual scene (Listing 4-1). The structure of the file describes recorded motion of monitored individuals in vector notation. Each entry contains information about *time* at which the motion has been registered in a form of a *number in a sequence* (counter attribute) and time (in seconds) elapsed since the beginning of the recording (tpf attribute). These values are critical for satisfactory approximation of recorded movements within the simulated visual scene.

a) *Total number* of recognised individuals in monitored area in a form of a real number. This number is required to match the total number of agents parameter delivered in stream input XML file setting the visual scene (Listing 4-1)
b) *Identifiers* of any individual being under surveillance. These identifiers need to match the total number of agents parameter delivered previously in stream input XML file setting the visual scene (Listing 4-1)
c) *Movement of an agent* recorded at particular frame, specified by “counter” attribute.
d) *Number of seconds* that passed since the beginning of the monitoring of the individual specified by “tpf” attribute.
e) *Viewing direction* of individual in a form of a unit vector at the time of recording the movement.
f) *Walking direction* of individual in a form of a unit vector at the time of recording the movement.


```
<video>
  <movementSet class="VideoState">
    <agents number="3">
      <agent id="agent-ID0">
        <movement counter="0" counterID="agent-ID0-0" frameRate="0.49727827" time="20-10-2017-10-
50-07" tpf="0.0" tpfSinceLast="2.0109465">
          <globalPosition>
            <vectorX>-78.60532</vectorX>
            <vectorY>4.3999987</vectorY>
            <vectorZ>-61.354675</vectorZ>
          </globalPosition>
          <walkDirection direction="">
            <vectorX>0.0</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.0</vectorZ>
          </walkDirection>
          <viewDirection direction="none" directionValue="0.0">
            <vectorX>-0.87773573</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.48221964</vectorZ>
```

```
          &lt;/viewDirection&gt;
        &lt;/movement&gt;
      <movement counter="1" counterID="agent-ID0-1" frameRate="1.9120132" time="20-10-2017-10-50-
07" tpf="2.0109465" tpfSinceLast="0.52300894">
          &lt;globalPosition&gt;
            &lt;vectorX&gt;-78.60532&lt;/vectorX&gt;
            &lt;vectorY&gt;4.399999&lt;/vectorY&gt;
            &lt;vectorZ&gt;-61.354675&lt;/vectorZ&gt;
          &lt;/globalPosition&gt;
          &lt;walkDirection direction=""&gt;
            &lt;vectorX&gt;0.0&lt;/vectorX&gt;
            &lt;vectorY&gt;0.0&lt;/vectorY&gt;
            &lt;vectorZ&gt;0.0&lt;/vectorZ&gt;
          &lt;/walkDirection&gt;
          &lt;viewDirection direction="none" directionValue="0.0"&gt;
            &lt;vectorX&gt;-0.87773573&lt;/vectorX&gt;
            &lt;vectorY&gt;0.0&lt;/vectorY&gt;
            &lt;vectorZ&gt;0.48221964&lt;/vectorZ&gt;
          &lt;/viewDirection&gt;
        &lt;/movement&gt;
[…]
```

Listing 4-2. An excerpt of streamlined XML file describing the movement of agents delivered by another module of the general framework

The detection of any modification of world data triggers an update of the global state of visual scene automatically.

4.3.2. Simulation Loop

At the core of the simulation is the *simulation loop,* which reads the incoming information and adds the response to an action or event on the scene.

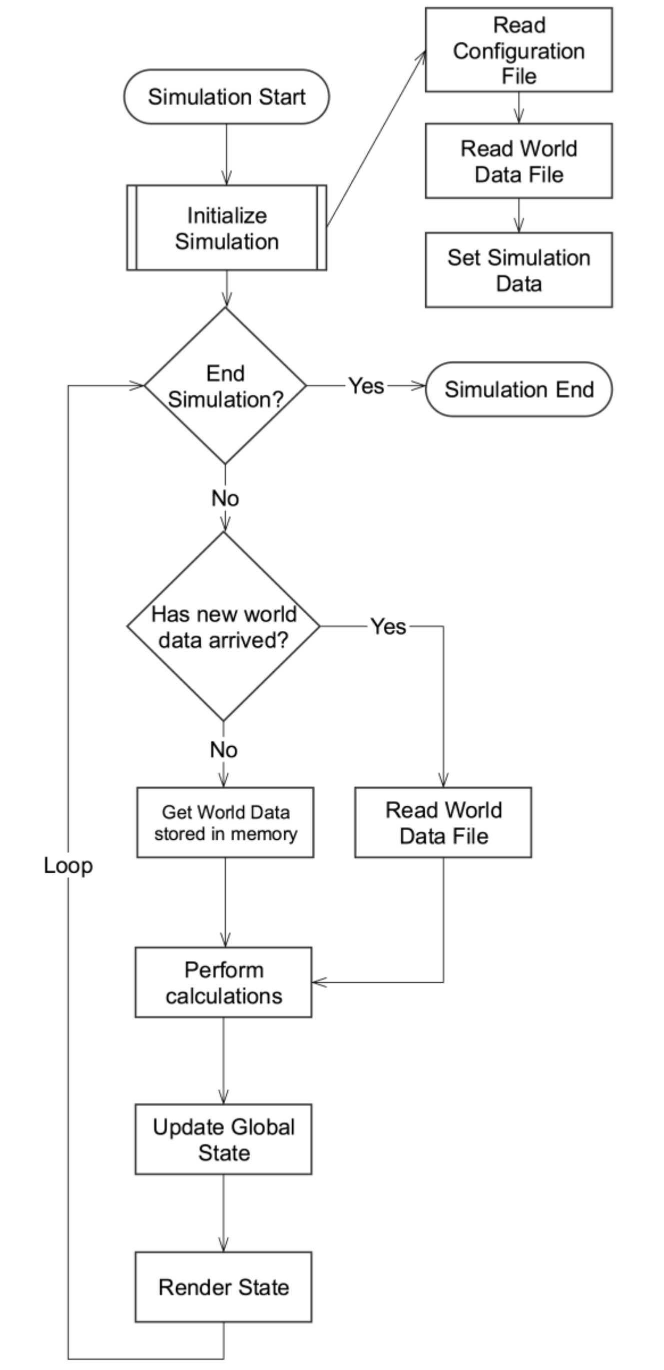


Figure 4-3. The algorithm of the simulation loop.

The simulation loop is similar to the *game loop* used to drive many interactive games (Nystrom, 2014). Through repetitive execution of the scanning algorithm of the simulator it periodically updates the global state of the system. In case of 3D simulation like in our case, the simulator loop is responsible also for *rendering* 2D images accordingly so that the scene may visually reflect the changes. A single execution of methods in this block will be referenced as *simulation cycle* or *simulation step*. A simplified algorithm of the simulator cycle is shown on Figure 4-3.

Rendering is a process in which geometrical forms of objects are being assembled out of set of *points* (vertices) and *graphics primitives* that hold information on how those points need to be connected in order to generate a shape in the image frame (Lengyel, 2003). An example of rendering of a simple box as 3D graphics is presented in Figure 4-4.

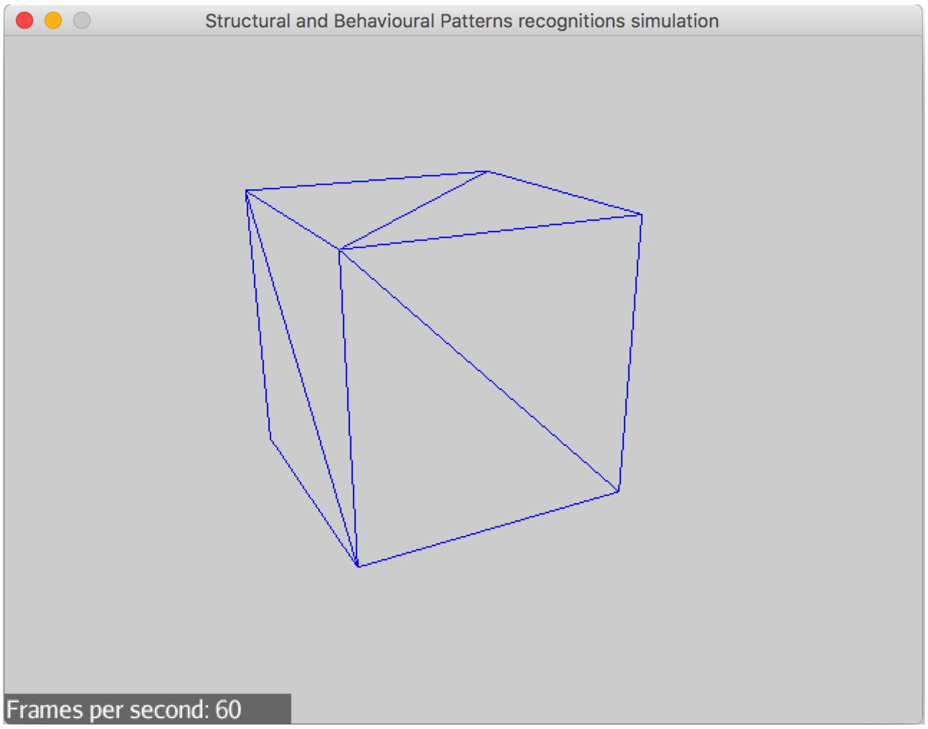


Figure 4-4. The cube is formed out of triangles made out of three points (vertices) connected using three line segments (edges)

The frequency of updates depends on multiple factors, such as technical specification of hardware, number and complexity of the operations within each loop step and geometric data that needs to be re-rendered at each loop step. Considering the latest technological advancements of graphics cards, central processing units (CPU) and other hardware components, one can easily achieve fast rendering beyond the necessary for subsequent analysis. To balance this, some delay to the updates has been introduced, so that simulation runs at constant rate of 30 frames per second (FPS). The value of FPS has been chosen to meet the average frame rates of industrial CCTV cameras (IPVM, 2015).

4.3.3. Human body in motion

Individuals play a central role in providing the operational semantics of pattern language due to their impact on the dynamics of the entire visual scene. For the sake of clarity, in this section the individuals will be named agents and a distinction between their conceptual, functional and analytical appearance on the scene will be made. At the very basic level, the agents have been represented by their absolute position within the three-dimensional scene in a similar way to how locations are described with latitude and longitude within Global Positioning System (GPS) (Official U.S. Government, 2015). Starting with this, the individual descriptions have been further expanded by specifying their *orientation* using a normalized vector. It is possible to add more elements of the body dynamics (i.e., speed, acceleration, etc.). However, in this research the main interest is in the geometric orientation of the individuals, which determine two important characteristics to be used to analyse the pattern of behaviour, namely the *direction of movement* and the *viewing direction* at any given moment.

Although these oversimplifications are acceptable at the start, they can cause difficulty in establishing the precise physical occupation of visual space by each individual if retained. To mitigate this, a series of models of the human body with increasing complexity have been developed. They were used for interpolating the invisible parts of the bodies. All these models were based on the use of spheres as their volume remains invariant to rotations, i.e. they can be oriented in three-dimensional scene without the need to change their shape. Additionally, this geometry was selected not just for the simulator, but also for the entire framework because the spheres could be easily positioned within the surrounding environment. Formula 4-1 provides the analytical basis for calculating invisible parameters of 3D bodies such as depth and thickness on the basis of estimation of 2D parameters.

$$\vec{u} = [u_x, u_y, u_z]$$

$$(p_x - u_x)^2 + (p_y - u_y)^2 + (p_z - u_z)^2 - r^2 = 0$$

where $\vec{u}$ is the *absolute (or global) position vector* of an agent, p is a point lying on the sphere surface and r is the radius of the sphere.

Formula 4-1. Equation for abstract representation of a spherical agent

The movement of a sphere with constant velocity can be calculated using the vector representing the orientation. If one considers the monitored area to be a flat surface without objects can change their positions vertically e.g. stairs or steps, then one can calculate the locations with Formula 4-2.

$$\hat{v} = 1/\parallel \vec{v} \parallel \vec{v}.$$

$$(p_x - (u_x + \hat{v}_x))^2 + (p_y - (u_y + \hat{v}_y))^2 + (p_z - (u_z + \hat{v}_z))^2 - r^2 = 0$$

where $\hat{v}$ is the unit vector representing the orientation of the sphere.

Formula 4-2. Calculating movement of a sphere in 3D space accounting the viewing direction.

Given that only the direction that an agent is facing is required, a unit vector is used. The floor is represented as a flat surface (a plane) so the values of u_y and $\hat{v}_y$ will always be equal to 0. The *viewing direction vector* is added to the global position vector of an agent recorded at previous frame. Performing this calculation in relation to FPS, one can achieve smooth transition from one location to another ultimately simulating linear movement of an agent.

With a sphere, one can calculate the position, orientation and approximate the overall volume of an agent. However, considering anatomical structure of a human body, the observable movements are built out of more atomic level of actions. When one perceives dynamic behaviour in a wider picture, it will quickly become clear that legs and hands play a significant role in the genesis of human activities. An individual can gesticulate, make poses and carry out daily activities with the use of the limbs according to the situations. To that end, an extension of a single sphere model to a more advanced geometry is needed so that it could also incorporate limbs as well.

Strategy adopted in this research is similar to hierarchical classification of actions outlined by Gong and Xiang (2011) in which three layers are considered:

a) The *atomic actions* upon which actions an action is formed. For example, in a kicking movement scenario, agent needs to *"move right leg in front of the right leg"*.

b) *Actions* are sequences of atomic actions that fulfil a function or purpose, e.g. "*picking up, walking*" etc.

c) *Activities* are composed of sequences of actions over space and time, e.g. "*an agent is moving towards the door*" or "*an agent picks up an object from the floor"*.

Therefore, for analysing more complex movements of the individuals one needs to account the graphical model and movements of the limbs. Tracking individual body components would also allow to infer more detailed information about events occurring on the scene and build up the knowledge on activities that invoke them. For example, if an agent is stretching its arm in the direction of a shelf one can interpret this as *"agent reaches out for an object on a shelf""* which would not be possible without tracking the movement of the hand.

4.3.4. Accounting the movements of the individual body limbs

Initially, several spheres located at particular offsets from each other were used to model the body with limbs. Few issues were quickly identified with this approach. Firstly, spheres were not only difficult to move in a regular manner along with other parts of the body, but also hard to track at the time of movement. Secondly, spheres had the tendency to overlap each other, which produced graphically confusing results.

In a second attempt, cubic geometries that would account better the extents of individual limbs in relation to the body have been tried. Although it eliminated the overlapping, the volume of the geometric calculations during rotations increased substantially. As a trade-off between the two extremes the decision has been made to use cylindrical representation of the body parts. Cylinders can be rotated along their length without the risk of complex geometric calculations as in the case of arm-twisting.

Geometric calculations provided by the graphic engine have been used to increase the efficiency of this solution. In order to make an object visible in 3D scene **jMonkeyEngine** needs to be attached to a *node* being part of the *scene graph*. The scene graph is simply a tree data structure consisting of nodes organised in an ancestor-descendant hierarchy. The engine utilises this concept to ease the process of storing and keeping track of the state of all objects on the scene. The nature of this data structure implies that the descendant nodes inherit all transformations from their ancestors, i.e. when a transformation is applied to a parent node it is also applied to all children nodes. The advantage of this assumption is that one can manipulate the physical appearance of a great number of objects (*spatial*) belonging to a particular group (*sub-tree*) in parallel by calculating the transformation of the group as represented by one of its member nodes only.

This allowed considering the limbs as a group and performing the geometric calculations for the entire group in a very efficient way. We used the concept of a scene graph to assemble the agent's body out of several geometries attached to a single tree node corresponding to the body). Because individual limbs have to be positioned relative to the *central node* representing the entire body, one could treat it as an origin point of a *local coordinate system*. All body parts are represented using nodes, which are attached to this central node with different offsets for each limb, thus forming separate sub-trees in the tree of nodes representing the body (Figure 4-5). Such configuration provides an efficient way to coordinate the movements of the limbs because the individual transformations of the body parts are dependent on the transformation of the body as a whole and can be calculated using the results of the calculation for the topmost node of the corresponding tree. This principle is utilised in simulation of agent's movements such as walking where it is necessary to relocate all body parts simultaneously.

The calculations in the case of using elliptic cylinders require taking into consideration both the radiuses and the height of the cylinders (see Appendix A1). Positioning the base of a cylinder at the origin of the XY Euclidean plane, where radius on the X-axis is r, radius on the Y-axis is s and its height h along Z-axis one can calculate its lateral surface using Formula 4-3.

$$x^2 + m^2y^2 = r^2$$
$$0 \leq z \leq h$$

where $m = r/s$ and is the ratio of the two radiuses. If the ratio is equal to 1, i.e. $r = s$ then $m = 1$ and the cylinder is a sphere.

Formula 4-3 Analytic description of elliptic cylinders in Cartesian coordinate space (Lengyel, 2003)

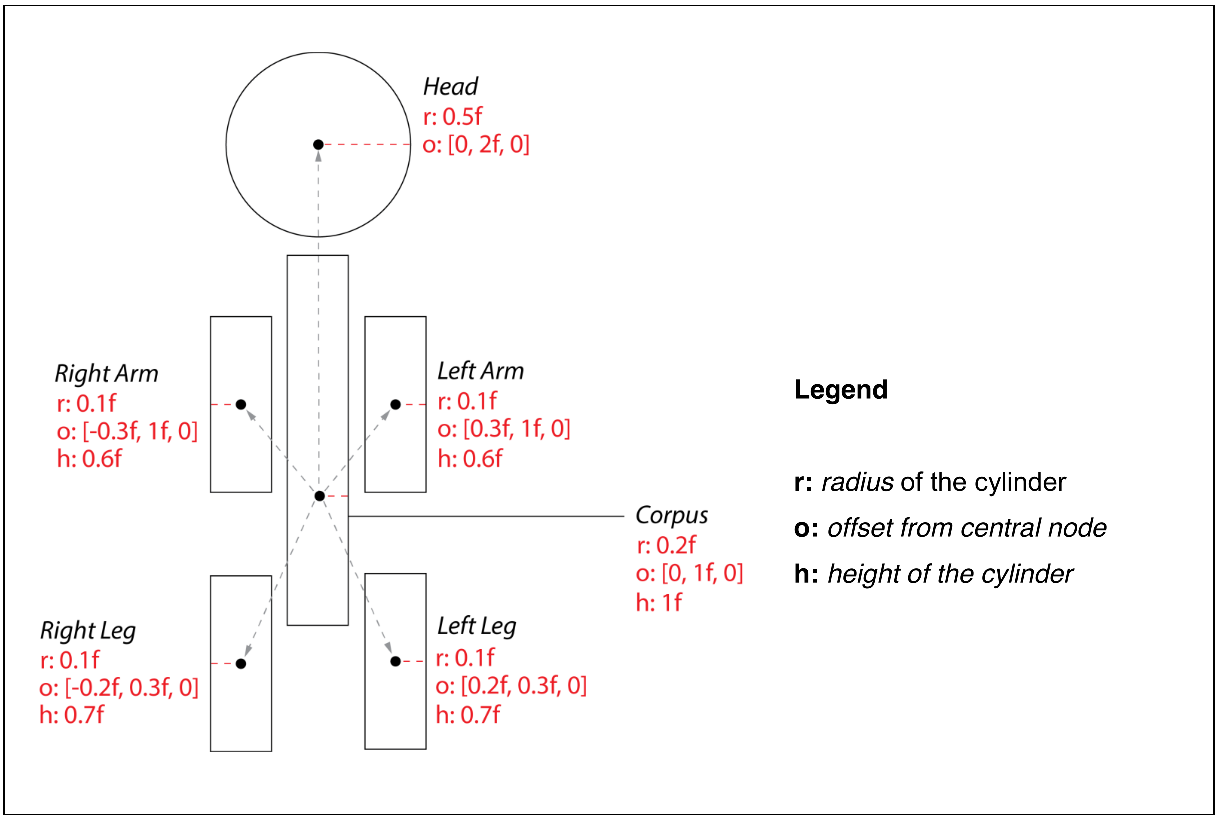


Figure 4-5. Structure of the representation of a body made out of primitive geometries representing the individual limbs

To construct a model, which corresponds to the structure presented in Figure 4-5, one has to calculate the individual cylinders relative to the central node. This can be carried out by adding an appropriate offset vector and then subtracting half of the cylinder height to correctly position its base (e.g., to avoid the situation when the arm is above the torso). Note that z-component of $\vec{u}$ is equal to 0 which means that only an offset of limb on XY Euclidean plane is considered.

$$\vec{l} = [l_x, l_y, l_z] \qquad \vec{u} = [u_x, u_y, 0]$$

$$\left(l_x + \left(u_x - \frac{h}{2}\right)\right)^2 + m^2\left(l_y + \left(u_y - \frac{h}{2}\right)\right)^2 = r^2$$

$$\left(l_x + \left(u_x - \frac{h}{2}\right)\right)^2 + m^2\left(l_y + \left(u_y - \frac{h}{2}\right)\right)^2 - r^2 = 0$$

where $\vec{l}$ is a position of a central node of the model in regards to global coordinate system of the scene, $\vec{u}$ is an offset vector, h is the height (length of a limb) and r is the radius of cylinder.

Formula 4-4. Description of a cylindrical limb included in the agent model presented in Figure 4-5

The head positioning can be calculated similarly but using the equation about sphere instead (Formula 4-5).

$$\vec{l} = [l_x, l_y, l_z] \qquad \vec{u} = [u_x, u_y, 0]$$
$$(p_x - (l_x + u_x))^2 + ((p_y - (l_y + u_y))^2 + ((p_z - (l_z + u_z))^2 = r^2$$
$$(p_x - (l_x + u_x))^2 + ((p_y - (l_y + u_y))^2 + ((p_z - (l_z + u_z))^2 - r^2 = 0$$

Formula 4-5. Describing a position of a head in the agent model presented in Figure 4-5

Structuring the model in this manner allows us to track not only the global position of an agent but also the movements of its limbs. This in turn creates the possibility to analyse the intentions of an individual through considering more complex dynamic patterns of behaviour, which involve limbs. However, the use of cylinders is inefficient due to the large number of linear transformations that needs to be executed in a relatively short time. Furthermore, geometrical manipulation of cylinders does not provide sufficient data for prognosis of the movement and thus is not very suitable for more fine analysis.

4.3.5. Employing body armature

The *armature* is a concept of building a skeleton of organic living beings consisting of a number of mutually dependent bones defined by line segments (Blender Foundation, 2014). Armatures are used in many 3D applications such as **Blender** modelling tool (Blender Foundation, 2015) and Microsoft **Kinect** (Microsoft, 2015) game engine because they provide an easier method for describing skeletal structure of humans and simulating movements and motion capturing of limbs. Considering all advantages provided by the armatures, they were adopted in the implementation of the simulator as a basis for the subsequent behaviour analysis.

Key for developing of a good body armature is the available data, which captures individual body part movements. The datasets of this kind aim to provide recording of ready-to-use motions commonly occurring in natural settings, such as in the kitchen (De la Torre et al., 2008). There are several extensive datasets developed using advanced technologies but they are pricy and cost-inefficient for the purpose of our implementation. Instead of this, free databases of pre-recorded files capturing human motions (see Carnegie Mellon (2015) and Hahne (2008)) have been used. Such files are typically presented in a special **bvh** format, which contain information about position and rotation of individual bones during fixed intervals of movement as shown in Listing 4-3.

```
HIERARCHY
ROOT Hips
{
	OFFSET 0.00000 0.00000 0.00000
	CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
	JOINT LHipJoint
	{
```

```
            OFFSET 0 0 0
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT LeftUpLeg
            {
                OFFSET 1.36306 -1.79463 0.83929
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT LeftLeg
                {
                    OFFSET 2.44811 -6.72613 0.00000
                    CHANNELS 3 Zrotation Yrotation Xrotation
                    JOINT LeftFoot

[…]
```

Listing 4-3. An excerpt of a **bvh** file. The recorded dynamic action consists of information about participating bone motions as well (Hahne, 2008)

One can graphically reconstruct an armature of an agent from data stored in **bvh** files since the description of bones contains information about their offsets from parent bones. Additionally, the transformation of skeleton's bones can be presented as sequences of visual frames, thus forming skeleton animation that can in turn be used for the subsequent analysis. At this stage **Blender** was adopted as an armature-modelling tool (Blender Foundation, 2015). This software application combines a great amount of specialised accessories for creating complex three-dimensional geometries. The geometry of an agent is based on the anatomical structure of armatures used in animations of human motions in public databases (Hahne, 2008). By assigning an armature to the geometry it is possible to alter spatial properties of vertices according to the movement of a bone that they surround as shown in Figure 4-6.a and Figure 4-6.b. The final version of an agent's geometry equipped with an armature capable of manipulating its surface has been presented in Figure 4-7.

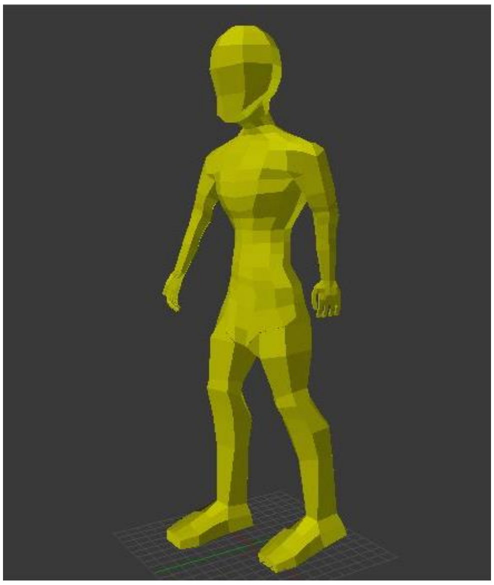


Figure 4-6.a. The final result of modelling a humanoid agent in **Blender**

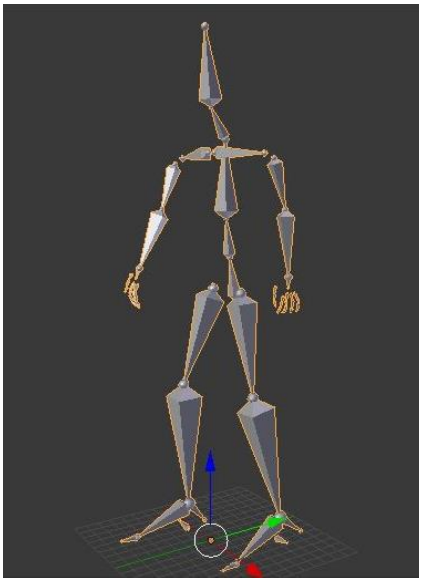


Figure 4-6.b. The armature of a human agent relative to the sculpted geometry

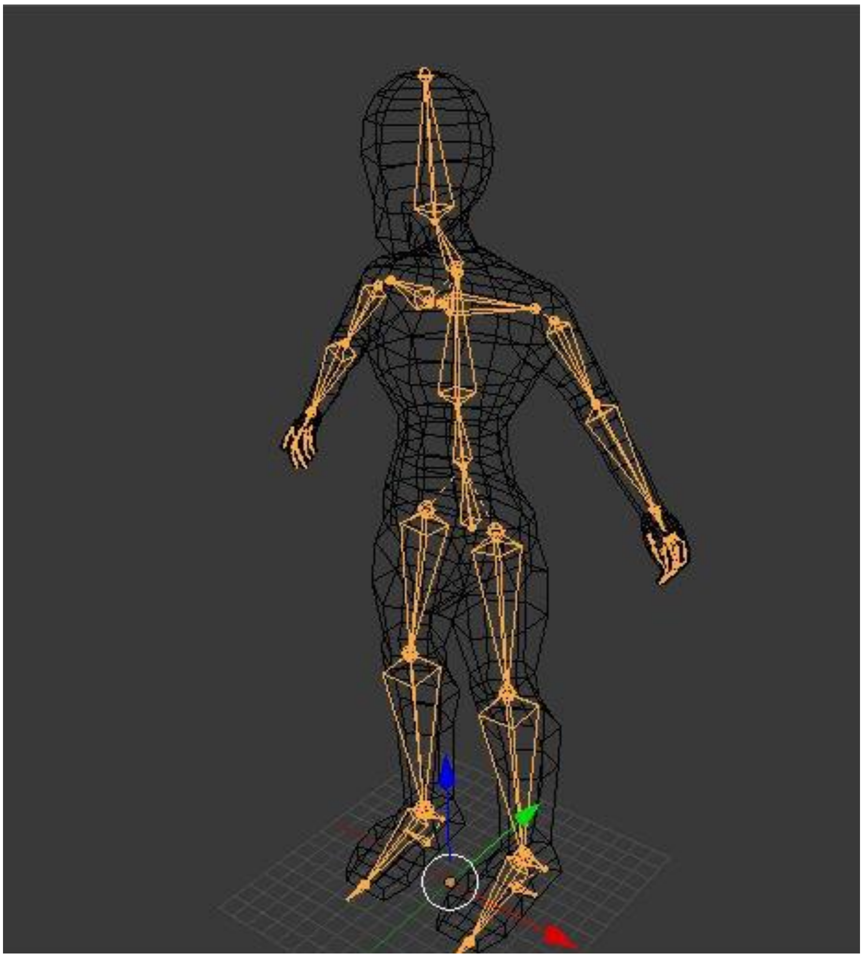


Figure 4-7. The armature combined with agent's geometry in wireframe preview mode

4.3.6. Animating individual movements

The algorithmic nature of dynamic patterns requires processing a very extensive set of movements constituting the dynamic activity of an agent. Let us consider a scenario in which a person needs to pick up an object from the floor. In order to exhibit such behaviour the person needs to bend or kneel down, stretch an arm into the direction of the object and then return to the original position holding the object. In this scenario, one can identify three key poses of the body, namely kneeling down, stretching an arm and standing up. Between them one needs to simulate movements, which complete the motion. In order to simulate such behaviour with armature one has to specify first the *key frames*, i.e. the starting and ending transition points of the involved movements. The remaining frames in the set represent the intermediate positions of the body in transition between the different poses.

For the purpose of the pattern analysis, a fixed range of movements (Table 4-1) has been adopted. Note that the situations in which different movements occur have been classified into three categories depending on their importance.

Table. 4-1. Movements which are used in the animation

Animation Type	Potential movements involved
Normal	Walk (Fig.4-8a), Look Left, Look Right, Look Up, Look Down, Idle, Reach Out (Fig.4-8b)
Dangerous	Run (Fig.4-8c), Punch (Fig.4-8d), Kick (Fig.4-8e)
Suspicious	Run, Reach Right, Pick Up (Fig.4-8f), Drop Down, Head Shake

The process of creating animations was conducted as follows (Figures 4-8a to 4-8d):

1) Identification of animations required by different case scenarios.
2) Recognition of the key poses that form the entire animation.
3) Manipulation of the individual bone location, rotation and scale to achieve desired pose in **Blender**.
4) Creating key frames in **Blender** at specific frames and encapsulating individual animations as a data blocks.
5) Exporting entire 3D object model along with the stored data block animations to a format recognizable by **jMonkeyEngine**.

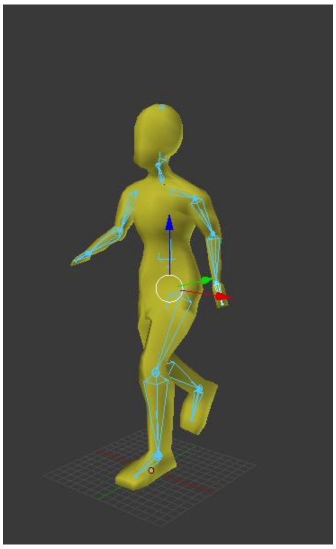


Figure 4-8a. Walk

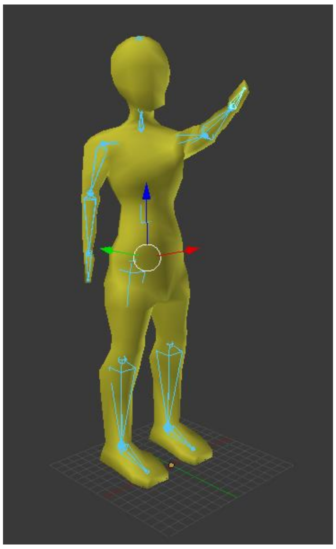


Figure 4-8b. Reach Out

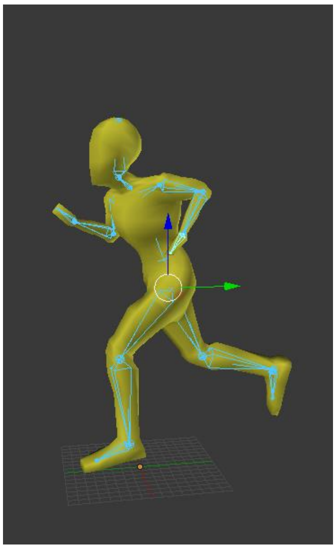


Figure 4-8c. Run

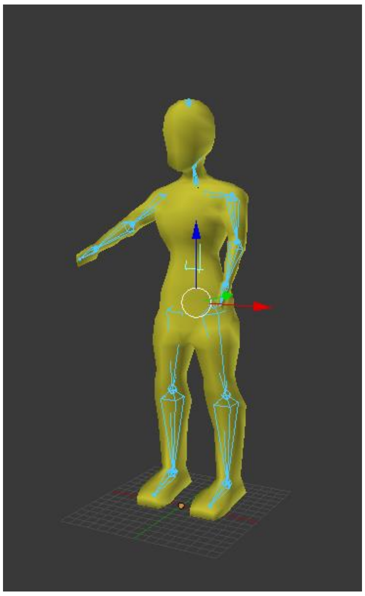


Figure 4-8d. Punch

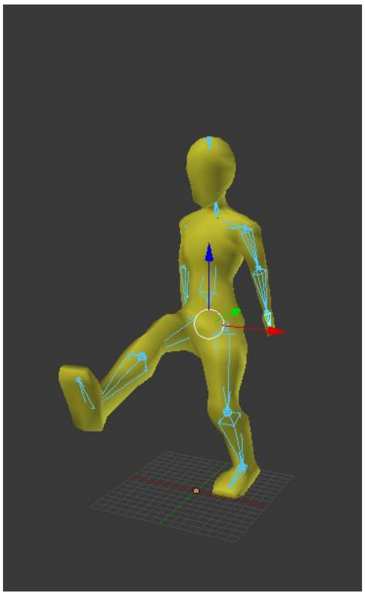


Figure 4-8e. Kick

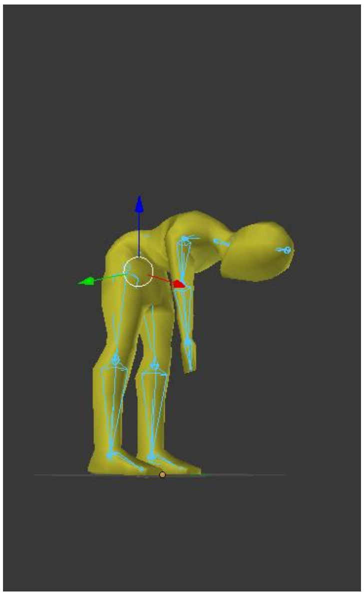


Figure 4-8f. Pick Up

A sample file generated by Blender is shown on Listing 4-4. Figure 4-9 shows the actual animation produced using jMonkeyEngine on the base of it.

```
[…]
<animations >
    <animation name="Walk" length="1.3333333333333333" >
      <tracks >
        <track bone="Hips" >
          <keyframes >
            <keyframe time="0.0" >
              <translate z="0.000000" x="0.000000" y="0.000000" />
              <rotate angle="0.224691" >
                <axis z="-0.087907" x="-0.770964" y="-0.630782" />
              </rotate>
              <scale z="1.000000" x="1.000000" y="1.000000" />
            </keyframe>
            <keyframe time="0.041666666666666664" >
              <translate z="0.000000" x="0.000000" y="0.000000" />
              <rotate angle="0.224691" >
                <axis z="-0.087907" x="-0.770964" y="-0.630782" />
              </rotate>
              <scale z="1.000000" x="1.000000" y="1.000000" />
            </keyframe>
            <keyframe time="0.08333333333333333" >
              <translate z="0.000000" x="0.000000" y="0.000000" />
              <rotate angle="0.224691" >
                <axis z="-0.087907" x="-0.770964" y="-0.630782" />

[…]
```

Listing 4-4. XML file of the model's skeletal animations exported from Blender

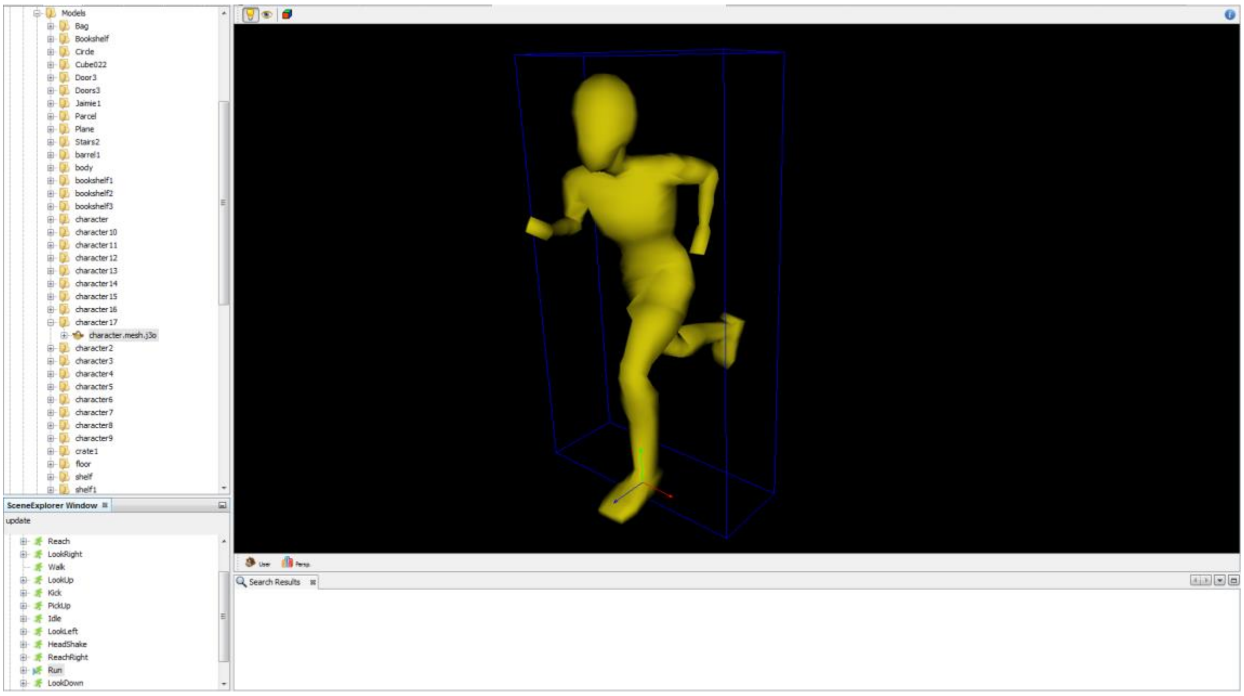


Figure 4-9. Animated movements in **jMonkeyEngine** based on the information exported from **Blender**

4.3.7. Setting up the scene

The initial construction of the visual scene in **jMonkeyEngine** can be made by using boxes as they can be used as containers holding various objects of interest (see Figure 4-10).

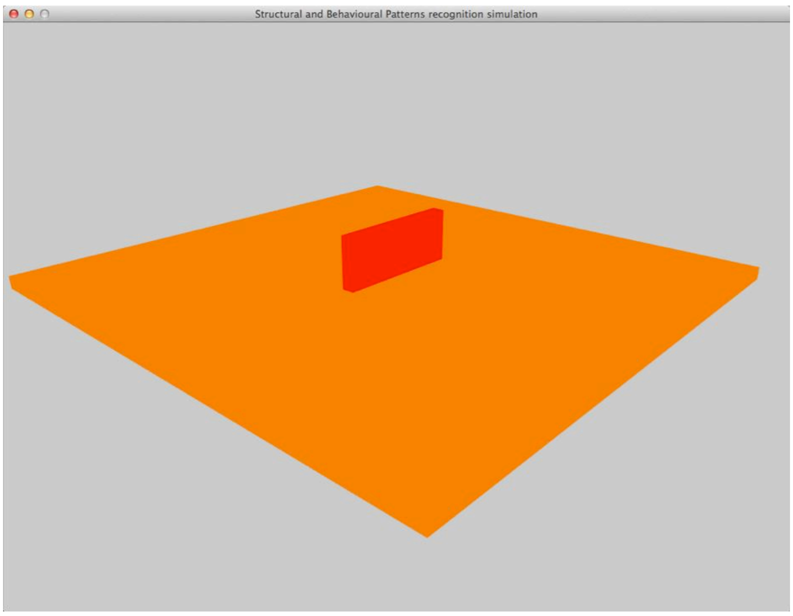


Figure 4-10. The visual scene formed out of basic geometric shapes. The orange flat cube represents the floor while the red one a wall

The floor can be made out of an appropriately scaled single box, which delimitation the entire visual scene. The walls can be represented with vertically scaled boxes positioned at the edges of the floor. Other objects of interest, such as shelves, can be located within the premises of the simulated scene.

The scale of the visual scene is measured in world units and it is the responsibility of the modeller to specify the exact proportions. The monitored area is always a restricted space that comprises a section of a building or ground area such as room, corridor, hall, pathway, etc. In such a case the most practical measure that can be adopted for a world unit is centimetre, i.e. one world unit in visual scene is equivalent to one centimetre in real world. In Figure 4-10, the floor is forty centimetres wide, forty centimetres long and one centimetre tick, which means the overall size is 40x1x40 in world units (WU).

There are several different types of *materials* that define physical properties of objects. The two basic types are *unshaded* and *shaded* materials as shown in Figure 4-11. The former does not require any *light sources* to be visible, which means that it appears flat no matter from which angle they are being looked at. The latter, however, requires additional illumination to be visible from different angles. The shading and coloration of such objects may vary depending on the viewing angle. The simple geometries used for the implementation of the visual scene shown in Figure 4-10 have been made of unshaded material.

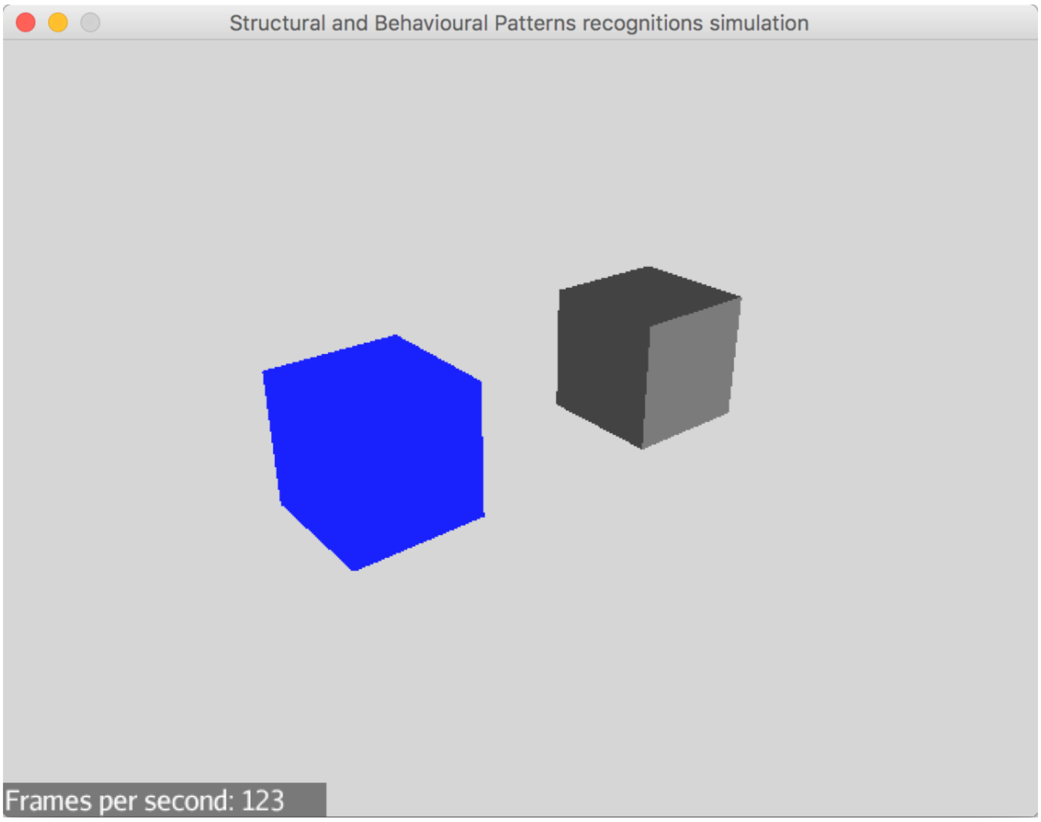


Figure 4-11. Two cubes made of different materials. The blue cube is made of *unshaded* material while the other is made of *shaded* one

Although different sources of light may influence the process of extracting features from live video feed, they do not have an impact on the simulation of dynamic movements in three-dimensional space. In the analysis of the behavioural patterns this research is mostly concerned about the dimensions, locations, orientations and approximated physical appearance of objects identified in monitored area.

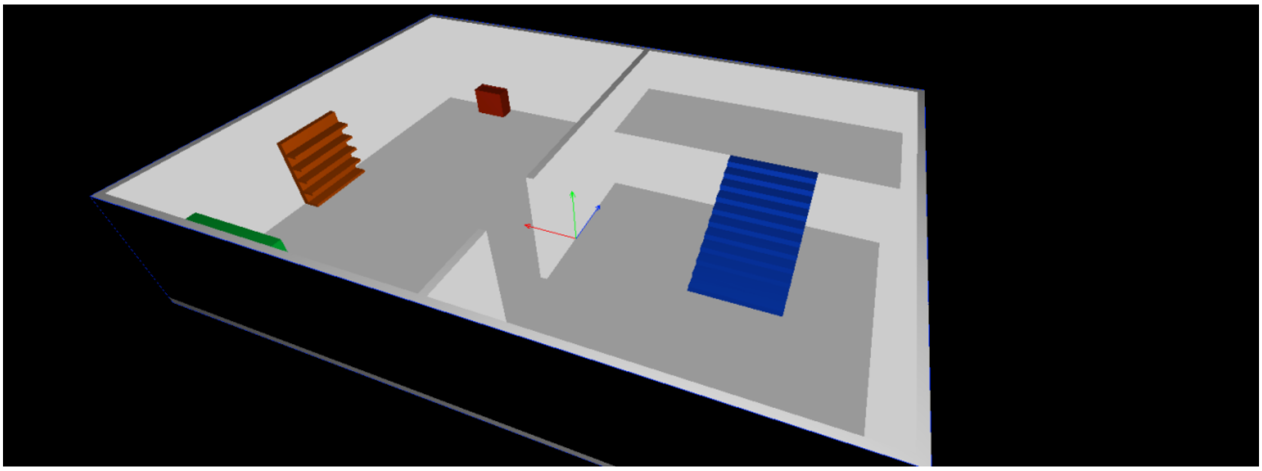


Figure 4-12. The final version of a visual scene recreated in the graphics engine from the exported XML file generated by **Blender**

Since the simulator depends on the input data streamlined from other modules, the materials can be interchangeably used solely for the purpose of aesthetic enhancement of the graphics. Therefore, the in-depth study of mechanisms behind lighting, shading and materials in 3D graphics has been left out of the scope of the research. When development process reached a more matured stage, high-level methods have been used to increase the level of graphical details, but they do not affect the execution of simulation of movements. As in the case of an agent mode, the final three-dimensional scene has been modelled with **Blender** as shown in Figure 4-12.

The exported XML file contains the information of all vertices comprising the overall geometry of a closed space. To preserve dimensions of the scene between different mediums and maintain the consistency, the unit measurements were set to the default value of one meter.

4.4. Implementation of the simulator

4.4.1. Utilization of engine core libraries for critical functionality of the simulator

The implementation of physical properties of the visual scene, linear movements and collision detection mechanisms relies on libraries provided with **jMonkeyEngine** (jMonkeyEngine, 2014). Table 4-2 summarizes the libraries used in the implementation of the simulator.

Table 4-2. jMonkeyEngine library functions used for implementation of the simulator (jMonkeyEngine, 2014)

Function in simulation	What is it used for?	Formula	Class / Library
Vector; A mathematical construct that is of significant importance in 3D engines and modelling tools for calculations related to three-dimensional space.	Calculating objects positions in the boundaries of visual scene, simulating their dynamic movement, defining spatial properties and operations related to collision detection mechanics.	$\vec{v} = [v_x, v_y, v_z]$ (Lengyel, E., 2003) See Appendix for more details	**Class Vector3f** `com.jme3.math.Vector3f`
Matrix; A mathematical construct used for storing and executing transformation operations on objects in 3D space.	A compact mathematical construct used for encapsulation of three-dimensional transformations of spatial properties of objects visible in visual scene. Matrices are a convenient way of storing a mathematical description of transformation that one has to apply to an object in visual scene in order, for instance, to simulate movement such as rotation.	$M = [a_{ij}] = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix}$ (Lengyel, E., 2003) See Appendix for more details	**Class Matrix3f** `com.jme3.math.Matrix3f`
Quaternion; A mathematical construct that is used by 3D engines to define orientations of objects.	Calculating objects rotations in more efficient manner than matrices since quaternions require less storage space, fewer arithmetic operations and less memory for computations.	$q = w + xi + yj + zk$ (Hamilton, 1853) $q = \cos\left(\frac{\alpha}{2}\right) + i\left(x * \sin\left(\frac{\alpha}{2}\right)\right) + j\left(y * \sin\left(\frac{\alpha}{2}\right)\right) + k\left(z * \sin\left(\frac{\alpha}{2}\right)\right)$ (Baker) See Appendix for more details	**Class Quaternion** `com.jme3.math.Quaternion`

Function in simulation	What is it used for?	Formula	Class / Library
Sphere Geometry; Approximation of object spatial volume in visual scene; refer to section 5.3.3. for justification of selecting sphere geometry for approximation.	Approximating the logical and physical shape of objects and their parts in the visual scene. Example: sphere surrounding agent hand, sphere surrounding a package, etc. Calculating distances between occupancies of two or more independent objects.	$(p_x - c_x)^2 + (p_y - c_y)^2 + (p_z - c_z)^2 = r^2$ (Lengyel, E., 2003) See Appendix for more details	**Class SphereCollisionShape** `com.jme3.bullet.collision.shapes.SphereCollisionShape`
Agent linear movement; Simulation of each agent movement within the boundaries of the visual scene while accounting the laws of physics. In the current version a constant velocity is taken into account when simulating movements to reduce the overall complexity of the implementation and dynamic behavioural patterns.	Calculating agent location at next frame in respect of time and on a basis of viewing direction, walking direction and velocity while accounting the physical properties of solid objects. The library equips agent geometry with physical properties and performs collision detection checks to prevent it from going through solid objects in visual scene. A primitive geometry in a form of pink sphere (see Figure 4-17) has been selected to represent an approximated shape of an agent to minimize the number of collision detection calculations and tests.	$x(t) = x_0 + v_0 t$ $v(t) = \dot{x}(t) = \frac{d}{dt}x(t)$ $v(t) = v_0 + a_0 t$ $a(t) = \dot{v}(t) = \ddot{x}(t) = \frac{d^2}{dt^2}x(t)$ $d = \int_{t_1}^{t_2} v(t)dt$ $d = \int_0^t (v_0 + a_0 t)dt = v_0 t + \frac{1}{2}a_0 t^2$ $x(t) = x_0 + v_0 t + \frac{1}{2}a_0 t^2$ (Lengyel, E., 2003)	**Class BetterCharacterControl** `com.jme3.bullet.control.BetterCharacterControl`
Ray Casting; Determination of object location in relation to agent position; collision detection and generation of events used for recognition of dynamic patterns	Calculating a location of a point in which ray intersects a geometry representing an object in visual scene. Ray tracing technique is used in simulator to estimate distance between two different objects and determine whether they are within each other line-of-sight, i.e. if no other physical objects lie between them. Intersections result in the generation of logical events that are essential for accumulating necessary data for dynamic pattern recognition. Example: rays casted towards left or right side of an agent to determine if it walks along object.	$R(x) = \vec{u} + x\vec{v}$ (Lengyel, E., 2003)	**Class Ray** `com.jme3.math.Ray`

Function in simulation	What is it used for?	Formula	Class / Library
Ghosting; Determination of proximity between approximated volumes of two independent objects in the visual scene; Collision detection and generation of events data used for recognition of dynamic patterns	Calculating objects approximated volumes and positions in the boundaries of visual scene. Ghost controls are also used for defining spatial properties and operations related to collision detection mechanics.	$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_p + r_q)^2]}}{B^2}$ (Lengyel, E., 2003) See Appendix for more details.	**Class GhostControl** `com.jme3.bullet.control.GhostControl`

4.4.2. Simulating movements

Movements play an essential role in reconstructing the object trajectories and as such are the cornerstone of the simulation-based behaviour analysis. There are two different methods that have been adopted for simulating movements in the 3D space of the visual scene. The first uses directly analytical data received from other modules of the framework in XML format. The agent movement from one location to another is rendered using current location and the orientation of individual in visual scene. The second method relies on skeleton animations. The bone movement play two different roles in this case. Primarily they provide information about the location of corresponding body parts and their rotation at particular time of the movement. This information is critical for positioning the bounding boxes that are used for establishing relationships between body parts and other objects in the scene. Bounding boxes are described in more details later in this chapter. The secondary function is to visualise the movements.

The implementation of linear movements relies on libraries provided with **jMonkeyEngine** (see Table 4-2). The position is determined on the base of current location, velocity and forward direction vector that may rotate according to the changes of the viewing direction vector as shown in Figure 4-13.

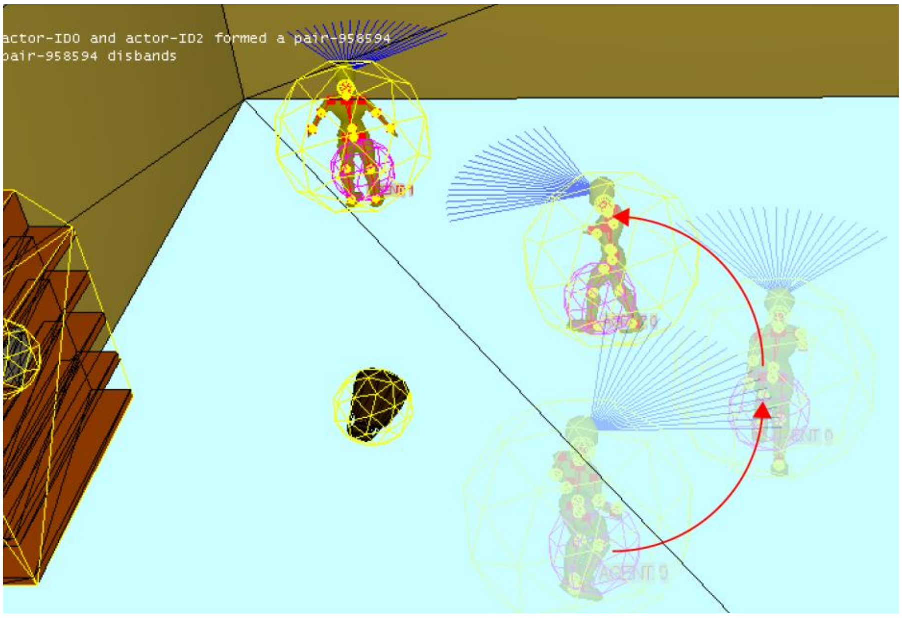


Figure 4-13. An agent moving along curved path

The key components that dictate how agents change their location in three-dimensional space are:

- ***The walking direction.*** This is a normalized vector calculated in other modules of the framework that is taken as input data. It describes in which direction the agent is about to move since the previous move. The magnitude of this vector defines the speed at which an agent moves, but since only constant velocity is considered at the time of this writing, a default constant value has been used in the simulator.

- ***The viewing direction.*** This normalized vector is also calculated externally. It describes the current orientation of the individual, which is important in many dynamic patterns and provides support for calculating agent's rotation in three-dimensional scene.

Depending on the purpose of the simulation, the trajectory information does not have to come from a live camera feed. With the use of built-in physics in the engine one can also calculate the movements of an agent using data coming straight from the keyboard. This would make the simulator an application that is self-contained and fully independent on input data. For the purpose of this research, this approach has been adopted. It proved to be fully adequate in testing and experimentation for the purpose of the subsequent pattern analysis. When the simulator is part of a framework an additional adapter for feeding in the input data into the simulator in real time will be necessary.

4.4.3. Collision detection

The analysis of dynamic behaviour is based on events occurring in the three-dimensional space during the simulation. Most of the internal events can be interpreted as collisions between various objects on the scene. The two commonly used in computer games collision detection techniques that allow registering an instance when two different geometries share a mutual surface point have been adopted in the implementation. *Ray casting* is a technique based on the concept of emitting a ray from a given point in space towards a specific direction to determine if any object lies in its path (Reese and Johnson, 2015). It is particularly useful in situations when it is required to determine if an object is within a certain distance from an agent. The other technique, *ghosting* is based on the idea of surrounding more complex geometries with simpler ones in order to reduce the number of operations required for detecting volumetric collisions with an object. The responsibility of ghosts is only to report details of the event and not to influence physical attributes of the objects.

4.4.3.1. Implementation of agent's "sight sense" for spatial reconnaissance

The *sight sense* of an agent utilises ray casting technique for approximation of the vision field and probing neighbouring areas for objects. Using the method presented by Edén (2014), rays are being rendered from agent's head geometry to emulate the central and peripheral vision of human sight sense (Figure 4-14).

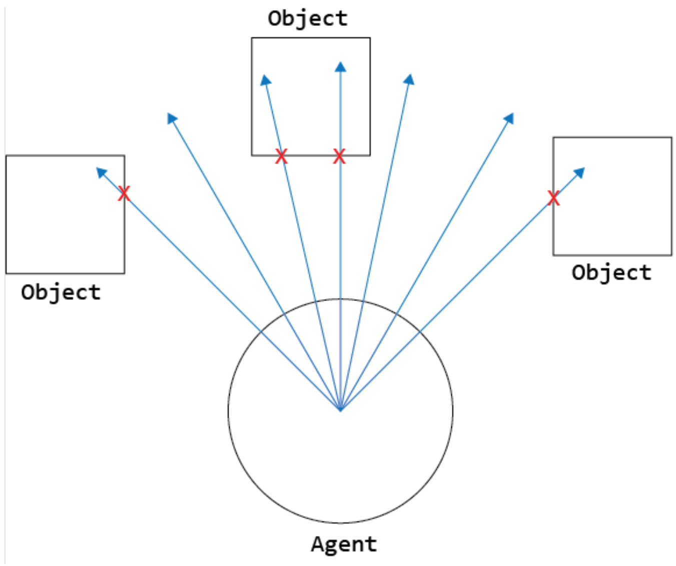


Figure 4-14. Schematic view of the sense of sight of an agent; the rays are depicted with blue arrows

To accommodate situations in which an agent might change body position (e.g., while bending), the arc, which emits the rays, is repositioned and orientated relative to the spine and the head with the use of a novel formula presented in Formula 4-6. The method utilises the equation for rotating a vector around an arbitrary axis defined by unit vector as presented in Formula 4-18. Such implementation avoids counterintuitive behaviour of the detection mechanism e.g. head oriented at the floor does not prevent am agent from detecting objects located in front of it. Each 'sight' ray origin point and direction are calculated in several steps assuming that the following data is available at the time of calculation:

a) $\vec{u}_h$ - location of agent's *head* in relation to the origin of global coordinate space specified in vector notation.
b) $\vec{u}_p$ - location of agent's *central node* (local origin point) in relation to the origin of global coordinate space specified in vector notation.
c) $\vec{v}_{dir}$ – a normalized *viewing direction vector* an agent.
d) θ – *head orientation* angle relative to Y axis.
e) $\vec{p}_s$– agent's *spine* defined as a difference between head and central node, i.e. $\vec{u}_h - \vec{u}_p$ (assuming the vector is normalized).

Knowing the viewing direction vector of an individual $\vec{v}_{dir}$, the spine vector $\vec{p}_s$, the head orientation angle θ and the angular distance between the rays expressed by angle δ, one can calculate the orientation matrices R_1, R_2, R_3, R_4 where each matrix represents different stage of overall rotations that need to be made in order to position the viewing direction relative to the orientation of agent's head and body.

$$R_1 = \begin{bmatrix} (1-\cos 90°){u_x}^2 + \cos 90° & (1-\cos 90°)u_x u_y - \sin 90°\, u_z & (1-\cos 90°)u_x u_z + \sin 90°\, u_y \\ (1-\cos 90°)u_x u_y + \sin 90°\, u_z & (1-\cos 90°){u_y}^2 + \cos 90° & (1-\cos 90°)u_y u_z - \sin 90°\, u_x \\ (1-\cos 90°)u_x u_z - \sin 90°\, u_y & (1-\cos 90°)u_y u_z + \sin 90°\, u_x & (1-\cos 90°){u_z}^2 + \cos 90° \end{bmatrix}$$

$$R_2 =$$

$$\begin{bmatrix} (1-\cos 90°){(R_1 v_{dir})_x}^2 + \cos 90° & (1-\cos 90°)(R_1 v_{dir})_x (R_1 v_{dir})_y - \sin 90°\,(R_1 v_{dir})_z & (1-\cos 90°)(R_1 v_{dir})_x (R_1 v_{dir})_z + \sin 90°\,(R_1 v_{dir})_y \\ (1-\cos 90°)(R_1 v_{dir})_x (R_1 v_{dir})_y + \sin 90°\,(R_1 v_{dir})_z & (1-\cos 90°){(R_1 v_{dir})_y}^2 + \cos 90° & (1-\cos 90°)(R_1 v_{dir})_y (R_1 v_{dir})_z - \sin 90°\,(R_1 v_{dir})_x \\ (1-\cos 90°)(R_1 v_{dir})_x (R_1 v_{dir})_z - \sin 90°\,(R_1 v_{dir})_y & (1-\cos 90°)(R_1 v_{dir})_y (R_1 v_{dir})_z + \sin 90°\,(R_1 v_{dir})_x & (1-\cos 90°){(R_1 v_{dir})_z}^2 + \cos 90° \end{bmatrix}$$

$$R_3 = \begin{bmatrix} (1-\cos\theta){p_{s_x}}^2 + \cos\theta & (1-\cos\theta)p_{s_x} p_{s_y} - \sin\theta\, p_{s_z} & (1-\cos\theta)p_{s_x} p_{s_z} + \sin\theta\, p_{s_y} \\ (1-\cos\theta)p_{s_x} p_{s_y} + \sin\theta\, p_{s_z} & (1-\cos\theta){p_{s_y}}^2 + \cos\theta & (1-\cos\theta)p_{s_y} p_{s_z} - \sin\theta\, p_{s_x} \\ (1-\cos\theta)p_{s_x} p_{s_z} - \sin\theta\, p_{s_y} & (1-\cos\theta)p_{s_y} p_{s_z} + \sin\theta\, p_{s_x} & (1-\cos\theta){p_{s_z}}^2 + \cos\theta \end{bmatrix}$$

$$R_4 = \begin{bmatrix} (1-\cos\delta){p_{s_x}}^2 + \cos\delta & (1-\cos\delta)p_{s_x} p_{s_y} - \sin\delta\, p_{s_z} & (1-\cos\delta)p_{s_x} p_{s_z} + \sin\delta\, p_{s_y} \\ (1-\cos\delta)p_{s_x} p_{s_y} + \sin\delta\, p_{s_z} & (1-\cos\delta){p_{s_y}}^2 + \cos\delta & (1-\cos\delta)p_{s_y} p_{s_z} - \sin\delta\, p_{s_x} \\ (1-\cos\delta)p_{s_x} p_{s_z} - \sin\delta\, p_{s_y} & (1-\cos\delta)p_{s_y} p_{s_z} + \sin\delta\, p_{s_x} & (1-\cos\delta){p_{s_z}}^2 + \cos\delta \end{bmatrix}$$

where $\vec{u} = [0,1,0]$

Formula 4-6. Calculating orientation matrices. The formula takes into consideration the *current orientation of head and spine* when the ray is a part of agent's estimated eyesight.

Calculating the final ray direction vector is then

$$\vec{r}_{final} = R_4(R_3(R_2\vec{p}_s))$$

Substituting this vector into the general ray tracing equation yields

$$R(x) = \vec{u}_h + x\vec{r}_{final}$$

As reported by Gasiorowski, Vassilev and Ouazzane (2016) in the paper where general framework was first presented, the above formulas allows approximation of the perceptual system of humans taking into account the mechanisms of the peripheral vision, body and head positioning. Although useful for estimating the focus of agent's attention, it may be difficult to detect objects solely on that basis. For this reason, a set of additional rays is being generated that are orthogonal to agent's viewing direction vector (Figure 4-15).

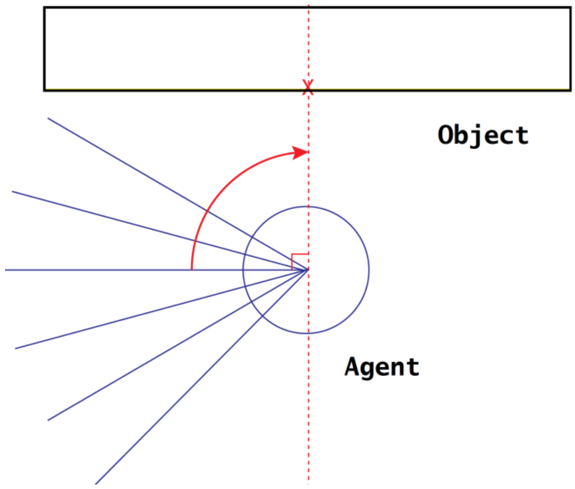


Figure 4-15. Schematic view of ray casting used for detecting objects located in close proximity. The blue rays indicate the sight sense while the red dashed rays depict the ray casting boundary

On the basis of the general Formula 4-6 one can calculate the additional side rays as follows

$$\vec{r}_{side} = R_{side}\vec{v}_{dir}$$

where

$$R_{side} = \begin{bmatrix} (1-\cos\pm 90°){u_x}^2 + \cos\pm 90° & (1-\cos\pm 90°)u_x u_y - \sin\pm 90°u_z & (1-\cos\pm 90°)u_x u_z + \sin\pm 90°u_y \\ (1-\cos\pm 90°)u_x u_y + \sin\pm 90°u_z & (1-\cos\pm 90°){u_y}^2 + \cos\pm 90° & (1-\cos\pm 90°)u_y u_z - \sin\pm 90°u_x \\ (1-\cos\pm 90°)u_x u_z - \sin\pm 90°u_y & (1-\cos\pm 90°)u_y u_z + \sin\pm 90°u_x & (1-\cos\pm 90°){u_z}^2 + \cos\pm 90° \end{bmatrix}$$

and $u = [0,1,0]$

For vertical rays, i.e. rays casted up and down the head, one needs to know also the agent's location:

$$R(x) = \vec{u}_p + x\vec{u}$$

where $\vec{u}_p$ is the central node point of individual (physical location of individual in world space) and $\vec{u}$ is either $[0,1,0]$ or $[0,-1,0]$ depending on the direction one wishes to cast ray towards.

4.4.3.2. Implementation of agent's "ghost" for object detection

The simulator utilises the "ghost" technique to detect potential collisions in situations when either the data from ray casting is insufficient or there is a need to account not just the surface but also the volume of an object. For example, considering the anatomical structure of human body it would be difficult to detect an object that came into contact with a limb using ray casting since in such a case it is impossible to determine the exact direction of the rays. The emission of multiple rays around each limb also would not provide efficient solution, because the number of potential collisions would grow exponentially and this could quickly reduce the computational capacity of the simulator.

In the real world, the individuals *interact* with surrounding objects by using their body parts, such as hands or legs, represented by the bones in our model. The "ghosts" will simply *encapsulate* all dynamic and static objects as shown in Figure 4-16. The centre of the sphere is set to coincide with the central point of the object and the radius is set to cover completely its shape. Such non-physical spherical ghost geometries could potentially lead to collisions, which can be detected by calculating the distances between the interacting ghosts.

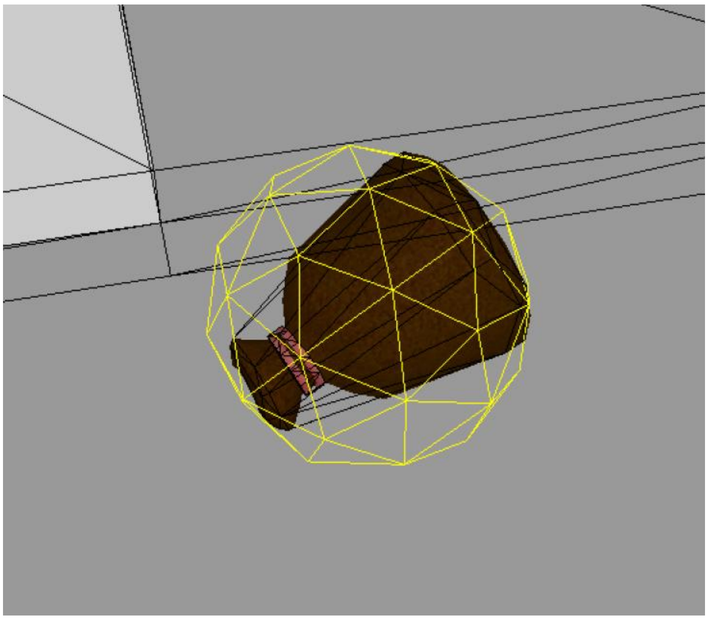


Figure 4-16. Positioning of the centre of ghost sphere

The simulator utilises this technique to report potential collisions between the body parts of individual and previously identified objects in 3D scene. In order to follow the movement of individual limbs one first needs to extract the location of the bones from the input stream. To set the ghost spheres at the correct locations one also needs to translate their coordinates from the body model to the world space. For this purpose a transformation matrix, which preserves the scale and orientation was constructed. So the transformation matrix is a product of two matrices – the first one describes current spatial definition of a bone in model space, and the second calculates the translation which positions the model space in world space coordinates according to Formula 4-10 (see Figure 4.7).

$$T = \begin{bmatrix} 1 & 0 & 0 & \mathrm{u.x} \\ 0 & 1 & 0 & \mathrm{u.y} \\ 0 & 0 & 1 & \mathrm{u.z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} \mathrm{v.x} & 0 & 0 & 0 \\ 0 & \mathrm{v.y} & 0 & 0 \\ 0 & 0 & \mathrm{v.z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \vec{p} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\vec{o} = T(R(S\vec{p}))$$

Where,

T : translation matrix

$\vec{u}$: vector defining an offset vector of a bone in relation to origin point in the world space coordinate system

R : rotation matrix that is a product of three matrices defining current orientation of a bone in relation to X-, Y- and Z-axis

γ : current rotation angle of a bone around X-axis

β : current rotation angle of a bone around Y-axis

α : current rotation angle of a bone around Z-axis

S : scale matrix

$\vec{v}$: vector, whose components are defining the current scale of a bone along each axis in a model space

$\vec{p}$: a zero vector written in *homogenous coordinates* notation

$\vec{o}$: a final position of a bone in world coordinates and a centre position point of a ghost sphere surrounding it

Formula 4-7. Translating a bone from *model space* to *world space*

The above transformation has been implement in the simulator using the built-in functions of **jMonkeyEngine** (jMonkeyEngine, 2014). The final result of this ghost-equipped model is shown on Figure 4-17. The red cubes represent the bones and the red line segments depict the armature connecting them. The rays emitted from the head are shown in blue and the ghost spheres surrounding the body and the limbs are highlighted with yellow wireframes. As opposed to ghost spheres that are used to report only logical collisions, the pink sphere is used by the engine to detect physical collisions between solid boundaries of the visual scene such as floors and an agent in order to prevent from going through them.

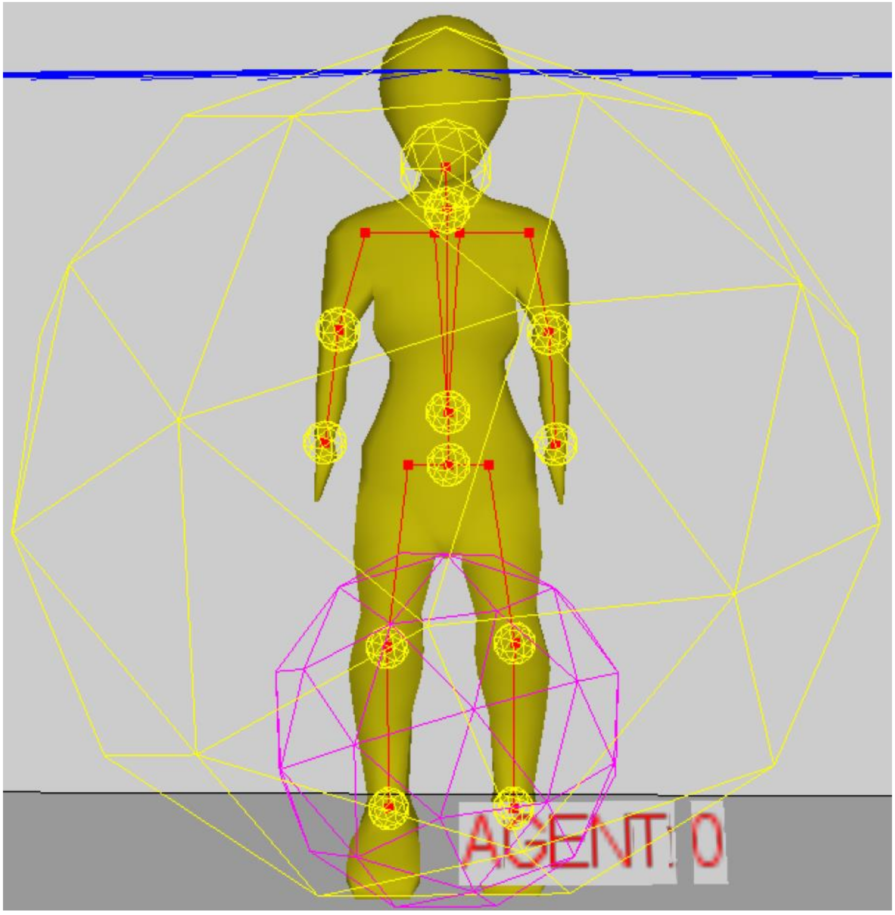


Figure 4-17. Final body model equipped with bones armature, ray casting vision and spherical ghosts attached to the limbs

4.4.3.3. Pairing and grouping of individuals

Critical feature of the approach adopted in this research is that the groups are treated as separate entities, which can exhibit their own behaviour. The group behavioural patterns always involve at the very least two different agents. In our ontology we made a further distinction between pairs and groups due to the need to distinguish antisymmetric relations in pairs from asymmetric relations within groups. As a result, the groups can be formed by individuals joining pairs or by merging two groups. Two agents participating in a pair may be labelled as *first* and *second* for convenience, although these labels do not necessarily reflect the positions of the same agents within groups. Similarly, two groups which merge into a group can be labelled as *first* and *second pair* although this in no way determines the position of their members in the groups.

The characteristics of the group should depend on the characteristics of the participating individuals. The simulator is able to derive the location of the groups automatically. By interpolation of the values of the location points of all individual members one can determine the *midpoint* of the group (see Formula 4-8). The basic calculations needed are already implemented by the built-in functions of **jMonkeyEngine** (jMonkeyEngine, 2014). The code in Listing 4-4 presents only the Java method, which utilises Formula 4-8 in the calculations. The agents belonging to a group are being registered in a data structure (array) in the order they have joined the group.

$$G_M = \begin{bmatrix} \sum_{i=2}^{n} \big((1 - s) * u_i.x\big) + (s * u_{i-1}.x) \\ \sum_{i=2}^{n} \big((1 - s) * u_i.y\big) + (s * u_{i-1}.y) \\ \sum_{i=2}^{n} \big((1 - s) * u_i.z\big) + (s * u_{i-1}.z) \end{bmatrix}$$

Where,

G_M: the midpoint of a Group. The location is represented in vector notation.

n : number of Group members

$\vec{u}_i$: the location of a Group member

s: the interpolation factor which value is $0 \leq s \leq 1$. The factor specifies the linear interpolation distance between the location of a group member $\vec{u}_i$ and a group member $\vec{u}_{i-1}$. The group members are ordered in a data structure by the time they joined a group.

Formula 4-8. Calculating the midpoint of a Group.

```
    /**
     * Calculating the mid point of all agents in the group.
     * @param aAgents
     * @return
     */
    private Vector3f getGroupMidPointPosition(ArrayList<Spatial> aAgents) {
        Vector3f groupMidPointPosition = Vector3f.ZERO;
        for(int i=0;i<aAgents.size();i++) {
              if(i > 0) {
                   Vector3f currPosition = aAgents.get(i).
                                              getWorldTranslation().clone();
                   if( groupMidPointPosition.length() != 0) {
                        groupMidPointPosition = FastMath.interpolateLinear(.5f,
                                                         groupMidPointPosition,
                                                                  currPosition);
                   } else {
                        Vector3f prevPosition = aAgents.get(i-1).
                                              getWorldTranslation().clone();
                        groupMidPointPosition = FastMath.interpolateLinear(.5f,
                                                prevPosition, currPosition);
                   }
              }
```

```
        }
        return groupMidPointPosition;
    }
```

Listing 4-4. Implementation of the method for calculating midpoint location of a group in Java

Using the same idea one can also calculate the average viewing direction of the group on the basis of known viewing direction vectors of its individual members. Some of the agents may “look” or even “walk” in completely opposite directions than others, but they still remain members of the group due to their close proximity (see Figure 4-18).

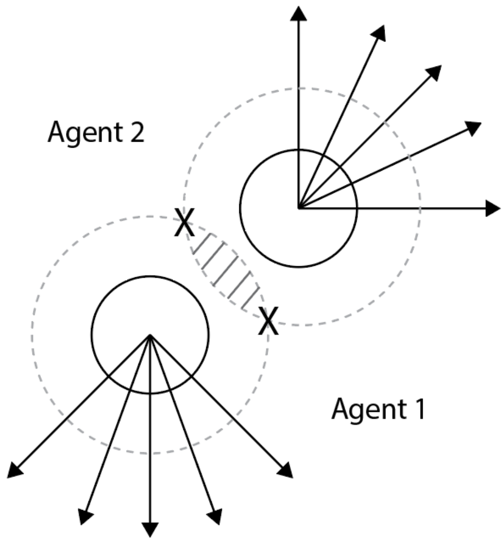


Figure 4-18. Schematic view of two agents in the same group facing opposite directions

To account this possibility, calculations of *average viewing* and *walking directions* of groups are based on *majority*, i.e. most members, whose viewing directions are facing similar direction (Listing 4-5).

```
    /**
     * Calculates the average view direction of all agents in the group.
     * @param aAgents
     * @return
     */
    private Vector3f getGroupViewDirection(ArrayList<Spatial> aAgents) {
        Vector3f groupViewDirection = Vector3f.ZERO;
        for(int i=0;i<aAgents.size();i++) {
              if(i > 0) {
                   Vector3f currViewDirection = aAgents.get(i).
                                  getControl(CharacterControl.class).
                                             getViewDirection().clone();
                   if(groupViewDirection.length() != 0) {
                      groupViewDirection = FastMath.interpolateLinear(.5f,
                                                      groupViewDirection,
                                                       currViewDirection);
                   } else {
```

```
                    Vector3f prevViewDirection = aAgents.get(i-1).

                                   getControl(CharacterControl.class).
                                            getViewDirection().clone();
                    groupViewDirection = FastMath.
                                   interpolateLinear(.5f,
                                                    prevViewDirection,
                                                    currViewDirection);
               }
           }
       }

       // Adjusting the average view direction within the group
       ArrayList<String> ids = new ArrayList<String>();
       Vector3f finalGroupViewDirection = Vector3f.ZERO;
       for(int j=0;j<aAgents.size();j++) {
           Vector3f viewDirection = aAgents.get(j).
                                    getControl(CharacterControl.class).
                                            getViewDirection().clone();
           int isOpposite = SimMath.
                        getVectorDirectionProximity(groupViewDirection,
                                                        viewDirection);
           if(isOpposite == 1) {
              finalGroupViewDirection = FastMath.
                                    interpolateLinear(.5f,
                                               finalGroupViewDirection,
                                                        viewDirection);
             ids.add(aAgents.get(j).
                             getControl(CharacterControl.class).getID());
           }
       }

       return finalGroupViewDirection;
   }
```

Listing 4-5. Implementation of the method for calculating the average viewing and walking directions of a group in Java

The central location of a group in combination with the average viewing direction provides sufficient means for capturing group events in a similar manner as those of individual events. The advantage of this solution is that one can capture the behavioural patterns of the groups and the individual agents in parallel (Figure 4-19).

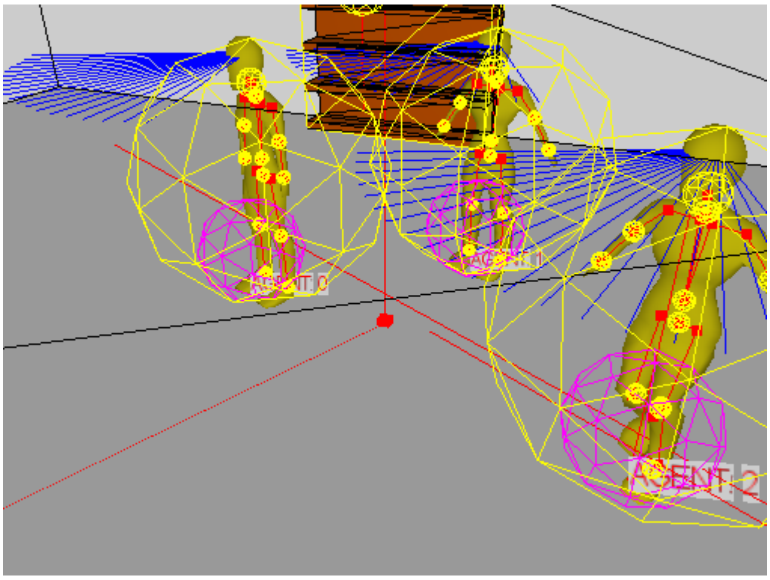


Figure 4-19. A group of three agents

The *midpoint of a group* is represented by red cube stretched to cover all members of the group, while the *average viewing direction* is depicted with a red line segment emitted from it.

4.5. The 3D Simulation as a Basis for Event-driven Dynamic Behaviour Analysis

The fundamental principle behind the simulation-based event-driven dynamic behaviour pattern analysis is to simulate the physical conditions, under which the events are registered by the simulator for further analysis. The ray casting and ghosting techniques have been adopted to support the detection of potential collisions between the geometries of objects located within the space of visual scene, which can generate events by simply accounting the laws of physics. In addition, a range of techniques for generating individual and group events purely have been logically developed by accounting the proximity between the geometries.

Currently, the *individual dynamic behavioural patterns* are recognised using the following information:

a) The set of *properties describing the agent* a participating in a pattern at time t when event e occurs.
b) The set of *properties describing an object* o participating in a pattern at time t when event e occurs.
c) The *location of the agent* $\vec{u}_t$ defined in global coordinate space at the time t when event e occurs.
d) The *location of the object* $\vec{v}_t$ defined in global coordinate space at the time t when event e occurs.
e) The unit vector defining a *walking direction* from agent towards an object $\overrightarrow{dir}_t$ at the time t when event occurs.
f) The unit vector defining a *viewing direction of an agent* $\overrightarrow{vdir}_t$ at time t.

The application incorporates two different methods for deriving behavioural patterns during the simulation which are executed simultaneously at each step of the simulation loop in order to detect potential occurrence:

- ***By analysing the entries in the event log for discovering patterns.*** The approach is inspired by the architecture developed by Sung, Gleicher and Chenney (2004) in which the behaviour of an agent in simulation is analysed and then selected on a basis of *sampling* of the output of various tracing functions in a batch. Once the number of entries in the buffer reaches certain pre-defined size, the events in the batch are analysed for recognising the potential pattern. In some cases only a few event entries in the buffer suffice to recognise a pattern. More information on this method is presented in section 2.3.3.
- ***By measuring the quantitative increments for estimating the critical distance to qualitative changes.*** This method is based on using various quantitative estimations of qualitative changes, e.g. calculating the angle between the viewing direction and the front of a Static Object, calculating the distance between the ghosts of an object and a group of objects, etc. The calculated values in this case are checked against pre-defined *thresholds* to determine the behaviour pattern. The approach is inspired by the methodology presented by Maik et al. (2010) in which human poses are determined by comparing measurements against minimum and maximum angular variations of the joints of the human body. More information on this method is presented in section 2.2.

Both methods of analysis are parameterised, i.e. the criteria for *recognising a pattern* as well as the criteria for *capturing an event* can be adjusted. The parameters can be configured prior to the simulation, but they can be adjusted later on at runtime using a separate interface. This is possible because the algorithms for pattern recognition are always invoked only during a single step of the game loop, while the thresholds are updated at the beginning of the next cycle of the loop.

The process of analysis is presented in Figure 4-20. During the experimentation it was observed that without appropriate timing mechanisms the *event logging* methods are executed too quickly, which breaks the simulation into discrete fragments that are too small. To address this issue an additional synchronisation of the simulation loop has been enforced to smoothen the flow.

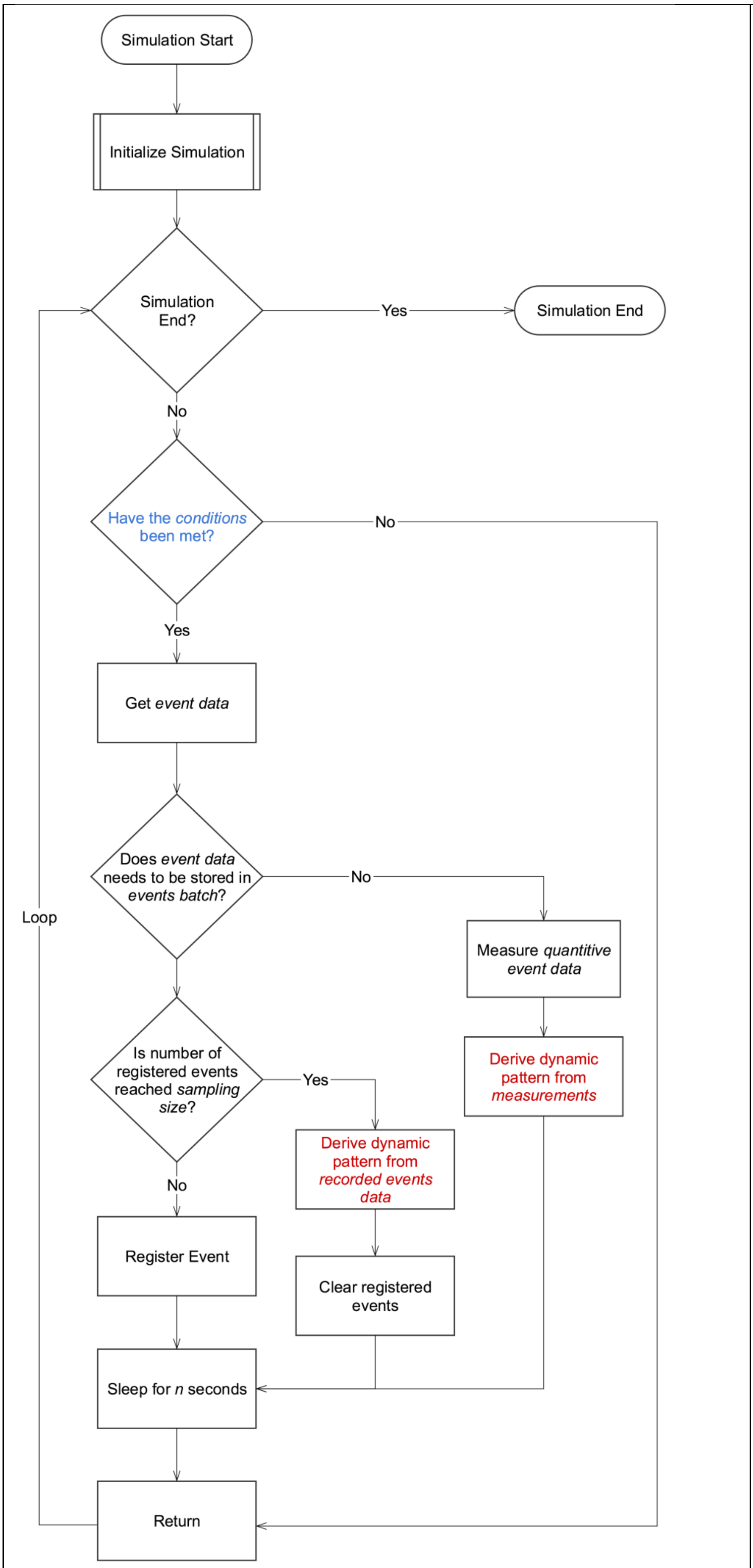


Figure 4-20. Top-level loop cycle of the simulator

4.6. Event Logger

The major role of the simulation is to generate an informative log of events occurring within the visual scene so that they can be analysed further by pattern matching techniques. The event logger has the architecture of "*observer*" design pattern, with attached individual loggers to each object within the visual scene. The individual observers log all events related to the observed objects. This allows further extension of the logging module without changing the existing code of the simulator.

The simulator log will be parsed by the analyser for recognising and classification of the behaviour patterns according to the grammar of the pattern language. The behaviour analysis can be continued solely based on the logs, while the original video data can be used to increase the precision of the simulation. This approach gives the opportunity to incorporate purely symbolic techniques for behaviour analysis.

In its current version, the simulator generates two types of log files with time-stamped entries describing all captured events. The first is a text-based log file in which each line hints a frame of the movie that a given pattern was recognized (Listing 4-6). The second one is an XML file, which contains more details about the event, such as spatial properties of participating entities (Listing 4-7).

```
...
414 :: pair-349746 DISBANDS
415 :: agent-ID1 MOVES TOWARDS static-Counter_ID4
466 :: agent-ID1 MOVES AWAY FROM static-Counter_ID4
490 :: agent-ID1 and agent-ID2 FORMS A pair-321282
699 :: agent-ID0 and pair-321282 FORMS A group-321282
745 :: participant-agent-ID0 MOVES TOWARDS static-Bookshelf_ID2
747 :: group-321282 MOVES TOWARDS static-Bookshelf_ID2
756 :: participant-agent-ID0 MOVES TOWARDS static-Bookshelf_ID2
765 :: group-321282 MOVES TOWARDS static-Bookshelf_ID2
768 :: participant-agent-ID0 MOVES TOWARDS static-Bookshelf_ID2
780 :: participant-agent-ID0 MOVES TOWARDS static-Bookshelf_ID2
...
```

Listing 4-6. An excerpt of the text-based log file generated by simulator.

```
<entry id="event-2" timeStamp="25/02/2017 10:51:17" type="Event">
        <Event-initiator>Agent-ID2</Event-initiator>
        <group id="976693" type="group">
          <actor id="Agent-ID2">
            <location>
              <vectorX>82.19113</vectorX>
              <vectorY>4.3999987</vectorY>
              <vectorZ>50.879482</vectorZ>
            </location>
            <viewDirection>
              <vectorX>-0.87773573</vectorX>
              <vectorY>0.0</vectorY>
              <vectorZ>0.48221964</vectorZ>
            </viewDirection>
```

```xml
        </actor>
        <actor id="participant-Agent-ID1 in pair with pair-976693">
          <location>
            <vectorX>121.19113</vectorX>
            <vectorY>4.3999987</vectorY>
            <vectorZ>21.879484</vectorZ>
          </location>
          <viewDirection>
            <vectorX>-0.87773573</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.48221964</vectorZ>
          </viewDirection>
        </actor>
      </group>
      <Result-Group-Event>Agent-ID2 JOINS group-976693</Result-Group-Event>
</entry>
```

Listing 4-7. An example of a single event entry extracted from the detailed XML log file generated by the simulator

The more detailed XML file is suitable for additional processing by other modules of the visual analytics framework and can be used independently by external applications.

4.7. Summary

In this chapter, the architecture of the software application capable of simulating three-dimensional closed micro-worlds using partial information about the real world has been presented. The simulator allows utilising the limited data extracted from live video in real time to approximate activities and events that may occur in areas monitored by surveillance cameras. There are several advantages of this approach. Firstly, the simulator reduces the complexity of data processing by eliminating the need of having precise data to derive patterns of behaviour. Secondly, the control over the precision of the simulation can be maintained through fine tuning a large number of parameters, such as frequency of updating the global state, frequency of capturing the events, as well as by varying of the thresholds that dictate how dynamic patterns are being recognised and logged. Lastly, the simulator can also visualise its input data in 3D so that one can immediately validate the functionality and estimate the efficiency of the analysis through a simple observation.

Although the simulator has been developed especially for the purpose of dynamic behaviour analysis in video surveillance, since it has been implemented as a standalone, self-contained and independent module it has the potential for direct application in computer games where they need more sophisticated engines, capable of simulating intelligent behaviour. By analysing gamer's behaviour it is possible to respond to the actions more accurately and in more versatile manner.

Chapter 5

Experimental validation of the simulator

In this chapter, the functionality of the simulator of dynamic behaviour patterns is presented, evaluated and validated. The simulator has been initially verified through visual observation of the simulated agent's movements invoked by keyboard-controlled input (Section 5.4 and Appendix A.3). Subsequently, the simulation has been validated systematically using pre-recorded XML files containing data extracted from the visual scene and stored in external XML files (Section 5.5 and Appendix A.8).

5.1. Methodology of conducting the experiments

As explained in the previous chapters, the simulator plays a dual role in the process of derivation of behavioural patterns. Firstly, it synchronises the logical simulation with the incoming physical data through constant updating of the global state of the simulated world. Secondly, it provides operational semantics of the pattern language built according to the ontology of the visual scene (see Appendix A.5). Our experiments show that the difference between the physical data input and the simulated data is so insignificant that it does not make any impact on the algorithmic nature of the dynamic patterns of behaviour when the simulator is properly configured. The empirical conclusion is that *the dynamic patterns are relatively robust and do not depend on the precision of data recorded by video camera.* Second conclusion from the experimentation is that *the updates of the input data by varying the parameters of the simulator allow us to achieve satisfactory precision without compromising the efficiency*. The simulator is completely capable of approximating real data at stable 30fps frame rate for small-sized groups. This proves successfully our hypothesis that we can replace completely the real physical data with approximated simulation data and still be able to capture the essence of dynamic behaviour of both individuals and groups on the visual scene.

The basis for our experimental methodology is the simultaneous maintenance of two independent outputs – graphical output from the simulator, created by rendering the generated behaviour to the 3D animation console, and symbolic output from the event logger, created by posting information about detected events during the simulation in the event console. This allows us to compare the two outputs in order to verify the functionality of the simulator, to validate the detected events, recognised patterns and classified behaviour as well as to evaluate the quality and the performance of the application in general. According to this methodology, the *detected events* must correspond to the *observed events*, the *recognised patterns* must correspond to the *observed patterns* and the *classified behaviour* must correspond to the *observed behaviour* in 3D visual scene. In other words, we perform validation, verification and overall evaluation of the quality by simply visual observation and comparison (Figure 5-1). For example, if at the time of appearance of a message on the event console stating that given agent walks towards specific static object we can observe such activity through the simulation console we can validate the pattern and verify the functionality behind the simulation by comparing the textual message with the graphical animation.

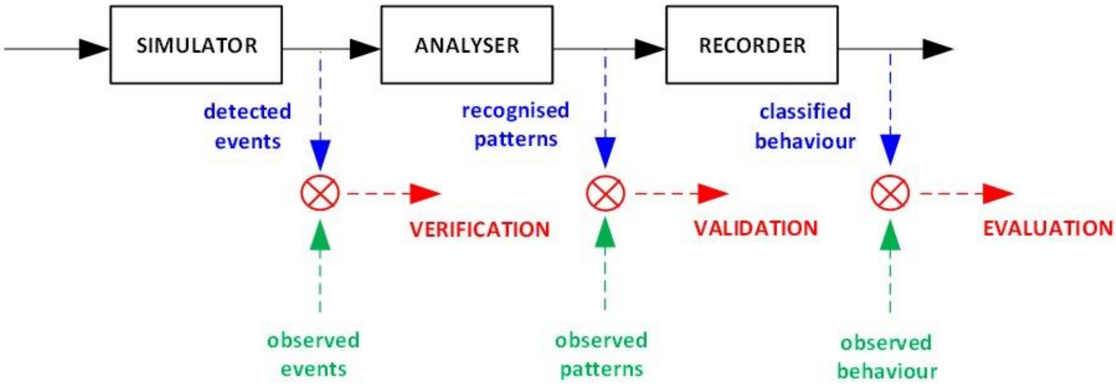


Figure 5-1. General scheme of the methodology for empirical assessment through visual observation and comparison

The simulator has an intuitive interface, which allows to observe the agents movements and to follow the flow of messages appearing on the event console in real-time (Figure 5-2).

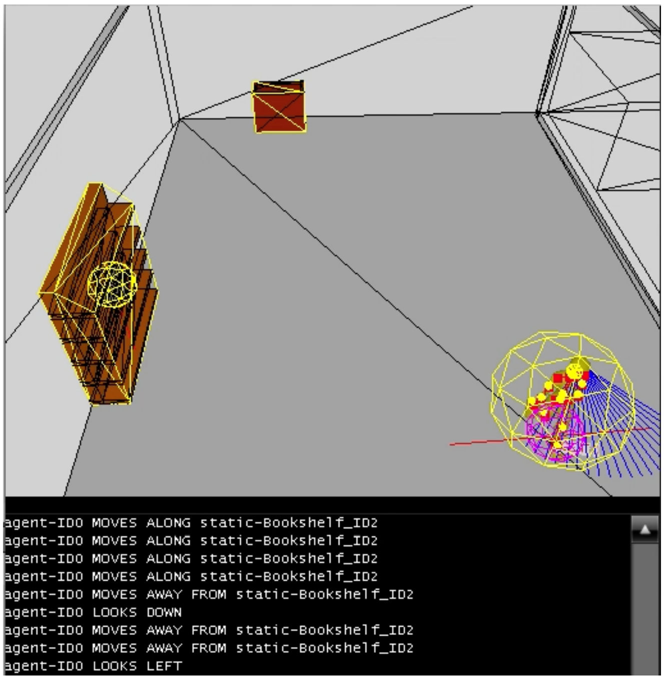


Figure 5-2. The simulator console

5.2. Parameters of the dynamic behaviour patterns and empirical optimisation of the pattern recognition

The pattern recognition has been experimented with different variations of the numerical parameters, which govern the execution of event capturing operations. To determine whether each behaviour pattern can be captured within an established amount of time with a given parameter we have performed a number of interactive simulations using both keyboard controlled and file streamlined inputs. If a specific combination of parameters does not produce satisfactory results after certain amount of time, it can be adjusted accordingly through varying the value of the parameter. In order to eliminate the need of manual editing the configuration file and restarting the simulation each time the parameters change, a separate Graphical User Interface (GUI) has been produced. It allows to change the parameters in real-time by inserting desired values in appropriate boxes located at the right-hand side of the console window (Figure 5-3).

Figure 5-3. The parameters panel of the simulator

To examine the influence of parameters and select their optimal values we can conduct as many simulations as necessary before achieving satisfactory results. After sufficiently long experimentation the empirically established optimal values for the parameters have been set as their default values. The entire process of tuning the simulator has been depicted on the flowchart diagram in Figure 5-4.

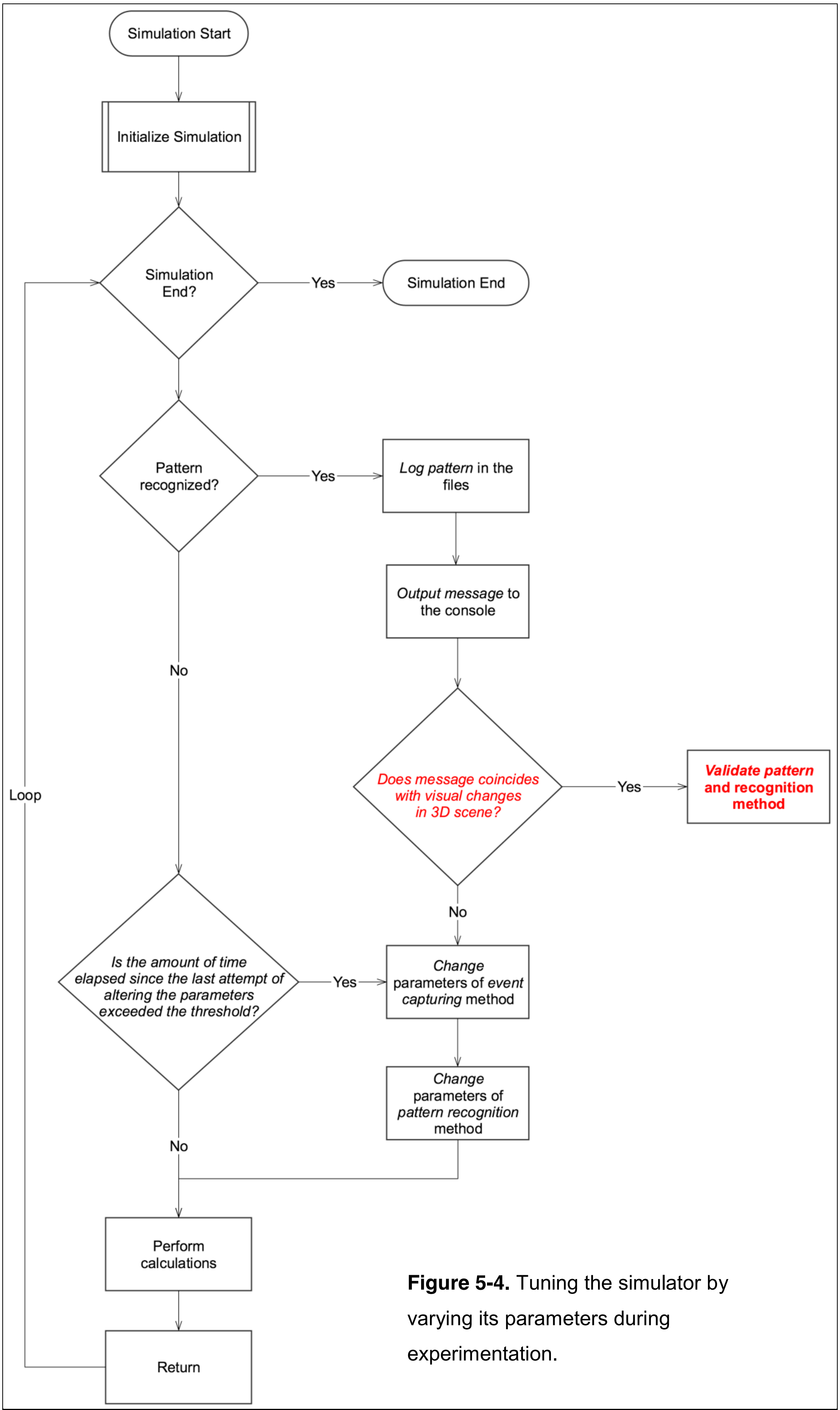


Figure 5-4. Tuning the simulator by varying its parameters during experimentation.

5.3. Configuration of the simulator and algorithmic control of the execution

Every *event capturing* and *pattern recognition* operation is performed in a single step of the simulation loop. They are synchronised and controlled by the simulator parameters which fall under four categories:

- ***Intervals***. The *frequency of capturing events* controls the time for taking spatial measurements or event sampling, i.e. the time that needs to pass from previous execution of the operation before the next one can take place. The parameter is measured in seconds (1.0f = 1 second).

- ***Cool down periods***. The *minimum time period* that needs to elapse before resuming the recording of event sampling or taking spatial measurements, e.g. to register the event when an agent moves its head left it is necessary to take continuous angular measurements and check them against the predefined thresholds for a certain period of time. It is also important to prevent initiation of taking the next sample before the current operation finishes. The parameter is measured in seconds (1.0f = 1 second).

- ***Distance thresholds***. The parameters that usually define a range with *minimum* and *maximum* values, e.g. in some cases the distance reached by an agent must fall within specific range in order to trigger registration of the event or recognition of the pattern. This parameter can be measured in different dimensions depending on the context - centimetres (1.0f = 1cm), decimetres (1.0f = 1dm), angle degrees (1.0f = 1°) and in some cases also percentages (1.0f = 100%).

- ***Sampling sizes***. The *minimum number of events* that needs to be recorded before the pattern recognition can be conducted. The parameter represents the total size of the events batch and is measured in integers (1 = size of the batch is 1).

The configuration file which sets the initialisation values of the parameters requires specifying of their name and value and grouping them in parameter sets for data validation purpose by corresponding Java class (Listing 6-1). The names of the parameters have been constructed specifically to hint at their role in the control of the simulation.

```
<config>
	<parameterSet class="WorldAdvancedLoggerState">
		<parameter>
			<name>walkTowardsInterval</name>
			<value>0.10</value>
		</parameter>
		<parameter>
			<name>walkTowardsSamplingSize</name>
			<value>5</value>
		</parameter>
		<parameter>
			<name>walkTowardsDistanceTreshold</name>
			<value>80</value>
		</parameter>
…
```

```
        <parameterSet class="GroupsManagerState">
                <parameter>
                        <name>groupFormSetInterval</name>
                        <value>1.5</value>
                </parameter>
                <parameter>
                        <name>groupFormProximityDistance</name>
                        <value>50</value>
                </parameter>
                <parameter>
                        <name>groupFormProximityLeavingDistance</name>
                        <value>70</value>
                </parameter>
…
```

Listing 5-1. Excerpt of a configuration file with default parameter initialisation

At the beginning of the simulation cycle the simulator determines if each event capturing operation can be executed. The elapsed time is checked against the interval parameters and cooling down period parameters. During execution the simulator evaluates if the corresponding event has occurred under the conditions specified using the threshold parameters set, e.g. whether an agent is within a specific range from a given static object. When the pattern recognition depends on historical data recorded in the events batch, the sampling size parameter is used to estimate if the number of accumulated events is sufficient to conduct their analysis. Similar structure is developed for all operations related to the pattern recognition because only one type of parameters is used at each step of the simulation cycle. The loop of the simulator picks up a suitable parameter for checking by traversing the hierarchy of parameters shown in Figure 5-5.

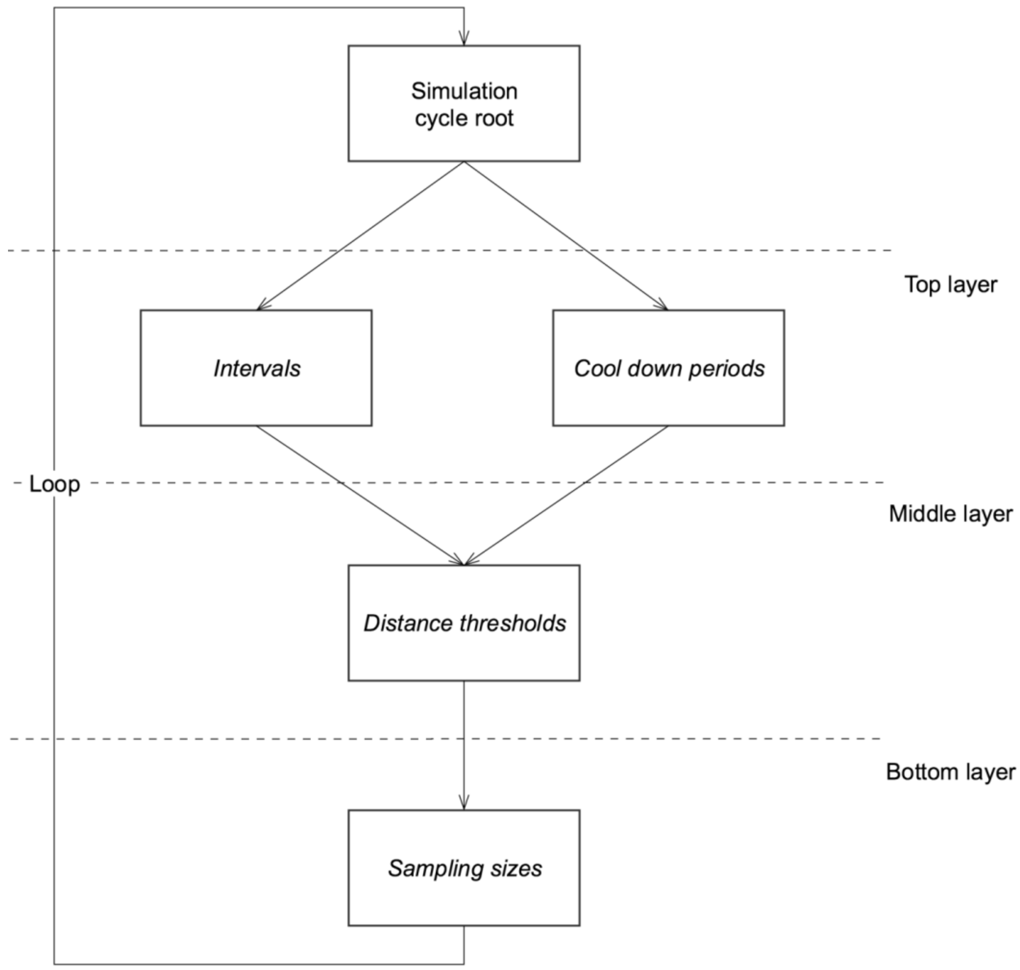


Figure 5-5. A hierarchy of parameters organised according to the points in time when they are being used in the simulation loop cycle

At the top of the hierarchy are the interval parameters and cool down period parameters that regulate the pace of pattern recognition algorithm's execution. The time-per-frame (TPF) variable in each cycle provides information about how much time has passed since the rendering of the previous frame. The value is used to define the frequency of event capturing by keeping track of the elapsed time as described by Reese and Johnson (2015) (Formula 5-1). Consequently, this parameter has an impact on how fast a given pattern can be recognised

```
time = 0
for each simulation cycle
        if time < interval
                time = time + tpf
        else
                execute algorithm
                time = 0
```

Formula 5-1. Pseudo code of execution of an operation controlled by interval parameter (Reese and Johnson, 2015)

The conditions that need to be met in order to execute an operation are also controlled by parameters defining thresholds. In most cases, they declare minimum and maximum distances that agent needs to reach in order to trigger the analysis of the event in order to recognise a pattern (Formula 5-2). Most of the parameters of this kind are located in the middle layer of the hierarchy, but in few cases, thresholds are used in sampling and pattern recognition processes as well, i.e. when dynamic behavioural patterns are being analysed using accumulated events stored in the events batch.

```
time = 0
for each simulation cycle
        if cool down period has passed
                if time < cool down period
                        time = time + tpf
                else
                        cool down period has passed
                        time = 0
```

Formula 5-2. Pseudo code of execution of an operation controlled by cooling down period parameter (Reese and Johnson, 2015)

The size of the event batch is controlled by the parameters in sampling size parameter category. Since the sizes are always checked after certain conditions are met, immediately before triggering the execution of an operation, these parameters are located at the bottom of the hierarchy. The pseudo code of the controlled execution in such a case is presented in Formula 5-3.

```
initiate algorithm
        if conditions have been met against thresholds
                get event data
                if event data needs to be stored in event batch
                        if events batch size >= sampling size
                                derive a pattern from event samples
                        else
                                record event
```

Formula 5-3. Pseudo code of an operation for pattern recognition, which uses historical events registered in the events batch which is controlled by threshold parameters

Controlling the execution of pattern recognition algorithms using parameters allows unifying the initialisation of the simulator and the tuning of the algorithms for recognition themselves. In fact the parameter pane of the simulator has been used for both purposes. Both types of parameters can be also amended at runtime without stopping the simulation, which supports flexibility and greatly reduces the overall time for tuning and analysis.

5.4. Experiment setup

During simulation the simulator produces two different outputs – graphical output, visualized on the display of the computer, and an in-memory event stream, stored for convenience locally as an XML file for verification and testing purposes. The graphical output is convenient for testing and tuning of the entire analyser but in production environment can be completely switched off to maximise the efficiency, which would substantially reduce the computational requirements. The simulator works in three different modes, accepting different types of input data:

- interactive mode, in which the simulator accepts input directly from the keyboard; this allows quick visual validation and generation of test movies.

- batch mode, in which the simulator engine executes a pre-recorded XML file containing information about the objects on the scene, their location and trajectories of movement; this can be used for more thorough experimentation and analysis.

- stream processing mode in which the simulator accepts live XML-formatted stream of data about the visual scene and generates an XML-formatted stream of patterns to be used for further processing in enterprise applications.

The screen shots shown on the following figures illustrate the basic principles of visualisation. The topology of the physical space is depicted in grey and is divided by straight lines. The agents modelled after the Blender armature are shown in yellow, encapsulated in their ghost-capsule and emitting their rays which are shown in blue. Key points of interest used for calculations during the simulation are shown in red.

The event stream produced by the simulator is used by the analyser to analyse the patterns of dynamic behaviour and potentially to generate notifications which can be used for different purpose in different scenarios. Appendix A.8 contains the XML file recorded during one of the simulation sessions and used for validation of the simulator and the analyser.

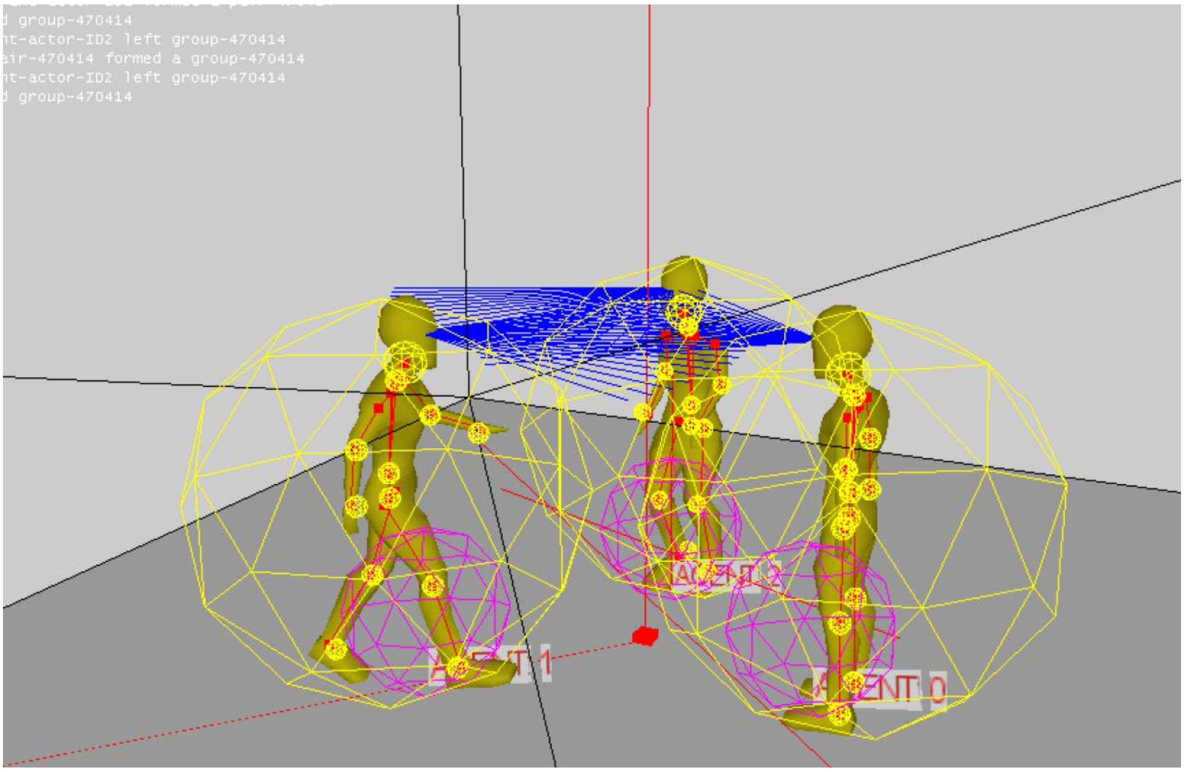


Figure 5-6. Gathering of a group of three agents

5.5. Test data for validation of the simulator

By running several experiments we will estimate the accuracy of the simulator. For this purpose we will compare the calculated movements of the agents on the scene and the information included in the XML video supplied at the input. The set of movies was recorded at 30 frames-per-second rate (FPS) and stored in XML formatted files which contain the spatio-temporal information about each agent participating in the scenario (Table 5-1). The test cases have been selected on the basis of the number of agents participating in the scenario, variety of movements exhibited by them (slow, fast, oscillating, erratic, etc.), and the reconstructed trajectories of movement (e.g. agent moving around ground floor, agent climbing the stairs, etc.) Each entry in the file represents a movie frame that encapsulates essential information for simulating agent movement accordingly (see section 4.3.1 and Appendix A.8).

Table 5-1. The video files used in the validation of the simulator, pattern analyser and experimentations.

File	Video Length (in frames)	Scenario
001.xml	281 frames	The video file contains footage of an agent moving towards a **static-Counter_ID4** in a straight manner and subsequently moving away from it.
002.xml	686 frames	The video file contains footage of a pair of agents walking next to each other around the visual scene. At one point the members of a pair move away from each other and start walking in different directions, browsing the premises of the virtual scene (building). They later meet up and form up a pair again.
003.xml	264 frames	The video file contains footage of movements of two agents whose viewing directions change slowly; one of the agents moves towards and later along **static-Bookshelf_ID2** while being in a pair
004.xml	856 frames	The video file contains footage of movements of two agents whose viewing directions change slowly; the agents are walking around the visual scene; at one point the agents were in a pair, then moved away from each other and met up again and formed a pair again.
005.xml	1128 frames	The video file contains footage of movements of two agents whose viewing directions change slowly; the agents are walking around the visual scene; one of the agents climbs **static-Stairs_ID1** at one point in the video
006.xml	808 frames	The video file contains footage of a pair waiting for a third member to join them. The third member (not being in a group or pair) walks from a different room and moves towards a pair. The group consisting of three agents is formed at the end of the video after an agent joins the pair.
007.xml	1479 frames	The video file contain footage of four agents walking around a visual scene (building). They are walking around in two different rooms separated by the wall. At one point, the agents come together and meet up in one of the rooms to form a group consisting of four members. The movie ends with a group members being orientated at each other (i.e. averagely directed at the group center point).

008.xml	1466 frames	The video file contains footage of movements of three agents whose viewing directions change slowly; the agents form pairs, groups while walking around the visual scene.
009.xml	875 frames	The video file contains a footage of a crowd consisting of nine agents walking around, forming pairs, groups and browsing the premises of the visual scene. The visual scene is cluttered for the entire duration of the video.
010.xml	857 frames	The video file contains footage of erratic movements of an agent whose viewing direction rapidly changes.
011.xml	2186 frames	The video file contains footage of movements of three agents whose viewing directions change slowly; the agents form pairs, groups while walking around the visual scene; at one point one of the agents climbs stairs, walks around the 1st floor and walks down
012.xml	1583 frames	The video file contains footage of movements of two agents whose viewing directions change rapidly; the agents exhibit erratic movements while walking around the visual scene; the agents form a pair, walk away from each other and meet up again later on at the end of the video.

5.6. Validation scenario

The validation of the simulator is based on comparison of spatial values stored in the video file against the ones that are being simulated for particular frame, i.e. calculating the difference between agent's global position vector components stored for a given frame in the XML video file and the global position vector calculated for that frame during the simulation (replaying). The difference is expressed in percentage values (eq. 5-1) and is calculated for every frame.

$$\frac{|\Delta v|}{\frac{\sum v}{2}} * 100 = \frac{|v_1 - v_2|}{\frac{(v_1 + v_2)}{2}} * 100 = Percentage\ Difference \qquad \textbf{(eq. 5-1)}$$

The following diagrams depict how the error (presented on vertical scale) fluctuated over the duration of the videos (measured in frames presented on horizontal scale). The x, y and z components of a measured global position vectors have been written in a self-explanatory notation: the first part indicates which agent has been monitored and the second part highlights the component (e.g. **agent-ID0-vectorX**). The case scenarios were replayed with 30 frames-per-second rate.

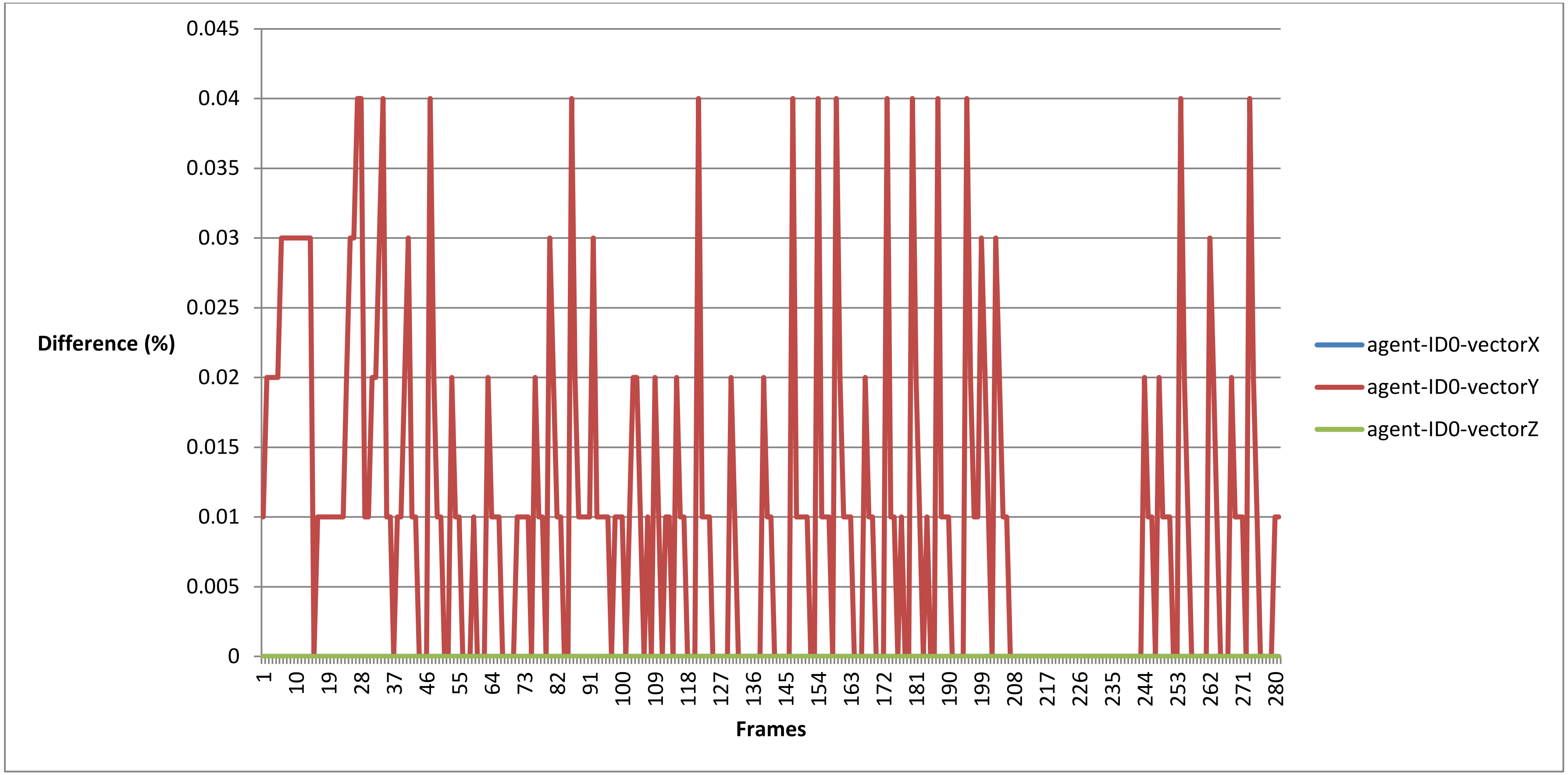


Figure 5-7. Validation of simulator on a basis of 001.xml

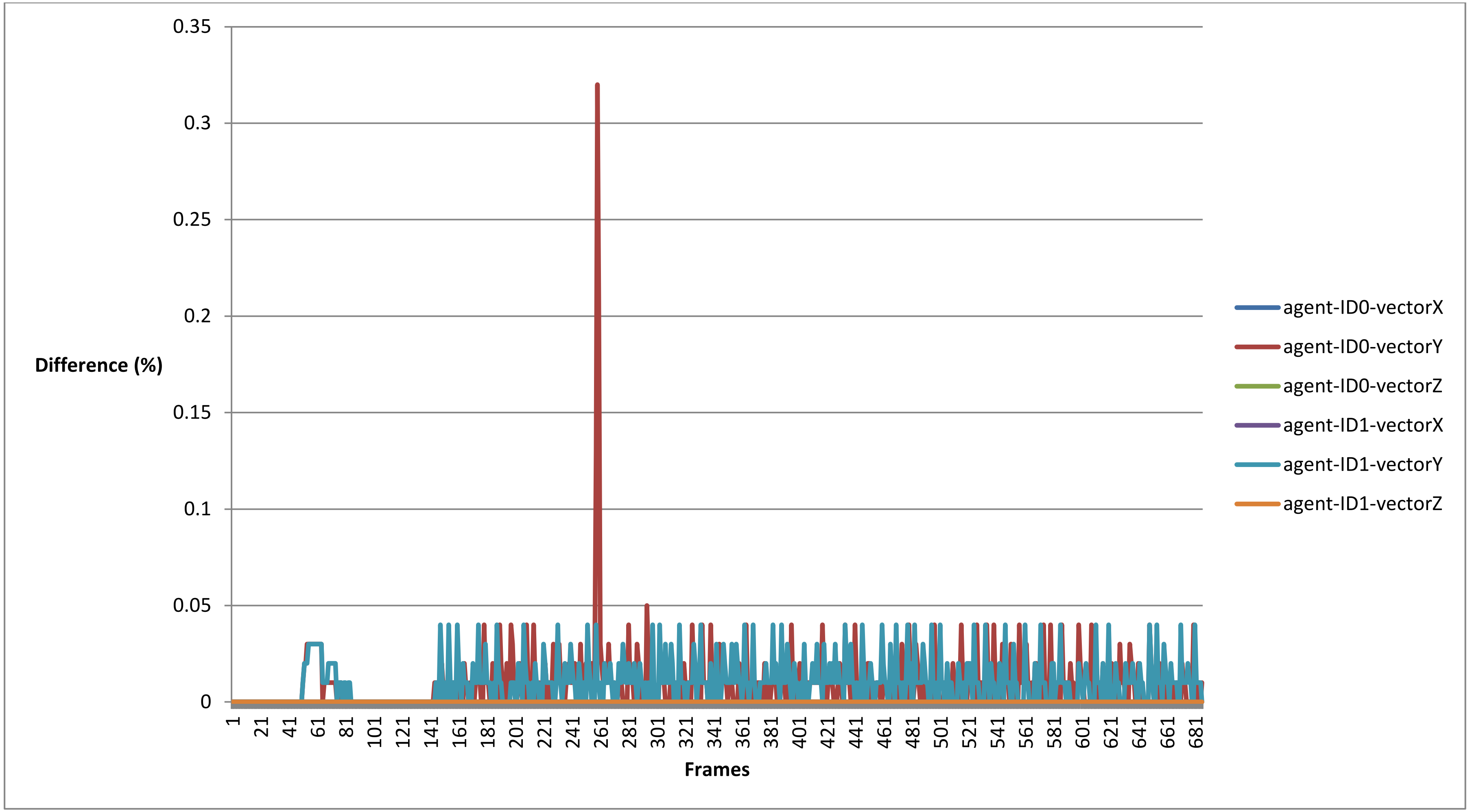


Figure 5-8. Validation of simulator on a basis of 002.xml

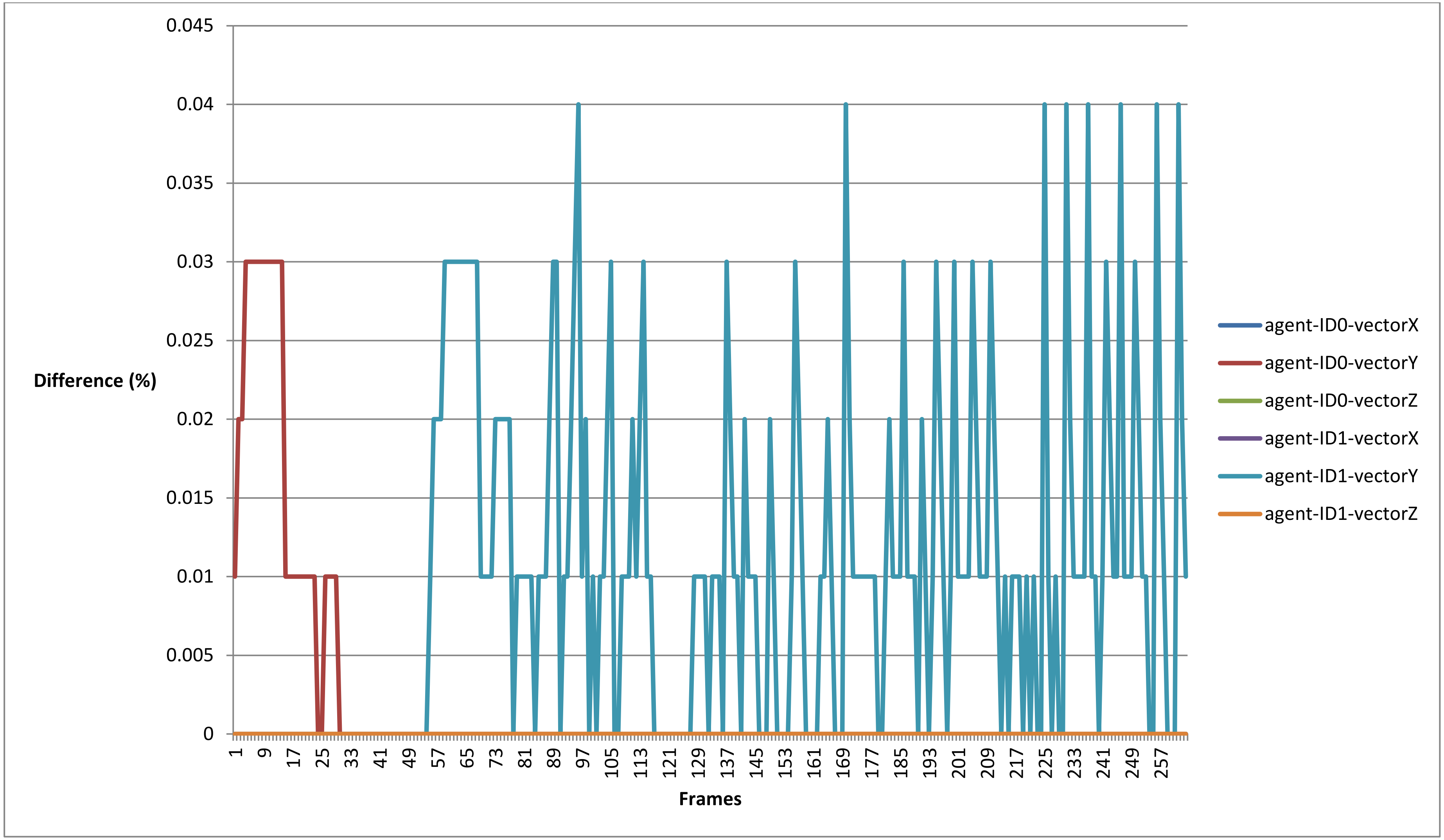


Figure 5-9. Validation of simulator on a basis of 003.xml

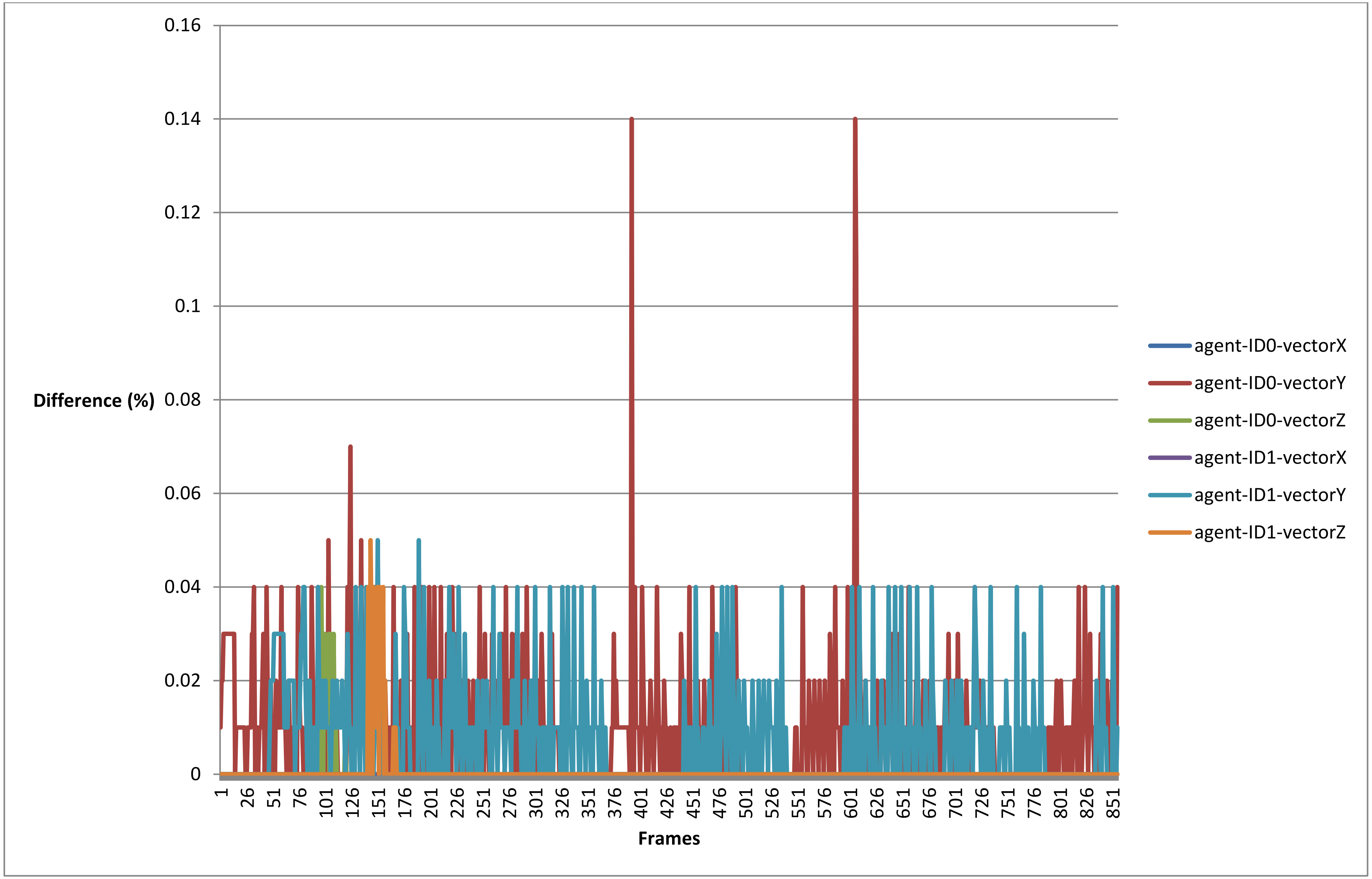


Figure 5-10. Validation of simulator on a basis of 004.xml

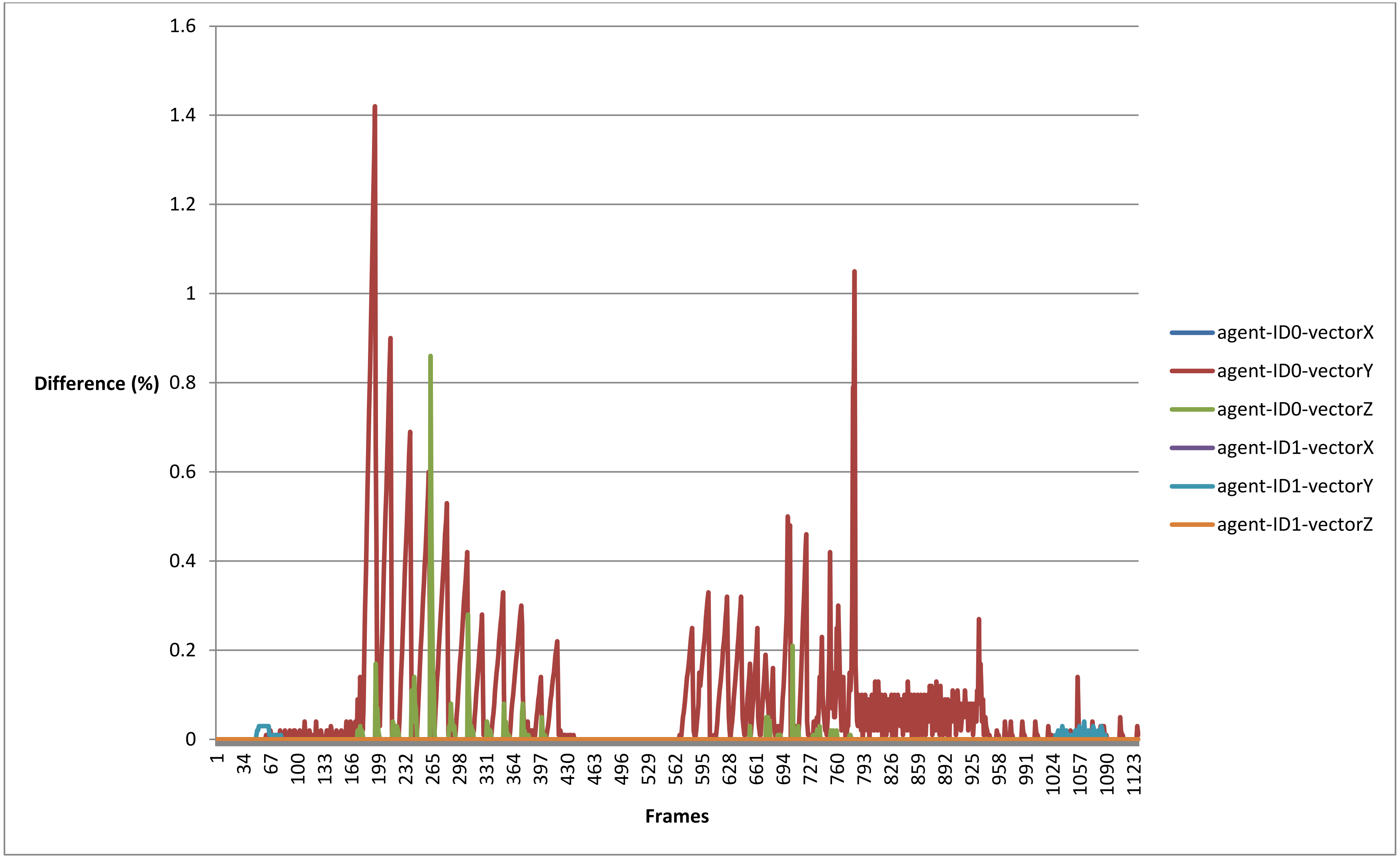


Figure 5-11. Validation of simulator on a basis of 005.xml.

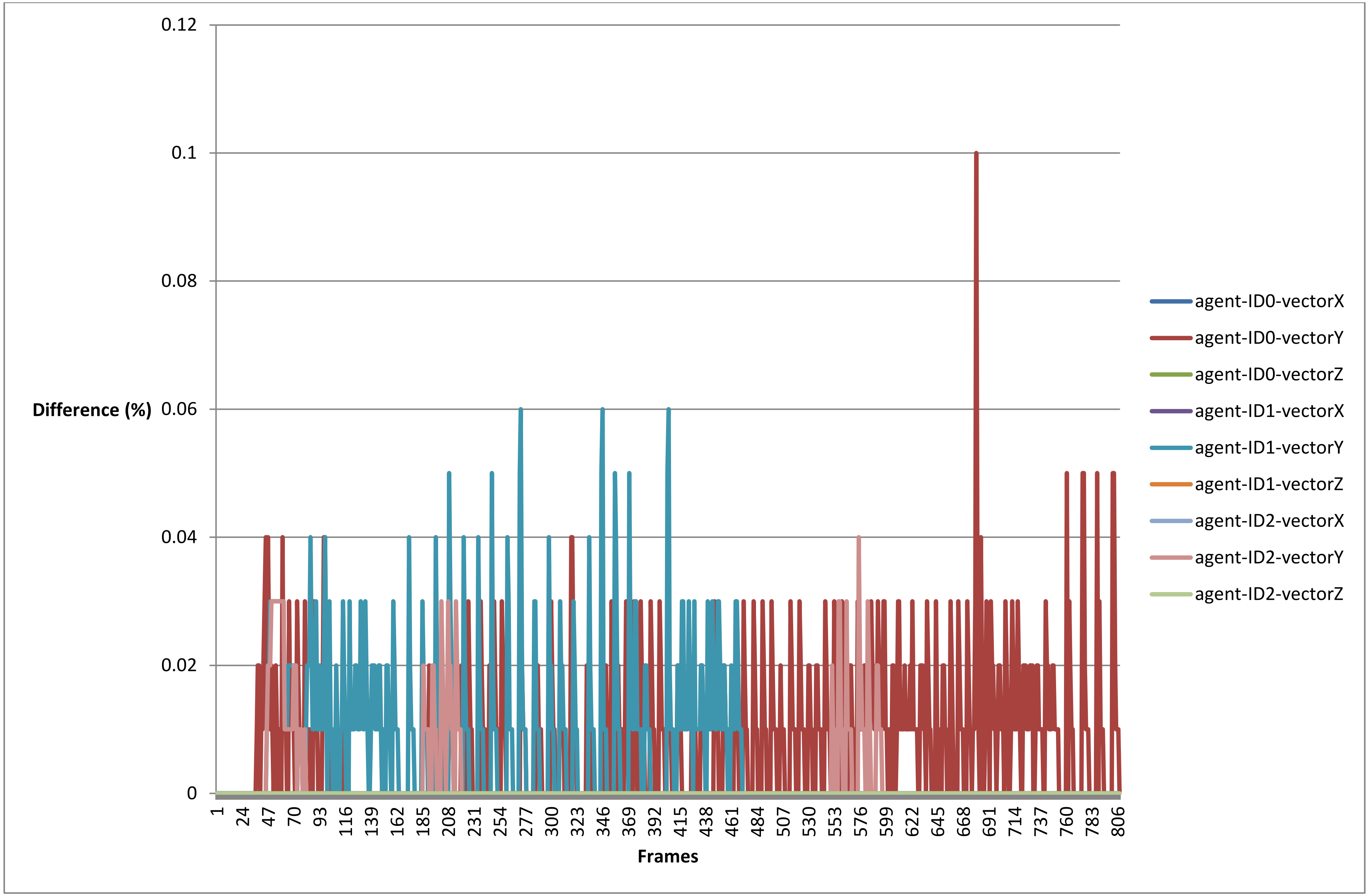


Figure 5-12. Validation of simulator on a basis of 006.xml.

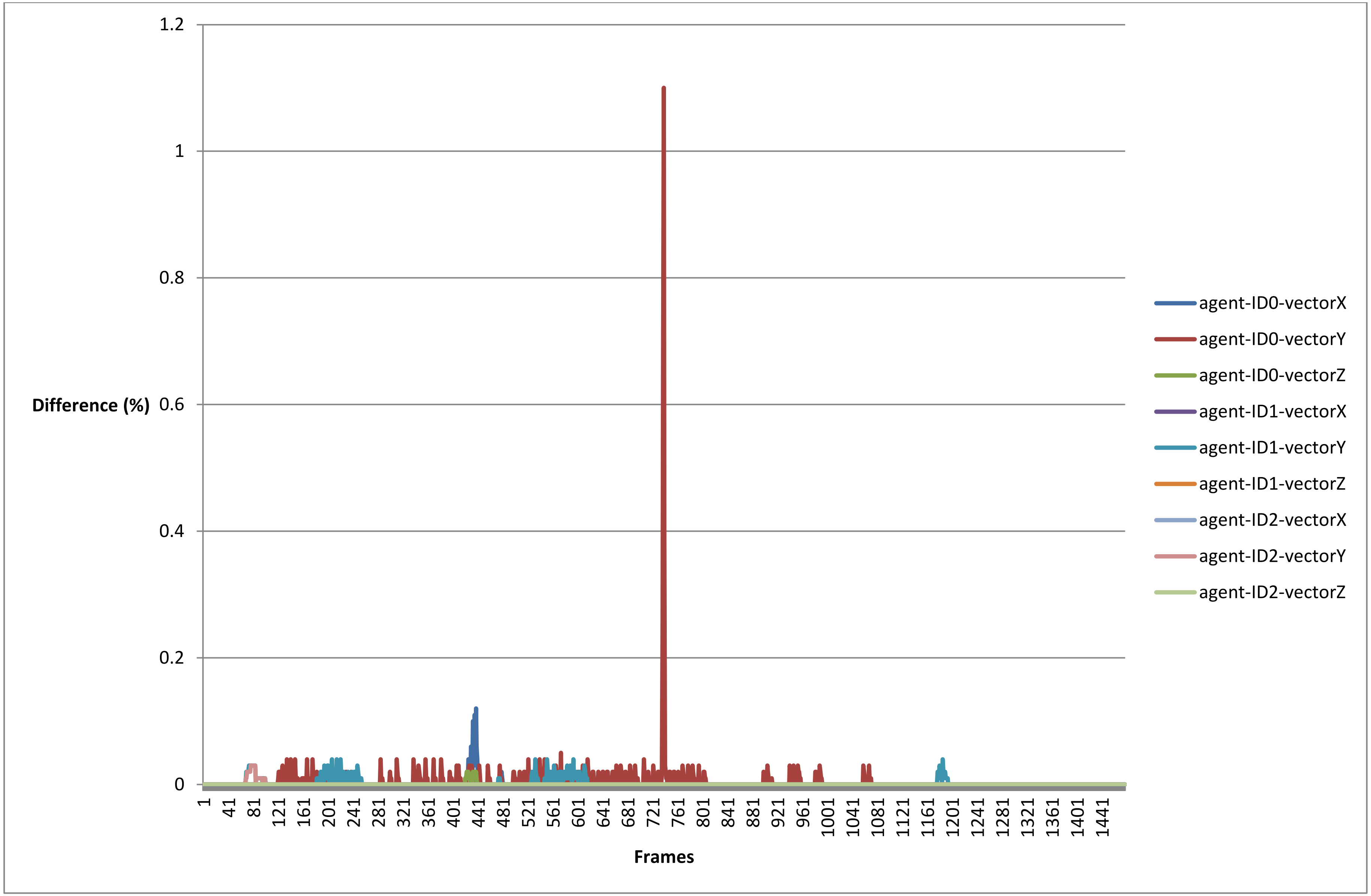


Figure 5-13. Validation of simulator on a basis of 007.xml.

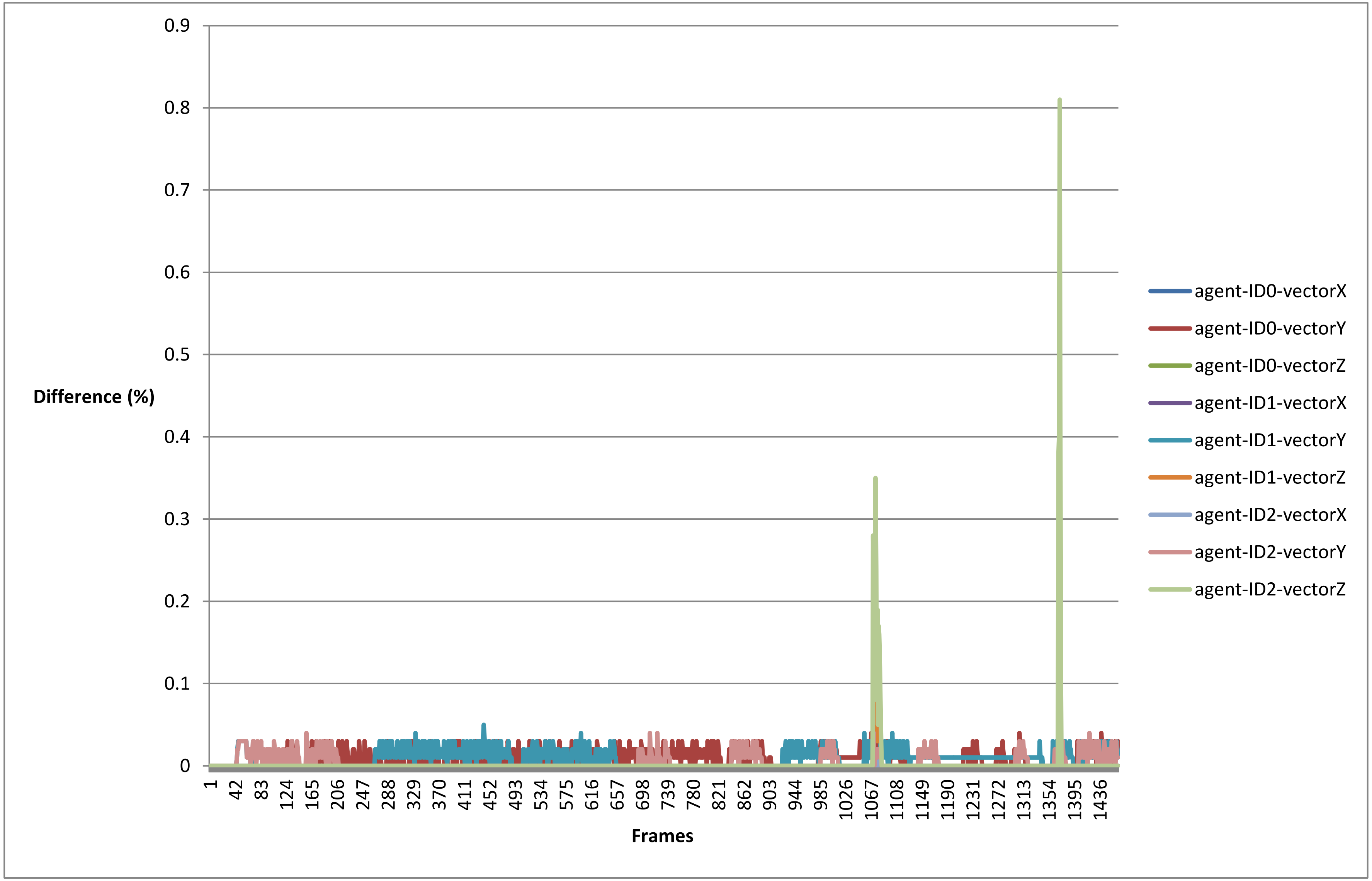


Figure 5-14. Validation of simulator on a basis of 008.xml.

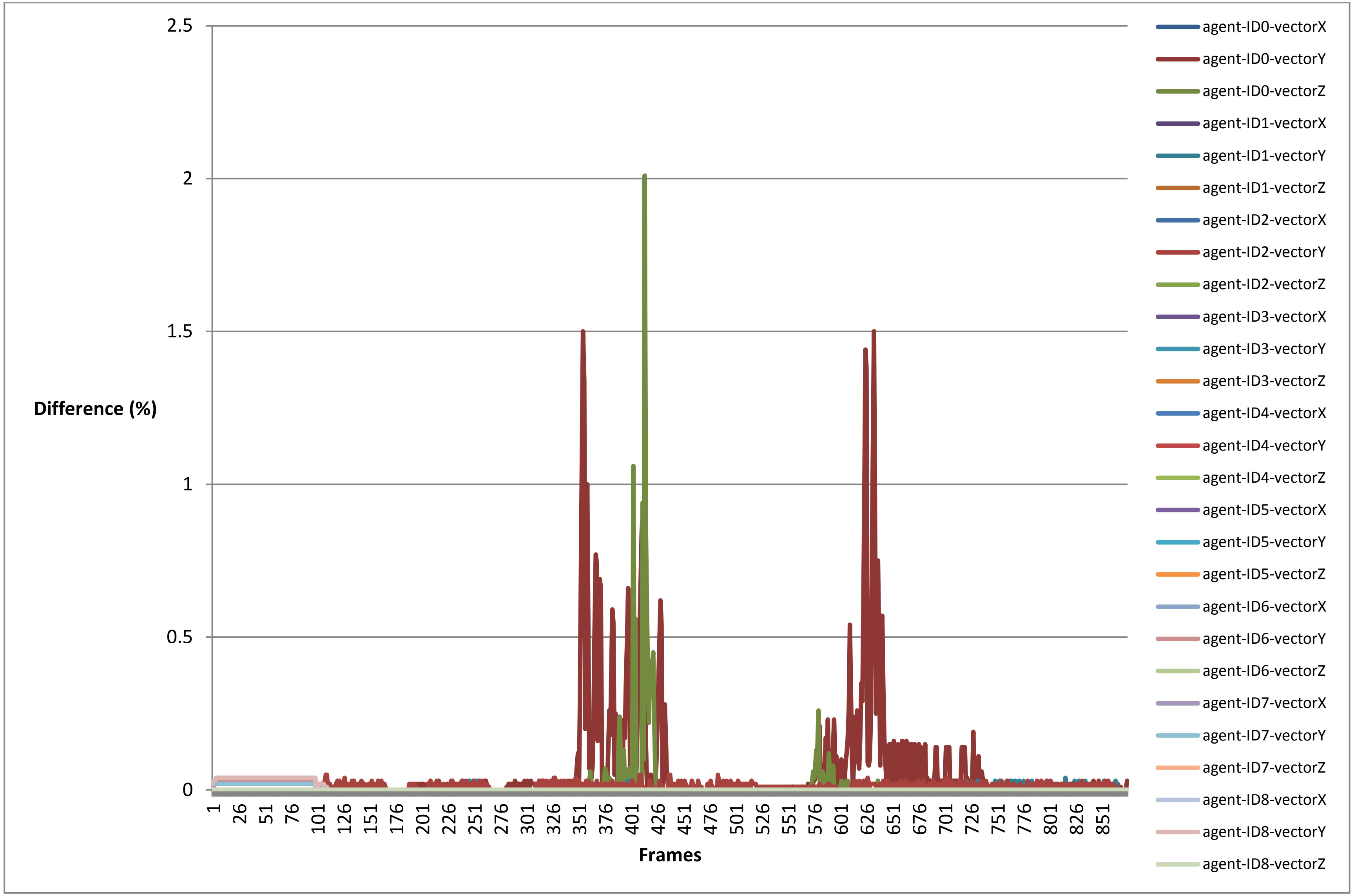


Figure 5-15. Validation of simulator on a basis of 009.xml.

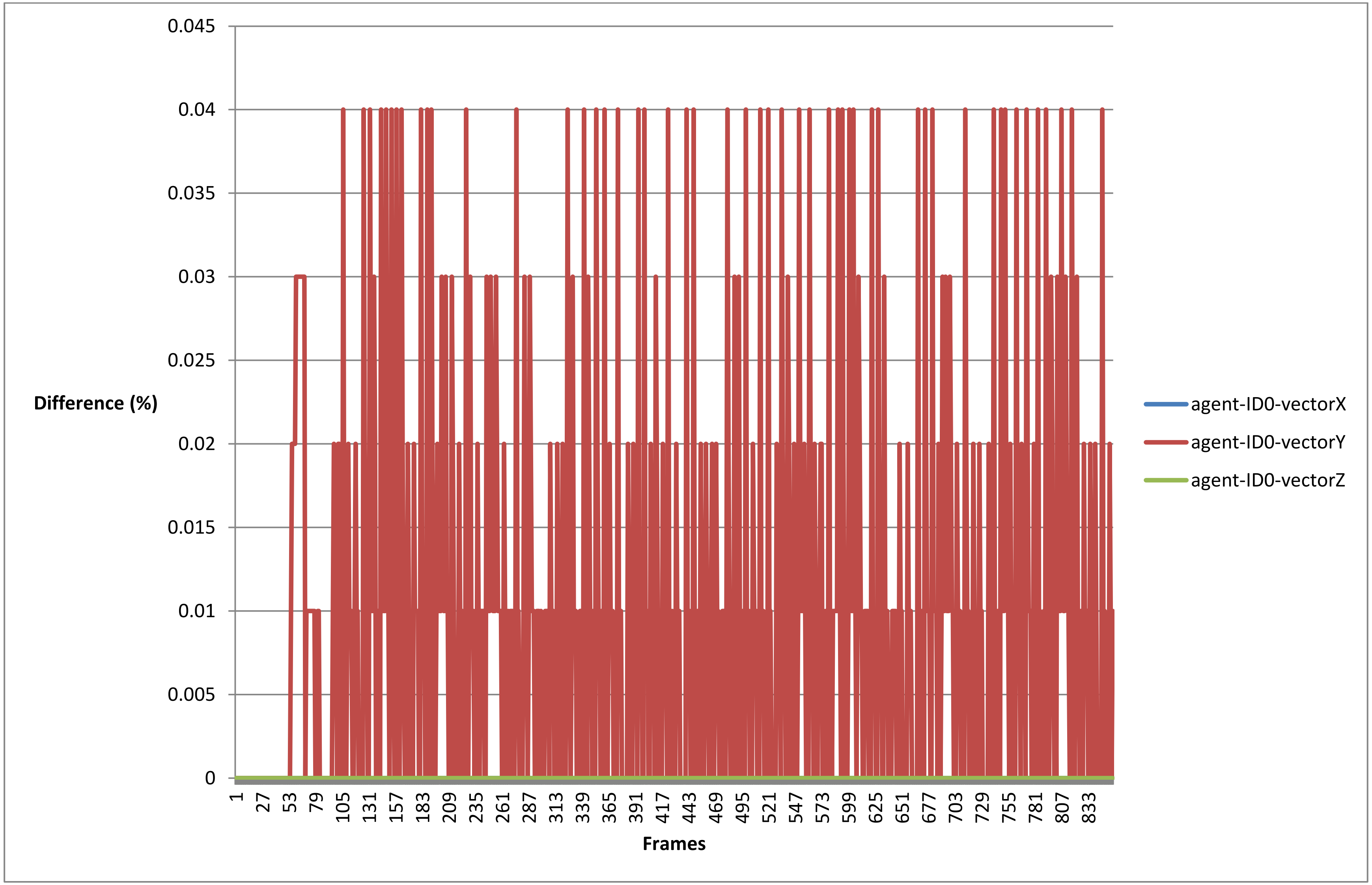


Figure 5-16. Validation of simulator on a basis of 010.xml.

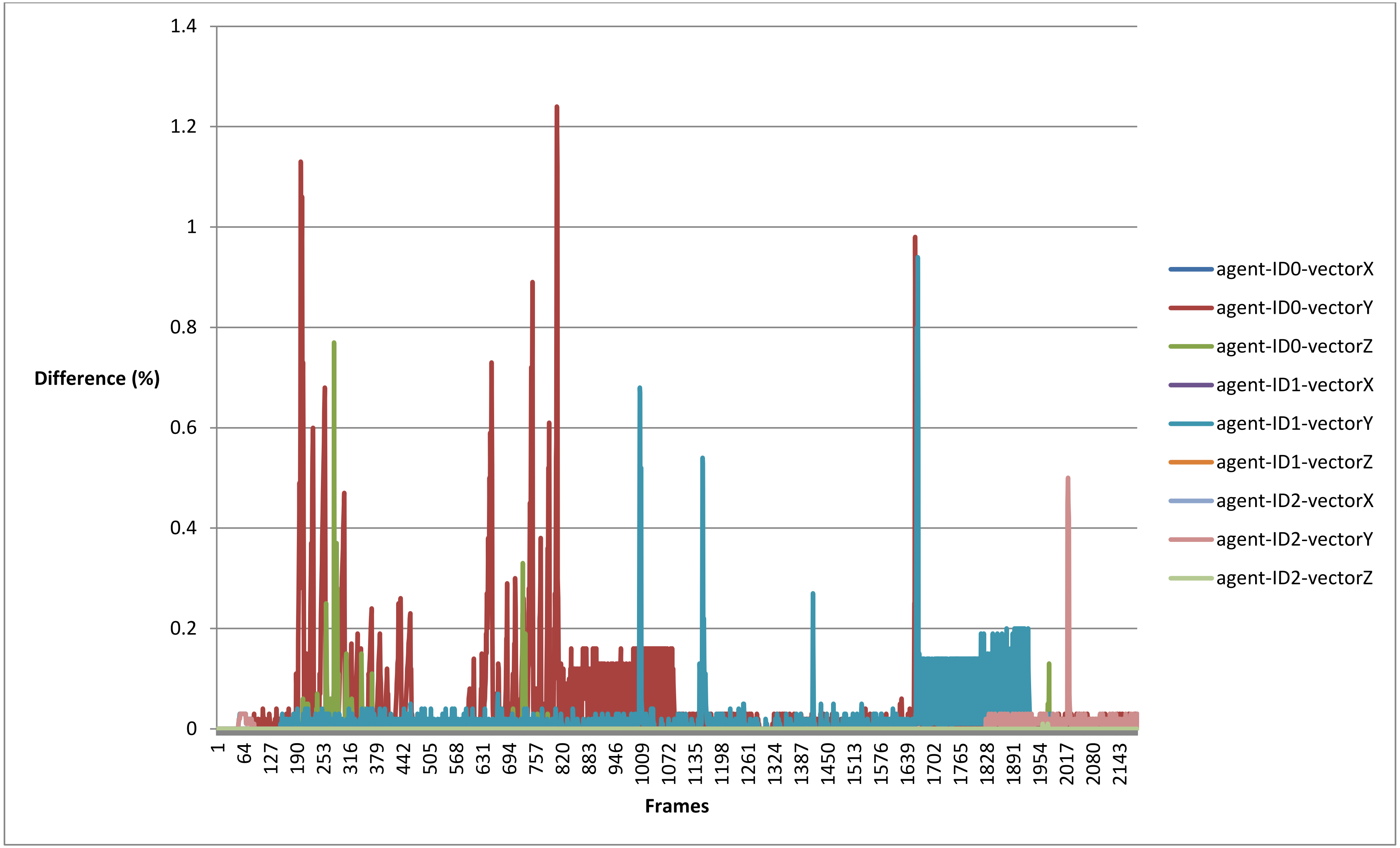


Figure 5-17. Validation of simulator on a basis of 011.xml.

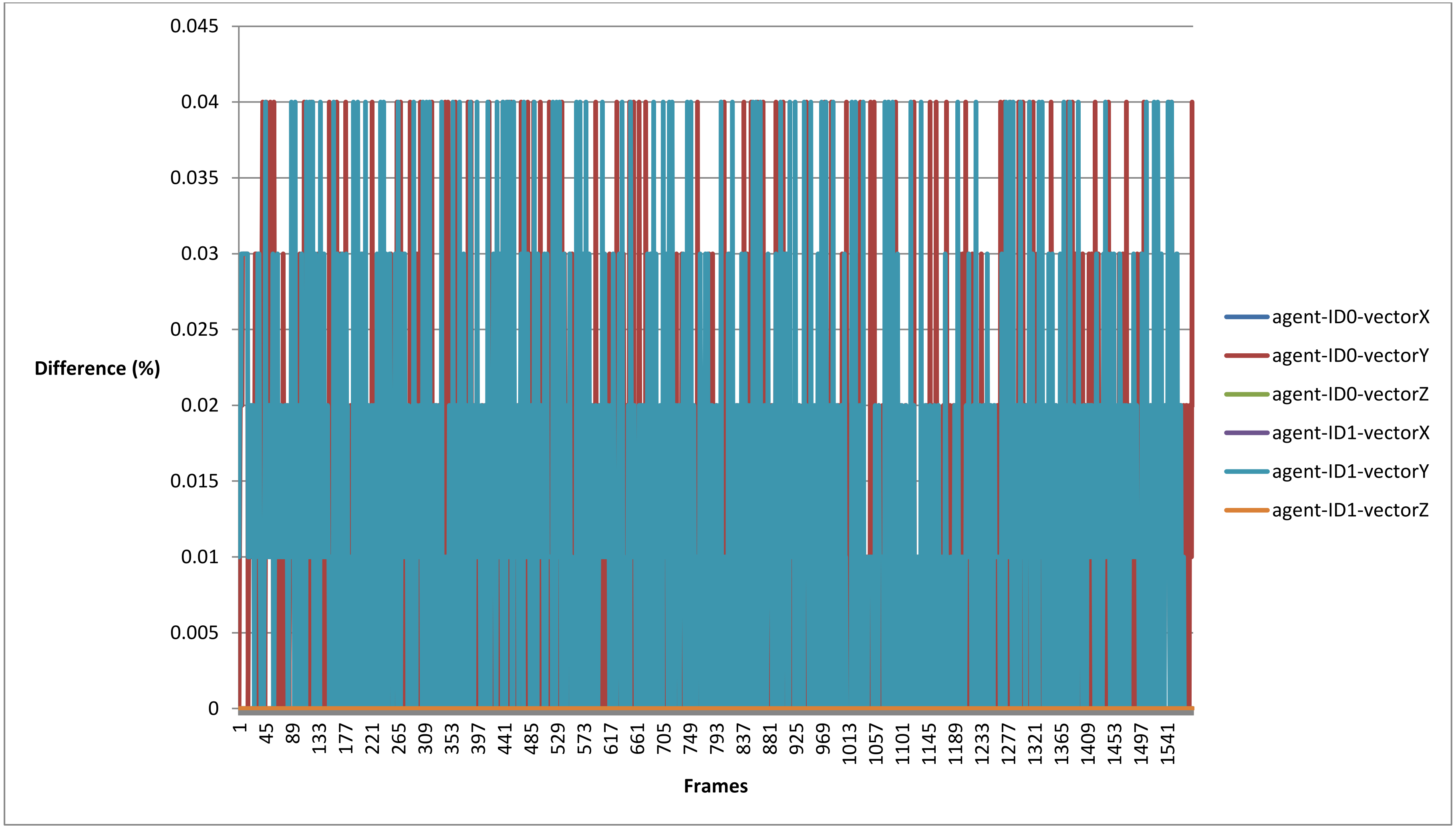


Figure 5-18. Validation of simulator on a basis of 012.xml.

Analysis of the experimental results

The results show that errors in the calculation of global position vectors of agents at given frames remain between 1% and 2% in most of the simulated scenarios. However, it was observed that error was maximum when agent was climbing the stairs in the video 005.xml (1.42%; Figure 5-11), 009.xml (2.10%; Figure 5-15) and 011.xml (1.24%; Figure 5-17). The minimum was observed when agents were walking on a flat surface (floor) and is 0.01%. The sudden spikes in errors in Figures 5-12, 5-13, 5-14, 5-15 and 5-16 suggest that when data for greater number of agents is delivered at the input, the simulator is capable of calculating their global position vectors at every frame but the constituent components may appear slightly deviated from their original recorded values in the XML video file. Such errors only appear when more than one agent participates in a replayed case scenario, which leads to the conclusion that the cause is related to the amount of data that needs to be parsed and extracted from the XML during the simulation. Although the errors increase in such instances, the accuracy of the simulator remains on average between 98% and 99%, which is deemed acceptable. Of course in all experiments the number of agents on the scene was limited to 10 due to the limitation of the computational resource available (an Apple iMac with 16GB RAM). More precise analysis can be done when an adequate server equipment is available since in a more realistic setting the simulation will be executed on a more powerful server equipment.

5.7. Summary

In this chapter the behavioural patterns that can be inferred by analysing the log file of the simulator have been presented in a systematic way. We have proven that by utilising 3D programming concepts it is possible to generate events that can provide sufficiently rich data to analyse the patterns of dynamic behaviour. Since these patterns are constructed only from body movements they are independent on the specific area of application. The simulator has the capability of capturing events related to the dynamic behaviour of both individuals and groups of individuals thanks to the logical rather than statistical relationships between participating entities. The patterns can be further enriched by contextualisation, which is outside of the scope of this research, but is planned for a separate research project within the current simulation-based event-driven framework. In the next chapter, we will look closely at various parameters controlling the general flow of simulations and will analyse the impact they have on the pattern recognition process.

Chapter 6

Experimental analysis and evaluation of the patterns of individual and group dynamic behaviour

The process of simulation, event log generation and pattern analysis is controlled by a large number of parameters. This chapter concludes our research with detailed description of these parameters and reporting of the results of the extensive experiments we have conducted using the developed software. It is a result of a large number of experimental simulations of different scenarios in which we have performed visual verification of the simulation, validation of the recognised patterns and empirical estimation of the influence of their parameters on the overall performance and quality of the analysis.

6.1. Experimental setup and validation of the pattern analyser

The validation and accuracy of the pattern analyser was made through replaying the pre-recorded case scenarios stored in XML video files and analysis of the log that was produced as a result of the simulation. A set of several different movies with various lengths where dynamic behaviours are presented in various combinations and within diverse scenarios were recorded at 30 frames per second (Table 5-1). The movies contain spatial information of participating agents, such as global position vectors, at given frames (see section 4.3.1). The pattern analyser was configured to report exact movie frame at which the pattern has been recognized. It was validated through the analysis of the vectors describing spatial properties of an agent and changes of the components of these vectors over several frames prior to the video frame at which a pattern has been recognized. The patterns are always being recognized on a basis of data contained in a sequence of frames. The patterns that have been implemented in and that can be recognized by the simulator are the following:

- “Somebody is walking towards something” – recognized when an individual agent moves towards a static object located in the visual scene.
- “Somebody is walking way from something” – recognized when an individual agent moves away from a static object located in the visual scene.
- “Somebody walks alongside something” – recognized when an individual agent moves alongside a static object located in the visual scene.
- “Somebody climbs something” – recognized when an individual agent climbs a static object located in the visual scene.
- “Somebody punches something / somebody” – recognized when an individual agent punches a static object or other agent located in the visual scene.

- “Somebody kicks something / somebody” – recognized when an individual agent kicks a static object or other agent located in the visual scene.
- “Something picks up / reaches out for / carry something” - recognized when an individual agent interacts with a dynamic object located in the visual scene with hands limbs.
- “Somebody holds something over / places something on top of something” – recognized when an individual agent holds a dynamic object over a static object located in the visual scene or places a dynamic object on top of a static object located in the visual scene.
- “Somebody passes over something to somebody” – recognized when an individual agent carrying a dynamic object passes it to another individual agent with hands limbs.
- “Somebody drops down something” – recognized when an individual agent drops down carried dynamic object on the floor.
- “Somebody looks up / down right / left” – recognized when an individual agent orientates head in a given direction.
- “Somebody and somebody form a Pair” – recognized when two individual agents are within close proximity and form a pair.
- “Somebody joins a Group” and “Two Pairs / Groups merge” – recognized when an individual agent joins already established group or two already established pairs or groups merge.
- “Somebody leaves a Group” and “Group disbands” – recognized when participant leaves the group or when there is no more participants left in the group after agent departure.
- “Somebody makes a handshake with somebody” – recognized when an individual agent shakes hands with other individual agent.
- “Group / Pair moves towards something” – recognized when a pair or group of agents move towards a static object located in the visual scene.
- “Group / Pair moves away from something” – recognized when a pair or group of agents move away from a static object located in the visual scene.
- “Group / Pair moves alongside something” – recognized when a pair or group of agents move alongside a static object located in the visual scene.
- “Group / Pair climbs something” – recognized when a pair or group of agents climb a static object located in the visual scene.

A more detailed description of the implemented patterns can be found in Appendix A.2. Table 6-1 summaries the results of the validation for each selected pattern under optimal conditions, i.e. at the frame rate the video was recorded (30 FPS) and without omitting any frames from the video during the simulation.

Table 6-1. Pattern Analyser validation experiments.

Pattern / Video Specification	**Recorded Video XML File**	**Verbal Interpretation**	**Analyser Output in Log File**	**Validation**
Label: *“Walking towards something”* **Video File:** 001.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 281 Frames	**Frame 0:** ***Counter_ID4 Global Position:*** (141.60583, 4.3999634, 129.92328) ***Agent-ID0 Global Position:*** (161.19113, 4.3976526, 51.879482) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 3:** ***Agent-ID0 Global Position:*** (161.19113, 4.3976526, 51.879482) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 23:** ***Agent-ID0 Global Position:*** (154.18465, 4.3967037, 60.56711) ***Agent-ID0 Viewing Direction:*** (-0.48266762, 0.0, 0.87748957) **Frame 281:** ***Agent-ID0 Global Position:*** (136.80289, 4.398989, 88.1191) ***Agent-ID0 Viewing Direction:*** (-0.09739578, 0.0, 0.9967296)	The global position of Agent-ID0 got closer to global position of Counter_ID4 in a sequence of 20 frames between Frame 3 and Frame 23 of the video; The distance between Agent-ID0 and Counter_ID4 decreased while the viewing direction vector remained approximately the same, directed at Counter_ID4.	23 :: agent-ID0 MOVES TOWARDS static-Counter_ID4	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.
Label: *“Walking towards something” (while agent is in a group)* **Video File:** 003.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 264 Frames	**Frame 0:** ***Bookshelf_ID2 Global Position:*** (191.95218, 4.4000015, -16.40506) ***Bookshelf_ID2 Global (Volume Boundary Point):*** (191.95218, 4.4000015, 7.7467536) ***Bookshelf_ID2 Global Position (Volume Boundary Point):*** (191.95218, 4.4000015, -40.556873) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 35:** ***Agent-ID1 Global Position:*** (152.80832, 4.3964806, 24.350977)	The global position of Agent-ID1 got closer to global position of Bookshelf_ID2 between Frame 35 and 45. While walking, the viewing direction vector of Agent-ID1 was directed at Bookshelf_ID2 before Agent-ID1 found himself next to it (this is indicated by the x and z component of viewing direction: -0.05518721 and -0.999955 respectively).	45 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

	Agent-ID1 Viewing Direction: (-0.05518721, 0.0, -0.999955) **Frame 45:** ***Agent-ID1 Global Position:*** (152.57428, 4.389238, 20.110752) ***Agent-ID1 Viewing Direction:*** (-0.05518721, 0.0, -0.999955) **Frame 264:** ***Agent-ID1 Global Position:*** (148.50789, 4.3999987, -53.56827) ***Agent-ID1 Viewing Direction:*** (-0.05518721, 0.0, -0.999955)			
Label: *"Walking away from something"* **Video File:** 001.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 281 Frames	**Frame 0:** ***Counter_ID4 Global Position:*** (141.60583, 4.3999634, 129.92328) ***Agent-ID0 Global Position:*** (161.19113, 4.3976526, 51.879482) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 95:** ***Agent-ID0 Global Position:*** (142.48766, 4.3979836, 86.538315) ***Agent-ID0 Viewing Direction:*** (-0.9498612, 0.0, 0.31736398) **Frame 105:** ***Agent-ID0 Global Position:*** (138.66302, 4.3909364, 85.987625) ***Agent-ID0 Viewing Direction:*** (-0.7507819, 0.0, -0.6627846) **Frame 281:** ***Agent-ID0 Global Position:*** (136.80289, 4.398989, 88.1191) ***Agent-ID0 Viewing Direction:*** (-0.09739578, 0.0, 0.9967296)	The global position of Agent-ID0 was closer to global position of Counter_ID4 at frame 95 and more distant at frame 105. Furthermore there is a clear change in viewing direction in a sequence of 10 frames between frame 95 and 105; The components values (x, y, z) of viewing direction vector indicate a change in the orientation of an agent into opposite direction in a sequence of 10 frames.	105 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

Label: *"Walking along something" (while agent is in a group)* **Video File:** 003.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 264 Frames	**Frame 0:** ***Bookshelf_ID2 Global Position:*** (191.95218, 4.4000015, -16.40506) ***Bookshelf_ID2 Global Position (Volume Boundary Point):*** (191.95218, 4.4000015, 7.7467536) ***Bookshelf_ID2 Global Position (Volume Boundary Point):*** (191.95218, 4.4000015, -40.556873) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 120:** ***Agent-ID1 Global Position:*** (150.91478, 4.3909535, -9.957732) ***Agent-ID1 Viewing Direction:*** (-0.05518721, 0.0, -0.999955) **Frame 133:** ***Agent-ID1 Global Position:*** (150.6252, 4.3948503, -15.205059) ***Agent-ID1 Viewing Direction:*** (-0.05518721, 0.0, -0.999955) **Frame 264:** ***Agent-ID1 Global Position:*** (148.50789, 4.3999987, -53.56827) ***Agent-ID1 Viewing Direction:*** (-0.05518721, 0.0, -0.999955)	The X, Y and Z components of a global position of an Agent-ID1 indicate that he is located closely next to Bookshelf-ID2 between frames 120 and 133 within a distance defined by **walkAlongDistanceToObject** parameter (set to 60 at the time). The X and Z component of a global position of an Agent-ID-1 at frame 120 and 133 indicate that he travelled along the space occupied by the volume of the Bookshelf-ID2, which pivot (central) point is located at (191.95218, 4.4000015, -16.40506) and the boundary starts at point (191.95218, 4.4000015, 7.7467536) and ends at point (191.95218, 4.4000015, -40.556873); The analyser identified Agent-ID1 as participant-agent-ID1 because at the time he was part of the group (pair-535734).	133 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

Label: *"Climbing something up"* **Video File:** 005.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 1128 Frames	**Frame 0:** ***Stairs_ID1 Global Position:*** (-114.16765, 4.1388817, -19.553867) ***Stairs_ID1 Global Position (Volume Boundary Point):*** (-85.739976, 4.1388816, -23.555876) ***Stairs_ID1 Global Position (Volume Boundary Point):*** (-142.57341, 4.1388816, -23.555876) ***Agent-ID0 Global Position:*** (-78.60532, 4.3999987, -61.354675) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 127:** ***Agent-ID0 Global Position:*** (-109.088104, 4.393914, -30.396585) ***Agent-ID0 Viewing Direction:*** (-0.33066198, 0.0, 0.94531393) **Frame 214:** ***Agent-ID0 Global Position:*** (-116.956635, 20.517485, 0.6147314) ***Agent-ID0 Viewing Direction:*** (-0.22846565, 0.0, 0.9750688) **Frame 1128:** ***Agent-ID0 Global Position:*** (-36.784275, 4.398902, -53.2181) ***Agent-ID0 Viewing Direction:*** (-0.9043657, 0.0, -0.4302044)	The global position of Agent-ID0 has changed within a period of 87 frames between frame 127 and 214. The global position of Stairs_ID1 indicates that Agent-ID0 climbed them up since his global position was not only within the boundaries of the Stairs_ID1 between frames 127 and 214 but also the value of y component of his global position increased from 4.393914 to 20.517485.	214 :: agent-ID0 CLIMBS static-Stairs_ID1 UP	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

Label: *"Forming a pair"* **Video File:** 004.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 856 Frames	**Frame 0:** ***Agent-ID0 Global Position:*** (161.19113, 4.3999987, 51.879482) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 430:** ***Agent-ID0 Global Position:*** (149.63788, 4.396135, 63.178993) ***Agent-ID0 Viewing Direction:*** (-0.45466504, 0.0, -0.89232) ***Agent-ID1 Global Position:*** (126.57549, 4.3929315, 14.1280155) ***Agent-ID1 Viewing Direction:*** (0.46358812, 0.0, 0.8877161) **Frame 442:** ***Agent-ID0 Global Position:*** (147.43553, 4.3983274, 58.85669) ***Agent-ID0 Viewing Direction:*** (-0.45466504, 0.0, -0.89232) ***Agent-ID1 Global Position:*** (128.82101, 4.393213, 18.427898) ***Agent-ID1 Viewing Direction:*** (0.46358812, 0.0, 0.8877161) **Frame 856:** ***Agent-ID0 Global Position:*** (156.2939, 4.3999643, 92.23968) ***Agent-ID0 Viewing Direction:*** (0.4520256, 0.0, 0.8936589) ***Agent-ID1 Global Position:*** (107.31645, 4.3997803, -5.176361) ***Agent-ID1 Viewing Direction:*** (-0.36949524, 0.0, -0.9308196)	The global position of Agent-ID0 got closer to global position of Agent-ID1 in a sequence of 12 frames on X and Z axis. The viewing direction vectors indicate that Agent-ID0 and Agent-ID1 were walking towards each other until they met and formed a pair at frame 442 because they found themselves within proximity defined by **groupFormProximityDistance** parameter (set to 50 at the time)	442 :: agent-ID0 and agent-ID1 FORMS A pair-931849	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

Label: *"Forming a group"* **Video File:** 006.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 808 Frames	**Frame 0:** ***Agent-ID0 Global Position:*** (-78.60532, 4.3999987, -61.354675) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID2 Global Position:*** (94.19113, 4.3999987, 41.879482) ***Agent-ID2 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 64:** ***Agent-ID0 Global Position:*** (-60.42729, 4.398239, -47.536617) ***Agent-ID0 Viewing Direction:*** (0.91928446, 0.0, 0.39733076) ***Agent-ID1 Global Position:*** (136.5312, 4.3974657, 27.941086) ***Agent-ID1 Viewing Direction:*** (-0.9448742, 0.0, 0.33191645) ***Agent-ID2 Global Position:*** (94.19113, 4.399998, 41.879482) ***Agent-ID2 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 76:** ***Agent-ID0 Global Position:*** (-60.42729, 4.398239, -47.536617) ***Agent-ID0 Viewing Direction:*** (0.91928446, 0.0, 0.39733076) ***Agent-ID1 Global Position:*** (133.07945, 4.3968735, 29.153582) ***Agent-ID1 Viewing Direction:*** (-0.9448742, 0.0, 0.33191645) ***Agent-ID2 Global Position:*** (94.19113, 4.399998, 41.879482) ***Agent-ID2 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 654:** ***Agent-ID0 Global Position:*** (97.49183, 4.393773, -0.43898976) ***Agent-ID0 Viewing Direction:*** (0.41836187, 0.0, 0.90990615) ***Agent-ID1 Global Position:*** (138.1249, 4.399999, 28.664574) ***Agent-ID1 Viewing Direction:***	The global position of Agent-ID1 got closer to global position of Agent-ID2 in a sequence of 12 frames on X and Z axis between frame 64 and frame 76. The viewing direction vectors indicate that Agent-ID1 was walking towards Agent-ID2 until they met and formed a pair at frame 76 while Agent-ID0 remained far away from both of them. In a sequence of 12 frames, between frames 654 and 663, Agent-ID0 was walking towards Agent-ID1 and Agent-ID2 until he joined them (as indicated by continuous change in x and z components values of Agent-ID0 global position). A previously formed pair by Agent-ID1 and Agent-ID2 was transformed into a group with the same ID because the global position of Agent-ID0 got closer to global position of both – Agent-ID1 and Agent-ID2.	76 :: agent-ID1 and agent-ID2 FORMS A pair-82920 **663 :: agent-ID0 and pair-82920 FORMS A group-82920**	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

	(-0.9908853, 0.0, 0.14526859) ***Agent-ID2 Global Position:*** (109.63152, 4.399998, 41.971382) ***Agent-ID2 Viewing Direction:*** (-0.51210994, 0.0,-0.86063784) **Frame 663:** ***Agent-ID0 Global Position:*** (99.0117, 4.398138, 2.8666258) ***Agent-ID0 Viewing Direction:*** (0.41836187, 0.0, 0.90990615) ***Agent-ID1 Global Position:*** (138.1249, 4.399999, 28.664574) ***Agent-ID1 Viewing Direction:*** (-0.9908853, 0.0, 0.14526859) ***Agent-ID2 Global Position:*** (109.63152, 4.399998, 41.971382) ***Agent-ID2 Viewing Direction:*** (-0.51210994, 0.0, -0.86063784) **Frame 808:** ***Agent-ID0 Global Position:*** (106.53381, 4.3981676, 19.226778) ***Agent-ID0 Viewing Direction:*** (0.41836187, 0.0, 0.90990615) ***Agent-ID1 Global Position:*** (138.1249, 4.399999, 28.664574) ***Agent-ID1 Viewing Direction:*** (-0.9908853, 0.0, 0.14526859) ***Agent-ID2 Global Position:*** (109.63152, 4.399998, 41.971382) ***Agent-ID2 Viewing Direction:*** (-0.51210994, 0.0, -0.86063784)			
Label: *"Group moving towards something"* **Video File:** 008.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 1466 Frames	**Frame 0:** ***Bookshelf_ID2 Global Position:*** (191.95218, 4.4000015, -16.40506) ***Bookshelf_ID2 Global (Volume Boundary Point):*** (191.95218, 4.4000015, 7.7467536) ***Bookshelf_ID2 Global Position (Volume Boundary Point):*** (191.95218, 4.4000015, -40.556873) ***Agent-ID0 Global Position:*** (-78.60532, 4.3999987, -61.354675) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:***	Agent-ID0, Agent-ID1 and Agent-ID2 formed a group at frame 699. In a sequence of 20 frames between frame 727 and 747, the global position of Agent-ID0 and Agent-ID1 got closer to global position of Bookshelf_ID2 while Agent-ID2, although he remained in the same global position, his viewing direction vector changed direction and was directed at Bookshelf_ID2 at frame 747. In addition, the viewing direction vectors of all members in a group were mutually and approximately directed at Bookshelf_ID2 at frame 747.	699 :: agent-ID0 and pair-321282 FORMS A group-321282 745 :: participant-agent-ID0 MOVES TOWARDS static-Bookshelf_ID2 **747 :: group-321282 MOVES TOWARDS static-Bookshelf_ID2**	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

	(-0.87773573, 0.0, 0.48221964) ***Agent-ID2 Global Position:*** (94.19113, 4.3999987, 41.879482) ***Agent-ID2 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 727:** ***Agent-ID0 Global Position:*** (119.801155, 4.3943334, -7.2493052) ***Agent-ID0 Viewing Direction:*** (0.89841163, 0.0, 0.44250482) ***Agent-ID1 Global Position:*** (150.96687, 4.3999996, 41.70991) ***Agent-ID1 Viewing Direction:*** (0.42690766, 0.0, -0.90592736) ***Agent-ID2 Global Position:*** (132.19481, 4.3997817, 27.882788) ***Agent-ID2 Viewing Direction:*** (0.9811067, 0.0, -0.20096081) **Frame 747:** ***Agent-ID0 Global Position:*** (127.71792, 4.398066, -7.731458) ***Agent-ID0 Viewing Direction:*** (0.98695326, 0.0, -0.98695326) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID2 Global Position:*** (132.19481, 4.3997817, 27.882788) ***Agent-ID2 Viewing Direction:*** (0.59290737, 0.0, -0.80710375) **Frame 1466:** ***Agent-ID0 Global Position:*** (152.01909, 4.399999, -16.2053) ***Agent-ID0 Viewing Direction:*** (0.9920849, 0.0, 0.13682014) ***Agent-ID1 Global Position:*** (171.03581, 4.399999, 12.165615) ***Agent-ID1 Viewing Direction:*** (-0.9451574, 0.0, -0.33110297) ***Agent-ID2 Global Position:*** (126.99999, 4.399996, 30.33306) ***Agent-ID2 Viewing Direction:*** (-0.41366825, 0.0, 0.91204894)			

Label: *"Group moving along something"* **Video File:** 008.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 1466 Frames	**Frame 0:** ***Bookshelf_ID2 Global Position:*** (191.95218, 4.4000015, -16.40506) ***Bookshelf_ID2 Global (Volume Boundary Point):*** (191.95218, 4.4000015, 7.7467536) ***Bookshelf_ID2 Global Position (Volume Boundary Point):*** (191.95218, 4.4000015, -40.556873) ***Agent-ID0 Global Position:*** (-78.60532, 4.3999987, -61.354675) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID2 Global Position:*** (94.19113, 4.3999987, 41.879482) ***Agent-ID2 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 1000:** ***Agent-ID0 Global Position:*** (154.01247, 4.3956547, -31.826445) ***Agent-ID0 Viewing Direction:*** (-0.27433953, 0.0, -0.96316725) ***Agent-ID1 Global Position:*** (148.03925, 4.3954268, 8.053694) ***Agent-ID1 Viewing Direction:*** (-0.65464604, 0.0, -0.7578869) ***Agent-ID2 Global Position:*** (133.065, 4.398327, 2.7674255) ***Agent-ID2 Viewing Direction:*** (-0.978954, 0.0, -0.21119836) **Frame 1013:** ***Agent-ID0 Global Position:*** (152.5729, 4.392029, -36.88059) ***Agent-ID0 Viewing Direction:*** (-0.27433953, 0.0, -0.96316725) ***Agent-ID1 Global Position:*** (148.42871, 4.392576, 3.6613858) ***Agent-ID1 Viewing Direction:*** (0.5183277, 0.0, -0.8569069) ***Agent-ID2 Global Position:*** (132.23915, 4.3984885, 2.25052) ***Agent-ID2 Viewing Direction:*** (-0.978954, 0.0, -0.21119836)	Agent-ID0, Agent-ID1 and Agent-ID2 formed a group at frame 699. While moving around the visual scene, in a sequence of 13 frames, between frame 1000 and 1013, all members of the group-321282 travelled along Bookshelf_ID2 (as indicated by their global position and the boundary points of the Bookshelf_ID2). The viewing direction vector of Agent-ID0, Agent-ID1 and Agent-ID2 hints the approximate mutual direction of movement of the whole group.	699 :: agent-ID0 and pair-321282 FORMS A group-321282 [...] **1013 :: group-321282 MOVES ALONG static-Bookshelf_ID2**	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

	Frame 1466: ***Agent-ID0 Global Position:*** (152.01909, 4.399999, -16.2053) ***Agent-ID0 Viewing Direction:*** (0.9920849, 0.0, 0.13682014) ***Agent-ID1 Global Position:*** (171.03581, 4.399999, 12.165615) ***Agent-ID1 Viewing Direction:*** (-0.9451574, 0.0, -0.33110297) ***Agent-ID2 Global Position:*** (126.99999, 4.399996, 30.33306) ***Agent-ID2 Viewing Direction:*** (-0.41366825, 0.0, 0.91204894)			
Label: *"Leaving a group / Group disbands"* **Video File:** 004.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 856 Frames	**Frame 0:** ***Agent-ID0 Global Position:*** (161.19113, 4.3999987, 51.879482) ***Agent-ID0 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) ***Agent-ID1 Global Position:*** (161.19113, 4.3999987, 21.879484) ***Agent-ID1 Viewing Direction:*** (-0.87773573, 0.0, 0.48221964) **Frame 211:** ***Agent-ID0 Global Position:*** (161.06638, 4.392574, 77.39287) ***Agent-ID0 Viewing Direction:*** (-0.2552031, 0.0, 0.9684146) ***Agent-ID1 Global Position:*** (131.9717, 4.3967886, -4.8048387) ***Agent-ID1 Viewing Direction:*** (-0.5527537, 0.0, -0.835116) **Frame 856:** ***Agent-ID0 Global Position:*** (156.2939, 4.3999643, 92.23968) ***Agent-ID0 Viewing Direction:*** (0.4520256, 0.0, 0.8936589) ***Agent-ID1 Global Position:*** (107.31645, 4.3997803, -5.176361) ***Agent-ID1 Viewing Direction:*** (-0.36949524, 0.0, -0.9308196)	The distance between global position of Agent-ID0 and global position of Agent-ID1 (who were in a group "pair-843069") exceeded the proximity threshold parameter for a leaving distance (**groupFormProximityLeavingDistance** set to 70 at the time) at frame 211 and it is reflected in the XML Video File.	0 :: agent-ID0 and agent-ID1 FORMS A pair-843069 **211 :: pair-843069 DISBANDS**	The output of the analyser corresponds to what is in the input XML file and what the simulator visualizes.

Every pattern included in table 6-1 has been tested in a similar manner in different scenarios recorded in XML files. The summary of the results of this experiment are presented in Table 6-2. The movies were replayed at their original frame rate (30 FPS). The "N/A" label in the table indicates that no such pattern was recorded in the movie so pattern analyser was not able to register it. The numbers highlight the frame of the video at which the pattern has been recognized. The "Match" indicates that the spatial information encapsulated in the video file corresponds to what has been output.

Table 6-2. Summary of the pattern validation in different scenarios.

File	001.xml	002.xml	003.xml	004.xml	005.xml	006.xml	007.xml	008.xml	009.xml	010.xml	011.xml	012.xml
"Walking towards something"	23 frame Match	468 frame Match	N/A	243 frame Match	6 Frame Match	4 frame Match	17 frame Match	44 frame Match	373 frame Match	75 frame Match	56 frame Match	874 frame Match
"Walking towards something" (while agent is in a group)	N/A	144 frame Match	45 frame Match	191 frame Match	N/A	337 frame Match	N/A	804 frame Match	105 frame Match	N/A	922 frame Match	583 frame Match
"Walking away from something"	105 frame Match	330 frame Match	N/A	493 frame Match	763 frame Match	N/A	61 frame Match	115 frame Match	260 frame Match	567 frame Match	157 frame Match	921 frame Match
"Walking along something" (while agent is in a group)	N/A	N/A	133 frame Match	148 frame Match	N/A	35 frame Match	N/A	939 frame Match	23 frame Match	N/A	850 frame Match	501 frame Match
"Climbing something up"	N/A	N/A	N/A	N/A	214 frame Match	N/A	N/A	N/A	338 frame Match	N/A	282 frame Match	N/A
"Forming a pair"	N/A	9 frame Match	1 frame Match	697 frame Match	N/A	72 frame Match	128 frame Match	102 frame Match	555 frame Match	N/A	1274 frame Match	1130 frame Match
"Forming a group"	N/A	N/A	N/A	N/A	N/A	663 frame Match	223 frame Match	699 frame Match	89 frame Match	N/A	1795 frame Match	N/A
"Group moving towards something"	N/A	N/A	N/A	N/A	N/A	N/A	N/A	747 frame Match	477 frame Match	N/A	N/A	368 frame Match
"Group moving along something"	N/A	213 frame Match	91 frame Match	N/A	N/A	N/A	N/A	1013 frame Match	545 frame Match	N/A	N/A	105 frame Match
"Leaving a group / Group disbands"	N/A	233 frame Match	160 frame Match	211 frame Match	N/A	N/A	256 frame Match	414 frame Match	480 frame Match	N/A	2004 frame Match	809 frame Match

6.2. Analysis of the speed of replaying the xml file at different frame rates on the pattern recognition process

The movies were replayed at different frame rates in order to determine how this factor influence the operation of the dynamic pattern analyser. The tests have been conducted to check if any of the patterns would become unaccounted for depending on the frame rate movies have been played at. These tests also helped establishing whether fluctuations in frame rate values have direct impact on the pattern recognition process carried out by pattern analyser. Frame rates may be related to computational power of the machine the application is run at. Lower frame rates are expected during the operation of simulator on machines with less computational resources.

Table 6-3. The results of replaying XML videos at different frame rates.

Pattern / Video Specification	Replaying fps rate	Recognized?	Time difference	Pattern recognition log (movie frame)
Label: *"Walking towards something"* **Pattern Recognized at Frame (replaying at 30fps):** 23 **Video File:** 001.xml **Recording Frame Rate:** 30fps **Video Duration:** 281 Frames	5fps	yes	10.56%	18 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	10fps	yes	3.38%	21 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	15fps	yes	0.93%	23 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	20fps	yes	0.95%	25 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	25fps	yes	0.96%	28 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	30fps	yes	0.44%	23 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	60fps	yes	4.63%	23 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
Label: *"Walking towards something" (while agent is in a group)* **Pattern Recognized at Frame (replaying at 30fps):** 45 **Video File:** 003.xml **Recording Frame Rate:** 30fps **Video Duration:** 264 Frames	5fps	yes	16.91%	35 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	10fps	yes	7.03%	39 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	15fps	yes	4.35%	53 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	20fps	yes	1.08%	47 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	25fps	yes	0.27%	47 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	30fps	yes	0.39%	45 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2

	60fps	yes	1.64%	57 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
Label: *"Walking away from something"* **Pattern Recognized at Frame (replaying at 30fps):** 105 **Video File:** 001.xml **Recording Frame Rate:** 30fps **Video Duration:** 281 Frames	5fps	yes	27.92%	104 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	10fps	yes	16.32%	105 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	15fps	yes	10.64%	136 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	20fps	yes	6.05%	148 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	25fps	yes	2.87%	168 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	30fps	yes	0.31%	105 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	60fps	yes	0.14%	173 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
Label: *"Walking along something" (while agent is in a group)* **Pattern Recognized at Frame (replaying at 30fps):** 133 **Video File:** 003.xml **Recording Frame Rate:** 30fps **Video Duration:** 264 Frames	5fps	yes	28.75%	135 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	10fps	yes	16.92%	134 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	15fps	yes	9.83%	135 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	20fps	yes	5.57%	147 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	25fps	yes	2.25%	139 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	30fps	yes	0.27%	133 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	60fps	yes	0.16%	152 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
Label: *"Climbing something up"* **Pattern Recognized at Frame (replaying at 30fps):** 214 **Video File:** 005.xml **Recording Frame Rate:** 30fps **Video Duration:** 1128 Frames	5fps	yes	30.04%	178 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	10fps	yes	18.74%	194 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	15fps	yes	11.57%	210 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	20fps	yes	6.44%	211 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	25fps	yes	2.91%	225 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	30fps	yes	0.22%	214 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	60fps	yes	0.08%	229 :: agent-ID0 CLIMBS static-Stairs_ID1 UP

Label: *"Forming a pair"* **Pattern Recognized at Frame (replaying at 30fps):** 442 **Video File:** 004.xml **Recording Frame Rate:** 30fps **Video Duration:** 856 Frames	5fps	yes	33.35%	441 :: agent-ID0 and agent-ID1 FORMS A pair-784540
	10fps	yes	22.15%	450 :: agent-ID0 and agent-ID1 FORMS A pair-220853
	15fps	yes	14%	445 :: agent-ID0 and agent-ID1 FORMS A pair-225276
	20fps	yes	8.25%	463 :: agent-ID0 and agent-ID1 FORMS A pair-463337
	25fps	yes	3.67%	457 :: agent-ID0 and agent-ID1 FORMS A pair-174722
	30fps	yes	0.13%	442 :: agent-ID0 and agent-ID1 FORMS A pair-931849
	60fps	yes	0.09%	437 :: agent-ID0 and agent-ID1 FORMS A pair-422328
Label: *"Forming a group"* **Pattern Recognized at Frame (replaying at 30fps):** 663 **Video File:** 006.xml **Recording Frame Rate:** 30fps **Video Duration:** 808 Frames	5fps	yes	34.01%	641 :: agent-ID0 and pair-967058 FORMS A group-967058
	10fps	yes	22.9%	643 :: agent-ID0 and pair-381274 FORMS A group-381274
	15fps	yes	14.81%	650 :: agent-ID0 and pair-869367 FORMS A group-869367
	20fps	yes	8.63%	647 :: agent-ID0 and pair-193866 FORMS A group-193866
	25fps	yes	3.88%	654 :: agent-ID0 and pair-790242 FORMS A group-790242
	30fps	yes	0.72%	663 :: agent-ID0 and pair-82920 FORMS A group-82920
	60fps	yes	0.52%	674 :: agent-ID0 and pair-284956 FORMS A group-284956
Label: *"Group moving towards something"* **Pattern Recognized at Frame (replaying at 30fps):** 747 **Video File:** 008.xml **Recording Frame Rate:** 30fps **Video Duration:** 1466 Frames	5fps	yes	34.24%	741 :: group-265393 MOVES TOWARDS static-Bookshelf_ID2
	10fps	yes	23.15%	743 :: group-115637 MOVES TOWARDS static-Bookshelf_ID2
	15fps	yes	15.01%	744 :: group-613447 MOVES TOWARDS static-Bookshelf_ID2
	20fps	yes	8.77%	740 :: group-954204 MOVES TOWARDS static-Bookshelf_ID2
	25fps	yes	4.84%	762 :: group-237177 MOVES TOWARDS static-Bookshelf_ID2
	30fps	yes	4.5%	747 :: group-321282 MOVES TOWARDS static-Bookshelf_ID2
	60fps	yes	4.91%	761 :: group-336909 MOVES TOWARDS static-Bookshelf_ID2
Label: *"Group moving along*	5fps	yes	34.63%	1013 :: group-265393 MOVES ALONG static-Bookshelf_ID2

something" **Pattern Recognized at Frame (replaying at 30fps):** 1013 **Video File:** 008.xml **Recording Frame Rate:** 30fps **Video Duration:** 1466 Frames	10fps	yes	23.62%	1013 :: group-115637 MOVES ALONG static-Bookshelf_ID2
	15fps	yes	15.43%	1013 :: group-613447 MOVES ALONG static-Bookshelf_ID2
	20fps	yes	9.09%	1016 :: group-954204 MOVES ALONG static-Bookshelf_ID2
	25fps	yes	6.06%	1013 :: group-237177 MOVES ALONG static-Bookshelf_ID2
	30fps	yes	5.81%	1013 :: group-321282 MOVES ALONG static-Bookshelf_ID2
	60fps	yes	6.07%	1013 :: group-336909 MOVES ALONG static-Bookshelf_ID2
Label: *"Leaving a group / Group disbands"* **Pattern Recognized at Frame (replaying at 30fps):** 211 **Video File:** 004.xml **Recording Frame Rate:** 30fps **Video Duration:** 856 Frames	5fps	yes	30.55%	189 :: pair-932176 DISBANDS
	10fps	yes	18.88%	189 :: pair-73227 DISBANDS
	15fps	yes	11.71%	205 :: pair-206910 DISBANDS
	20fps	yes	6.63%	214 :: pair-532756 DISBANDS
	25fps	yes	2.95%	223 :: pair-883997 DISBANDS
	30fps	yes	0.25%	211 :: pair-843069 DISBANDS
	60fps	yes	0.14%	207 :: pair-51170 DISBANDS

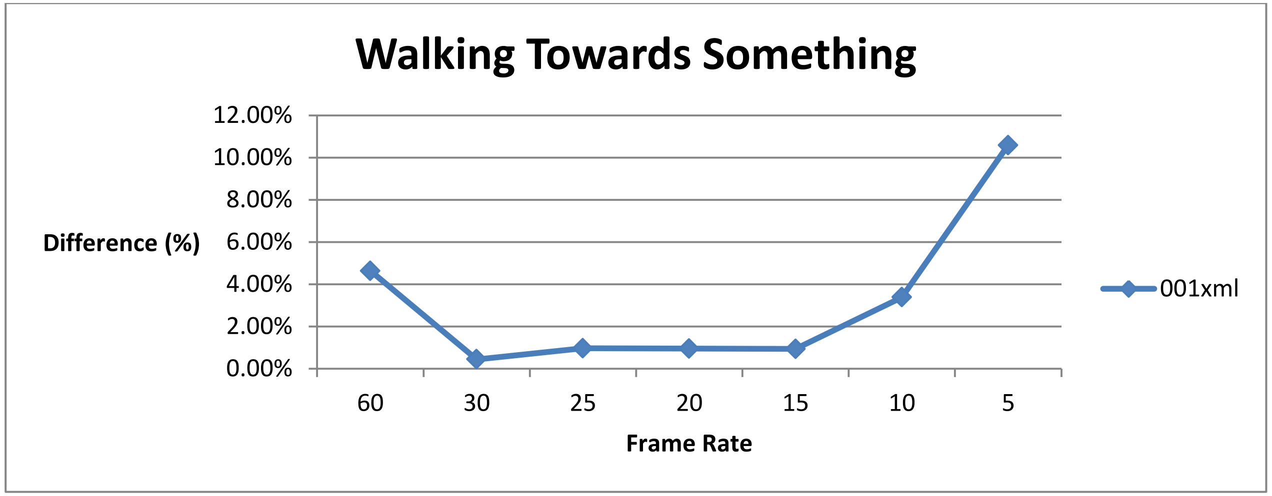


Figure 6-1. The errors calculated by simulator during the recognition of *"walking towards something"* pattern from the movie replayed at different frame rates.

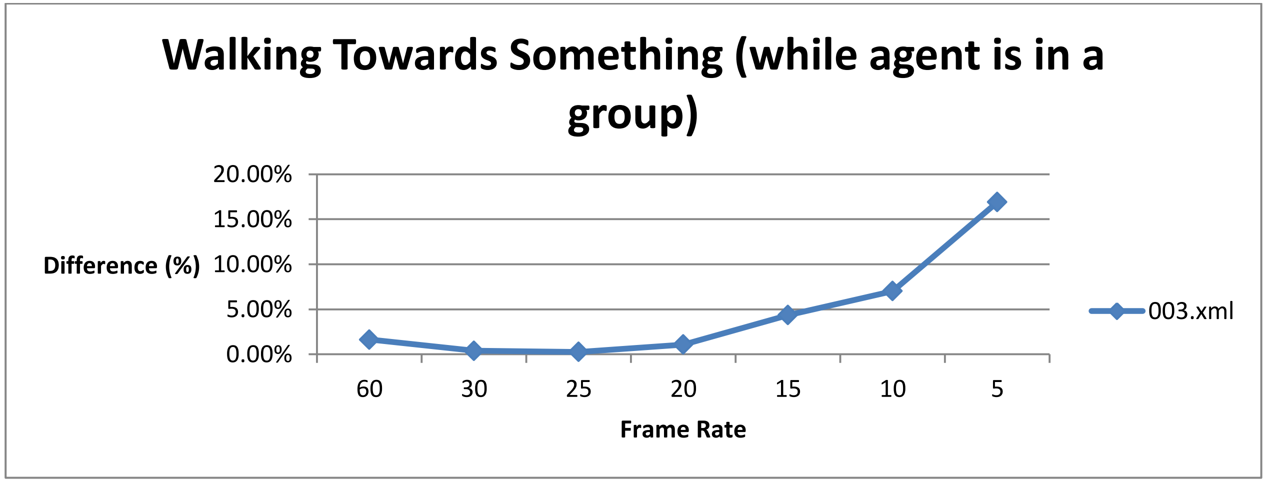


Figure 6-2. The errors calculated by simulator during the recognition of *"walking towards something (while agent is in a group)"* pattern from the movie replayed at different frame rates.

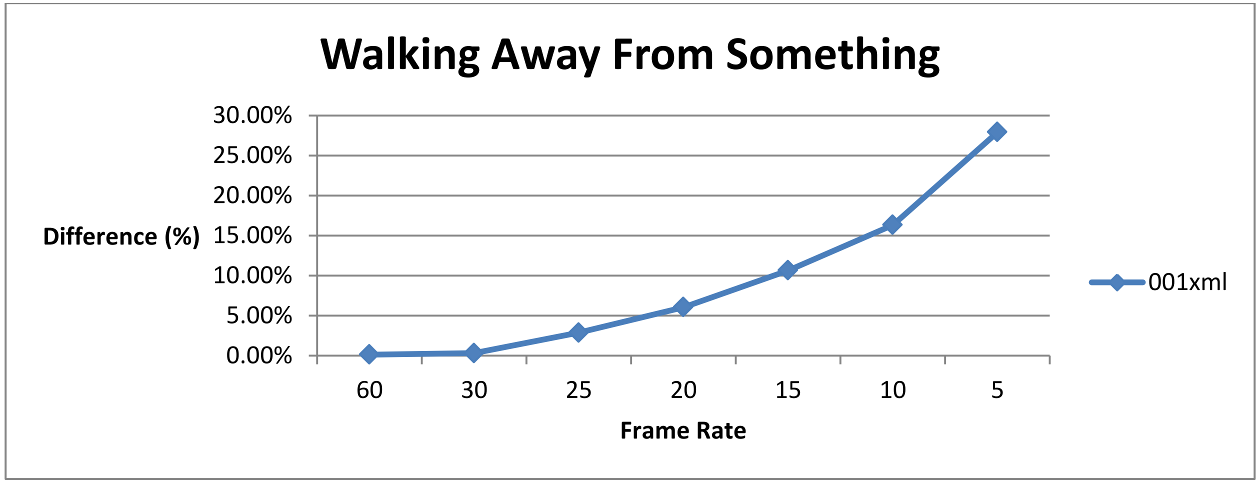


Figure 6-3. The errors calculated by simulator during the recognition of *"walking away from something"* pattern from the movie replayed at different frame rates.

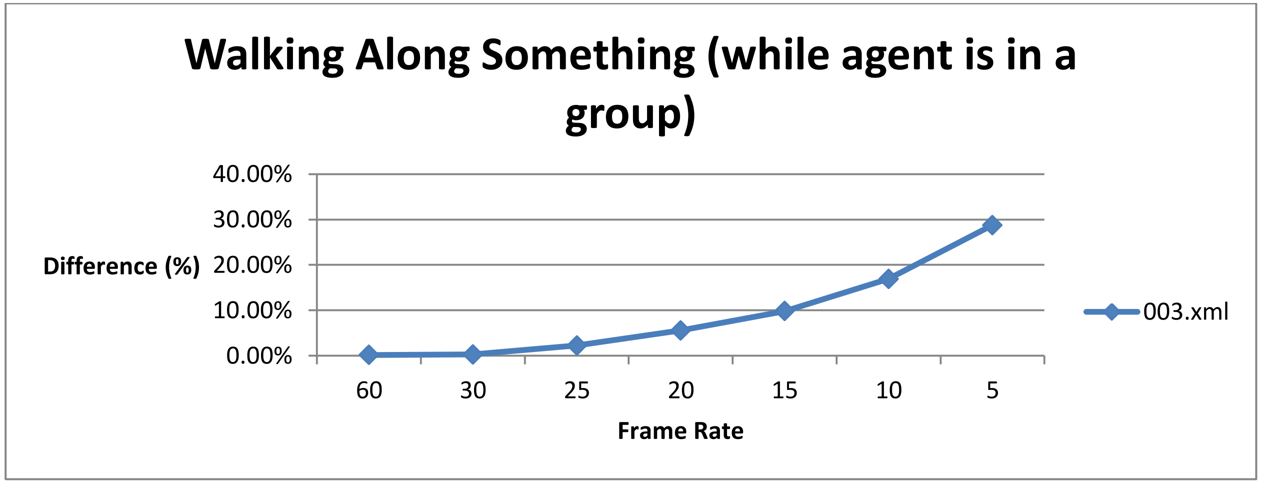


Figure 6-4. The errors calculated by simulator during the recognition of *"walking along something (while agent is in a group)"* pattern from the movie replayed at different frame rates.

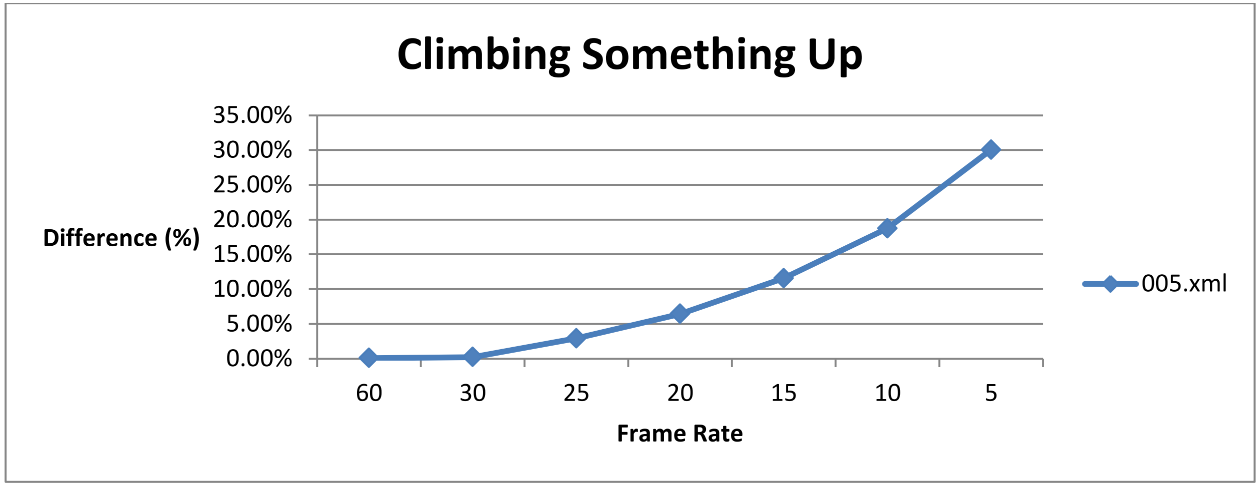


Figure 6-5. The errors calculated by simulator during the recognition of *“climbing something up”* pattern from the movie replayed at different frame rates.

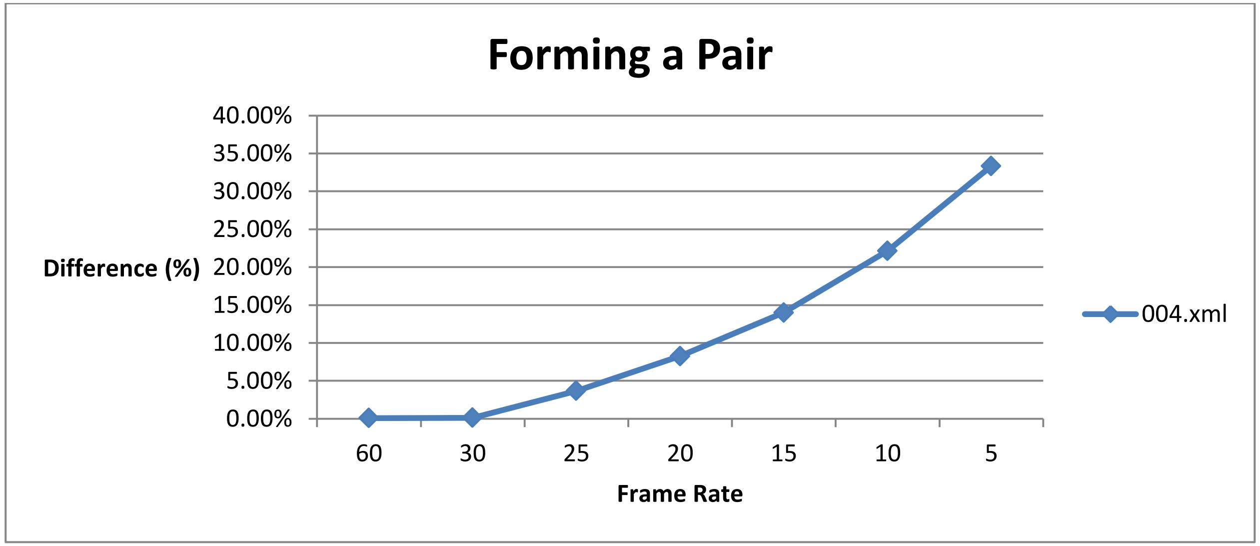


Figure 6-6. The errors calculated by simulator during the recognition of *“forming a pair”* pattern from the movie replayed at different frame rates.

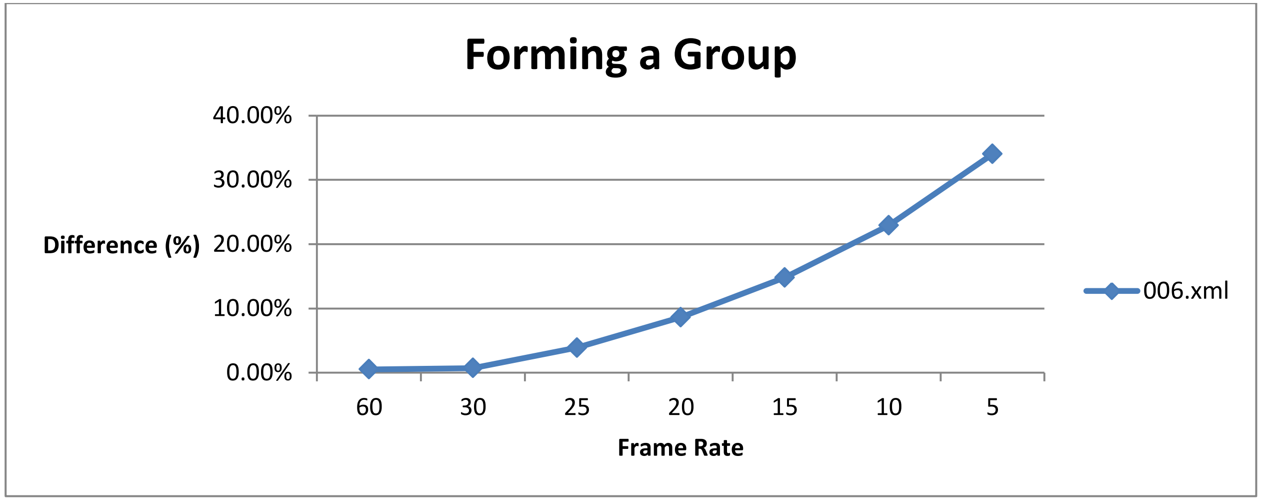


Figure 6-7. The errors calculated by simulator during the recognition of *"forming a group"* pattern from the movie replayed at different frame rates.

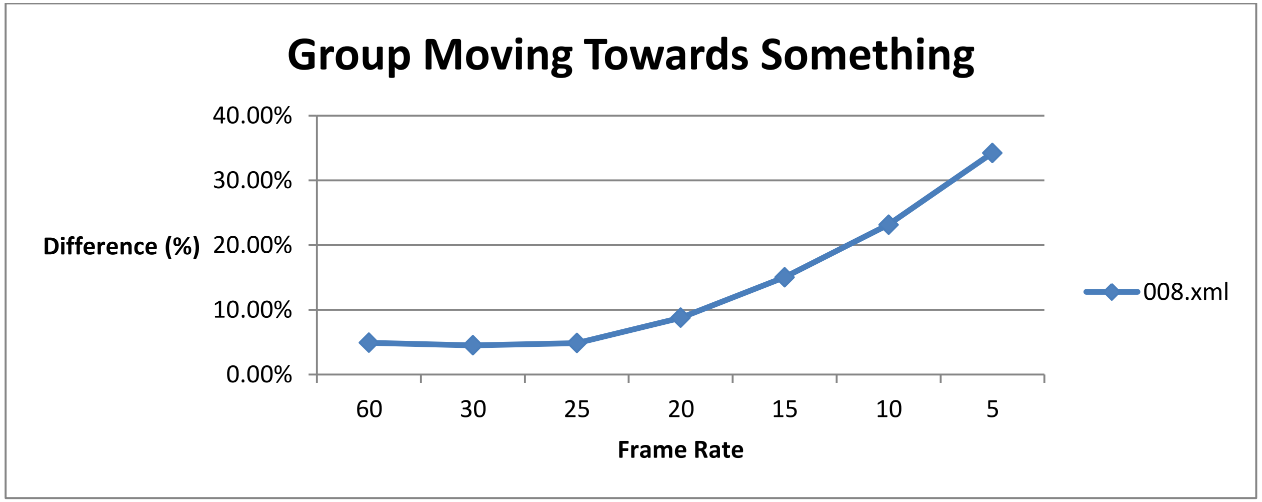


Figure 6-8. The errors calculated by simulator during the recognition of *"group moving towards something"* pattern from the movie replayed at different frame rates.

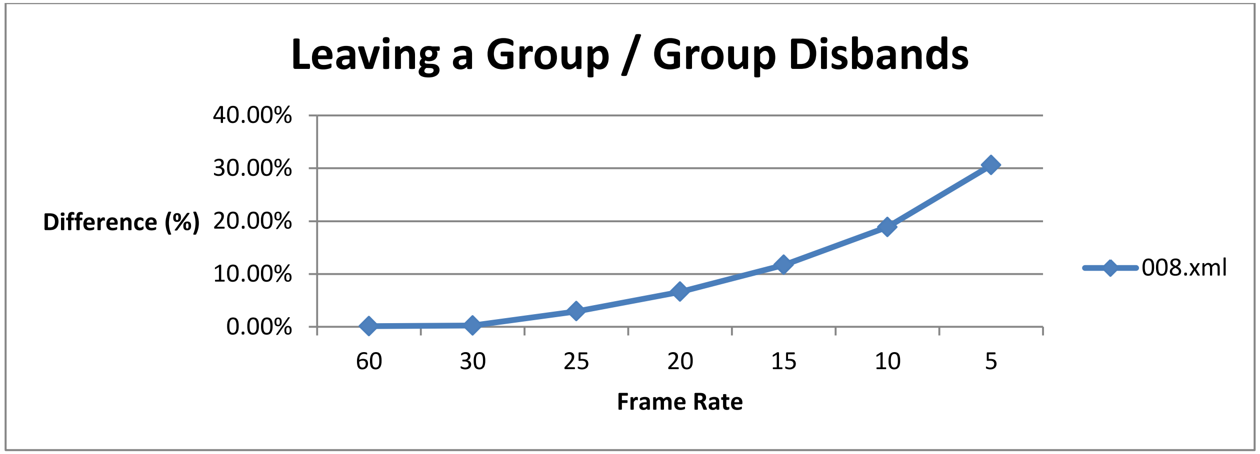


Figure 6-9. The errors calculated by simulator during the recognition of *"leaving a group / group disbands"* pattern from the movie replayed at different frame rates.

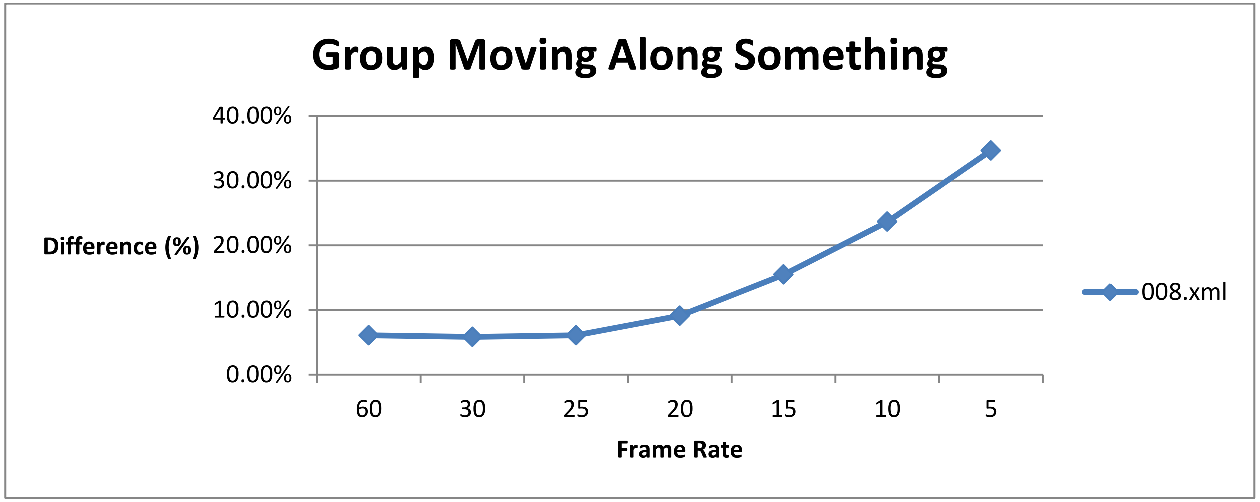


Figure 6-10. The errors calculated by simulator during the recognition of *"group moving along something"* pattern from the movie replayed at different frame rates.

Analysis of the experimental results

The results show that although the patterns were always recognized, they were reported with bigger delays when movies were replayed at lower frame rates than recorded. The error expressed in time difference (in percentage) between the time stamp recorded in the video xml file at the frame, at which the pattern was recognized and simulation clock time shows how much delay was made in reporting a pattern while playing a movie at given frame rate. The closer score to 0% the more instantaneous recognition of the pattern is. From Table 6-3 and Figures 6-2 – 6-10, one can observe that errors linearly increase as the frame rate drops.

6.3. Influence of the events of skipping frames while replaying the video xml file on pattern recognition process

The movies were replayed with skipped frames in order to determine how this factor influence the operation of the dynamic pattern analyser. The tests have been conducted to check if any of the patterns would become unaccounted for depending on the percentage of frames being dropped from the video file during the simulation. The percentage values in "Skipped frames (%)" column in Table 6-4 indicate how many frames were dropped from every 1 sec of the movie. The "Time Difference" column show how much was the delay in reporting the pattern. Dropped frames may be related to computational power of the machine the application is run at or the incorrect output generated by external module suppling the simulator with information about visual scene.

Table 6-4. The results of skipping frames while replaying XML video file.

Pattern / Video Specification	**Skipped frames (%)**	**Recognized?**	**Time difference**	**Pattern recognition log (movie frame)**
Label: *"Walking towards something"* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 23 **Video File:** 001.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 281 Frames	50%	yes	0.22%	30 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	66%	yes	0.38%	33 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	76%	yes	0.50%	29 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	83%	yes	0.41%	35 :: agent-ID0 MOVES TOWARDS static-Counter_ID4
	90%	yes	0.41%	40 :: agent-ID0 MOVES TOWARDS static-Counter_ID4

Label: *“Walking towards something” (while agent is in a group)* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 45 **Video File:** 003.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 264 Frames	50%	yes	0.45%	56 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	66%	yes	1.82%	45 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	76%	yes	8.37%	54 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	83%	yes	7.59%	62 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
	90%	yes	9.4%	61 :: participant-agent-ID1 MOVES TOWARDS static-Bookshelf_ID2
Label: *“Walking away from something”* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 105 **Video File:** 001.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 281 Frames	50%	yes	0.26%	138 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	66%	yes	0.38%	144 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	76%	yes	0.42%	138 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	83%	yes	0.35%	144 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
	90%	yes	0.37%	150 :: agent-ID0 MOVES AWAY FROM static-Counter_ID4
Label: *“Walking along something” (while agent is in a group)* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 133 **Video File:** 003.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 264 Frames	50%	yes	0.07%	76 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	66%	yes	3.37%	79 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	76%	yes	1.47%	83 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	83%	yes	2.34%	78 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	90%	yes	4.48%	76 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
Label: *“Climbing something up”* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 214 **Video File:** 005.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 1128 Frames	50%	yes	0.30%	239 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	66%	yes	0.29%	239 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	76%	yes	0.22%	240 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
	83%	yes	0.23%	240 :: agent-ID0 CLIMBS static-Stairs_ID1 UP

	90%	yes	0.27%	246 :: agent-ID0 CLIMBS static-Stairs_ID1 UP
Label: *"Forming a pair"* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 442 **Video File:** 004.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 856 Frames	50%	yes	0.12%	441 :: agent-ID0 and agent-ID1 FORMS A pair-784540
	66%	yes	0.21%	439 :: agent-ID0 and agent-ID1 FORMS A pair-272499
	76%	yes	0.13%	442 :: agent-ID0 and agent-ID1 FORMS A pair-248187
	83%	yes	0.11%	440 :: agent-ID0 and agent-ID1 FORMS A pair-411900
	90%	yes	0.14%	485 :: agent-ID0 and agent-ID1 FORMS A pair-327859
Label: *"Forming a group"* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 663 **Video File:** 006.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 808 Frames	50%	yes	2.94%	666 :: agent-ID0 and pair-553430 FORMS A group-553430
	66%	yes	0.13%	669 :: agent-ID0 and pair-423484 FORMS A group-423484
	76%	yes	1.05%	642 :: agent-ID0 and pair-829015 FORMS A group-829015
	83%	yes	0.91%	647 :: agent-ID0 and pair-193252 FORMS A group-193252
	90%	yes	0.17%	671 :: agent-ID0 and pair-78999 FORMS A group-78999
Label: *"Group moving towards something"* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 747 **Video File:** 008.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 1466 Frames	50%	yes	0.07%	767 :: group-980969 MOVES TOWARDS static-Bookshelf_ID2
	66%	yes	0.09%	758 :: group-931738 MOVES TOWARDS static-Bookshelf_ID2
	76%	yes	0.06%	751 :: group-813781 MOVES TOWARDS static-Bookshelf_ID2
	83%	yes	0.07%	768 :: group-250335 MOVES TOWARDS static-Bookshelf_ID2
	90%	yes	0.06%	760 :: group-993611 MOVES TOWARDS static-Bookshelf_ID2
Label: *"Group moving along something"* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 1013 **Video File:** 008.xml	50%	yes	0.12%	1253 :: group-980969 MOVES ALONG static-Bookshelf_ID2
	66%	yes	0.07%	1256 :: group-931738 MOVES ALONG static-Bookshelf_ID2
	76%	yes	0.04%	1256 :: group-813781 MOVES ALONG static-Bookshelf_ID2

Recording Frame Rate: 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 1466 Frames	83%	yes	0.04%	1232 :: participant-agent-ID1 MOVES ALONG static-Bookshelf_ID2
	90%	yes	0.04%	1257 :: group-993611 MOVES ALONG static-Bookshelf_ID2
Label: *"Leaving a group / Group disbands"* **Pattern Recognized at Frame (replaying at 30fps without skipping frames):** 211 **Video File:** 004.xml **Recording Frame Rate:** 30fps **Replaying Frame Rate:** 30fps **Video Duration:** 856 Frames	50%	yes	0.25%	211 :: pair-60931 DISBANDS
	66%	yes	0.43%	209 :: pair-937323 DISBANDS
	76%	yes	0.28%	209 :: pair-502407 DISBANDS
	83%	yes	0.22%	210 :: pair-5668 DISBANDS
	90%	yes	0.30%	209 :: pair-378774 DISBANDS

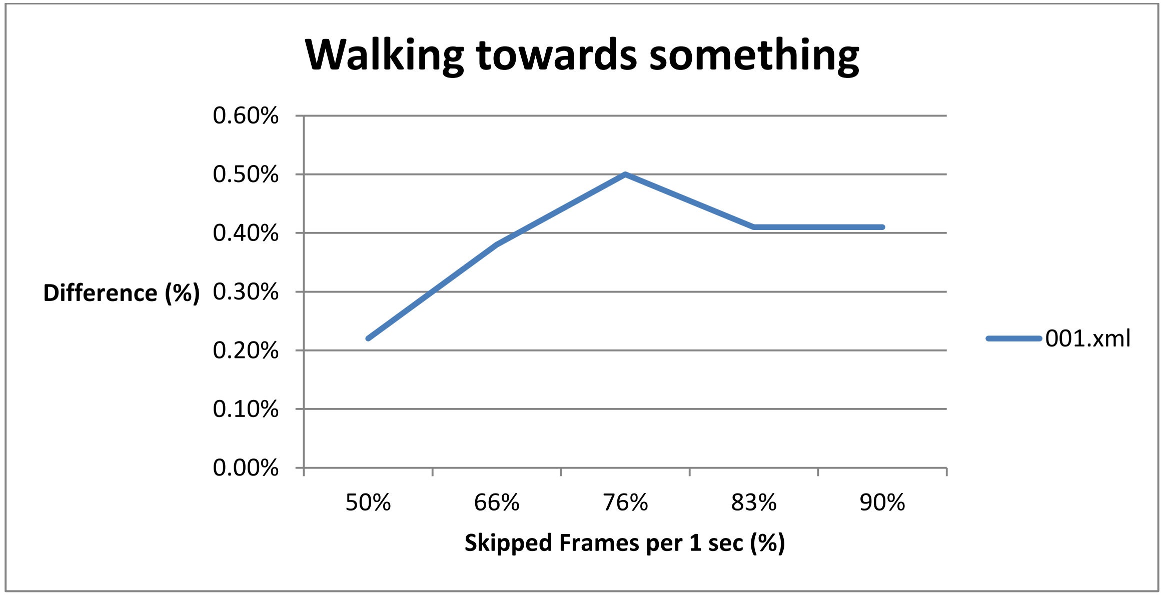


Figure 6-11. The errors calculated by simulator during the recognition of *"walking towards something"* pattern while skipping certain amount of frames from the movie.

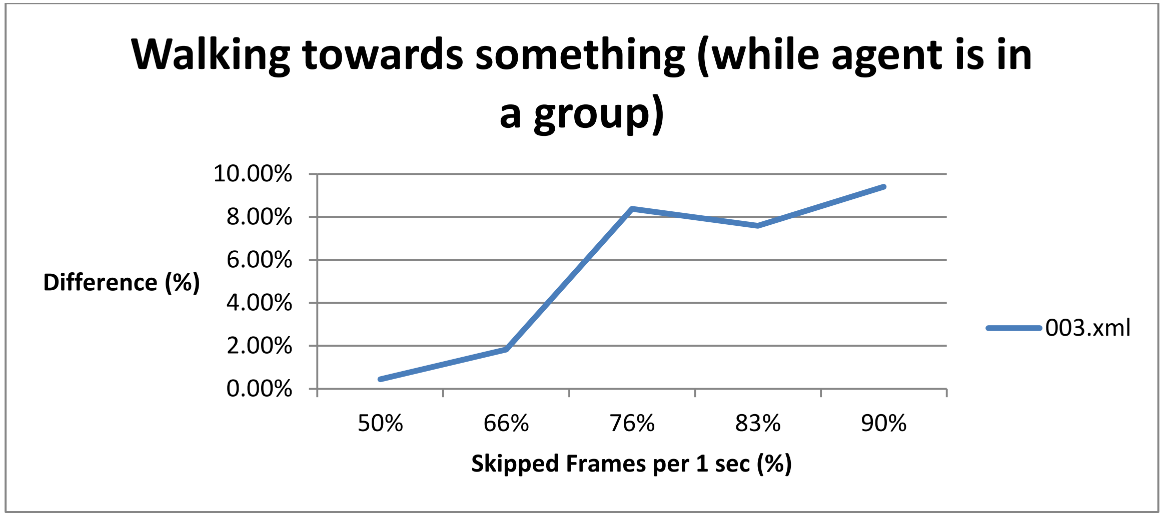


Figure 6-12. The errors calculated by simulator during the recognition of *"walking towards something (while agent is in a group)"* pattern while skipping certain amount of frames from the movie.

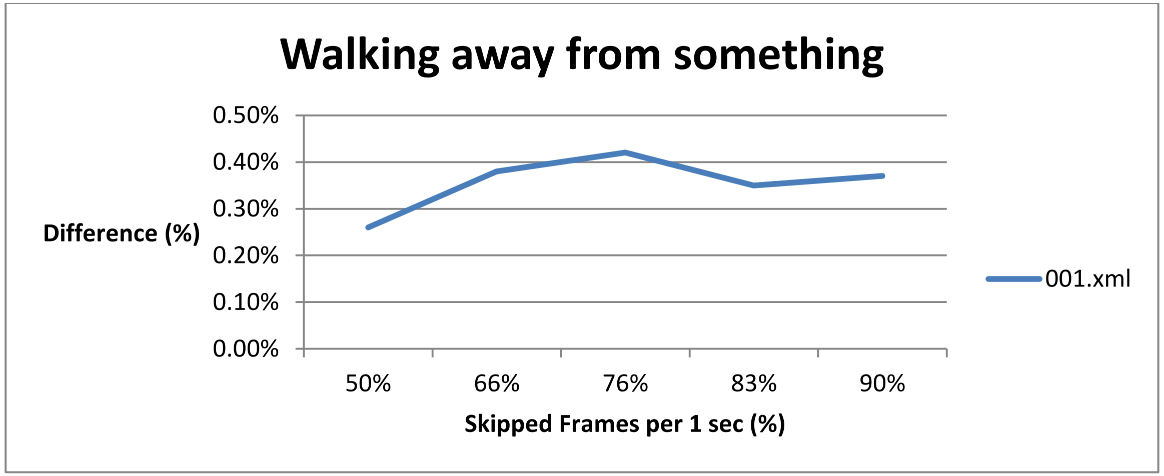


Figure 6-13. The errors calculated by simulator during the recognition of *"walking away from something"* pattern while skipping certain amount of frames from the movie.

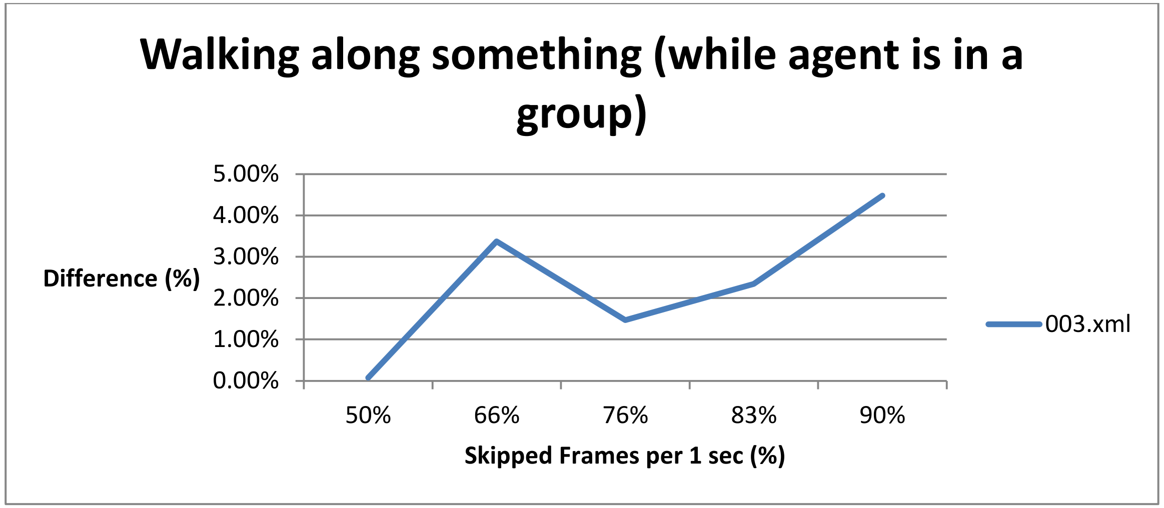


Figure 6-14. The errors calculated by simulator during the recognition of *"walking along something (while agent is in a group)"* pattern while skipping certain amount of frames from the movie.

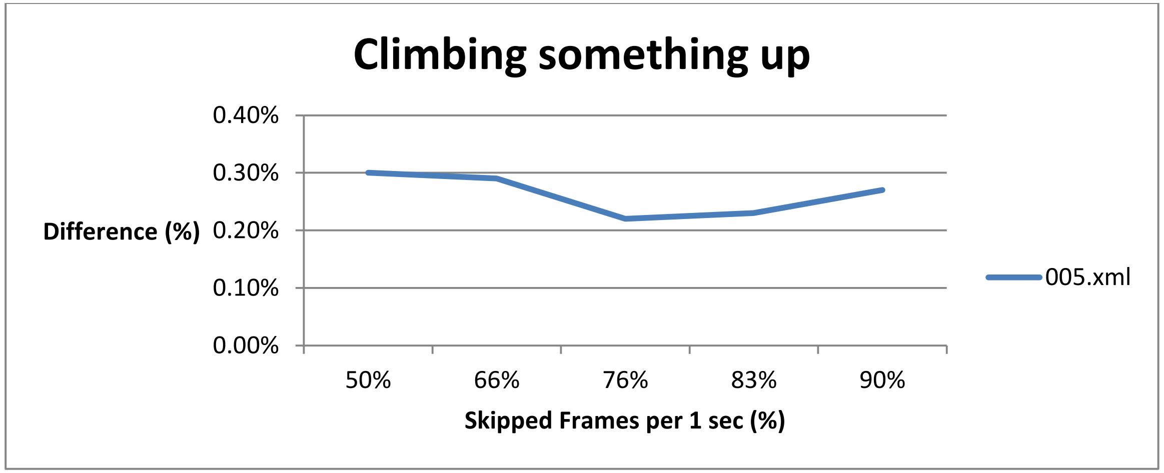


Figure 6-15. The errors calculated by simulator during the recognition of *"climbing something up"* pattern while skipping certain amount of frames from the movie.

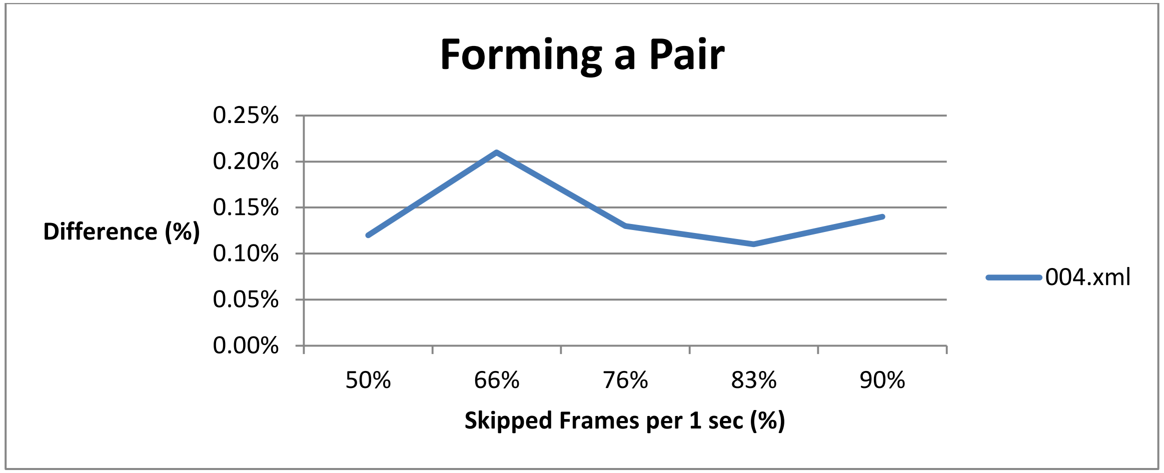


Figure 6-16. The errors calculated by simulator during the recognition of *"forming a pair"* pattern while skipping certain amount of frames from the movie.

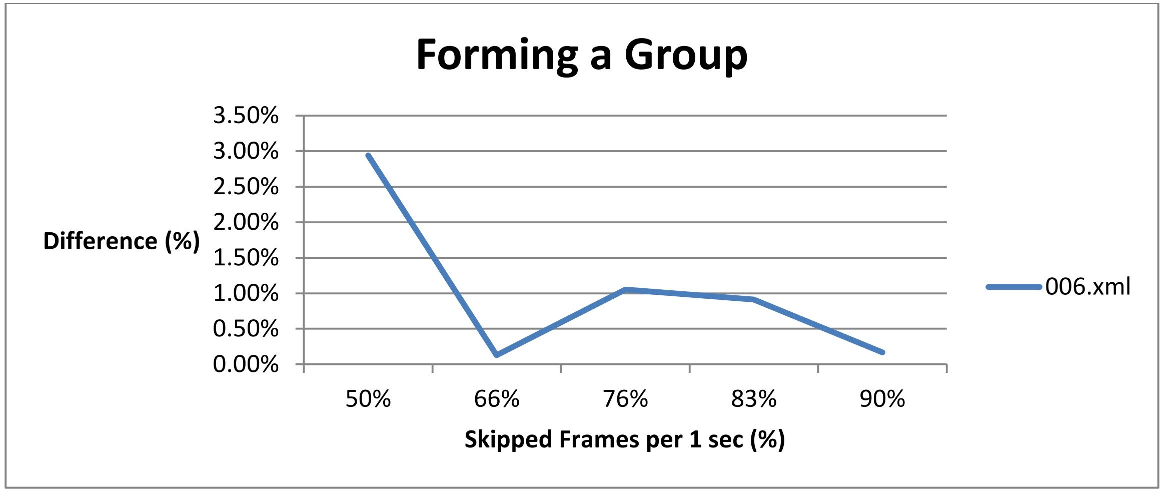


Figure 6-17. The errors calculated by simulator during the recognition of *"forming a group"* pattern while skipping certain amount of frames from the movie.

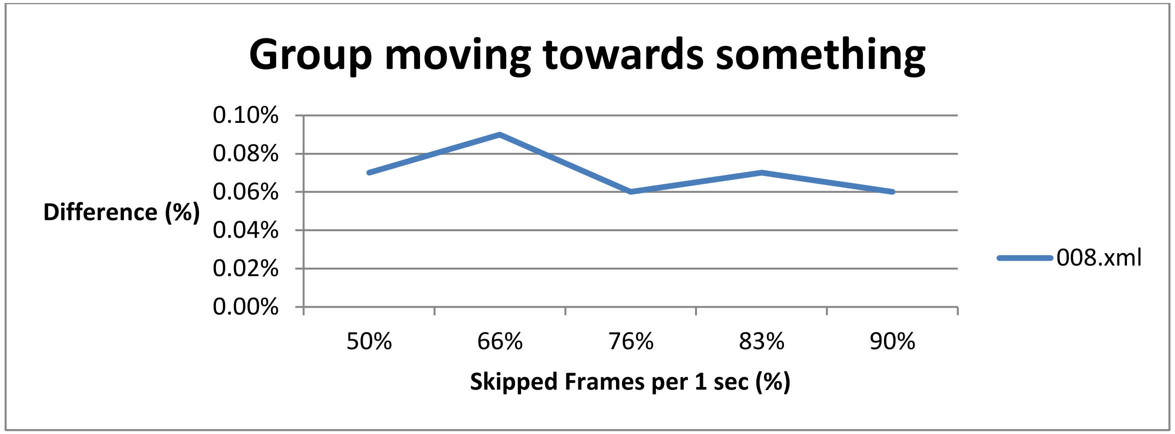


Figure 6-18. The errors calculated by simulator during the recognition of *"group walking towards something"* pattern while skipping certain amount of frames from the movie.

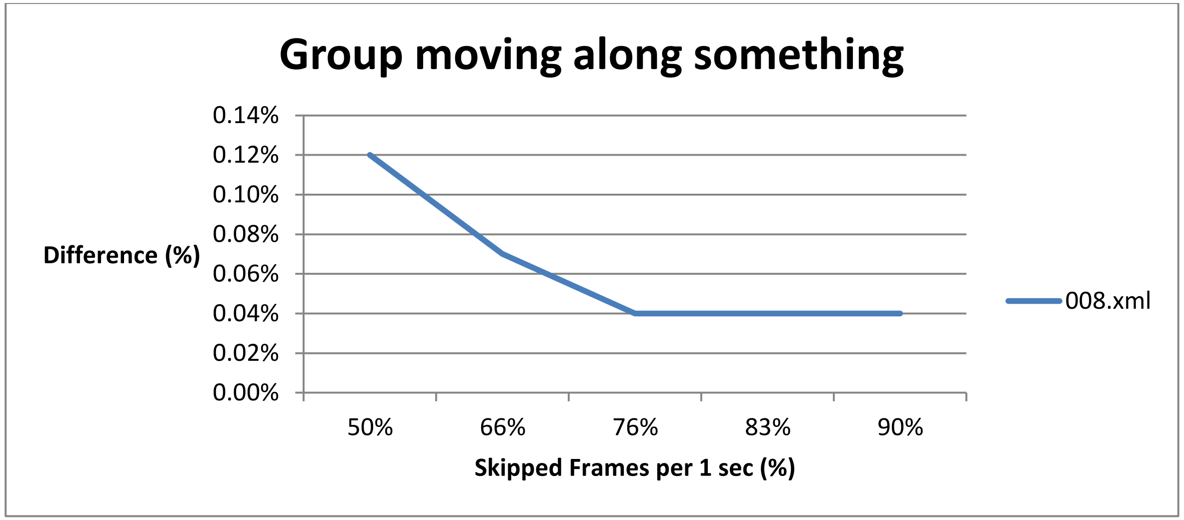


Figure 6-19. The errors calculated by simulator during the recognition of *"group walking along something"* pattern while skipping certain amount of frames from the movie.

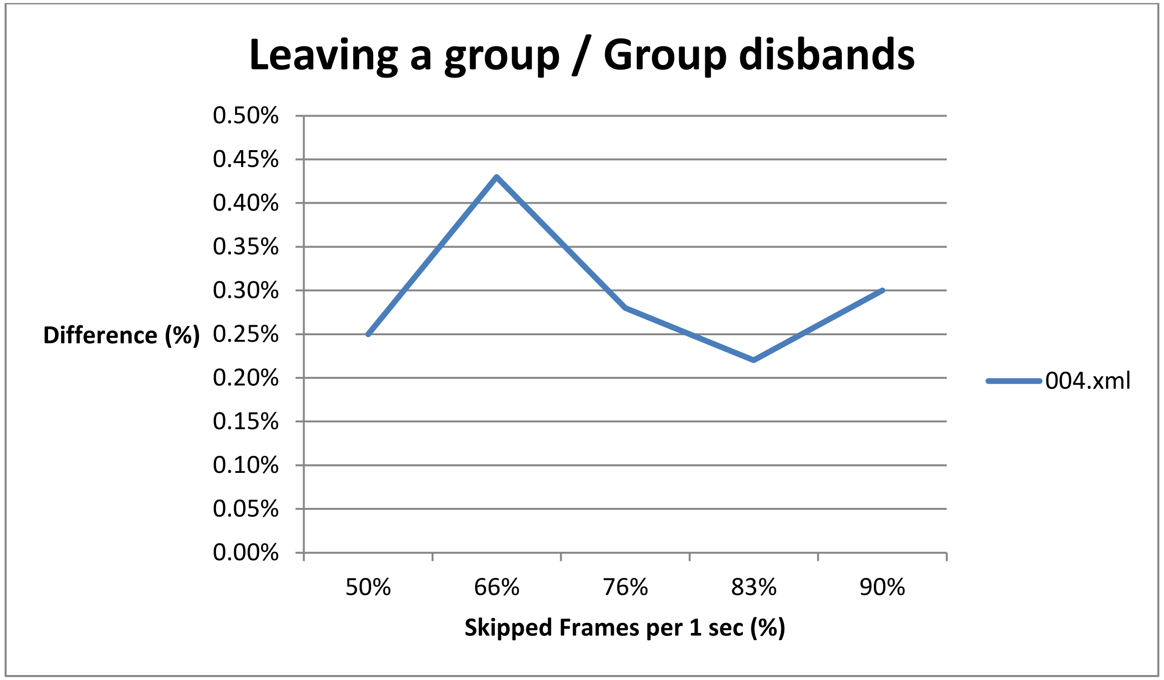


Figure 6-20. The errors calculated by simulator during the recognition of *"leaving a group / group disbands"* pattern while skipping certain amount of frames from the movie.

Analysis of the experimental results

The pattern recognition process relies on events captured during the simulation. These experiments were conducted to determine how pattern analyser module handles the recognition process in case when several frames from the video are not being processed during the simulation. The results show that the pattern analyser is still being supplied with sufficient information to successfully recognize a pattern in situations when simulator is not being able to process a certain percentage of frames from the video. The errors in this set of experiments are fluctuating between 0% and 10% as clearly indicated in "Time difference" column of Table 6-4 (Figures 6-11 – 6-20) and are not linear as in case of experiments carried out with replaying videos at different frame rates (section 6.2). This is due to the fact that each pattern emerges as a result of the execution of an algorithm following different logic. In some cases, the algorithms require fewer data to be processed in order to recognize a pattern (e.g. “walking towards something” pattern) and they are much faster. In other scenarios, the algorithms require to process more information before the pattern can be successfully recognized (e.g. “walking towards something while being in a group” pattern).

6.4. Summary

In this chapter the methodology for experimental analysis of the results of interactive simulation of different scenarios for event-driven individual and group dynamic behaviour pattern recognition was presented using the developed software. There are three series of the experiments. First one is based on comparing the expected results with the observed behaviour through reading the input and monitoring the visual output of the simulator. The second one is based on analysis of the output of the pattern analyser against data enclosed in external video files in order to validate the functionality of the pattern analyser module. This has been done in a way so that the relation between different video spatial values and log entries could have been established. The third series of experiments focused on replaying different case scenarios stored in external files under different conditions: different frame rates and skipping certain percentage of frames. Considering the computational power of different machines and possibility of partial data loss before delivery as a simulator input, the experiments have been conducted in a systematic and pragmatic way to determine how these two factors influence the operation of the pattern analyser.

Two separate consoles have been implemented and used to monitor the runtime behaviour of the software application. In order to facilitate the analysis a large number of configurable parameters, which control the execution have been implemented. They allow one to optimize the performance and improve the quality of the overall process of behaviour analysis at runtime. A separate parameter configuration console has been implemented to automate this process. We systematically applied the above methodology to validate the most important patterns of individual and group dynamic behaviour. The results are fully satisfactory and proved unanimously the initial hypothesis of this research.

Chapter 7

Conclusion and recommendations for future work

7.1. Reflection on the research

Dynamic behaviour analysis of individuals and groups is an area of extensive research in both research community and industry due to its wide applicability to various areas such as video surveillance and security, accidents and safety management, business customer insight and video games programming. The literature provides countless examples of presenting and tackling the complexity of this challenging task but the problem still remains difficult. Two main factors impact the real-time performance of the video analysis: the enormous volume of visual data, which has to be processed in real-time, and the need to combine video data processing with complex analytic symbolic data processing. Most researchers in the field concentrate on direct visual processing of image frames recorded by surveillance cameras and are mostly concerned with identification of single actions of humans executed in laboratory environments on the basis of statistical data. Such approach in combination with the latest technological advancement may address some of the issues of extracting critical visual features from image frames, but when it comes to more complex activities this soon becomes an overwhelming task. Our research hypothesis was that by combining partial information extracted from videos with simulated information generated by 3D simulation we can make this analysis feasible.

In the beginning of our research we had certain anticipation about the potential outcomes but we were not absolutely certain we can prove it feasible. The goal we formulated was to create a framework which includes two separate but tightly coupled components – 3D simulator and behaviour pattern analyser, combining two very different areas of modelling – graphical modelling and logical modelling. However, since the input data we assumed is limited to general spatial information and dynamic characterisation, i.e. locations and viewing directions of individuals, there was a concern that it might not be sufficient to simulate the scene dynamics adequately and the two outputs will diverge. There was also a risk that the simulation could lead to generation of noisy data, unsuitable for the purpose of pattern analysis.

Although our approach is still dependent on some data extracted through image processing methods, this information is minimal and therefore the framework reduces the need of performing complex data processing of the original video footage. Our simulator can be also enhanced to utilise more visual information, which would increase the accuracy of simulation and would allow the analyser to lower the granularity of recognised patterns and as a result, to analyse the behaviour beyond the simple dynamics based on physical movements only.

As the development progressed and reached more mature stages, we became more confident that our initial assumptions were correct. We managed to control effectively the event capturing process that has direct impact on the recognition of patterns in real-time and as a result we succeeded in pursuing our goal. Through extensive experimentations in the end of the research we were able to prove that is it possible to derive many useful behavioural patterns by combining very limited physical data and generated through simulation artificial data.

7.2. Originality and contributions to the knowledge

As a result of this research we have successfully implemented a prototype of a software tool for model-driven individual and group dynamic behaviour analysis based on 3D simulation and obtained some very encouraging experimental results which prove the applicability of the framework to areas where the video analytics is limited to dynamic behaviour. They can be summarized as follows:

1) ***A self-contained, standalone software framework for analysis of dynamic patterns of visual behaviour with the potential of applicability in other areas such as video games programming, video surveillance and security, accidents and safety management, business customer insight, and user experience enhancement.*** In our approach we are replacing the video signal from video footage with streamlined information extracted from it and combine it with simulated data propelled by minimal information about the visual scene. This has a number of advantages over standard image processing techniques: we can reduce the requirements for processing of large volume of video data in real time, we can define behavioural patterns on different level of granularity and we can combine visual features extracted from the image frames with symbolic data processing to conduct analysis which is fit for purpose. As a side effect of the research, we have created a modular engine for video analytics that can be utilised as a separate software module pluggable into various systems, enhancing their ability to analyse the behaviour, such as game engines, intelligent cameras, robot planning systems, etc. The final source of the simulator contains approximately 15,000 lines of programming code which can be used as a Java library. The potential applicability of the framework is supported by its modular structure, which allows processing the information flow in standard XML format, as well as a separate configuration file allowing easy configuration. The simulator can be used in conjunction with visual pattern recognition modules capable of producing the minimal information required for further analysis of the visual scene, such as object identification, description and classification, location, directions and trajectories of movements. The framework was presented first during the World Congress in Computer Science in June 2016 in Las Vegas, USA - see Gasiorowski, Vassilev and Ouazzane (2016).
2) ***An ontology of a visual scene.*** During the analytical phase of our research as a necessary step towards formulating the patterns of behaviour we developed a basic ontology of the visual scene, which can be used as a starting point for many applications which require representation of 3D visual information. Our ontology define key concepts, relations and visual appearance as well as an event-driven model of a constraint dynamic world which can be easily adopted in the conceptualisation and construction of various systems for analysis of visual scene. Especially interesting, we believe, is our analysis of the group behaviour on the basis of the behaviour of individual members of the group. The concepts of pairing and grouping we defined are based on the logical analysis of the group behaviour as social gathering of

individuals rather than statistical formation with spatial distribution in the physical space. This theoretical stance lays down the ground for automation of the analysis of social behaviour, which could be the next step in video analytics. For the first time our approach to group dynamics was reported during the World Congress in Computer Science in 2016 in Las Vegas, USA – see Gasiorowski, Vassilev & Ouazzane (2016). Further extension of this ontology is currently under development by other members of the Video Analytics research group at the Cyber Security Research Centre of London Metropolitan University and will be published in the future.

3) ***A method for simulation of dynamic behaviour based on combination of ray casting and ghosting techniques.*** During our work on the simulator we implemented a useful method for logical analysis by attaching bounding spheres to individual agent's limbs. By using the ray casting techniques commonly used in computer games we were able to establish meaningful relationships with other objects on the scene and thus, to avoid collisions and logical inconsistencies in the simulation. Instead of analysing such features on the base of extracted information from the video stream, our method is capable of recognising dynamic patterns of the individuals on the basis of simulation events. This approach allows achieving higher degree of precision in the simulation and higher level of independence of the agent's behaviour. It was introduced first in the publication Gasiorowski, Vassilev & Ouazzane (2016) and is presented in details in an article currently awaiting publication in IEEE Transactions on Visual Simulation – Gasiorowski, Vassilev & Ouazzane (forthcoming).

4) ***Algorithms for approximation of dynamic data about movements suitable for behaviour pattern analysis.*** During our work on the simulator we have implemented two original algorithms, which may be useful in different contexts**.** The first algorithm estimates the sight sense of an agent with regards to its spine and head orientation (Formula 4-6 in chapter 4). The algorithm uses information on current head location, bottom parts of its back and calculates a vector representing the approximation of a spine and viewing direction. This algorithm can be used to smoothen the behaviour of game agents in video games and other simulated environments. The experimental analysis reported in Chapter 5 shows that this algorithm is robust and efficient so that it can be used in real time. The algorithm is described in our forthcoming publication Gasiorowski, Vassilev & Ouazzane (forthcoming). The second original algorithm determines which agent is leaving a group and when purely on the basis of the distance to the other members. It is a part of the recognition method of the "Somebody leaves a Group" pattern in chapter 6 (See Formula A.2-1 in Table. A.2-14; A.2.) The algorithm was intensively tested in various scenarios and proved to work in almost 100% of the cases with an absolute adequacy and perfectly acceptable accuracy. Both algorithms are programmed as a self-contained and entirely reusable library, so that they can be easily imported and used in applications which account the position of the individuals within the group and the parts of their body, respectively.

7.3. Recommendations for future development

The framework we have implemented can be enhanced in several directions that can increase the accuracy of the recognition and the usability of the software in other projects. In this section we indicate the identified room for improvement and present a few suggestions about potential future work.

1) ***Introduction of physical shapes.*** Currently the objects are entirely bound to a logical capsule, which is used for performing fast and efficient calculations of locations and directions. Furthermore, we are artificially adding approximation of objects shapes to allow capturing events and processing of the simulated data rather than real data. These limitations could be lifted up if we introduce more realistic physical shapes of the objects to account their real dimensions with higher precision.

2) ***Accounting the spatial boundaries of the visual scene.*** Currently the methods for recognising the patterns on the basis of event capturing do not account designating boundaries of space such as walls and ceilings. The prototype implementation considers only the floor as a physical boundary to support discovering of patterns which involve climbing on top/jumping from other objects or walking upstairs/downstairs. Considering the ceilings and the walls would allow recognition of more fine grained patterns of dynamic behaviour which involve traversing of partitioned spaces, such as continuing walking along a corridor after leaving a room, or changing the floor of a building.

3) ***Implementation of a separate adaptor of dynamic movements.*** The current implementation depends on the input of the trajectories through interactive navigation controlled from the keyboard or loaded from a file. As it is at the moment the simulator updates the location and orientation of all agents on the scene by editing an external file located in the application. In order to automate this process it is essential to perform the updates and streamline the reconstructed trajectories directly in the simulator's memory rather than in a file. For this purpose a separate adapter can be developed. Such architecture would also allow interpolation of the values of different registered locations while accounting the viewing directions and relative position of the limbs even in situations when input signal temporarily cease to be delivered.

4) ***Advanced utilisation of the agent's sight sense.*** The implemented method for estimation of the sight range can be used for recognising situations when the agent focuses on a particular object. This requires timing the attention of an agent and detecting possible collision between rays and other objects in the scene. Such a possibility would allow more fine grained analysis of the behaviour beyond the analysis based only on physical movements.

5) ***Consideration of multiple and potentially moving cameras.*** As a future analysis we can consider multiple cameras distributed within the boundaries of physical space being under surveillance. It is an obvious candidate for extending the framework, but would require changing of some of the underlying mathematical theory behind the 3D simulation to accommodate the relativity of the different viewing points. It would also require additional synchronisation of the analysis performed at different places.

References

Abdul-Azim, H.A., Hemayed, E.E., 2015. Human action recognition using trajectory-based representation. Egyptian Informatics Journal 16, 187–198. doi:10.1016/j.eij.2015.05.002

Afsar, P., Cortez, P., Santos, H., 2015. Automatic Human Action Recognition from Video Using Hidden Markov Model, in: 2015 IEEE 18th International Conference on Computational Science and Engineering (CSE). Presented at the 2015 IEEE 18th International Conference on Computational Science and Engineering (CSE), pp. 105–109. doi:10.1109/CSE.2015.41

Afzal, M.M., Ouazzane, K., Vassilev, V., Patel, Y., 2016. Incremental Reconstruction of Moving Object Trajectory. Presented at the VISUAL 2016, The First International Conference on Applications and Systems of Visual Paradigms, pp. 24–29.

Afzal, M.M., Ouazzane, K., Vassilev, V., Patel, Y., 2017. Incremental Reconstruction of Moving Object Trajectory. The First International Conference on Applications and Systems of Visual Paradigms 24–29.

Ahad, M.A.R., Tan, J.K., Kim, H., Ishikawa, S., 2009. Human activity analysis: Concentrating on Motion History Image and its variants, in: ICCAS-SICE, 2009. Presented at the ICCAS-SICE, 2009, pp. 5401–5406.

Akita, K., 1984. Image sequence analysis of real world human motion. Pattern Recognition, Knowledge Based Image Analysis 17, 73–83. doi:10.1016/0031-3203(84)90036-0

Alexandre, M.U.Z., David, R.C., INNOCENTI, E., n.d. Y. A post-processed 3D visualization tool for forest fire simulations. st International ICST Confer-ence on Simulation Tools and Techniques for Communication, Network and System., Network and System.

AlGadhi, Saad AH, and H. Mahmassani. "Simulation of crowd behaviour and movement: fundamental relations and application." Transportation Research Record 1320.1320 (1991): 260-268.

Ali, S., Shah, M., 2007. A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07. Presented at the IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07, pp. 1–6. doi:10.1109/CVPR.2007.382977

Ali, S., Shah, M., 2008. Floor Fields for Tracking in High Density Crowd Scenes, in: Forsyth, D., Torr, P., Zisserman, A. (Eds.), Computer Vision – ECCV 2008, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–14.

Allen, J.F., 1983, "Maintaining knowledge about temporal intervals", Communications of the ACM, 26(11): 832–843.

Aloysius, L.H.W., Dong, G., Zhiyong, H., Tan, T., 2004. Human posture recognition in video sequence using pseudo 2-D hidden Markov models, in: Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th. Presented at the Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th, p. 712–716 Vol. 1. doi:10.1109/ICARCV.2004.1468915

Anjum, N., Cavallaro, A., 2007. Single camera calibration for trajectory-based behaviour analysis, in: IEEE Conference on Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. Presented at the IEEE Conference on Advanced Video and Signal Based Surveillance, 2007. AVSS 2007, pp. 147–152. doi:10.1109/AVSS.2007.4425301

Autodesk, 3D Design, Engineering & Entertainment Software, 2017. [Online]. Available at: https://www.autodesk.co.uk/ [Accessed 3 Nov 2017].

Azorín-López, J., Saval-Calvo, M., Fuster-Guilló, A., García-Rodríguez, J., 2013. Human behaviour recognition based on trajectory analysis using neural networks, in: The 2013 International Joint Conference on Neural Networks (IJCNN). Presented at the The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. doi:10.1109/IJCNN.2013.6706724

Azorin-López, J., Saval-Calvo, M., Fuster-Guilló, A., Oliver-Albert, A., 2014. A predictive model for recognizing human behaviour based on trajectory representation, in: 2014 International Joint Conference on Neural Networks (IJCNN). Presented at the 2014 International Joint Conference on Neural Networks (IJCNN), pp. 1494–1501. doi:10.1109/IJCNN.2014.6889883

Baker, M.J., n.d. Maths - Quaternion Functions [Online] Available at: http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/functions/index.htm [Accessed 20 Oct 2015].

Baker, M.J., n.d. Transformations using Quaternions. Available at: http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/transforms/ [Accessed 21 Dec 2015].

Bao, R., Iwamoto, K., 2016. Fast 2D-to-3D matching with camera pose voting for 3D object identification, in: 2016 IEEE International Conference on Image Processing (ICIP). Presented at the 2016 IEEE International Conference on Image Processing (ICIP), pp. 664–668. doi:10.1109/ICIP.2016.7532440

Baum, L. E., Petrie, T., Soules, G. and Weiss, N., 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. The Annals of Mathematical Statistics, vol. 41, no. 1, pp. 164–171.

Bélanger, J., Venne, P., Paquin, J.N., 2010. The What, Where and Why of Real-Time Simulation. Planet RT.

Ben, X., Huang, X., Zhuang, Z., Yan, R., Xu, S., 2013. Agent-based approach for crowded pedestrian evacuation simulation. IET Intelligent Transport Systems 7, 56–67. doi:10.1049/iet-its.2011.0236

Bhaskar, P.K., Yong, S.P., Jung, L.T., 2015. Enhanced and effective parallel optical flow method for vehicle detection and tracking, in: 2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC). Presented at the 2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC), pp. 138–143. doi:10.1109/ISMSC.2015.7594042

Biswas, K.K., Basu, S.K., 2011. Gesture recognition using Microsoft Kinect, in: 2011 5th International Conference on Automation, Robotics and Applications (ICARA). Presented at the 2011 5th International Conference on Automation, Robotics and Applications (ICARA), pp. 100–103. doi:10.1109/ICARA.2011.6144864

Blender Foundation, 2015. [Online]. Blender Software. Available at: https://www.blender.org/ [Accessed 5 Nov 2015].

Blender Foundation, 2014. Introduction — Blender Manual [Online] Available at: https://docs.blender.org/manual/en/dev/rigging/armatures/introduction.html [Accessed 25 Sep 2014].

Bobick, A.F., Davis, J.W., 2001. The recognition of human movement using temporal templates. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 257–267. doi:10.1109/34.910878

Bogdanovych, A., Bauer, M., Simoff, S., 2009. Recognizing Customers' Mood in 3D Shopping Malls Based on the Trajectories of Their Avatars, in: Filipe, J., Cordeiro, J. (Eds.), Enterprise Information Systems, Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 745–757.

Bouttier, F. and Courtier, P., 1999. Data assimilation concepts and methods, Meteorological Training Course Lecture Series, ECMWF, 1999, pp. 75.

Cabello, F., León, J., Iano, Y., Arthur, R., 2015. Implementation of a fixed-point 2D Gaussian Filter for Image Processing based on FPGA, in: Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2015. Presented at the Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2015, pp. 28–33. doi:10.1109/SPA.2015.7365108

Callan, D.R., 1998. The Essence of Neural Networks, 01 edition. ed. Prentice Hall, Harlow.

Carnegie Mellon, 2015. Motion Capture Database [Online]. Available at: http://mocap.cs.cmu.edu [Accessed 11 Apr 2015].

Cernekova, Z., Nikolaidis, N., Pitas, I., 2006. Temporal Video Segmentation by Graph Partitioning, in: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings. Presented at the 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, pp. II–II. doi:10.1109/ICASSP.2006.1660316

Cewe A., Kobierowska J., Nahorska H., Pancer I., 2015 Tablice matematyczne [Mathematical tables]. Wydawnictwo Podkowa, Gdansk, Poland.

Chen, H.T., Chou, C.L., Tsai, W.J., Lee, S.Y., 2011. 3D ball trajectory reconstruction from single-camera sports video for free viewpoint virtual replay, in: 2011 IEEE Visual Communications and Image Processing (VCIP). Presented at the 2011 IEEE Visual Communications and Image Processing (VCIP), pp. 1–4. doi:10.1109/VCIP.2011.6115930

Chen, M.Y. and Hauptmann, A., 2009. Mosift: Recognizing human actions in surveillance videos.

Chen, Z., 2012. Mining individual behaviour pattern based on significant locations and spatial trajectories, in: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops). Presented at the 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 540–541. doi:10.1109/PerComW.2012.6197563

Crytek Inc., Cry Engine, 2017. [Online]. Available at: http://www.crytek.com/ [Accessed 3 Nov 2017]

Cucchiara, R., Prati, A., Vezzani, R., 2005. Posture classification in a multi-camera indoor environment, in: IEEE International Conference on Image Processing 2005. Presented at the IEEE International Conference on Image Processing 2005, p. I-725-8. doi:10.1109/ICIP.2005.1529853

De la Torre, F., Hodgins, J., Bargteil, A., Martin, X., Macey, J., Collado, A., Beltran, P., Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213, 2008. Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC)Database. [Online]. Available at: http://www.cs.cmu.edu/~ftorre/cmu-mad.pdf [Accessed 22 Feb 2015].

Deloura, DeLoura, M., 2000. Game Programming Gems, Har/Cdr edition. ed. Charles River Media, Boston, Mass.

Demisse, G.G., Aouada, D., Ottersten, B., 2015. Template-based statistical shape modelling on deformation space, in: 2015 IEEE International Conference on Image Processing (ICIP). Presented at the 2015 IEEE International Conference on Image Processing (ICIP), pp. 4386–4390. doi:10.1109/ICIP.2015.7351635

Dorea, C.C., Pardas, M., Marques, F., 2005. A motion-based binary partition tree approach to video object segmentation, in: IEEE International Conference on Image Processing 2005. Presented at the IEEE International Conference on Image Processing 2005, p. II-430-3. doi:10.1109/ICIP.2005.1530084

Edén, R., 2014. jMonkeyEngine 3.0 Cookbook. Packt Publishing.

Efros, A.A., Berg, A.C., Mori, G., Malik, J., 2003. Recognizing action at a distance, in: Ninth IEEE International Conference on Computer Vision, 2003. Proceedings. Presented at the Ninth IEEE International Conference on Computer Vision, 2003. Proceedings, pp. 726–733 vol.2. doi:10.1109/ICCV.2003.1238420

Encyclopedia Britannica, 2017. Computer Simulation. [Online]. Available at: https://www.britannica.com/technology/computer-simulation [Accessed 23 Sep 2017].

Epic Games Inc., Unreal Engine, 2017. [Online]. Available at: https://www.unrealengine.com/ [Accessed 3 Nov 2017]

Fitzgibbon, A., Zisserman, A., 1998. Automatic 3D model acquisition and generation of new images from video sequences, in: Signal Processing Conference (EUSIPCO 1998), 9th European. Presented at the Signal Processing Conference (EUSIPCO 1998), 9th European, pp. 1–8.

Fredrick, H., n.d. Why Doesn't GPS Work Inside a Building? [Online]. Available at: https://itstillworks.com/doesnt-gps-work-inside-building-18659.html [Accessed 4 Nov 2017].

Gao, W.H., 2010. Research on client behavior pattern recognition system based on Web log mining, in: 2010 International Conference on Machine Learning and Cybernetics. Presented at the 2010 International Conference on Machine Learning and Cybernetics, pp. 466–470. doi:10.1109/ICMLC.2010.5581016

García-Zubia, J., Irurzun, J., Angulo, I., Hernández, U., Castro, M., Sancristobal, E., Orduña, P., Ruiz-de-Garibay, J., 2010. SecondLab: A remote laboratory under Second Life, in: IEEE EDUCON 2010 Conference. Presented at the IEEE EDUCON 2010 Conference, pp. 351–356. doi:10.1109/EDUCON.2010.5492556

Gasiorowski, P., Vassilev, V. and Ouazzane, K., 2016, January. Simulation-based Visual Analysis of Individual and Group Dynamic Behaviour. In Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV) (p. 303). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Gasiorowski, P., Vassilev, V. and Ouazzane, K., forthcoming, Model-based Event-driven Simulation of Individual and Group Dynamic Behavior in Bound Spaces, IEEE Transaction of Visual Simulation (to be published).

Gong, S. and Xiang, T., 2003. Scene Events Recognition Without Tracking. Acta Automatica Sinica, 29(3), pp.321-321.

Gong, S., Xiang, T., 2011. Visual analysis of behaviour from pixels to semantics. Springer, London; New York.

Guo, R.Y., Huang, H.J., 2008. A mobile lattice gas model for simulating pedestrian evacuation. Physica A: Statistical Mechanics and its Applications 387, 580–586. doi:10.1016/j.physa.2007.10.001

Hahne, B., 2008. Motion Capture - cgspeed [Online] Available at: https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture [Accessed 11 Apr 2015].

Hamilton, W.R., 1853. Lectures on quaternions: containing a systematic statement of a new mathematical method, of which the principles were communicated in 1843 to the Royal Irish academy, and which has since formed the subject of successive courses of lectures, delivered in 1848 and subsequent years, in the halls of Trinity college, Dublin. Hodges and Smith, Dublin.

Hariyono, J., Jo, K.H., 2015. Pedestrian action recognition using motion type classification, in: 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF). Presented at the 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF), pp. 129–132. doi:10.1109/CYBConf.2015.7175919

Hassan, M., RajKumar, R., Isa, D., Arelhi, R., 2011. Kalman Filter as a pre-processing technique to improve the support vector machine, in: 2011 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT). Presented at the 2011 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT), pp. 107–112. doi:10.1109/STUDENT.2011.6089335

He, X., Wang, Y., Cao, Y., 2012. Researching on AI path-finding algorithm in the game development, in: 2012 International Symposium on Instrumentation Measurement, Sensor Network and Automation (IMSNA). Presented at the 2012 International Symposium on Instrumentation Measurement, Sensor Network and Automation (IMSNA), pp. 484–486. doi:10.1109/MSNA.2012.6324627

Helbing, D., Farkas I., and Vicsek, T., 2000. Simulating dynamical features of escape panic. Nature, vol. 407, no. 6803, pp. 487-490. [Online]. Available at: http://dx.doi.org/10.1038/35035023 [Accessed 2 Apr 2014].

Hluchy, L., Kvassay, M., Dlugolinský, S., Schneider, B., Bracker, H., Kryza, B., Kitowski, J., 2011. Handling internal complexity in highly realistic agent-based models of human behaviour, in: 2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI). Presented at the 2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 11–16. doi:10.1109/SACI.2011.5873089

Horn, B.K. and Schunck, B.G., 1981. Determining optical flow. Artificial intelligence, 17(1-3), pp.185-203.

Hospedales, T., Gong, S. and Xiang, T., 2009, September. A markov clustering topic model for mining behaviour in video. In 2009 IEEE 12th International Conference on Computer Vision (pp. 1165-1172). IEEE.

Hu, C.-H., Wo, S.-L., 2010. An efficient method of human behaviour recognition in smart environments, in: 2010 International Conference on Computer Application and System Modeling (ICCASM). Presented at the 2010 International Conference on Computer Application and System Modeling (ICCASM), pp. V12–690–V12–693. doi:10.1109/ICCASM.2010.5622464

Huang, L., Wong, S.C., Zhang, M., Shu, C.-W., Lam, W.H.K., 2009. Revisiting Hughes' dynamic continuum model for pedestrian flow and the development of an efficient solution algorithm. Transportation Research Part B: Methodological 43, 127–141. doi:10.1016/j.trb.2008.06.003

Hughes, R.L., 2002. A continuum theory for the flow of pedestrians. Transportation Research Part B: Methodological 36, 507–535. doi:10.1016/S0191-2615(01)00015-7

IEEE Standard Ontologies for Robotics and Automation, 2015. . IEEE Std 1872-2015 1–60. doi:10.1109/IEEESTD.2015.7084073

IPVM – ipvideomarket, 2015. Average Frame Rate Video Surveillance 2011 [Online]. Available at: https://ipvm.com/reports/recording-frame-rate--whats-actually-being-used [Accessed 2 Oct 2015].

Iuppa, N., Borst, T., 2009. End-to-End Game Development: Creating Independent Serious Games and Simulations from Start to Finish, 1 edition. ed. Focal Press, Cambridge, England.

Jiang, Y.-G., Dai, Q., Xue, X., Liu, W., Ngo, C.-W., 2012. Trajectory-Based Modeling of Human Actions with Motion Reference Points, in: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (Eds.), Computer Vision – ECCV 2012, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 425–438. doi:10.1007/978-3-642-33715-4_31

Jia-Wen, Z., 2015. A Novel Sports Video Background Segmentation Algorithm Based on Graph Partition, in: 2015 8th International Conference on Intelligent Computation Technology and Automation (ICICTA). Presented at the 2015 8th International Conference on Intelligent Computation Technology and Automation (ICICTA), pp. 65–68. doi:10.1109/ICICTA.2015.25

jMonkeyEngine, 2014. GitHub Repository - GitHub. [Online]. Available at: https://github.com/jMonkeyEngine/jmonkeyengine [Accessed 17 Mar 2014].

Joselli, M., Silva, J.R. d, Clua, E., 2014. An Architecture for Real Time Crowd Simulation Using Multiple GPUs, in: 2014 Brazilian Symposium on Computer Games and Digital Entertainment. Presented at the 2014 Brazilian Symposium on Computer Games and Digital Entertainment, pp. 1–10. doi:10.1109/SBGAMES.2014.13

Kar, A., 2010. Skeletal tracking using microsoft kinect. Methodology, 1, pp.1-11.

Kawamoto, K., 2010. A data assimilation method for estimating the parameters of a social force model for pedestrian motion analysis, in: World Automation Congress (WAC), 2010. Presented at the World Automation Congress (WAC), 2010, pp. 1–5.

Kelly, D., Smyth, B., Caulfield, B., 2013. Uncovering Measurements of Social and Demographic Behavior From Smartphone Location Data. IEEE Transactions on Human-Machine Systems 43, 188–198. doi:10.1109/TSMC.2013.2238926

Kendra, C., 2016, Understanding Body Language [Online]. Available at: http://psychology.about.com/od/nonverbalcommunication/ss/understanding-body-language_8.htm [Accessed 11 Nov 2016].

Khorbotly, S., Hassan, F., 2011. A modified approximation of 2D Gaussian smoothing filters for fixed-point platforms, in: 2011 IEEE 43rd Southeastern Symposium on System Theory. Presented at the 2011 IEEE 43rd Southeastern Symposium on System Theory, pp. 151–159. doi:10.1109/SSST.2011.5753797

Khronos Group Inc., 2017. The Industry’s Foundation for High Performance Graphics [Online]. Available at: https://www.khronos.org/opengl [Accessed at: 3 Nov 2017]

Kim, H., Jung, K., Oh, T., Myung, H., 2015. Image-based localization using image database and local 3D maps, in: 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Presented at the 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 32–33. doi:10.1109/URAI.2015.7358922

Kowalczyk R., Niedzialomski K., Obczynski C., 2011. Matematyka dla studentów i kandydatów na wyższe uczelnie [Mathematics for students and candidates for higher education]. Wydawnictwo Naukowe PWN, Warsaw, Poland.

Kowalski R., Sergot M. (1986). A Logic-based Calculus of Events. New Generation Computing 4: 67–95

Kusterer, R., 2013. jMonkeyEngine 3.0 Beginner’s Guide. Packt Publishing, Birmingham, U.K.

Kvassay, M., Hluchý, L., Schneider, B., Bracker, H., 2012. Towards causal analysis of data from human behaviour simulations, in: 2012 4th IEEE International Symposium on Logistics and Industrial Informatics. Presented at the 2012 4th IEEE International Symposium on Logistics and Industrial Informatics, pp. 41–46. doi:10.1109/LINDI.2012.6319507

Lengyel, E., 2003. Mathematics for 3D Game Programming and Computer Graphics, 2nd Revised edition edition. ed. Charles River Media, Hingham, Mass.

Lengyel, E., 2016. Foundations of Game Engine Development, Volume 1: Mathematics, 1 edition. ed. Terathon Software LLC.

Lepetit, V. and Fua, P., 2005. Monocular model-based 3D tracking of rigid objects. Now Publishers Inc.

Lester P., 2005, A* Pathfinding for Beginners [Online]. Available at: http://www.policyalmanac.org/games/aStarTutorial.htm [Accessed 16 Apr 2014].

Li, H., Tang, W., Simpson, D., 2004. Behaviour based motion simulation for fire evacuation procedures, in: Theory and Practice of Computer Graphics, 2004. Proceedings. Presented at the Theory and Practice of Computer Graphics, 2004. Proceedings, pp. 112–118. doi:10.1109/TPCG.2004.1314460

Lin, D., Grimson, E., Fisher, J., 2010. Modeling and estimating persistent motion with geometric flows, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8. doi:10.1109/CVPR.2010.5539848

Lu, W.-L., Little, J.J., 2006. Simultaneous Tracking and Action Recognition using the PCA-HOG Descriptor, in: The 3rd Canadian Conference on Computer and Robot Vision (CRV'06). Presented at the The 3rd Canadian Conference on Computer and Robot Vision (CRV'06), pp. 6–6. doi:10.1109/CRV.2006.66

Maik, V., Paik, D.T., Lim, J., Park, K., Paik, J., 2010. Hierarchical pose classification based on human physiology for behaviour analysis. IET Computer Vision 4, 12–24. doi:10.1049/iet-cvi.2008.0086

Mashtalir, S., Mashtalir, V., 2016. Sequential temporal video segmentation via spatial image partitions, in: 2016 IEEE First International Conference on Data Stream Mining Processing (DSMP). Presented at the 2016 IEEE First International Conference on Data Stream Mining Processing (DSMP), pp. 239–242. doi:10.1109/DSMP.2016.7583549

Mehran, R., Oyama, A., Shah, M., 2009. Abnormal crowd behaviour detection using social force model, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009. Presented at the IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009, pp. 935–942. doi:10.1109/CVPR.2009.5206641

Microsoft, 2015. Kinect - Windows App Development [Online] Available at: https://developer.microsoft.com/en-us/windows/kinect [Accessed 7 Apr 2015].

Milam, M.B., Mushambi, K., Murray, R.M., 2000. A new computational approach to real-time trajectory generation for constrained mechanical systems, in: Proceedings of the 39th IEEE Conference on Decision and Control, 2000. Presented at the Proceedings of the 39th IEEE Conference on Decision and Control, 2000, pp. 845–851 vol.1. doi:10.1109/CDC.2000.912875

Mizerski W., Sadowski W., Garbarczyk A., Tokarska B., Mazur K., 2014. Tablice matematyczne [Mathematical tables]. Grupa Wydawnicza Adamantan s.c., Warsaw, Poland.

Mlynek, J., Martinec, T., 2014. Mathematical model of composite manufacture and calculation of robot trajectory, in: 2014 16th International Conference on Mechatronics - Mechatronika (ME). Presented at the 2014 16th International Conference on Mechatronics - Mechatronika (ME), pp. 345–351. doi:10.1109/MECHATRONIKA.2014.7018282

Mukherjee, S., Biswas, S.K., Mukherjee, D.P., 2011. Recognizing Human Action at a Distance in Video by Key Poses. IEEE Transactions on Circuits and Systems for Video Technology 21, 1228–1241. doi:10.1109/TCSVT.2011.2135290

Mulayim, A.Y., Yilmaz, U., Atalay, V., 2003. Silhouette-based 3-D model reconstruction from multiple images. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 33, 582–591. doi:10.1109/TSMCB.2003.814303

Muramatsu, M., Irie, T., Nagatani, T., 1999. Jamming transition in pedestrian counter flow. Physica A: Statistical Mechanics and its Applications 267, 487–498. doi:10.1016/S0378-4371(99)00018-7

Musse, S.R., Thalmann, D., 2002. Hierarchical model for real time simulation of virtual human crowds, in: 5th IEEE EMBS International Summer School on Biomedical Imaging, 2002. Presented at the 5th IEEE EMBS International Summer School on Biomedical Imaging, 2002, p. 13 pp.-. doi:10.1109/SSBI.2002.1233994

Naji, H.A.H., Wu, C., Zhang, H., Li, L., 2017. Towards understanding the impact of human mobility patterns on taxi drivers' income based on GPS data: A case study in Wuhan #x2014; China, in: 2017 4th International Conference on Transportation Information and Safety (ICTIS). Presented at the 2017 4th International Conference on Transportation Information and Safety (ICTIS), pp. 1152–1160. doi:10.1109/ICTIS.2017.8047916

Navarro, A., Pradilla J.V., Rios O., 2012. Open Source 3D Game Engines for Serious Games Modeling, Modeling and Simulation in Engineering, Prof. Catalin Alexandru (Ed.), ISBN: 978-953-51- 0012-6, InTech. [Online]. Available at: http://www.intechopen.com/books/modeling-and-simulation-in- engineering/open-source-3d-game-engines-for-serious-games-modeling [Accessed 2 Nov 2017]

Nevatia, R., Zhao, T., Hongeng, S., 2003. Hierarchical Language-based Representation of Events in Video Streams, in: Conference on Computer Vision and Pattern Recognition Workshop, 2003. CVPRW '03. Presented at the Conference on Computer Vision and Pattern Recognition Workshop, 2003. CVPRW '03, pp. 39–39. doi:10.1109/CVPRW.2003.10038

Nguyen, V., Blumenstein, M., 2011. An Application of the 2D Gaussian Filter for Enhancing Feature Extraction in Off-line Signature Verification, in: 2011 International Conference on Document Analysis and Recognition. Presented at the 2011 International Conference on Document Analysis and Recognition, pp. 339–343. doi:10.1109/ICDAR.2011.76

Nystrom, R., 2014. Game Programming Patterns. Genever Benning.

Official U.S. government, n.d. GPS.gov: GPS Overview [Online]. Available at: http://www.gps.gov/systems/gps/ [Accessed 17 Aug 2015].

Torus Knot Software Ltd, Ogre3D, 2016. [Online]. Available at: https://www.ogre3d.org/ [Accessed 3 Nov 2017]

Oikonomopoulos, A., Patras, I., Pantic, M., Paragios, N., 2007. Trajectory-Based Representation of Human Actions, in: Huang, T.S., Nijholt, A., Pantic, M., Pentland, A. (Eds.), Artifical Intelligence for Human Computing, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 133–154. doi:10.1007/978-3-540-72348-6_7

Onofri, L., Soda, P., 2012. Combining video subsequences for human action recognition, in: 2012 21st International Conference on Pattern Recognition (ICPR). Presented at the 2012 21st International Conference on Pattern Recognition (ICPR), pp. 597–600.

Oosten, J. van, 2012. “Understanding Quaternions. 3D Game Engine Programming.” [Online]. Available at: http://mathworld.wolfram.com/PolarCoordinates.html [Accessed 19 Oct 2015].

OptiTrack, n.d. OptiTrack [Online]. Available at: https://www.optitrack.com/ [Accessed 8 Apr 2015].

Oxford Dictionary, 2015. Behaviour - definition of Behaviour in English [Online]. Available at: https://en.oxforddictionaries.com/definition/behaviour [Accessed 3 Jan 2015].

Oxford Dictionary, 2015. Pattern - definition of Behaviour in English [Online]. Available at: https://en.oxforddictionaries.com/definition/behaviour [Accessed 3 Jan 2015].

Parrish, K., 2016. DirectX 12: what is it, and why it matters to PC gamers. Tech Radar. [Online]. Available at: http://www.techradar.com/news/gaming/directx-12-what-is-it-and-why-it-matters-to-pc-gamers-1318636 [Accessed 2 Nov 2017].

Pelechano, N., Allbeck, J.M., Badler, N.I., 2007. Controlling Individual Agents in High-density Crowd Simulation, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA ’07. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 99–108.

Perez, G.J., Tapang, G., Lim, M., Saloma, C., 2002. Streaming, disruptive interference and power-law behaviour in the exit dynamics of confined pedestrians. Physica A: Statistical Mechanics and its Applications 312, 609–618. doi:10.1016/S0378-4371(02)00987-1

PhysX Technology, 2017. [Online]. NVIDIA GeFORCE. Available at: https://www.geforce.com/hardware/technology/physx/technology [Accessed 17 Nov 2017].

Qin, Q., Wei, J., 2010. An agent-based approach for crowd dynamics simulation, in: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS). Presented at the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), pp. 78–82. doi:10.1109/ICICISYS.2010.5658843

Qiu, W., Bandara, A., 2015. GPS Trace Mining for Discovering Behaviour Patterns, in: 2015 International Conference on Intelligent Environments. Presented at the 2015 International Conference on Intelligent Environments, pp. 65–72. doi:10.1109/IE.2015.17

Rai, S., Hu, X., 2013. Behaviour Pattern Detection for Data Assimilation in Agent-Based Simulation of Smart Environments, in: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). Presented at the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), pp. 171–178. doi:10.1109/WI-IAT.2013.106

Rao, A.S., Gubbi, J., Marusic, S., Stanley, P., Palaniswami, M., 2013. Crowd density estimation based on optical flow and hierarchical clustering, in: 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI). Presented at the 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 494–499. doi:10.1109/ICACCI.2013.6637221

Rao, K.S., Fernandes, S.L., Haniben, P., Prajna, Pratheek, Devadiga, R.V., Kottary, S., Surakshitha, T.P., 2015. A review on various state of art technique to recognize occluded face images, in: 2015 2nd International Conference on Electronics and Communication Systems (ICECS). Presented at the 2015 2nd International Conference on Electronics and Communication Systems (ICECS), pp. 595–601. doi:10.1109/ECS.2015.7124977

Reese, R., Johnson, J., 2015. jMonkeyEngine 3.0 Game Development: A Practical Guide. P8Tech.

Ren, Z., Yuan, J., Meng, J., Zhang, Z., 2013. Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. IEEE Transactions on Multimedia 15, 1110–1120. doi:10.1109/TMM.2013.2246148

Rosch, E., Mervis, C., Gray, W., Johnson, D. and Boyes-Braem, P., 2004. Basic objects in natural categories. Cognitive psychology: Key readings, 448.

Rossmann, J., Hempe, N., Tietjen, P., 2009. A flexible model for real-time crowd simulation, in: IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009. Presented at the IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009, pp. 2085–2090. doi:10.1109/ICSMC.2009.5346308

Rymaszewski, M., Au, W.J., Wallace, M., Winters, C., Ondrejka, C., Batstone-Cunningham, B., Rosedale, P., 2006. Second Life: The Official Guide, Pap/Cdr edition. ed. Sybex, San Francisco, Calif. : Chichester ; Hoboken, N.J.

Saboia, P., Goldenstein, S., 2011. Crowd Simulation: Improving Pedestrians’ Dynamics by the Application of Lattice-Gas Concepts to the Social Force Model, in: 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images (Sibgrapi). Presented at the 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images (Sibgrapi), pp. 41–47. doi:10.1109/SIBGRAPI.2011.11

Saitou, M., Kojima, A., Kitahashi, T., Fukunaga, K., 2006. A Dynamic Recognition of Human Actions and Related Objects, in: First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC’06). Presented at the First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC’06), pp. 306–309. https://doi.org/10.1109/ICICIC.2006.5

Sasiadek, J. Z. and Duleba, I., 2000. 3D Local Trajectory Planner for UAV, In: Journal of Intelligent and Robotic Systems, 29, 191-21
Sattler, T., Leibe, B., Kobbelt, L., 2011. Fast image-based localization using direct 2D-to-3D matching, in: 2011 International Conference on Computer Vision. Presented at the 2011 International Conference on Computer Vision, pp. 667–674. doi:10.1109/ICCV.2011.6126302
Savage P., 2015. Unreal Engine 4 is now free. [Online]. PC Gamer. Available at: http://www.pcgamer.com/unreal-engine-4-is-now-free/ [Accessed 3 Nov 2017]
Second Life Wiki, n.d. [Online]. Available at: http://wiki.secondlife.com/wiki/Main_Page [Accessed 3 Nov 2017].
Sellers, G., Wright, R.S., Haemel, N., 2013. OpenGL Superbible: Comprehensive Tutorial and Reference, 6 edition. ed. Addison Wesley, Upper Saddle River, NJ.
Shan, C., Wei, Y., Qiu, X., Tan, T., 2004. Gesture recognition using temporal template based trajectories, in: Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. Presented at the Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004, p. 954–957 Vol.3. doi:10.1109/ICPR.2004.1334687
Shanahan, M., Witkowski, M., 2000. High-Level Robot Control through Logic, in: Castelfranchi, C., Lespérance, Y. (Eds.), Intelligent Agents VII Agent Theories Architectures and Languages, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 104–121. doi:10.1007/3-540-44631-1_8
Sharma, S., Lohgaonkar, S., 2010. Simulation of agent behaviour in a goal finding application, in: IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the. Presented at the IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the, pp. 424–427. doi:10.1109/SECON.2010.5453811
Shen, C., Xie, R., Zhang, L., Song, L., 2015. Small group people behaviour analysis based on temporal recursive trajectory identification, in: 2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW). Presented at the 2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW), pp. 1–6. doi:10.1109/ICMEW.2015.7169864
Shila, D.M., Srivastava, K., O'Neill, P., Reddy, K., Sritapan, V., 2016. A multi-faceted approach to user authentication for mobile devices #x2014; Using human movement, usage, and location patterns, in: 2016 IEEE Symposium on Technologies for Homeland Security (HST). Presented at the 2016 IEEE Symposium on Technologies for Homeland Security (HST), pp. 1–6. doi:10.1109/THS.2016.7568944
Smith, R.T., Minton, R.B., 2001. Calculus, 2nd Ed edition. ed. McGraw-Hill Education, Boston.
Software Developer's Journal, n.d., Artificial Intelligence in Games - CodeProject [Online]. Available at: http://www.codeproject.com/Articles/14840/Artificial-Intelligence-in-Games [Accessed 16 Apr 2014].
Stover, C. and Weisstein, E. W., 2016. "Euclidean Space." [Online] Available at: http://mathworld.wolfram.com/EuclideanSpace.html [Accessed 24 May 2016].
Sun, M., Wang, J., 2008. Methodology of simulation science and technology, in: Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing, 2008. ICSC 2008. Presented at the Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing, 2008. ICSC 2008, pp. 989–994. doi:10.1109/ASC-ICSC.2008.4675509
Sung, M., Gleicher, M. and Chenney, S. (2004), Scalable behaviours for crowd simulation. Computer Graphics Forum, 23: 519–528. doi: 10.1111/j.1467-8659.2004.00783.x
Tajima, Y., Nagatani, T., 2001. Scaling behaviour of crowd flow outside a hall. Physica A: Statistical Mechanics and its Applications 292, 545–554. doi:10.1016/S0378-4371(00)00630-0
Tebjan, H., 2011, 3d Vector Mathematics [Online]. Available at: http://vvvv.org/documentation/3d-vector-mathematics [Accessed 15 Apr 2014].
Terzimehic, T., Silajdzic, S., Vajnberger, V., Velagic, J., Osmic, N., 2011. Path finding simulator for mobile robot navigation, in: 2011 XXIII International Symposium on Information, Communication and Automation Technologies (ICAT). Presented at the 2011 XXIII International Symposium on Information, Communication and Automation Technologies (ICAT), pp. 1–6. doi:10.1109/ICAT.2011.6102086
Tran, D., Yuan, J., 2011. Optimal spatio-temporal path discovery for video event detection, in: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3321–3328. doi:10.1109/CVPR.2011.5995416
Treuille, A., Cooper, S., Popović, Z., 2006. Continuum Crowds, in: ACM SIGGRAPH 2006 Papers, SIGGRAPH '06. ACM, New York, NY, USA, pp. 1160–1168. doi:10.1145/1179352.1142008
Urban, C. and Schmidt, B., 2001. PECS–Agent-Based Modelling of Human Behaviour. In Emotional and Intelligent–The Tangled Knot of Social Cognition, AAAI Fall Symposium Series, North Falmouth, MA. www. or. unipassau. de/5/publik/urban/CUrban01. pdf.
Varas, A., Cornejo, M.D., Mainemer, D., Toledo, B., Rogan, J., Muñoz, V., Valdivia, J.A., 2007. Cellular automaton model for evacuation process with obstacles. Physica A: Statistical Mechanics and its Applications 382, 631–642. doi:10.1016/j.physa.2007.04.006

Vieira, M.R., Barbosa, L., Kormáksson, M., Zadrozny, B., 2015. USapiens: A System for Urban Trajectory Data Analytics, in: 2015 16th IEEE International Conference on Mobile Data Management. Presented at the 2015 16th IEEE International Conference on Mobile Data Management, pp. 255–262. doi:10.1109/MDM.2015.35

Wang, C., Li, C., Liu, Y., Cui, J., Zhang, T., 2011. Behaviour-Based Simulation of Real-Time Crowd Evacuation, in: 2011 12th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics). Presented at the 2011 12th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics), pp. 456–461. doi:10.1109/CAD/Graphics.2011.62

Wang, M., Lu, H., 2012. Research on Algorithm of Intelligent 3D Path Finding in Game Development, in: 2012 International Conference on Industrial Control and Electronics Engineering (ICICEE). Presented at the 2012 International Conference on Industrial Control and Electronics Engineering (ICICEE), pp. 1738–1742. doi:10.1109/ICICEE.2012.460

Wang, X., Ji, Q., 2016. Hierarchical Context Modeling for Video Event Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence PP, 1–1. doi:10.1109/TPAMI.2016.2616308

Wang, Y., Huang, C., Shan, J., 2015. An initial study on college students' daily activities using GPS trajectories, in: 2015 23rd International Conference on Geoinformatics. Presented at the 2015 23rd International Conference on Geoinformatics, pp. 1–6. doi:10.1109/GEOINFORMATICS.2015.7378686

Weinland, D., Ronfard, R. and Boyer, E., 2006. Free viewpoint action recognition using motion history volumes. Computer vision and image understanding, 104(2), pp.249-257.

Weisstein, E. W., 2016. "Coordinates." [Online] Available at: Available at: http://mathworld.wolfram.com/Coordinates.html [Accessed 25 May 2016].

Weisstein, E. W., 2016. "Distance." [Online] Available at: Available at: http://mathworld.wolfram.com/Distance.html [Accessed 22 May 2016].

Weisstein, E. W., 2016. "Euclidean Plane." [Online] Available at: http://mathworld.wolfram.com/EuclideanPlane.html [Accessed 24 May 2016].

Weisstein, E. W., 2016. "Euler's Rotation Theorem." [Online] Available at: http://mathworld.wolfram.com/EulersRotationTheorem.html [Accessed 11 Mar 2015].

Weisstein, E. W., 2016."Perpendicular Vector." [Online] Available at: http://mathworld.wolfram.com/PerpendicularVector.html [Accessed 8 Sep 2015].

Weisstein, E. W., 2016. "Argand Diagram" [Online]. Available at: http://mathworld.wolfram.com/ArgandDiagram.html [Accessed 14 Apr 2014].

Weisstein, E. W., 2016. "Complex Number" [Online]. Available at: http://mathworld.wolfram.com/ComplexNumber.html [Accessed 22 Aug 2015].

Weisstein, E. W., 2016. "Euler Angles." [Online]. Available at: http://mathworld.wolfram.com/EulerAngles.html [Accessed 16 Jun 2016].

Weisstein, E. W., 2016. "Normal Vector" [Online]. Available at: http://mathworld.wolfram.com/NormalVector.html [Accessed 12 Aug 2014].

Weisstein, E. W., 2016. "Plane" [Online]. Available at: http://mathworld.wolfram.com/Plane.html [Accessed 16 Apr 2016].

Weisstein, E. W., 2016. "Polar Coordinates" [Online]. Available at: http://mathworld.wolfram.com/PolarCoordinates.html [Accessed 26 May 2016].

Weisstein, E. W., 2016. "Quaternion" [Online]. Available at: http://mathworld.wolfram.com/Quaternion.html [Accessed 14 Apr 2014].

Weisstein, E. W., 2016. "Sphere" [Online]. Available at: http://mathworld.wolfram.com/Sphere.html [Accessed 8 Jul 2016].

Weisstein, E. W., 2016. "Trigonometry" [Online]. Available at: http://mathworld.wolfram.com/Trigonometry.html [Accessed 26 May 2016].

Weisstein, E. W., 2016. "Vector Space" [Online] Available at: http://mathworld.wolfram.com/VectorSpace.html [Accessed 24 May 2016].

Williams, A., Barrus, S., Morley, R.K., Shirley, P., 2005. An Efficient and Robust Ray-Box Intersection Algorithm. Journal of Graphics Tools 10, 49–54. doi:10.1080/2151237X.2005.10129188

Wu, S., Moore, B.E., Shah, M., 2010. Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2054–2060. doi:10.1109/CVPR.2010.5539882

Xiang, T. and Gong, S., 2006. Beyond tracking: Modelling activity and understanding behaviour. International Journal of Computer Vision, 67(1), pp.21-51.

Xiang, T. and Gong, S., 2006. Model selection for unsupervised learning of visual context. International Journal of Computer Vision, 69(2), pp.181-201.

Xiang, X., Bao, W., Tang, H., Li, J., Wei, Y., 2016. Vehicle detection and tracking for gas station surveillance based on AdaBoosting and optical flow, in: 2016 12th World Congress on Intelligent Control and Automation (WCICA).

Presented at the 2016 12th World Congress on Intelligent Control and Automation (WCICA), pp. 818–821. doi:10.1109/WCICA.2016.7578324

Yang, J., Jiang, Y.G., Hauptmann, A.G. and Ngo, C.W., 2007, September. Evaluating bag-of-visual-words representations in scene classification. In Proceedings of the international workshop on Workshop on multimedia information retrieval (pp. 197-206). ACM.

Yordanova, K., 2011. Modelling Human Behaviour Using Partial Order Planning Based on Atomic Action Templates, in: 2011 7th International Conference on Intelligent Environments (IE). Presented at the 2011 7th International Conference on Intelligent Environments (IE), pp. 338–341. doi:10.1109/IE.2011.27

Yu, C.P., Le, H., Zelinsky, G., Samaras, D., 2015. Efficient Video Segmentation Using Parametric Graph Partitioning, in: 2015 IEEE International Conference on Computer Vision (ICCV). Presented at the 2015 IEEE International Conference on Computer Vision (ICCV), pp. 3155–3163. doi:10.1109/ICCV.2015.361

Zagal, J.P., Mateas, M., Fernández-Vara, C., Hochhalter, B. and Lichti, N., 2007. Towards an ontolqgical language for game analysis. Worlds in Play: International Perspectives on Digital Games Research, 21, p.21.

Zhang, D., 2015. Recognizing human actions via silhouette image analysis, in: The 27th Chinese Control and Decision Conference (2015 CCDC). Presented at the The 27th Chinese Control and Decision Conference (2015 CCDC), pp. 5870–5874. doi:10.1109/CCDC.2015.7161859

Zhang, L., Zhou, C., 2007. Robot Optimal Trajectory Planning Based on Geodesics, in: IEEE International Conference on Control and Automation, 2007. ICCA 2007. Presented at the IEEE International Conference on Control and Automation, 2007. ICCA 2007, pp. 2433–2436. doi:10.1109/ICCA.2007.4376799

Zhang, Z., Sun, R., Wang, X., Zhao, C., 2017. A Situational Analytic Method for User Behavior Pattern in Multimedia Social Networks. IEEE Transactions on Big Data PP, 1–1. doi:10.1109/TBDATA.2017.2657623

Zhao, M., Cai, W., Turner, S.J., 2015. Evaluation of Crowd Models in Low Density Scenarios Using Real-World Crowd Data, in: 2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT). Presented at the 2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 1–9. doi:10.1109/DS-RT.2015.25

Zhao, Z., Chen, M., Yang, L., Fan, Z., Ma, L., 2010. 2D to 3D video conversion based on interframe pixel matching, in: The 2nd International Conference on Information Science and Engineering. Presented at the The 2nd International Conference on Information Science and Engineering, pp. 3380–3383. doi:10.1109/ICISE.2010.5691623

Zhong, J., Luo, L., Cai, W., Lees, M., 2014. EA-based evacuation planning using agent-based crowd simulation, in: Simulation Conference (WSC), 2014 Winter. Presented at the Simulation Conference (WSC), 2014 Winter, pp. 395–406. doi:10.1109/WSC.2014.7019906

Zhou, C., Xu, Z., Huang, B., 2010. Activity Recognition from Call Detail Record: Relation Between Mobile Behavior Pattern and Social Attribute Using Hierarchical Conditional Random Fields, in: Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom). Presented at the Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom), pp. 605–611. doi:10.1109/GreenCom-CPSCom.2010.141

Zhu, H., Mastorakis, N.E., Zhuang, X.D., 2015. A Method of Effective Background Estimation for Video Sequences with Dense Moving Objects, in: 2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI). Presented at the 2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pp. 99–104. doi:10.1109/MCSI.2015.13

Zimmermann, R., Kim, S.H., Shahabi, C., n.d. Active Key Frame Selection for 3d Model Reconstruction from Crowdsourced Geo-Tagged Videos.

Appendix

A.1. Theoretical foundation of 3D graphical modelling

The operational semantics of dynamic patterns defined in a language require implementation. While the language is focused on formal description of *what* is happening in the scene (events), it does not provide details about spatial existence of individuals and objects identified in the scene. To that end, the language does not describe *how* visible changes that are the motor of dynamic patterns occur. Adequate theory allowing modelling the visual appearance of objects and their movements is required. A natural candidate tool for this task is mathematics from the area of analytical geometry. The elementary vector calculus and related areas are the main vehicle of developing a plausible graphical model of visual scene. In this appendix only the fundamentals upon which theory behind implementation of a simulator are presented.

Analytical geometry for modelling the visual appearance of the world and its dynamics

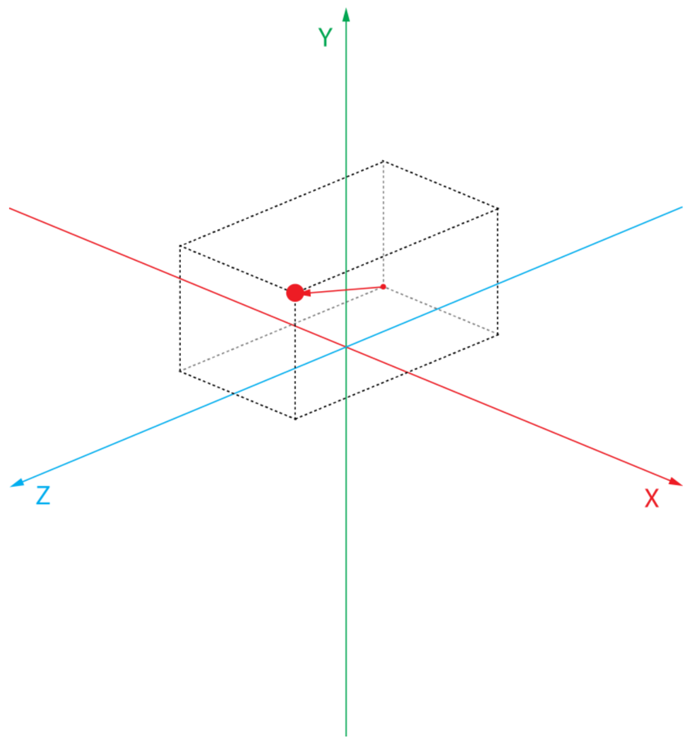


Figure A.1-1. A vector defined in three-dimensional Cartesian coordinate system

The visual scene can be perceived as a *space* in which one can observe the existence of objects, individuals and evolution of movements. The space is defined as *Euclidean* (or *Cartesian*) n-space of all n-tuples of real numbers $(x_1, x_2, \dots x_n)$ where the totality of this space is denoted by $\mathbb{R}^n$, i.e. $x_n \in \mathbb{R}^n$ (Stover and Weisstein, 2016). Every n-tuple is called a *point* (or *vector* because Euclidean space can be also considered a *vector space*). *"A vector space $\mathbb{V}$ is a set that is closed under finite vector addition and scalar multiplication"* (Weisstein, 2016). The Euclidean space $\mathbb{R}^n$ has a dimension of n and since the visual scene is considered to be three-dimensional it can be denoted by $\mathbb{R}^3$. The $\mathbb{R}^3$ space also encompasses *Euclidean plane,* which is a *two-dimensional Euclidean space denoted by* $\mathbb{R}^2$ (Weisstein, 2016). We will express vectors in a form of distances (i.e. *coordinates*) measured along three perpendicular axes, also called Cartesian coordinates (Weisstein, 2016) as shown in Figure A.1-1. A dashed bounding box visually illustrates the location, length and direction of the vector.

Vectors

Vectors are used to represent points in space, directions, locations of objects and their orientations (Lengyel, 2003). In this and following chapters, for the sake of clarity, a *point* is a vector that represents a point in three-dimensional space and a *direction vector* or simply *vector* is a vector that represents a direction in three-dimensional space. Points can be written as ordered 3-tuples (*triplets*) of *coordinates* (or *components*) in round brackets, e.g. $A(x, y, z)$ and vectors can be written as ordered triplets of coordinates in square brackets with an arrow accent, e.g. $\vec{u} = [x, y, z]$. In this appendix, points and vectors are represented using one of these notations. Vectors are represented in a form of a directed line segment in coordinate system (Figure A.1-1).

The central point is located at the origin O, which always lies at coordinate $(0,0,0)$ and where all three axes X, Y and Z intersect. In order to move from one location to another one has to specify the beginning and the end of a directed line segment. For example, if one wants to move from point A located at the origin O $(0,0,0)$ to point B located at coordinate $(5,5,2)$ then it can be done so by writing $\vec{u} = [0 \gg 5, 0 \gg 5, 0 \gg 2]$. This provides one with not only the distance between those two points but also the direction in which the movement should be made which in this case is 5 units right on X axis, 5 units up on Y axis and 2 units towards the screen on Z axis.

One can add and subtract vectors as well as multiply them by a scalar. If one assumes that $\vec{u} = [u_x, u_y, u_z]$ and $\vec{v} = [v_x, v_y, v_z]$ are two vectors and a is a real number then these operations can be performed with eq.A-1, eq.A-2 and eq.A-3.

$$\vec{u} + \vec{v} = [u_x + v_x, u_y + v_y, u_z + v_z] \quad \textbf{(eq. A-1)}$$

$$\vec{u} - \vec{v} = [u_x - v_x, u_y - v_y, u_z - v_z] \quad \textbf{(eq. A-2)}$$

$$a\vec{u} = [au_x, au_y, au_z] \quad \textbf{(eq. A-3)}$$

Vectors possess a set of arithmetic operations properties (Lengyel, 2003). For any given vectors $\vec{u}$, $\vec{v}$ and $\vec{w}$ and two real numbers (*scalars)* a and b then the following can be expressed:

1) $\vec{u} + \vec{v} = \vec{v} + \vec{u}$
2) $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$
3) $a(b\vec{v}) = (ab)\vec{v}$
4) $(a + b)\vec{v} = a\vec{v} + b\vec{v}$
5) $a(\vec{u} + \vec{v}) = a\vec{u} + a\vec{v}$

6) $1 * \vec{v} = \vec{v}$
7) $0 * \vec{v} = \vec{0}$

Where, $\vec{0}$ is a zero vector defined as $[0,0,0]$.

A vector, which starts at point $A(x_1, y_1, z_1)$ and finishes at point $B(x_2, y_2, z_2)$ can be calculated by subtracting their respective component values (see eq.A-4).

$$[x_2 - x_1, y_2 - y_1, z_2 - z_1] \qquad \textbf{(eq. A-4)}$$

It is also referred to as a *bound vector* and is written as $\overrightarrow{AB}$. If the beginning point of a vector is unknown then it is positioned at the origin of a coordinate system and finished at a point dictated by this vector. Such a vector is then called a *free vector*. Generally, if two points $A(x_1, y_1, z_1)$ and $B(x_2, y_2, z_2)$ are the *initial point* and *terminal point* of a vector respectively, then one can calculate that vector position and direction in 3D space (see eq.A-5).

$$\overrightarrow{AB} = [x_2 - x_1, y_2 - y_1, z_2 - z_1] \qquad \textbf{(eq. A-5)}$$

Wherein, there exists some equality between initial and terminal points of a vector (see eq.A-6, eq.A-7).

$$\overrightarrow{AA} = \vec{0} \qquad \textbf{(eq. A-6)}$$

$$\overrightarrow{BA} = -\overrightarrow{AB} \qquad \textbf{(eq. A-7)}$$

A free vector $[v_x, v_y, v_z]$ becomes a bound vector if one knows the location of its initial point $A(x_1, y_1, z_1)$. The terminal point of this vector will then have coordinates at $B(x_1 + v_x, y_1 + v_y, z_1 + v_z)$. For example, the free vector $\vec{u} = [1,2,3]$ can be a bound vector of two free vectors if $A(1,1,1)$ and $B(2,3,4)$ are known. This is because when one subtracts their components the resulting vector will be equal to $\vec{u}$: $\overrightarrow{AB} = [2 - 1, 3 - 1, 4 - 1] = [1,2,3] = \vec{u}$.

The midpoint of a vector $\overrightarrow{AB}$, where $A(x_1, y_1, z_1)$ is the initial point and $B(x_2, y_2, z_2)$ is the terminal point, is a point $S(x_s, y_s, z_s)$, whose components can be calculated with eq.A-8.

$$x_s = (x_1 + x_2)/2 \qquad y_s = (y_1 + y_2)/2 \qquad z_s = (z_1 + z_2)/2 \qquad \textbf{(eq. A-8)}$$

On the basis of Pythagoras theorem, one can calculate the *magnitude* (or *length*) of a vector $\vec{u} = [u_x, u_y, u_z]$ denoted by $|\vec{u}|$ (see eq. A-9) (Lengyel, 2003). However, $\| \vec{u} \|$ notation will be used in the following sections to distinguish the magnitude of a vector from *absolute value*. The magnitude of a vector in a context of visual scene is used to measure spatial distances between different objects and individuals.

$$\| \vec{u} \| = \sqrt{u_x^2 + u_y^2 + u_z^2}$$ **(eq. A-9)**

One can use eq. A-9 to calculate the distance between two points $A(x_1, y_1, z_1)$ and $B(x_2, y_2, z_2)$ (see eq. A-10) (Weisstein, 2016).

$$\| \overrightarrow{AB} \| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$ **(eq. A-10)**

A vector $\vec{v}$ that has at least one nonzero component can be normalized. The process of *normalization* can be useful in calculations that require only a direction of a vector and are not concerned about its magnitude. A normalized vector is often called *unit vector* and it is calculated by dividing its components by its magnitude, i.e. $\overrightarrow{v_n} = 1/\| \vec{v} \| \, \vec{v}$.

Occasionally, vectors are described with *standard basis vectors* notation for convenience and geometric analysis. Since a visual scene defined in three-dimensional space is considered, the *standard basis* V_3 consists of three unit vectors, where each lies on one of the coordinate axes (see eq. A-11) (Smith and Minton, 2001).

(eq. A-11)

$$\vec{\imath} = [1,0,0] \qquad \vec{\jmath} = [0,1,0] \qquad \vec{k} = [0,0,1]$$

Where, $\| \vec{\imath} \| = \| \vec{\jmath} \| = \| \vec{k} \| = 1$

For any vector $\vec{v} \in V_3$, one can write it in a standard basis (see eq. A-12) (Smith and Minton, 2001).

$$\vec{v} = [v_x, v_y, v_z] = v_x i + v_y j + v_z k$$ **(eq. A-12)**

Matrices

In 3D graphics, matrices, next to vectors, are of a fundamental significance. Matrices are used for performing transformations of vectors from one coordinate space to another (Lengyel, 2003). Up to this moment, the presented vector calculus was used to define attributes of vectors such as position and direction in three-dimensional space described in a coordinate system C consisting of three coordinate axes and origin. In this system, a vector is written as a triplet of components where each component specifies how many units it needs to navigate along each axis from the origin point to reach its final point.

However, in 3D graphics it is often necessary to *translate* a set of vertices in three-dimensional space to a different coordinate space (e.g. in order to appropriately display a three-dimensional space in a two-dimensional space of a display screen). Considering another coordinate system C', "*in which the components* *of vectors* $[x', y', z']$ *can be expressed as linear functions of coordinates* $[x, y, z]$", one can perform a linear transformation from coordinate system C to coordinate system C' with (see eq. A-13) (Lengyel, 2003).

$$x'(x, y, z) = u_1x + v_1y + w_1z + \overrightarrow{t_1}$$
$$y'(x, y, z) = u_2x + v_2y + w_2z + \overrightarrow{t_2}$$
$$z'(x, y, z) = u_3x + v_3y + w_3z + \overrightarrow{t_3}$$
(eq. A-13)

Where,

x', y', z' are coordinates of a vector in a new coordinate system C',

$\vec{t}$ is a vector which represents the translation from the origin of C to the origin of C'.

$\vec{u}$, $\vec{v}$ and $\vec{w}$ vectors represent how the orientation of the coordinate axes is changed when transforming from C to C'.

The *linear transformation* (i.e. *linear equation*) in eq. A-13 can be represented in a more compact way in a form of a *matrix* (see eq. A-14) (Lengyel, 2003).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \overrightarrow{t_1} \\ \overrightarrow{t_2} \\ \overrightarrow{t_3} \end{bmatrix}$$
(eq. A-14)

Therefore a matrix, in a context of visual scene, encapsulates information about visual dynamic changes that can be applied to objects in that visual scene. The matrices provide indication about the level of impact that a given transformation will have on object upon its application. This means that a matrix is a convenient mathematical instrument for describing movements and changes of physical properties of individuals and objects in a visual scene.

Matrix is the arrangement of numbers (*elements*) into a two-dimensional array of n rows and m columns $(n \times m)$. Their general representation has a form presented in eq. A-15.

$$M = [a_{ij}] = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix}$$
(eq. A-15)

Where,

$$i = 1, \dots, n$$
$$j = 1, \dots, m$$

The element residing at position(i, j) in that array is called *an entry* in the matrix (Lengyel, 2003). As with vectors, matrices possess similar set of arithmetic operations properties (Lengyel, 2003). For any given matrices $n \times m$ F, G and H and any two scalars a and b the following can be expressed;

1) $F + G = G + F$
2) $(F + G) + H = F + (G + H)$
3) $a(bF) = (ab)F$
4) $a(F + G) = aF + aG$
5) $(a + b)F = aF + bF$

There are cases of matrices that can be characterised when they satisfy certain conditions. The most important characteristics of matrices and their properties are as follow:

1) A matrix, which number of columns is equal to the number of rows (i.e. $n = m$) is called a *square matrix* (Lengyel, 2003).

2) The values of entries in a matrix satisfying a condition $i = j$ are called *main diagonal entries* (Lengyel, 2003).

3) The *transpose* of an $n \times m$ matrix M (which is denoted by M^T) is an $m \times n$ matrix for which entry (i, j) is equal to M_{ji}, i.e. $M_{ij}^T = M_{ji}$ (Lengyel, 2003).

4) A matrix $n \times m$ is called the *identity matrix* (which is denoted as I) when $MI = IM = M$ for any matrix M (Lengyel, 2003).

5) A $n \times m$ matrix M is *invertible* if there exists a matrix (which is denoted by M^{-1}), such that $MM^{-1} = M^{-1}M = I$. The matrix M^{-1} is then called the *inverse* of a matrix M (Lengyel, 2003).

Vectors can be perceived as a special type of matrix because they can be represented in a form of matrices that have one-column and n-rows, i.e. $n \times 1$ (sometimes called *column vectors* (Lengyel, 2003)) or matrices that have one-row and m-columns, i.e. $1 \times m$. Since only 3D vectors are considered in this research, they have matrix forms presented in eq. A-16.

$$\vec{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \qquad \vec{u}^T = [u_x u_y u_z]$$ **(eq. A-16)**

Where, $\vec{u}^T$ is a transpose of vector $\vec{u}$.

The most relevant arithmetic operations on matrices and vectors have been summarised in Table A-1.

Table A.1-1. Matrix operations for transformations

Operation	Equation and generic example	Comments
Multiplying a matrix by scalar	$\alpha M = [\alpha a_{ij}]$ $2\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 2a_{11} & 2a_{12} & 2a_{13} \\ 2a_{21} & 2a_{22} & 2a_{23} \\ 2a_{31} & 2a_{32} & 2a_{33} \end{bmatrix}$	- M is any type of matrix - α is a scalar
Adding two matrices	$(M+N)_{ij} = M_{ij} + MN_{ij}$ $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{bmatrix}$	- *condition*: the number of rows and columns in matrix M is equal to corresponding number of rows and columns in matrix N - the operation results in a matrix, which number of rows and columns is equal to the number of rows and columns in matrix M and N (before the operation)
Multiplying a matrix by a matrix	$(MN)_{ik} = \sum_{j=1}^{l} m_{ij} n_{jk}$ $\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} =$ $= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \end{bmatrix}$	- *condition*: the number of columns in matrix M is equal to the number of rows in matrix N - the operation results in a matrix, which number of rows is equal to number of rows in matrix M and number of columns equal to number of columns of N
Multiplying vector by a matrix	$\vec{u}M = [u_1 \dots u_n] \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} =$ $= \begin{bmatrix} a_{11}u_1 + a_{12}u_2 + & \cdots & + a_{1n}u_n \\ \vdots & \vdots & \vdots \\ a_{m1}u_1 + a_{m2}u_2 + & \cdots & + a_{mn}u_n \end{bmatrix}$ $\vec{v}N = [v_1 v_2 v_3] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11}v_1 + a_{12}v_2 + a_{13}v_3 \\ a_{21}v_1 + a_{22}v_2 + a_{23}v_3 \\ a_{31}v_1 + a_{32}v_2 + a_{33}v_3 \end{bmatrix}$	- $\vec{u}$ is a vector called *constant vector* (Lengyel, 2003) - $\vec{v}$ is a 3D constant vector - M is a matrix called *coefficient matrix* (Lengyel, 2003) - N is a 3x3 coefficient matrix

The elementary transformation matrices that are commonly used in 3D graphics are the *scaling matrix*, the *translation matrix* and the *rotation matrix.*

Scaling Transformation

In order to scale a vector $\vec{u}$ by a factor a in three-dimensional space, one can perform the operation with the use of a scaling matrix (see eq. A-17).

$$\vec{u}' = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} [u_x u_y u_z] = \begin{bmatrix} au_x \\ au_y \\ au_z \end{bmatrix} \qquad \textbf{(eq. A-17)}$$

Where, $\vec{u}'$ is the result vector of the transformation.

When one scales a vector by the same exact amount along three axes X, Y and Z, it is called a *uniform scale.* A uniform scale is always applied when main diagonal entries of the matrix are the same (see eq. A-17). Analogically, when diagonal entries are different, i.e. when one scales a vector by different amounts along three axes X, Y and Z then it is called a *non-uniform scale* (see eq. A-18).

$$\vec{u}' = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} [u_x u_y u_z] = \begin{bmatrix} au_x \\ bu_y \\ cu_z \end{bmatrix} \qquad \textbf{(eq. A-18)}$$

Where, $\vec{u}'$ is the result vector of the transformation.

Scaling is a convenient method of adjusting the size of the graphical world and all its elements. For instance, it might be necessary to change the proportions of objects in a three-dimensional space so that they correspond to their natural sizes relative to graphical world and its boundaries.

Translation Transformations

In order to translate a point A from one coordinate space C to another coordinate space C' there is a need to perform an operation similar to the one shown in eq. A-19.

$$A' = MA + \vec{t} \qquad \textbf{(eq. A-19)}$$

Where, A' is a point after the translation, M is some 3x3 matrix and $\vec{t}$ is a vector that translates origin point of coordinate space C to the origin point of coordinate space C'.

As noted by Lengyel (2003), in order to perform, for example, two translation operations such as the ones shown in eq.A-19, one would have to keep track of not only matrix components but also translation vector components at each stage of the procedure (see eq.A-20).

$$A' = M_2(M_1 A + \vec{t}_1) + \vec{t}_2 = (M_2 M_1)A + M_2\vec{t}_1 + \vec{t}_2 \qquad \textbf{(eq. A-20)}$$

This can produce several complexities and overheads if one considers greater number of translations to be performed in a relatively short amount of time, e.g. if there is a need to perform multiple translation operations in order to simulate movement of objects or individuals in a visual scene. To overcome this problem and represent this type of transformation in a more compact mathematical structure, one needs to extend vectors to *homogenous coordinates*, i.e. add the fourth component (often called the w-coordinate), to the existing three and setting its value to one (see eq.A-21) (Lengyel, 2003).

$$\vec{u} = [x, y, z, w] = [x, y, z, 1] \qquad \textbf{(eq. A-21)}$$

Since the multiplication operation of two matrices requires that the number of columns in multiplier matrix has to be equal to the number of rows in the multiplicand matrix (Table A-1), there is a need to extend 3x3 matrix to 4x4 matrix. The 4x4 matrix F that is used for translation transformation contains 3x3 matrix M and 3D translation vector $\vec{t}$ from eq. A-20 so that it has a form shown in eq. A-22.

$$F = \begin{bmatrix} M & \vec{t} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & \vec{t}_x \\ M_{21} & M_{22} & M_{23} & \vec{t}_y \\ M_{31} & M_{32} & M_{33} & \vec{t}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(eq. A-22)}$$

Multiplication of a 4D vector $\vec{u}$ (see eq. A-21) by 4x4 matrix F constructed in eq. A-22 results in a vector, whose components x, y and z are transformed in exactly the same way as in eq. A-19 leaving coordinate w set to 1. The advantage of this approach is that one can eliminate convoluted expressions (see eq.A-20) and enclose several translation transformations in a single mathematical structure. The multiplication of two types of transforms defined in 4x4 matrix M_1 and 4x4 matrix M_2 results in a matrix F that is equivalent to both, i.e. $F = M_1 M_2$ (Lengyel, 2003).

In section 4.1 a distinction between a point and a vector was made for multiple reasons. One of them is that a direction of a vector should remain the same even after the translation transformation (Lengyel, 2003). To achieve this, one has to extend a vector to homogenous coordinates and set its w-coordinate to 0. By doing so, one can force 4x4 matrix F to disregard its fourth column and apply only its upper left 3x3 part to a vector, leaving its direction intact (Lengyel, 2003).

If one considers the difference between two 4D homogeneous coordinates points $A(x_1, y_1, z_1, 1)$ and $B(x_2, y_2, z_2, 1)$, the resulting direction vector $\vec{u}$ will have its w-coordinate set to 0 (see eq. A-23).

$$\overrightarrow{AB} = [x_2 - x_1, y_2 - y_1, z_2 - z_1, 1 - 1] = [x', y', z', 0] = \vec{u} \qquad \textbf{(eq. A-23)}$$

Where, x', y', z' are the components of a new 4D vector $\vec{u}$ after the subtraction.

The value of vector $\vec{u}$ in eq. A-23 means that if one wants to translate a vector representing a point in three-dimensional space, there is a need to set its w-coordinate to 1 during multiplication operation. On the other hand, if one wants to translate a vector representing a direction in three-dimensional space without the risk of changing its direction, then there is a need to set its w-coordinate to 0 during multiplication operation.

Translation is the most fundamental operation that is used in the development of graphical world and its dynamic aspects. It allows positioning objects within world boundaries, dynamically move these objects from one location to another, correctly present three-dimensional space on a two-dimensional display screen, etc.

Orthogonal Matrices

3x3 matrices used for transformations in 3D graphics are *orthogonal* if and only if their inverse is equal to their transpose ($M^{-1} = M^T$) (Lengyel, 2003). Orthogonal matrices have the property of preserving magnitudes and angles during transformations of vectors. In case of visual scene, it means preserving the physical appearance of individuals and objects after their transformation in a coordinate system, e.g. the execution of dynamics of their movement does not result in deformation of their physical graphical appearance. In order for a matrix M to preserve a magnitude of a vector $\vec{u}$ it must satisfy the condition presented in eq. A-24.

$$\| M\vec{u} \| = \| \vec{u} \| \qquad \textbf{(eq. A-24)}$$

A matrix M, which preserves magnitudes and angles between vectors $\vec{u}$ and $\vec{v}$ has been presented in eq.A-25 (Lengyel, 2003).

$$(M\vec{u}) \cdot (M\vec{v}) = \vec{u} \cdot \vec{v} \qquad \textbf{(eq. A-25)}$$

The concept of an *angle between vectors* and its calculations in coordinate system has been presented in more details in section 4.1.4.

Analytical geometry allows one to calculate positions, sizes and establishing the boundaries of the world and its objects while *algebra* provides means to calculate a variety of transformations of those properties. However, those two branches of mathematics do not provide a simple method for performing calculations, which involve *angular measurements*. For example, it would be difficult to calculate orientations of individuals, i.e. a direction they are facing at any given point in time without measuring an angle between their last orientation and the current one. Hence it is necessary to introduce a mathematical apparatus that provides a convenient method for describing and calculating angles in three-dimensional space.

Trigonometry as a complementary tool for developing graphical model of the world

"The study of angles and of the angular relationships of planar and three-dimensional figures is known as trigonometry" (Weisstein, 2016). The concepts from the area of trigonometry are used to describe various

angular measurements in three-dimensional space in further sections. A great repository of trigonometry can be found in many mathematical text-books (Kowalczyk, Niedzialomski and Obczynski, 2011; Weisstein, 2016; Cewe et al., 2015). The focus of this section and all its sub-sections is made on presenting only crucial relationships between trigonometry and definitions of angles in three-dimensional space.

Trigonometric Pythagoras theorem

There exists an important relation between sine and cosine functions, which is shown in Figure A.1-2. From any given angle one can calculate the coordinates of a point lying on a circle’s circumference. Such coordinates are called *polar coordinates* (Weisstein, 2016). This property can be utilised, for instance, for approximation of individual’s focus in visual scene or finding correlation between objects by calculating their coordinates in situations where an angle is known only.

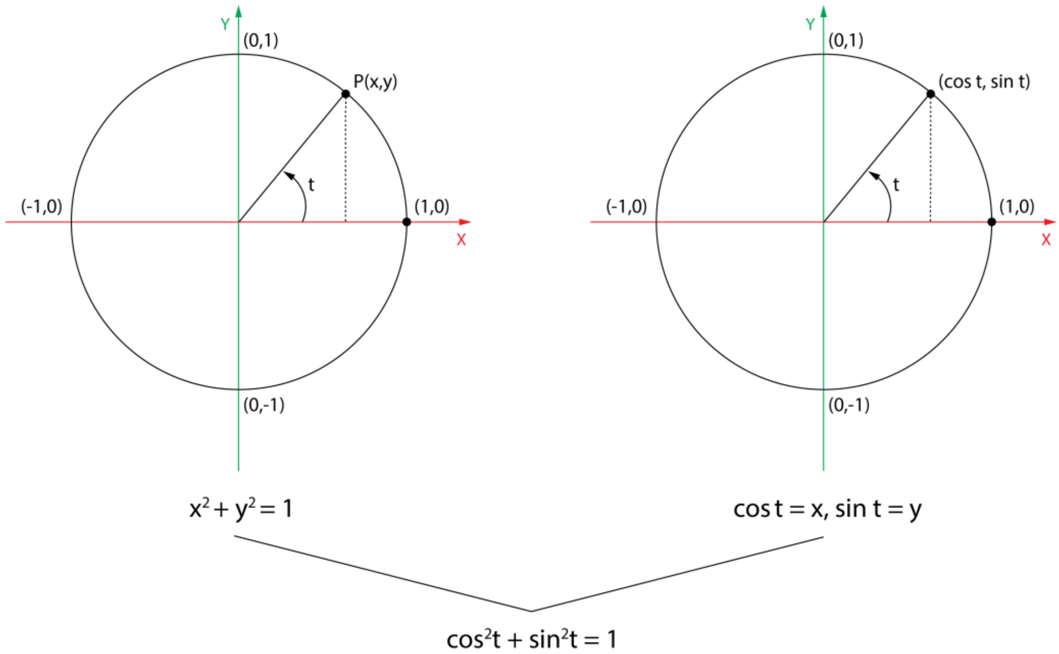


Figure A.1-2. The relation between sine and cosine functions for any given angle t (Kowalczyk, Niedzialomski and Obczynski, 2011)

For the point $P(x, y)$ that lies on a circle's circumference, from the definition of those functions we have $\cos t = x$ and $\sin t = y$. When one inputs these values into an equation for the length of a circle radius, eq. A-26 yields.

$$(\cos t)^2 + (\sin t)^2 = 1 \qquad \textbf{(eq. A-26)}$$

The values presented in eq. A-27 are for all arguments from the range of $[0,2\pi]$;

$$\tan t = \frac{\sin t}{\cos t} \qquad \cot t = \frac{\cos t}{\sin t} \qquad \tan t * \cot t = 1 \qquad \textbf{(eq. A-27)}$$

Utilisation of trigonometry in descriptions of three-dimensional world

Trigonometry provides a convenient mathematical method for describing, measuring and calculating angles on two-dimensional plane. Calculations, which involve angles, provide the ability to specify multiple angular characteristics of objects and their dynamics in three-dimensional space, e.g. one can calculate a *rotation* of an object with respect to coordinate system about any of the three axes X, Y or Z ultimately defining its *orientation*.

However, since trigonometry only deals with angles described in two-dimensional planes its concepts need to be extended to three-dimensional spaces in which visual scene is defined.

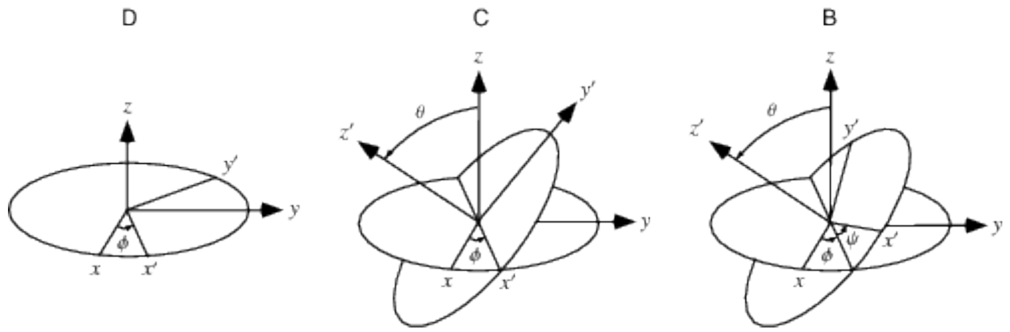


Figure A.1-3. Euler Angles (Weisstein, 2016)

According to Euler's rotation theorem, "*any arbitrary rotation can be described by only three parameters"* (Weisstein, 2016). In regards to visual scene, it means that any rotation in three-dimensional space can be described using three angles about all three axes X, Y, Z. Concepts of trigonometry that allow calculating angles defined on two-dimensional planes are fundamental components of calculating an arbitrary angle defined in three-dimensional space as it can be clearly shown in Figure A.1-3.

Trigonometry in crucial vector, matrices and quaternion calculations

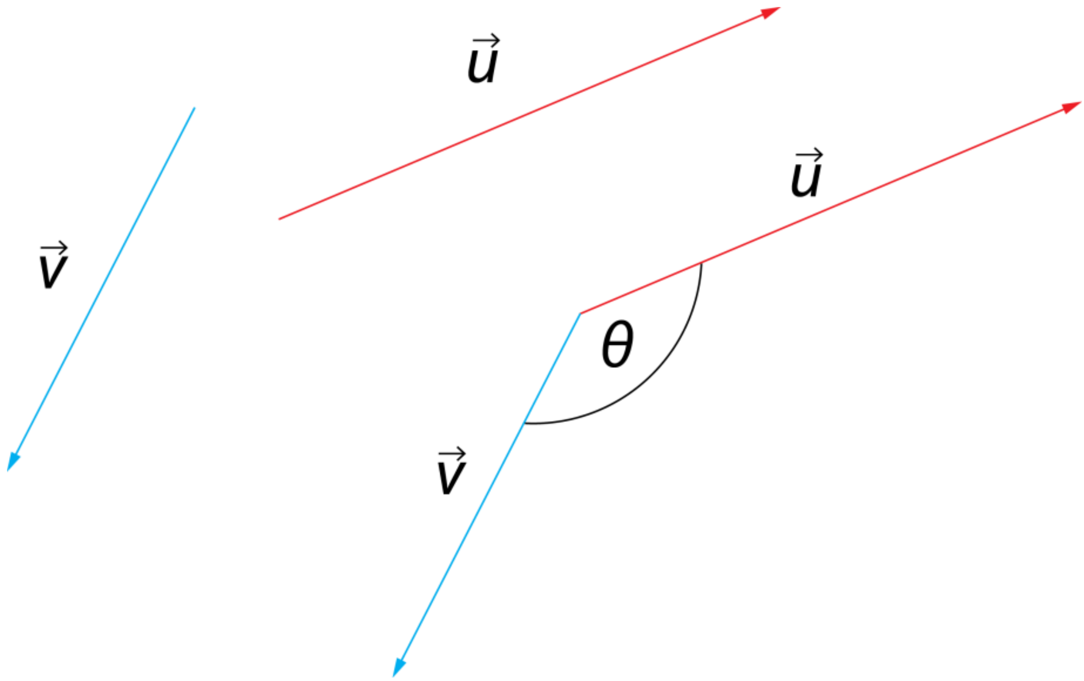


Figure A.1-4. *"An angle between two non-zero vectors $\vec{u}$ and $\vec{v}$ is a convex angle, whose one arm has a direction of vector $\vec{u}$ and a second arm has a direction of vector $\vec{v}$"* (Cewe et al., 2015)

An angle between two vectors, like the one shown in Figure A.1-4 is denoted by $\theta = \angle(\vec{u}, \vec{v})$ in further sections.

Dot product

The *dot* (or *scalar*) product of two non-zero vectors $\vec{u} = [u_x, u_y, u_z]$ and $\vec{v} = [v_x, v_y, v_z]$ is a real number. The equation, which needs to be satisfied by a dot product of two vectors, is presented in eq. A-28 (Lengyel, 2003).

$$\vec{u} \cdot \vec{v} = \| \vec{u} \| \| \vec{v} \| \cos(\angle(\vec{u}, \vec{v})) \qquad \textbf{(eq. A-28)}$$

Where, $\angle(\vec{u}, \vec{v})$ is an angle between vectors $\vec{u}$ and $\vec{v}$. If one of the vectors in that equation is equal to 0 then $\vec{u} \cdot \vec{v} = 0$.

The dot product can also be calculated with an alternative equation presented in eq.A-29 (Lengyel, 2003).

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{3} u_i \, v_i \qquad \textbf{(eq. A-29)}$$

Having two non-zero vectors, it is possible to calculate $\cos(\angle(\vec{u}, \vec{v})$. A cosine of an angle between two vectors can be derived from eq. A-28 as shown in eq. A-30.

$$\cos(\angle(\vec{u},\vec{v})) = \frac{\vec{u}\cdot\vec{v}}{\parallel\vec{u}\parallel\parallel\vec{v}\parallel} = \frac{u_x v_x + u_y v_y + u_z v_z}{\sqrt{{u_x}^2 + {u_y}^2 + {u_z}^2} * \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}} \quad \textbf{(eq. A-30)}$$

There are few properties of the scalar products. For any given vectors $\vec{u}$, $\overrightarrow{u_1}$, $\overrightarrow{u_2}$, $\vec{v}$ and any real numbers a and b the following laws can be expressed:

1) $\vec{u}\cdot\vec{u} \geq 0 \; and \; (\vec{u}\cdot\vec{u} = 0 \iff \vec{u} = \vec{0})$
2) $\parallel\vec{u}\parallel^2 = \vec{u}\cdot\vec{u}$
3) $\vec{u}\cdot\vec{v} = \vec{v}\cdot\vec{u}$
4) $(\overrightarrow{u_1} + \overrightarrow{u_2})\cdot\vec{v} = \overrightarrow{u_1}\cdot\vec{v} + \overrightarrow{u_2}\cdot\vec{v}$
5) $(a\vec{u})\cdot(b\vec{v}) = ab(\vec{u}\cdot\vec{v})$

In a two-dimensional plane, a vector is *perpendicular* (or *orthogonal*) to any given vector when one is rotated $90°$ counter clockwise and the other rotated $90°$ clockwise forming a right angle between them as shown in Figure A.1-5 (Weisstein, 2016).

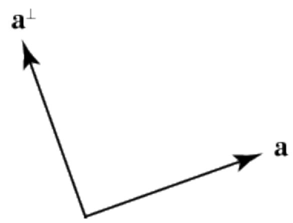


Figure A.1-5. The perpendicularity (or orthogonality) of two vectors in a two-dimensional plane (Weisstein, 2016)

In three-dimensional space, there are infinite numbers of vectors perpendicular to any given vector (Weisstein, 2016). When one of the corresponding components of two 3D vectors are equal (e.g. $\vec{v}_z = \vec{u}_z = 1$), then one can state that they are perpendicular on a two-dimensional plane defined in three-dimensional space. The perpendicularity of two vectors is denoted as $\vec{u} \perp \vec{v}$ ("$\vec{u}$ *is perpendicular to* $\vec{v}$") in further sections.

One can determine the difference in two vectors' directions by evaluating the value of their dot product. If the dot product of two vectors is positive then they are pointing in similar direction and when it is negative they are pointing in almost opposite directions. In addition, there exists formula for determining when two non-zero vectors are perpendicular to each other. $\vec{u} = [u_x, u_y, u_z]$ and $\vec{v} = [v_x, v_y, v_z]$ are perpendicular, if and only if their dot product is equal to 0, i.e. $u \cdot v = 0$ (see eq. A-31) (Lengyel, 2003).. This is due to the fact that cosine function at an angle $90°$ is equal to zero (Kowalczyk, Niedzialomski and Obczynski, 2011).

$$\vec{u} \perp \vec{v} \Leftrightarrow u \cdot v = 0 \Leftrightarrow u_x v_x + u_y v_y + u_z v_z = 0 \quad \textbf{(eq. A-31)}$$

The dot product is an essential component of pattern recognition algorithms (see Chapter 5). With the use of dot product it is possible to determine the average direction of individual's movement or aggregated flow motion of group of individuals.

Cross product

"Two non-zero vectors are parallel, when each of these vectors is a multiple of a second by a real number different than zero" (Cewe et al., 2015). The parallelism of two vectors is denoted by $\vec{u} \parallel \vec{v}$ ("*$\vec{u}$ is parallel to $\vec{v}$*") in further sections.

In order to check whether two vectors are parallel to each other one has to calculate a *determinant* (see eq.A-32). Two vectors are parallel, when their determinant is equal to 0.

$$det(\vec{u}, \vec{v}) = u_x v_y - u_y v_x \quad \textbf{(eq. A-32)}$$

Where, if

$$\vec{u} \parallel \vec{v} \Leftrightarrow det(\vec{u}, \vec{v}) = 0$$

then two vectors are parallel to each other.

However, eq. A-32 only applies to two-dimensional space. In three-dimensional space, to check whether two vectors are parallel to each other one can use the magnitude of their *cross product.* The cross product of two vectors is a vector that is perpendicular to both of them and can be calculated with eq. A-33 (Lengyel, 2003) while the determinant (the magnitude of the cross product) can be calculated with eq. A-34 (Mizerski et al., 2014). The determinant can be used to check whether two vectors are parallel by checking its value against 0.

$$\vec{u} \times \vec{v} = \left[u_y v_z - u_z v_y,\ u_z v_x - u_x v_z,\ u_x v_y - u_y v_x\right] \quad \textbf{(eq. A-33)}$$

$$\parallel \vec{u} \times \vec{v} \parallel = \parallel \vec{u} \parallel \parallel \vec{v} \parallel \sin(\angle(\vec{u}, \vec{v})) \quad \textbf{(eq. A-34)}$$

Where, if

$$\vec{u} \parallel \vec{v} \Leftrightarrow \parallel \vec{u} \times \vec{v} \parallel = 0$$

then two vectors are parallel to each other.

Eq.A-34 states that when angle $\angle(\vec{u}, \vec{v})$ is $0°$ or $180°$ its sin value is equal to 0 (Kowalczyk, Niedzialomski and Obczynski, 2011). In such a case, when this value is put back to the equation it will always result in a zero vector $\vec{0}$, whose magnitude is 0.

As it was mentioned before, cross product of two vectors is a vector perpendicular to both of them. However, on a closer examination it can be noticed that both different cases satisfy this condition as shown in Figure A.1-6.

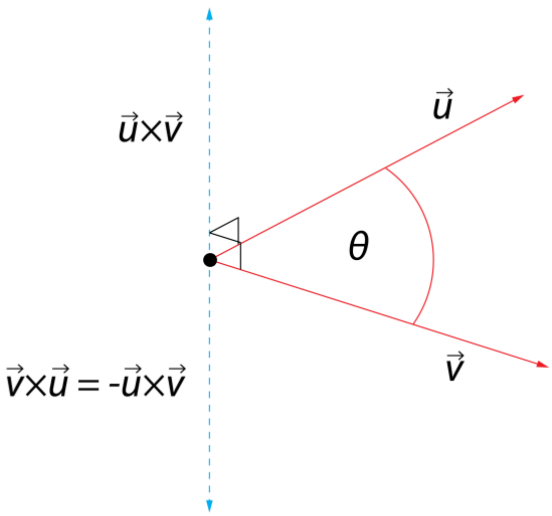


Figure A.1-6. Two different cases of a cross product satisfying its perpendicularity condition

The direction of the cross product of two vectors complies with the *right-hand rule*. The right hand rule states "*if you align the fingers of your right hand along the positive x-axis and then curl them toward the positive y-axis, your thumb will point in the direction of the positive z-axis."* (Smith and Minton, 2001). Following this rule, the positive z-axis (thumb) dictates the direction of a cross product of two vectors in three-dimensional space.

Cross product is usually used for finding the *normal vector* of an object surface at particular point when two distinct vectors defining that surface are known. *"The normal vector, often simply called the "normal" to a surface is a vector which is perpendicular to the surface at a given point. When normals are considered on closed surfaces, the inward-pointing normal (pointing towards the interior of the surface) and outward-pointing normal are usually distinguished"* (Weisstein, 2016).

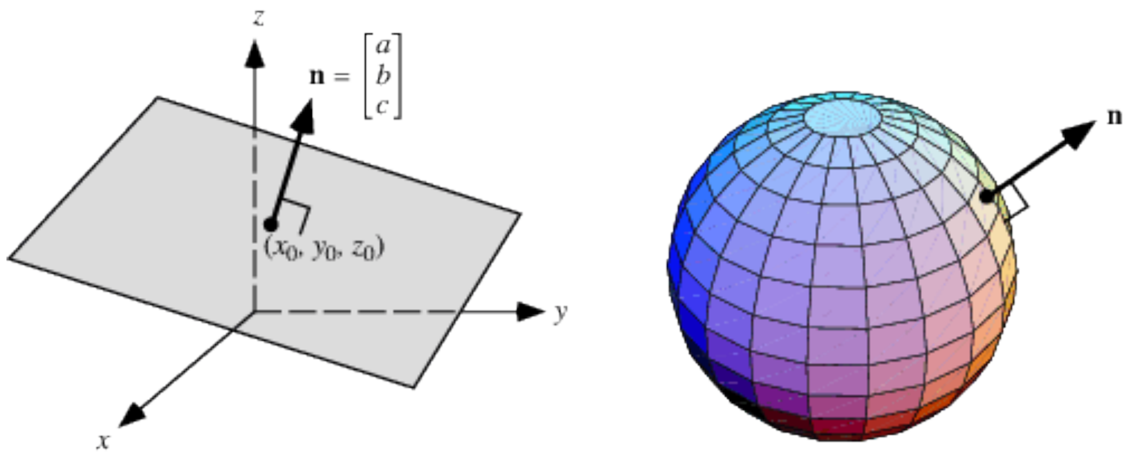


Figure A.1-7. Normal vector (Weisstein, 2016).

Rotations matrices

With the use of trigonometry it is possible to construct matrices that would provide means for rotating vectors about given axis of Cartesian coordinate system. In addition to a set of matrices presented in Table A-1, there are three matrices corresponding to rotation about individual axis (Lengyel, 2003). Eq. A-35 presents rotation about an X, Y and Z axis in three dimensions.

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$ **(eq. A-35)**

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In order to rotate a vector $\vec{u}$ about, for instance, X-axis one has to multiply it by a rotation matrix $R_x(\gamma)$ (see eq. A-35). The rotation around X-axis of a vector $\vec{u}$ is presented in eq. A-36.

$$\vec{u}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$ **(eq. A-36)**

Where, $\vec{u}'$ is the vector after the rotation.

Two subsequent rotations can be calculated using two separate multiplications of the vectors by rotation matrices. However, these two operations are equivalent for calculating a single rotation through multiplying vectors by one matrix, which is the product of the two. Following this logic, one can calculate a rotation around every axis by making a product out of three rotation matrices (see eq. A-35) and then multiplying this product by a vector. Rotating vector $\vec{u}$ about every axis in three-dimensional space with one matrix operation is demonstrated in eq. A-37.

$$R(\alpha,\beta,\gamma) = R_z(\alpha)R_y(\beta)R_x(\gamma) =$$ **(eq. A-37)**

$$= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

$$\vec{u}' = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

Where, $\vec{u}'$ is the vector after the rotation.

The order of the multiplication of matrices has the impact on the final position and orientation of the vector as it is presented in eq. A-38.

(eq. A-38)

$$R(\gamma,\beta,\alpha) = R_x(\gamma)R_y(\beta)R_z(\alpha) =$$

$$= \begin{bmatrix} \cos\beta\cos\alpha & -\cos\beta\sin\alpha & \sin\beta \\ \cos\gamma\sin\alpha + \sin\gamma\sin\beta\cos\alpha & \cos\gamma\cos\alpha - \sin\gamma\sin\beta\sin\gamma & -\sin\gamma\cos\beta \\ \sin\gamma\sin\alpha - \cos\gamma\sin\beta\cos\alpha & \sin\gamma\cos\alpha + \cos\gamma\sin\beta\sin\alpha & \cos\gamma\cos\beta \end{bmatrix}$$

In addition to general rotation matrices, there exists a method for constructing a matrix, which allows rotation around any arbitrary axis specified by unit vector (see eq. A-39) (Lengyel, 2003). This matrix proved to be useful during the development of supportive constructs for the approximation of viewing direction angle of individuals in relation to their body parts positions and orientations.

$$R_a = \begin{bmatrix} (1-\cos\alpha){u_x}^2+\cos\alpha & (1-\cos\alpha)u_x u_y - \sin\alpha\, u_z & (1-\cos\alpha)u_x u_z + \sin\alpha\, u_y \\ (1-\cos\alpha)u_x u_y + \sin\alpha\, u_z & (1-\cos\alpha){u_y}^2+\cos\alpha & (1-\cos\alpha)u_y u_z - \sin\alpha\, u_x \\ (1-\cos\alpha)u_x u_z - \sin\alpha\, u_y & (1-\cos\alpha)u_y u_z + \sin\alpha\, u_x & (1-\cos\alpha){u_z}^2+\cos\alpha \end{bmatrix}$$ **(eq. A-39)**

Where, u is a unit vector defining an axis of rotation and α is an angle of rotation.

Quaternions

Quaternions offer an alternative mathematical way of describing rotations in three-dimensional space. There is a one-to-one correspondence between quaternions and their matrix counterparts, but as noted by Lengyel (2003), quaternions have several crucial advantages:

a) Quaternions are faster to compute by machines because they require less storage space.

b) Quaternions allow storing and applying multiple separate rotations as a single mathematical entity that requires less computational operations.

c) Quaternions, as well as rotation matrices, prevent the occurrence of Gimbal Lock – a phenomenon of losing one degree of freedom in three-gimbal mechanism that occurs when two axes in that mechanism are set into parallel configuration decreasing the rotation about a third axis (Lepetit and Fua, 2005).

d) Quaternions can produce smoother and more realistic movement animations by interpolating between their values.

Quaternions are an extension of *complex numbers* (Weisstein, 2016). Complex numbers are written in algebraic form $z = a + bi$ where a and b are real numbers and i is an *imaginary number* equal to the square root of -1 ($\sqrt{-1}$) (Weisstein, 2016). In real number systems there is no such real number whose square is equal to -1 and therefore imaginary number is written as $i^2 = -1$. Complex numbers can be graphically interpreted on a *complex plane* also known as *Argand Diagram* as shown in Figure A.1-8 (Weisstein, 2016). It is important to indicate that the diagram shown in Figure A.1-8 cannot be interpreted similarly as two-dimensional Cartesian coordinate system. The purpose of Argand Diagram is to depict the real part of the complex number along a horizontal axis (*Re*) and its imaginary number part along vertical axis (*Im*).

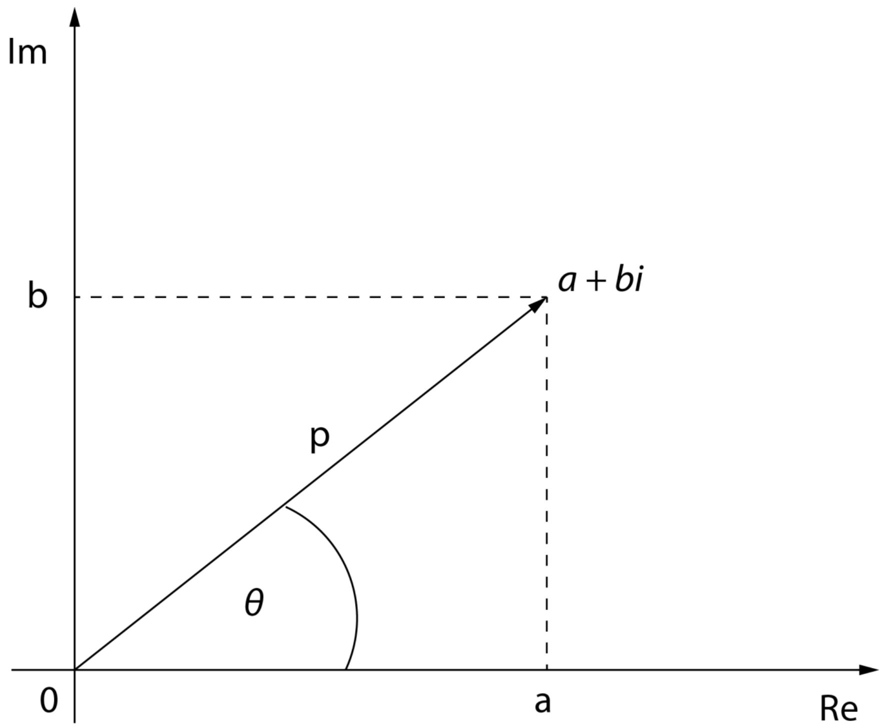


Figure A.1-8. Geometric interpretation of a complex number (Mizerski et al., 2014)

In Figure A.1-8 it can be seen that a vector whose *initial point* lies at the *origin* (0) of complex plane is directed at complex number ($a + bi$). The length of this vector is called *modulus* of a complex number (i.e. labelled as p in the diagram) and the angle between this vector and real number axis is called the *argument* of a complex number (i.e. labelled as θ in the diagram) (Weisstein, 2016). Figure A.1-8 shows that one can apply the Pythagorean theorem to write complex number in a trigonometric form as shown in eq. A-40 (Mizerski et al., 2014). The modulus p is usually written as an absolute value $|z|$.

$$z = p(\cos\theta + i\sin\theta) \quad \textbf{(eq. A-40)}$$

Where, $p = \sqrt{a^2 + b^2}$, $a = p\cos\theta$ and $b = p\sin\theta$

In order to write a general form of a quaternion, one needs to extend the concept on a complex number and complex plane into three dimensions by adding two imaginary numbers into its basic algebraic form (see eq.A-41).

$$q = w + xi + yj + zk \quad \textbf{(eq. A-41)}$$

Where, w, x, i, $w \in \mathbb{R}$ and, according to Hamilton (1853), imaginary components i, j, k follow the rules

1) $i^2 = j^2 = k^2 = -1$
2) $ij = -ji = k$
3) $jk = -kj = i$
4) $ki = -ik = j$

As noted by Oosten (2012), there exists a relationship between the rules formed by Hamilton (1853) and cross products of unit vectors defining a standard basis of a Cartesian coordinates space such that shown in eq. A-42.

$$\vec{\imath} = [1,0,0] \qquad \vec{\jmath} = [0,1,0] \qquad \vec{k} = [0,0,1]$$ **(eq. A-42)**

$$\vec{\imath} \times \vec{\jmath} = \vec{k} \qquad \vec{\jmath} \times \vec{k} = \vec{\imath} \qquad \vec{k} \times \vec{\imath} = \vec{\jmath}$$
$$\vec{\jmath} \times \vec{\imath} = -\vec{k} \qquad \vec{k} \times \vec{\jmath} = -\vec{\imath} \qquad \vec{\imath} \times \vec{k} = -\vec{\jmath}$$

In his works, as indicated by Oosten (2012), Hamilton (1853) noticed that imaginary numbers i, j, k can be used to represent three unit vectors $\vec{\imath}$, $\vec{\jmath}$, $\vec{k}$ of Cartesian coordinate system that follow the same rules, i.e.: $i^2 = j^2 = k^2 = -1$. Following this principle, vectors can be written in extended homogenous coordinates in a quaternion form in which one can identify a scalar and vector parts such that $q_v = v_w + v_x i + v_y j + v_z k$. Therefore, quaternions can be considered as generalisation of vectors.

A quaternion can also be specified in scalar-vector form (see eq. A-43).

$$q = s + \vec{v}$$ **(eq. A-43)**

Where, s is a scalar that is equal to w element and $\vec{v}$ is a vector containing x, y and z components.

Quaternions can be expressed more intuitively with a formula, which utilises trigonometric values of an angle. Quaternion corresponding to rotation around the unit axis X, Y and Z is shown in eq. A-44 (Baker).

$$q = \cos\left(\frac{\alpha}{2}\right) + i\left(x * \sin\left(\frac{\alpha}{2}\right)\right) + j\left(y * \sin\left(\frac{\alpha}{2}\right)\right) + k\left(z * \sin\left(\frac{\alpha}{2}\right)\right)$$ **(eq. A-44)**

Where, α is a rotation angle and x, y and z elements correspond to axis about which the rotation is to be calculated.

Quaternions can be also represented in a form of a 3x3 matrix (see eq.A-45) (Lengyel, 2003).

$$R_q = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix}$$ **(eq. A-45)**

Quaternions, as well as any other complex numbers, have *conjugates*. A conjugate of a quaternion is the same quaternion with the same lengths but inversed signs of imaginary numbers. A conjugate of a quaternion $q = w + v = w + xi + yj + zk$ is presented in eq. A-46.

$$\bar{q} = w - v = w - xi - yj - zk$$ **(eq. A-46)**

A conjugate of a quaternion has several useful properties that can be used in calculations (Baker):

a) They allow changing the order of multiplicands (see eq.A-47).

$$\bar{q}_x \bar{q}_y = \overline{(q_y q_x)}$$ **(eq. A-47)**

b) Multiplication of a quaternion by its conjugate results in a real number, which in turn allows finding the multiplicative inverse, i.e. one can easily find a rotation in a reversed direction to the rotation represented by a quaternion (see eq.A-48).

$$q\bar{q} = w^2 + x^2 + y^2 + z^2 = \| q \|^2$$ **(eq. A-48)**

c) They are used in a formula for rotation transformation of a vector in 3D space (see eq.A-49).

$$v' = q * v * \bar{q}$$ **(eq. A-49)**

Where, v' (a vector *after* the rotation operation) and v (a vector *before* the rotation operation) are vectors in three-dimensional space dictated by imaginary numbers in quaternion q (i.e. i, j, k).

To rotate a vector using this formula, it has to be first converted to a quaternion itself. The crucial operations using quaternions and vectors are presented in Table A-2.

Table A.1-2. Quaternions operations for transformations relevant to definition of dynamic behaviour in 3D space

Operation	Equation and example	Comments
Multiplication of quaternions	1. Method – using linear algebra $$q_1 = w_1 + x_1 i + y_1 j + z_1 k$$ $$q_2 = w_2 + x_2 i + y_2 j + z_2 k$$ $$q_1 q_2 = (w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) + + (w_1 x_2 + x_1 w_2 - y_1 z_2 - z_1 y_2) i + + (w_1 y_2 - x_1 z_2 - y_1 w_2 + z_1 x_2) j + + (w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2) k$$ 2. Method – using scalar-vector form $$q_1 = s_1 + \overrightarrow{v_1}$$ $$q_2 = s_2 + \overrightarrow{v_2}$$ $$q_1 q_2 = s_1 s_2 - \overrightarrow{v_1} \cdot \overrightarrow{v_2} + s_1 \overrightarrow{v_2} + s_2 \overrightarrow{v_1} + \overrightarrow{v_1} \times \overrightarrow{v_2}$$	- multiplication of quaternions is not commutative - operation allows to combine two distinct rotations
Quaternion normalization	$$q = w + xi + yj + zk$$ $$q_n = \frac{w + xi + yj + zk}{\sqrt{w^2 + x^2 + y^2 + z^2}}$$	- *condition*: $$w^2 + x^2 + y^2 + z^2 = 1$$ - normalized quaternions are useful when only pure rotations are required without any kind of undesired distortions such as expansions or contractions - q_n: is a normalized quaternion

Table A.1-2 cont. Quaternions operations for transformations relevant to definition of dynamic behaviour in 3D space

Operation	Equation and example	Comments
	1. Method – using the equation with a *conjugate* (Baker) $$v' = q * v * \bar{q}$$	- An example of a rotation of a $90°$ about Z axis.

3D rotation of a vector using a quaternion	$q = \cos\left(\frac{90}{2}\right) + i\left(0 * \sin\left(\frac{90}{2}\right)\right) + j\left(0 * \sin\left(\frac{90}{2}\right)\right) + k\left(1 * \sin\left(\frac{90}{2}\right)\right) =$ $= 0.7071 + i0 + j0 + k0.7071$ $q_v = 0 + i1 + j0 + k0$ $v' = (0.7071 + k0.7071) * (i1) * (0.7071 - k0.7071) =$ $= (i0.7071 + (k * i)0.7071) * (0.7071 - k0.7071) =$ $= (i0.7071 + j0.7071) * (0.7071 - k0.7071) =$ $= i0.5 + j0.5 + j0.5 - i0.5 = j$ 2. Method – using the matrix representation (Baker) $q = \cos\left(\frac{90}{2}\right) + i\left(0 * \sin\left(\frac{90}{2}\right)\right) + j\left(0 * \sin\left(\frac{90}{2}\right)\right) + k\left(1 * \sin\left(\frac{90}{2}\right)\right) =$ $= 0.7071 + i0 + j0 + k0.7071$ $R_q = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix} =$ $= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $v' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	- q_v: the quaternion representation of a vector [1,0,0] - v': the quaternion representation of a vector [0,1,0] after the rotation transformation - An example of a rotation of a $90°$ around Z axis using alternative method utilising matrix representation of quaternion. - j converts to vector [0,1,0] at the end of operation Quaternion values: - $w = 0.7071$ - $x = 0$ - $y = 0$ - $z = 0.7071$

Mathematical theories behind 3D geometries and processes used in the development of a graphical model

Objects in three-dimensional space are made out of vertices (points), lines (edges) and faces (planar surfaces). In their simplest case, they can form basic shapes that can be used for abstract representation of recognised individuals and objects in a visual scene. Due to the fact these shapes possess simplistic structure they can be utilised as a physical approximation of a more complex geometric shapes, ultimately reducing overall computational cost of their processing before displaying them on display screen.

Primitive geometries

Triangle geometry

A triangle surface is usually the atomic geometry that is drawn during the display process. Triangle is relatively simple, always *planar* (it can never be *non-planar*), can be used as a component of more complex polygonal shapes and is computed at a relatively fast speed. A triangle can be described with three vertices and a normal vector determining which direction it is orientated at. Knowing three positions of vector points that make a triangle ($\overrightarrow{p_0}, \overrightarrow{p_1}, \overrightarrow{p_2}$), the triangle normal vector $\vec{n}$ can be calculated with eq. A-50 (Lengyel, 2003).

$$\vec{n} = (\overrightarrow{p_1} - \overrightarrow{p_0}) \times (\overrightarrow{p_2} - \overrightarrow{p_0}) \qquad \textbf{(eq. A-50)}$$

Plane geometry

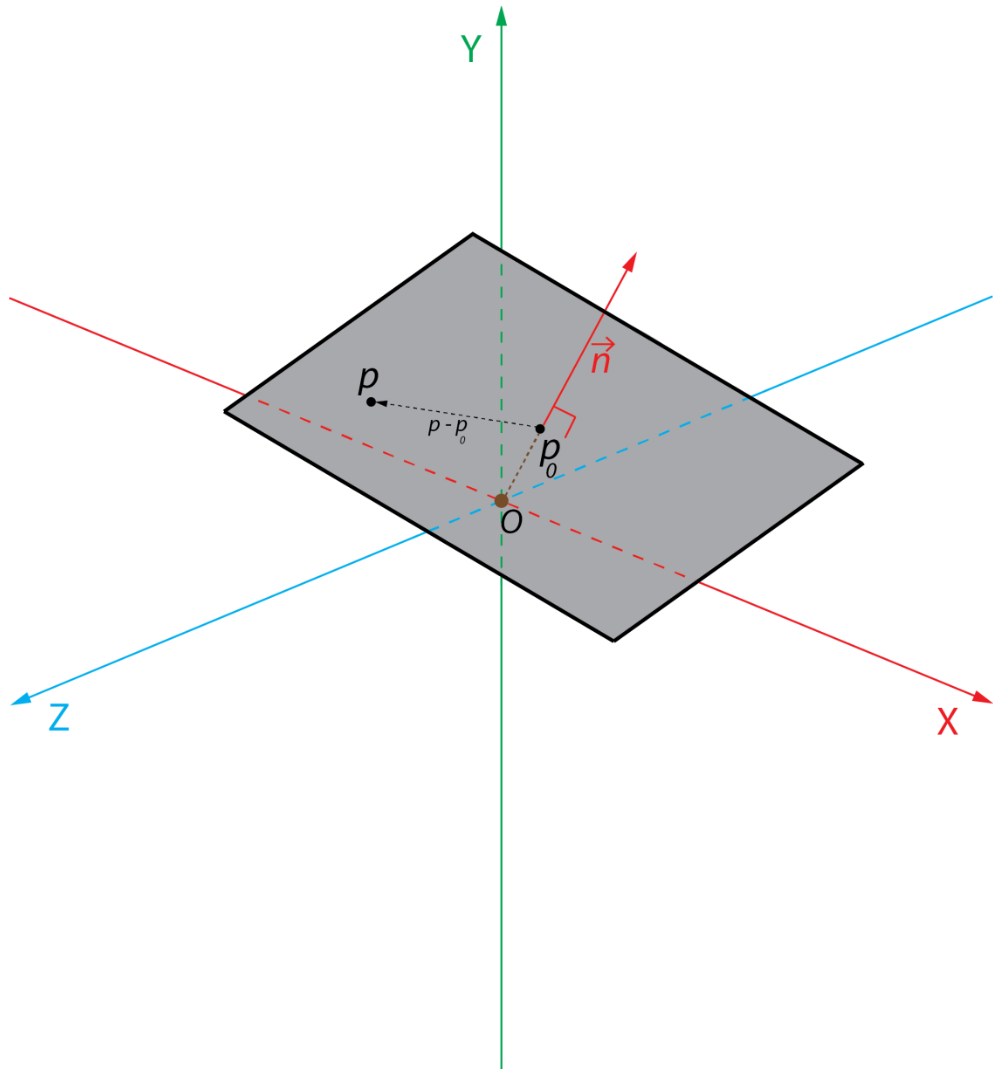


Figure A.1-9. Plane geometry in 3D space (Weisstein, 2016)

A plane can be defined in three-dimensional space as a non-zero normal vector $\vec{n}$ that goes through any point (e.g. p_0). Equation for calculating a plane surface in three-dimensional space is presented in eq. A-51 (Weisstein, 2016).

$$\vec{n} \cdot (p - p_0) = 0 \qquad \textbf{(eq. A-51)}$$

Where,

$$p \in S \qquad p = [x, y, z]$$
$$\vec{n} = [a, b, c] \qquad p_0 = [x_0, y_0, z_0]$$

and S is a set of points whose difference with point p_0 is always perpendicular to the normal vector $\vec{n}$ (Figure A.1-9).

The components of eq. A-52 are used to formulate a general plane equation (see eq. A-52) (Lengyel, 2003).

$$ax + by + zc + d = 0 \qquad \textbf{(eq. A-52)}$$

Where,

$$d \equiv -ax_0 - by_0 - cz_0$$
$$\vec{n} = [a, b, c] \qquad p_0 = [x_0, y_0, z_0]$$

From the general equation of a plane (see eq. A-52) one can derive three components of any point that lies on the plane where normal vector can go through that plane (see eq. A-53) (Weisstein, 2016). The exemplary normal vector is highlighted with red $\vec{n}$ vector in Figure A.1-9. One can also calculate the distance of a plane from the origin point of a Cartesian coordinate system with eq. A-54 (Weisstein, 2016).

$$x = -\frac{d}{a} \qquad y = -\frac{d}{b} \qquad z = -\frac{d}{c} \qquad \textbf{(eq. A-53)}$$

$$D = \frac{d}{\sqrt{a^2+b^2+c^2}} \qquad \textbf{(eq. A-54)}$$

Planes can be used to visually represent the natural boundaries of visual scene in three-dimensional space such as walls, floors and ceilings.

Box geometry

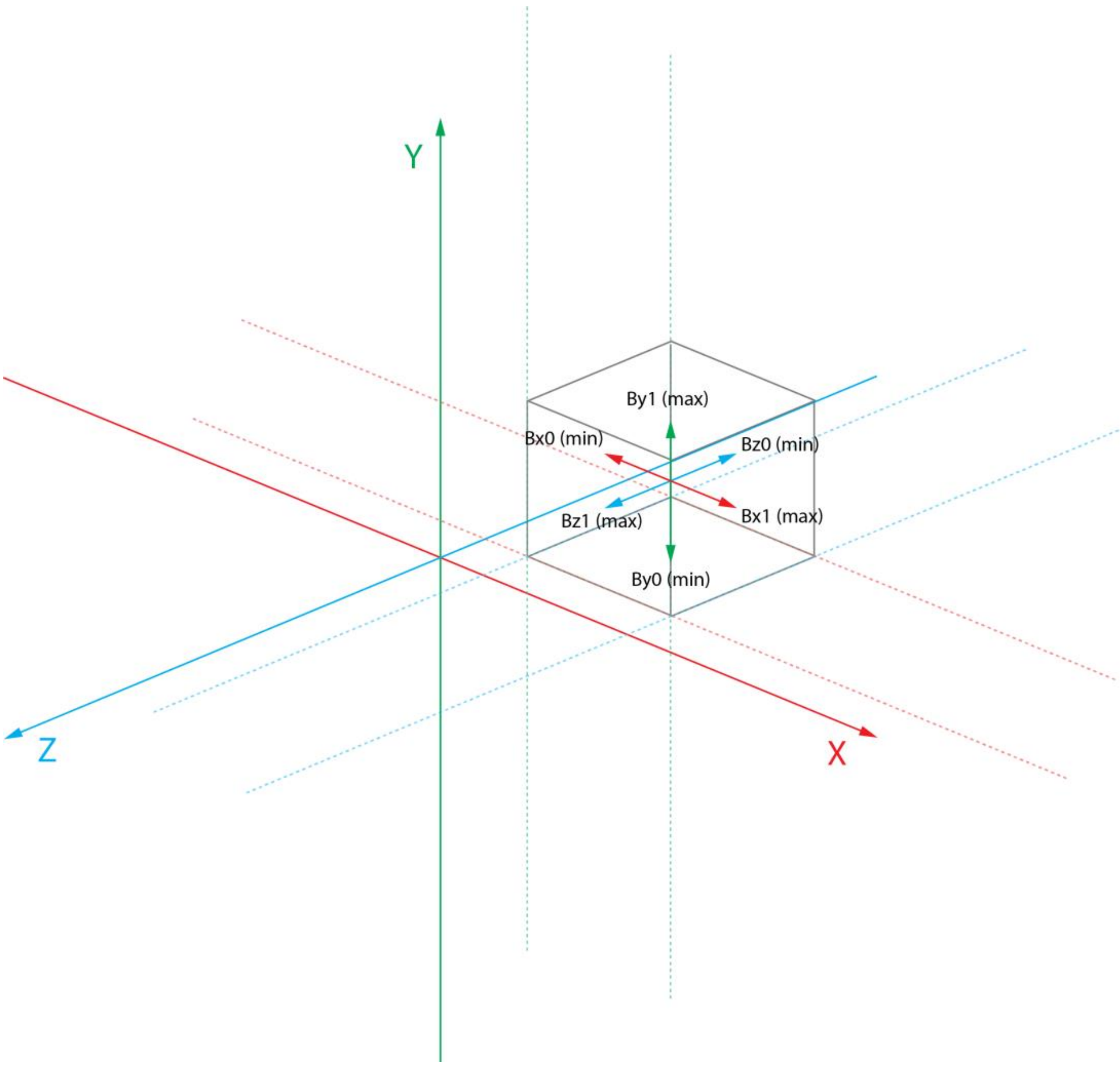


Figure A.1-10. Bounds of the box are defined with points specifying their minimum and maximum values along each axis in three-dimensional space.

As described by Lengyel (2003), a box in three-dimensional space is defined with six individual sides. Each side has minimum and maximum values that represent the extents of the box (see eq. A-55). Figure A.1-10 presents the minimum and maximum points of a box aligned along three axes X, Y and Z.

$$x = 0 \quad \text{and} \quad x = r_x \qquad y = 0 \quad \text{and} \quad y = r_y \qquad z = 0 \quad \text{and} \quad z = r_z \qquad \textbf{(eq. A-55)}$$

Where, r_x, r_y and r_z components constitute the dimensions of the box.

Sphere geometry

“A sphere is defined as the set of all points in three-dimensional Euclidean space $\mathbb{R}^3$ that are located at a distance r (the "radius") from a given point (the "center")” (Weisstein, 2016).

One can mathematically describe a sphere of a radius r in three-dimensional space positioned at the origin of a Cartesian coordinate system with eq. A-56 (Lengyel, 2003).

$$x^2 + y^2 + z^2 = r^2 \qquad \textbf{(eq. A-56)}$$

Where, x, y and z are the components of any point located at a fixed distance r from the centre of the sphere.

Substituting these components with a centre vector point of a sphere, one receives the equation presented in eq. A-57.

$$(p_x - c_x)^2 + \left(p_y - c_y\right)^2 + (p_z - c_z)^2 = r^2 \qquad \textbf{(eq. A-57)}$$

Where, p is any vector point being part of the sphere, c is the center (pivot) vector point of the sphere and r is the sphere’s radius.

Cylinder geometry

The lateral surface of an elliptical cylinder can be calculated on a basis of its radius and height. Positioning the base of a cylinder at origin of the XZ Euclidean plane in Cartesian coordinate system, whose radius on the X-axis is r, radius on the Z-axis is s and its height is h is defined along Y-axis (Figure A.1-11), one can describe the lateral surface of this cylinder with eq. A-58 (Lengyel, 2003).

$$\begin{gathered} x^2 + m^2 z^2 = r^2 \\ 0 \leq y \leq h \end{gathered} \qquad \textbf{(eq. A-58)}$$

Where, $m = r/s$ and is the ratio of two radiuses. If the ratio is equal to 1, i.e. $r = s$ then $m = 1$ and cylinder is circular.

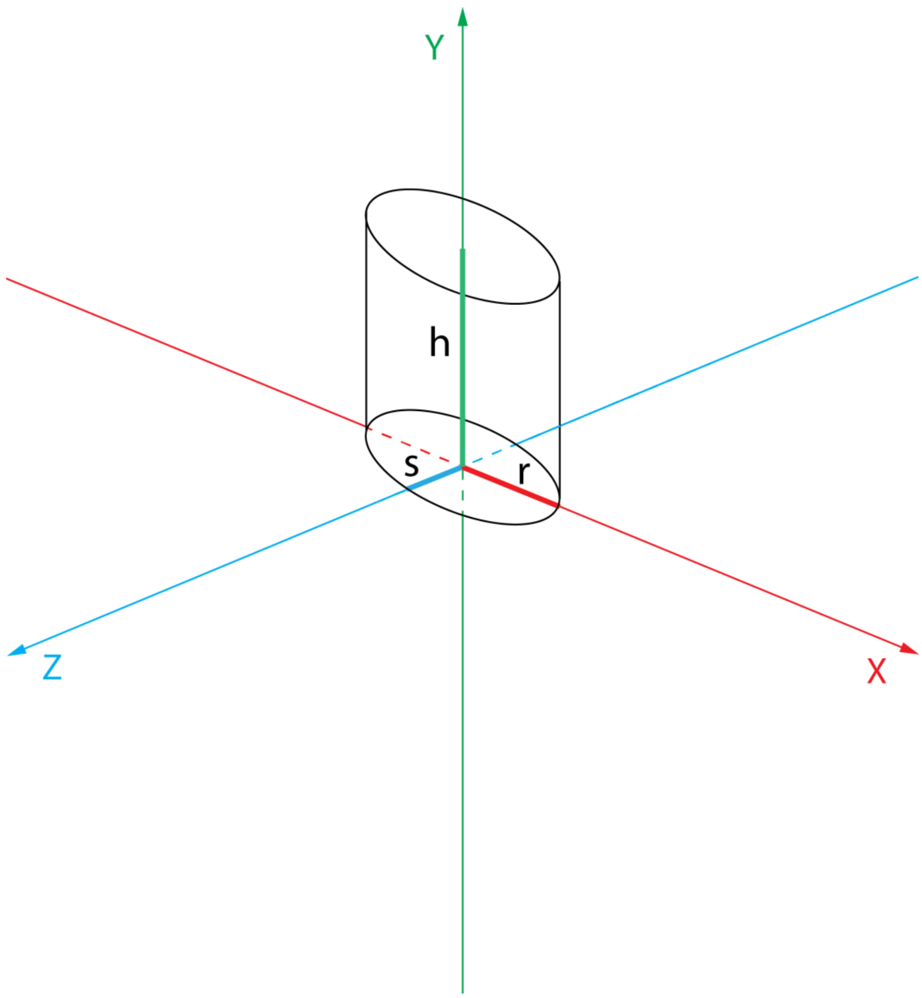


Figure A.1-11. Object space of an elliptical cylinder (Lengyel, 2003)

Ray Tracing

Ray tracing is a technique that determines when geometry comes into contact with a directed line (ray). A ray has its origin point, direction, but not a terminal point. The ray is emitted from a fixed origin point (*emission point),* towards a specific direction (*emission direction*) in three-dimensional space for the purpose of recognising geometries positioned in that direction as shown in Figure A.1-12. The directed line arrow in the diagram (Figure A.1-12) only represents the direction of the ray since it does not have a *terminal point*. In most practical cases, ray intersects geometry at two points: at *the inlet intersection point* and at *the outlet intersection point*. The duration of emission may vary in accordance to the requirements of graphical model of visual scene. It can last a certain amount of time or it can be instantaneous, i.e. ray can be emitted at one particular moment of time and cease its operation right afterwards.

Ray tracing technique has many uses in computer graphics ranging from collision detection and visibility tests to line-of-sights measures. This is a central method used in a simulation for detecting physical as well as non-physical (logical) collisions. The registry of such a collision is considered as an "*event*" contributing to the evolution of a dynamic pattern.

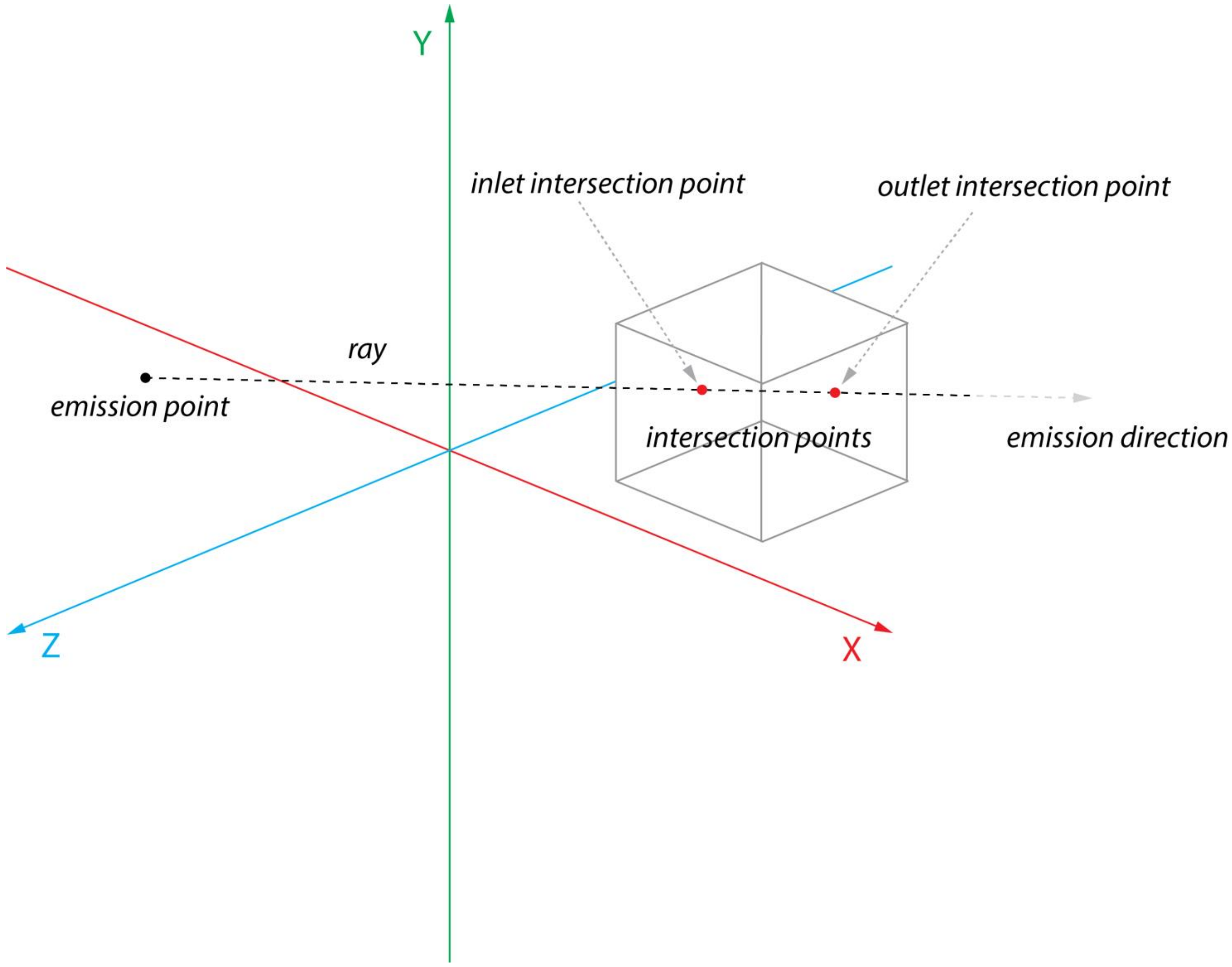


Figure A.1-12. Ray tracing technique

The importance of a collision event always means the relationship between one object and another at any given point in time. This relationship is registered with the use of a ray when it intersects the geometry, i.e. when ray and geometry's surface share a common point. Directed line (ray) equation used for finding intersection points with geometry surfaces is presented in eq. A-59.

$$R(x) = \vec{u} + x\vec{v} \quad \textbf{(eq. A-59)}$$

Where, $\vec{u}$ is the initial point of the line and $\vec{v}$ is the vector (usually normalized) dictating the direction of the line.

As profoundly described by Lengyel (2003), to find whether a ray defined by eq. A-59 intersects a geometry surface in three-dimensional space, one needs to find roots of a degree n polynomial in x. x is a scalar that provides a solution for finding any point lying on the line within a specific distance from ray's origin point. There are different degrees of polynomials that correspond to the level of complexity of the structure of three-dimensional geometry.

Table A.1-3. Solutions to degree one polynomial based on a table presented by Mizerski et al. (2014)

Equation	$ax + b = 0$		
Conditional values	$\mathrm{a} \neq 0$	$\mathrm{a} = 0, \quad \mathrm{b} \neq 0$	$\mathrm{a} = 0, \quad \mathrm{b} = 0$
Polynomial solutions	$x = -\frac{b}{2}$	$x \in \emptyset$	$x \in \mathbb{R}$
Rays and surface intersection	*1 solution found; ray intersects planar surface at one point*	*No solution found; ray does not intersect a planar surface*	*Infinite number of solutions found; ray covers entire length of planar surface. In practice it is a very rare case but needs to be accounted for.*

In this research the attention is focused on polynomials of degrees of one and two. They correspond to planar surfaces (planes potentially representing floors) and quadratic surfaces (spheres, cubes or cylinders depicting simplified geometric structures of static and dynamic objects) respectively. The degree of polynomials and their maximum number of real roots directly correspond to the maximum number of intersections that can be made by a ray with a given geometry surface. As it was stated by Lengyel (2003), one can easily find a solution for planar surfaces since the degree of their polynomials is one. This means that a ray, unless it covers the entire length of a planar surface, can intersect a plane at only one point as summarised in Table A-3.

Higher degree polynomials are possible but have been left out of the scope of this research because higher accuracy collision detection and more complex geometries require much greater computational memory usage that slows down the real-time performance as it was explained by Lengyel (2003).

The quadratic polynomial has a general form presented in eq. A-60.

$$ax^2 + bx + c = 0 \qquad \textbf{(eq. A-60)}$$

Real roots of polynomial are calculated with (eq. A-61).

$$x = \frac{-b \pm \sqrt{\Delta}}{2a} \qquad \textbf{(eq. A-61)}$$

Where, $\Delta = b^2 - 4\text{ac}$

By calculating the *discriminant* (Δ) of the polynomial, one can find out how many real roots it has and therefore determine whether a ray intersects a given object defined in three-dimensional space. It is important to note, that quantity given by Δ does not reveal the exact point of intersection, but only the information whether a ray intersects an object or not. All possible results that Δ can provide have been summarised in Table A-4.

In order to detect an intersection of a ray with a particular geometry, the strategy is to substitute ray equation into elements of that geometry surface equation. Figure A.1-13 illustrates different intersections of a ray with the sphere geometry based on the result of Δ. By doing so, one can determine points that lie on a ray and objects surface simultaneously, i.e. where the rays intersect the surface. The polynomial roots equations allow calculating the scalar component x of the ray equation defined in eq. A-59.

Table A.1-4. Determining number of intersections based on the value of discriminant

Equation	$ax^2 + bx + c = 0$		
Discriminant value	$\text{a} \neq 0, \quad \Delta > 0$	$\text{a} \neq 0, \quad \Delta = 0$	$\text{a} \neq 0, \quad \Delta < 0$
Polynomial roots	$x_1 = \frac{-b - \sqrt{\Delta}}{2a}$ $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$	$x_1 = x_2 = -\frac{b}{2a}$	$x \in \emptyset$
Rays and surface intersection	*2 real roots; ray intersects object's surface at two different points: entry point where ray intersects object's surface for the first time and exit point where ray intersects the same object's surface for last time*	*1 real root; ray is tangent to object surface. In practice it is a very rare case but needs to be accounted for.*	*No real roots; ray does not intersect an object.*

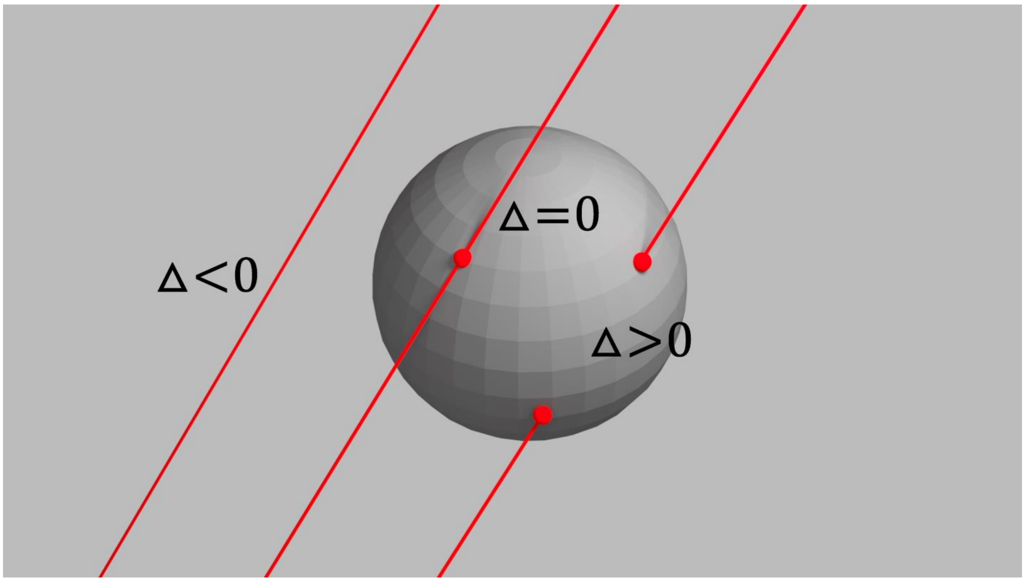


Figure A.1-13. Estimating the intersection between rays and a sphere based on the value of discriminant

Ray intersection with a plane geometry

In order to find at which point a ray specified by the equation eq. A-59 intersects a plane defined with the equation eq.A-51 one needs to substitute point p that lies on the plane with a ray equation as shown in eq.A-65.

$$\vec{n} \cdot (\vec{u} + x\vec{v} - p_0) = 0 \quad \textbf{(eq. A-65)}$$

To find the point at which the intersection occurs, eq. A-65 needs to be transformed so that x can be solved. By expanding the expression in the bracket in eq. A-65 yields the equation presented in eq. A-66.

$$\vec{n} \cdot (\vec{u} - p_0) + \vec{n} \cdot x * \vec{v} = 0 \quad \textbf{(eq. A-66)}$$

Leaving x on the left hand side in eq. A-66 results in eq. A-67.

$$x = -\frac{\vec{n} \cdot (\vec{u} - p_0)}{\vec{n} \cdot \vec{v}} = \frac{\vec{n} \cdot (p_0 - \vec{u})}{\vec{n} \cdot \vec{v}} \quad \textbf{(eq. A-67)}$$

Substituting the value of x obtained in eq. A-67 in a ray equation (see eq. A-59) produces the exact point of intersection.

Ray intersection with a box geometry

A box in a space is described with six plane equations corresponding to its sides (see eq. A-55). To simplify the illustration of a ray intersecting a box, the technique developed by Williams et al. (2005) is followed with the assumption that the box is aligned with each of the axis of Cartesian coordinate system. An alternative method for solving this problem is presented by Lengyel (2003) and is discussed later in this section.

Boxes aligned with the axis are often used as bounding boxes and are called "axis-aligned bounding boxes" (AABB). Analogically, when a box is not aligned along the axis of Cartesian coordinate system it is called "oriented bounding box" (OBB).

According to Williams et al. (2005), a box in space can be defined with a set of straight lines delimiting its boundaries. Therefore, bounds can be defined using straight-line analytical equation for each axis (see eq. A-68).

$$y = mx + b \qquad \textbf{(eq. A-68)}$$

where m is the slope, which dictates the orientation of the line and b is a point, where the line intersects the Y-axis.

Since the line defined by eq. A-68 is parallel to X-axis then $m = 0$. When AABB is considered, two points designating the minimum and maximum bounds of the box are required to describe a side on a given axis plane, i.e. the same principle applies when one needs to define the bounds along X-, Y- and Z- axis (Figure A.1-10).

Once one describes a box with their respective minimum and maximum values along each axis with a line equation, a point where a ray intersects a plane delimited by those values needs to be found. One can do this by substituting ray parametric equation into a line equation defining particular boundary. For instance, if $Bx0$ is the line equation for the x element of the box's minimum value boundary and $p = u + tv$ is a ray parametric equation then one can find where ray intersects this line using eq. A-69.

$$Bx0 = u_x + tv_x \qquad \textbf{(eq. A-69)}$$

Where, solving for t is then made by $tx0 = (Bx0 - u_x)/v_x$

$tx0$ was used on the left hand side of the equation eq. A-69 because one is required to do it for each boundary line. One needs to apply the same principle for y and z elements respectively so that six values where ray intersects each of the lines parallel to their corresponding axis can be revealed.

To calculate ray intersection values with each of the lines one has to solve six equations presented in eq. A-70.

$$tx0 = (Bx0 - u_x)/v_x \qquad \textbf{(eq. A-70)}$$

$$tx1 = (Bx1 - u_x)/v_x$$

$$ty0 = (By0 - u_y)/v_y$$
$$ty1 = (By1 - u_y)/v_y$$

$$tz0 = (Bz0 - u_z)/v_z$$
$$tz1 = (Bz1 - u_z)/v_z$$

Once these values are calculated, one knows where ray intersects planes specified by each side-line. However, there is still a need to find which of these values fall within the boundaries of a box, i.e. position where ray intersects a box's side described on a given axis plane. For each side there are minimum and maximum values where the ray may intersect a box. If the conditions in eq. A-71 are satisfied then one can find the minimum value where ray intersects a particular side of the box by comparing these values and selecting the value for which t is the greatest as shown in eq. A-72.

$$tx0 = (Bx0 - u_x)/v_x \quad \text{and} \quad ty0 = (By0 - u_y)/v_y \qquad \textbf{(eq. A-71)}$$

$$tx0 > ty0 \Rightarrow t_{min} = tx0 \quad \mathit{or} \quad ty0 > tx0 \Rightarrow t_{min} = ty0 \qquad \textbf{(eq. A-72)}$$

Where, t_{min} is the minimum value parameter for ray where it intersects this side of the box.

Finding the maximum value where a ray intersects this side of the box follows the same principle with the exception that the lowest value has to be selected. If the conditions in eq. A-73 are satisfied then one can find the maximum value where ray intersects a particular side of the box by comparing these values and selecting the value for which t is the lowest as shown in eq. A-74.

$$tx1 = (Bx1 - u_x)/v_x \quad \text{and} \quad ty1 = (By1 - u_y)/v_y \qquad \textbf{(eq. A-73)}$$

$$tx1 < ty1 \Rightarrow t_{max} = tx1 \quad \mathit{or} \quad ty1 < tx1 \Rightarrow t_{max} = ty1 \qquad \textbf{(eq. A-74)}$$

Where, t_{max} is the maximum value parameter for ray where it intersects this side of box.

The possibility of a ray not intersecting a box on this side can be confirmed with the conditionals specified in eq. A-75.

$$tx0 > ty1 \quad \text{or} \quad ty0 > tx1 \qquad \textbf{(eq. A-75)}$$

When t_{min} and t_{max} are known, one can expand the technique to include z element and therefore, cover the three-dimensional case of a box (see eq. A-76). The process follows the same principles described in eq. A-72.

If **(eq. A-76)**

$$tz0 = (Bz0 - u_z)/v_z \quad \text{and} \quad tz1 = (Bz1 - u_z)/v_z$$

Then

$$tz0 > t_{min} \Rightarrow t_{min} = tz0 \quad \textit{or} \quad tz1 < t_{max} \Rightarrow t_{max} = tz1$$

However, if at this stage of the process one of the following conditions specified in eq. A-77 is satisfied then no intersection occurs. Otherwise, if all tests are passed at the end of the process then an intersection has been found.

$$t_{min} > tz1 \quad \textit{or} \quad t_{max} < tz0 \qquad \textbf{(eq. A-77)}$$

Alternative approach of solving the problem of finding a point, at which ray intersects a box has been presented by Lengyel (2003). With the assumption that three planes defining a box are oriented away from ray's direction at the time of intersection, it is possible to determine which of them needs to be tested by investigating individual components of ray's direction vector. If a box is specified with six plane equations (see eq. A-55), one can perform several checks of a ray defined by the equation $p = u + tv$ in a following way:

- If $v_x = 0$ then a ray does not intersect either a plane $x = 0$ or $x = r_x$ because of the fact that ∥ x .
- If $v_x > 0$ then no intersection test with the plane $x = r_x$ needs to be conducted because it is an inner side of the box from ray's point of view.
- Analogically, if $v_x < 0$ then no intersection test with the plane $x = 0$ needs to be conducted for the similar reason.
- The same analysis is conducted on y- and z- elements of v.

Once the point of intersection of a given plane is found, the next step involves its examination to determine whether it falls within the boundaries of a given face of the box. In order to do that, one has to find t in ray equation $p\ (t) = u + tv$ by substituting a box side dimension and rearranging the terms. For instance, to find a point where ray intersects plane $x = r_x$ one has to find t with the equation eq. A-78.

$$t = \frac{r_x - u_x}{v_x} \qquad \textbf{(eq. A-78)}$$

In order to state that a ray intersects a box at this face, the conditions specified in eq. A-79 needs to be fulfilled.

$$0 \le p(t)_y \le r_y$$ **(eq. A-79)**

$$0 \le p(t)_z \le r_z$$

If both conditions in (eq.A-79) are satisfied, then a successful intersection has been found. No further tests need to be conducted since the closest point of contact has been determined. Otherwise, if any of these conditions are not satisfied, then no intersection occurred.

Ray intersection with a sphere geometry

To describe a sphere in three-dimensional space, all points defining its structural shape must fulfil the conditional equation presented in eq.A-80 (Lengyel, 2003).

$$(p_x - c_x)^2 + \left(p_y - c_y\right)^2 + (p_z - c_z)^2 = r^2$$ **(eq. A-80)**

Where, p is any point being part of the sphere, c is the center (pivot) point of the sphere and r is the sphere's radius.

By using the ray equation in eq. A-59, the components of the sphere can be written as shown in eq. A-81.

$$p_x = u_x + xv_x$$ **(eq. A-81)**

$$p_y = u_y + xv_y$$

$$p_z = u_z + xv_z$$

The components from eq. A-81 can be then used in the main sphere as shown in eq. A-82 and described by Lengyel (2003).

$$((u_x + xv_x) - c_x)^2 + \left((u_y + xv_y) - c_y\right)^2 + ((u_z + xv_z) - c_z)^2 = r^2$$ **(eq. A-82)**

By expanding the squared elements and writing them in a standard quadratic equation yield eq. A-83.

$$\left(v_x{}^2 + v_y{}^2 + v_z{}^2\right)x^2 + 2\left(v_x(u_x - c_x) + v_y\left(u_y - c_y\right) + v_z(u_z - c_z)\right)x + (u_x - c_x)^2 + (u_y - c_y)^2 +$$ **(eq. A-83)**

$$+ (u_z - c_z)^2 - r^2 = 0$$

The equation eq. A-83 is equal to the general quadratic polynomial form of eq. A-60, whose components are defined with eq. A-84 (Lengyel, 2003).

$$ax^2 + bx + c = 0$$ **(eq. A-84)**

Where,

$$a = {v_x}^2 + {v_y}^2 + {v_z}^2$$

$$b = 2\left(v_x(u_x - c_x) + v_y(u_y - c_y) + v_z(u_z - c_z)\right)$$

$$c = (u_x - c_x)^2 + (u_y - c_y)^2 + (u_z - c_z)^2 - r^2$$

which can be written in a shorthand notation;

$$a = \| \vec{v} \|^2$$

$$b = 2\vec{v} \cdot (\vec{u} - c)$$

$$c = \| \vec{u} - c \|^2 - r^2$$

The values of a, b and c in eq. A-84 determine the value of the discriminant and reveals how many real solutions the quadratic equation has for x. Therefore if $\triangle$ is less than 0 it is immediately known that a ray does not intersect the sphere's surface. Whereas, If $\triangle$ is equal or greater than 0 then a ray intersects at least one of the points of the sphere surface.

Ray intersection with a cylinder geometry

In order to detect a collision between cylinder and ray, one has to substitute the components of the cylinder equation (see eq.A-58) with a ray equation (see eq.A-59) as shown in eq.A-85 (Lengyel, 2003).

$$(u_x + xv_x)^2 + m^2(u_y + xv_y)^2 = r^2 \qquad \textbf{(eq. A-85)}$$

By expanding the squared elements of eq. A-85 and writing them in a standard quadratic equation one receives and equation presented in eq. A-86.

$$\left(v_x^2 + m^2 v_y^2\right)x^2 + 2\left(u_x v_x + m^2 u_y v_y\right)x + u_x^2 + m^2 u_y^2 - r^2 = 0 \qquad \textbf{(eq. A-86)}$$

The value of a discriminant reveals if the intersection occurred. Solving eq. A-86 gives an indication of how many real solutions the quadratic equation has for x. As a consequence, one can calculate the exact points of intersection by incorporating the x value into the ray equation. However, as explained by Lengyel (2003), in case of cylinders it is important that the points of intersection satisfy $0 \leq y \leq h$ condition.

Collision of two spheres

In order to mathematically describe a collision between two spheres in three-dimensional space, an example presented by Lengyel (2003) is followed. With the assumption that two spheres are moving at a constant linear velocity over a certain period of time, beginning at time $t = 0$ and finishing at time $t = 1$, it is possible to represent the initial and ending positions of first and second sphere centre points. The definition of velocity vectors representing the velocity of first and second sphere centre point is presented in eq. A-87.

$$\vec{v}_p = \vec{p}_2 - \vec{p}_1$$
$$\vec{v}_q = \vec{q}_2 - \vec{q}_1 \quad \textbf{(eq. A-87)}$$

Where,

$\vec{v}_p, \vec{v}_q$: velocity of first and second sphere
$\vec{p}_1, \vec{p}_2$: initial and ending position of first sphere centre point
$\vec{q}_1, \vec{q}_2$: initial and ending position of second sphere centre point

To calculate the position of first and second sphere centre points at time t, one has to add their respective velocities calculated with eq. A-87 multiplied by time factor as shown in eq. A-88 (Lengyel, 2003).

$$\vec{p}_t = \vec{p}_1 + t\vec{v}_p$$
$$\vec{q}_t = \vec{q}_1 + t\vec{v}_q \quad \textbf{(eq. A-88)}$$

Where,

$\vec{p}_t, \vec{q}_t$: position of first and second sphere center point at time t
$\vec{p}_1, \vec{q}_1$: initial position of first and second sphere center point
t : time factor where $t \in [0,1)$
$\vec{v}_p, \vec{v}_q$: velocity of first and second sphere (see eq. A-87)

One has to determine whether the distance d between the center points of both spheres at time $t \in [0,1)$ is ever equal to the sum of their radii as shown in eq. A-89.

$$d = r_p + r_q \quad \textbf{(eq. A-89)}$$

Where,

r_p, r_q : radius of the first and second sphere
d : distance between first sphere center point and second sphere center point

When the condition in eq. A-89 is fulfilled, the spheres are tangent to each other at time t and therefore collide with each other. The squared distance d between $\vec{p}_t$ and $\vec{q}_t$ is given by eq. A-90.

$$d^2 = \|\vec{p}_t - \vec{q}_t\|^2 \quad \textbf{(eq. A-90)}$$

Substituting the values in eq. A-90 produces the equation shown in eq. A-91.

$$d^2 = \left\|\vec{p}_1 + t\vec{v}_p - \vec{q}_1 - t\vec{v}_q\right\|^2 \quad \textbf{(eq. A-91)}$$

To utilise eq. A-91 in a convenient manner, it is rewritten in exactly the same way as described by Lengyel (2003). First, two additional helper values are defined as shown in eq. A-92 and substituted into the equation eq. A-93.

$$\begin{aligned} A &= \vec{p}_1 - \vec{q}_1 \\ B &= \vec{v}_p - \vec{v}_q \end{aligned} \quad \textbf{(eq. A-92)}$$

$$d^2 = \|A + tB\|^2 = A^2 + 2t(A \cdot B) + t^2B^2 \quad \textbf{(eq. A-93)}$$

If one uses the quadratic formula to solve for t in eq. A-93, the formulas shown in eq. A-94 are produced. The squared values in the equation (e.g. A^2) represent a dot product between the same vector (i.e. $A \cdot A$).

$$t_1 = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2(A^2 - d^2)}}{B^2} \quad \textbf{(eq. A-94)}$$

$$t_2 = \frac{-(A \cdot B) + \sqrt{(A \cdot B)^2 - B^2(A^2 - d^2)}}{B^2}$$

Time t_1 outputs time when two spheres are tangent while still approaching each other. Time t_2 is an instant when spheres are tangent but moving away from each other. Assuming that two spheres are not already intersecting with each other in the beginning, one has to only calculate the time when spheres are first tangent to each other, i.e. when they first collide. This is done by substituting squared distance value with the sum of the radii of the two spheres as shown in eq. A-95. If the calculated value of t is not within range of [0,1), then no collision occurred during the scrutinised time period.

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_p + r_q)^2]}}{B^2} \quad \textbf{(eq. A-95)}$$

In this appendix section, the fundamental formal theories behind graphical model of visual appearance of the scene and dynamic movements have been presented in an organised manner. At the very basic level, the linear algebra and vector calculus provide convenient mathematical constructs to describe spatial state of objects and dynamics occurring in the scene.

Trigonometry is a suitable area of mathematics providing convenient tools for measuring angles between various objects in three-dimensional space. In combination with vectors calculus, one can conveniently formulate calculations that involve angles between vectors defined in three-dimensional space, assess angular offsets between different objects, estimate the average orientation of individuals etc.

Matrices allow enclosing dynamic transformations, such as movements and rotations of objects, in a compact mathematical structure. Matrices not only provide a method of transforming objects in three-dimensional space, but also a structure that can be used for the analysis of dynamics observed in visual scene. The significance of matrices lies in the way they can be utilised to improve transformation calculations efficiency and speeds. Since they allow encapsulating several transformations in one matrix, the number of operations can be significantly reduced and therefore decrease computational cost.

Quaternions are often used in three-dimensional graphics programming because of their advantages over matrices: they occupy less storage space; require fewer arithmetic calculations to perform a transformation; reduce a chance of phenomenon occurrence called Gimbal Lock; are more easily interpolated between their values.

Ray tracing is a technique commonly used in 3D graphics programming to detect and recognise geometries lying in path of a ray. When a point being part of a specific object's surface also lies on a ray, the collision is successfully registered.

A.2. Patterns of Individual and Group Dynamic Behaviour

Two types of parameters of the patterns have been highlighted:

- *Blue parameters* are the parameters calculated during the simulation. These parameters utilise various techniques embedded in the simulator as described in the previous chapter

- *Red parameters* are the parameters, which are read from a configuration file prior to the beginning of the simulation. These parameters are adjustable during the simulation using a separate interface which will be presented in the next chapter.

Patterns of individual dynamic behaviour

Table. A.2-1. "*Somebody is walking towards something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location	- Walking straight - Calculating distance between Agent and Static Object - Calculating average walking direction	- ***Agent data***: location, walking direction - ***Static Object data***: location, orientation (towards agent) - ***walking towards distance threshold***: used for determining if Agent is within a specific radius from Static Object - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***walking towards interval***: how frequently events are being captured for this pattern - ***overall direction proximity coefficient***: used for calculating the threshold against which the overall direction proximity value is checked against. The coefficient is restricted to the range of [0,1]. - ***overall direction proximity***: the calculated sum of dot products stored in events batch - ***distance***: between Agent location and Static Object location

Description

Condition for registering event data for the pattern analysis

The events constituting a pattern of "walking towards" are registered when individual is located within a predefined distance threshold t_d from an object. In situations when several objects are located within this threshold, the closest one is selected.

$$\|\vec{v}_t - \vec{u}_t\| \leq t_d$$

$\vec{v}_t$: the location of a nearby object in a world space at the time t that is involved in the event
$\vec{u}_t$: the location of agent in a world space at the time t when event occurred
t_d : distance threshold where $t_d \in \mathbb{N} : t_d \in [1, N]$

Criterion for recognising a pattern from accumulated events

$$\sum_{i=1}^{n} (\vec{v}_i - \vec{u}_i) \cdot \overrightarrow{vd\imath r}_i \geq n - (nC)$$

$\vec{v}_i$: the location of an object in a world space recorded in entry i in *events batch*
$\vec{u}_i$: the location of an agent in a world space recorded in entry i in *events batch*
$\overrightarrow{vd\imath r}_t$: unit vector defining a walking direction of an agent recorded in entry i in *events batch*
n : total number of registered events where $n > 0$
C : threshold coefficient where $C \in \mathbb{N} : 0 < C \leq 1$

Algorithm structure (in pseudo code)

```
if events batch exists
    if distance ≤ walking towards distance threshold
        if events batch size ≥ sampling size
            for each event in events batch
                calculate dot product of direction vector and view direction vector
                add dot product value to overall direction proximity

            if overall direction proximity ≥ (events batch size * 1.0) - overall direction
proximity coefficient * (events batch size * 1.0)
                register a pattern
            clear events batch
        else
            select the closest static object
            if static object ID is equal to static object ID stored in last event in events
batch
                record static object and agent data
                add event to events batch
            else
                clear events batch
```

Output Message: *"Agent ID moves towards Static Object ID"*

Sources

Algorithm: Simulation.states / WorldAdvancedLogger.java
Event data bean: Simulation.beans / EventBean.java

Table. A.2-2. "*Somebody is walking away from something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location	- Walking straight - Calculating distance between Agent and Static Object	- ***Agent data***: location, walking direction - ***Static Object data***: location, orientation (towards agent) - ***walking away distance threshold***: used for determining if Agent is outside a specific radius of Static Object - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***walking away interval***: how frequently events are being captured for this pattern - ***distance***: between Agent location and Static Object location

Description

Condition for registering event data for the pattern analysis

The events constituting a pattern of "walking away" are registered when individual is located within a predefined distance threshold t_d from an object. In situations when several objects are located within this threshold, the closest one is selected.

$$\|\vec{v}_t - \vec{u}_t\| \leq t_d$$

$\vec{v}_t$: the location of an object in a world space at the time t that is involved in the event $\vec{u}_t$: the location of agent in a world space at the time t when event occurred t_d : distance threshold where $t_d \in \mathbb{N} : t_d \in [1, N]$ **Criterion for recognising a pattern from accumulated events** $(\vec{v}_0 - \vec{u}_0) \cdot \overrightarrow{vdir_0} > 0$ **and** $(\vec{v}_n - \vec{u}_n) \cdot \overrightarrow{vdir_n} < 0$ $\vec{v}_0$, $\vec{v}_n$: the location of an object in a world space recorded in *first* and *last* entry in *events batch* $\vec{u}_0, \vec{u}_n$: the location of an agent in a world space recorded in *first* and *last* entry in *events batch* $\overrightarrow{vdir_0}, \overrightarrow{vdir_n}$: the walking direction of an agent recorded in *first* and *last* entry in *events batch*
Algorithm structure (in pseudo code)
<pre>if events batch exists   if distance ≤ walking away distance threshold   if events batch size ≥ sampling size   if (dot product of first entry direction vector and first entry view direction vector > 0) and (dot product of last entry direction vector and last entry view direction vector < 0)   register a pattern   clear events batch   else   select the closest static object   if static object ID is equal to static object ID stored in last event in events batch   record static object and agent data   add event to events batch   else   clear events batch</pre>
Output Message: *"Agent ID moves away from Static Object ID"*
Sources
Algorithm: Simulation.states / WorldAdvancedLogger.java *Event data bean*: Simulation.beans / EventBean.java

Table. A.2-3. "*Somebody walks alongside something***" behavioural pattern**

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location, dimensions, shape	- Walking straight - Determining if a Static Object is by Agent's left / right side	- ***Agent data***: ID, location, walking direction - ***Static Object data***: ID, location, surface dimensions and shape - ***walking alongside ray's length limit***: used for delimiting ray's length and consequently defining a maximum distance at which collision may be detected - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***walking alongside interval***: how frequently events are being captured for this pattern

Description
Details of ray emission

$$r(x) = \vec{u}_p + x(R_{side}\vec{v}_{dir})$$

$r(x)$: point laying on the ray surface located at x distance from its *emission point*
$\vec{u}_p$: the location of a *central node* of an agent defined in global coordinates
R_{side}: a rotation matrix declared in section 4.4.2.1., i.e. rotation about Y-axis 90°
$\vec{v}_{dir}$: a normalized walking direction of an agent

Condition for registering event data for the pattern analysis

The events constituting a pattern of "walking alongside" are registered when a ray of a specified length intersects an object located to the left or right of the Agent.

$$p_1 \cdot \vec{n}_1 + (-p_0 \cdot \vec{n}_1) > 0$$

p_1 : point of intersection with an object geometry's triangle
$\vec{n}_1$: a normalized normal vector of additional triangle constructed during the operation
p_0 : point from which the ray has been emitted, e.g. agent's location

Criterion for recognising a pattern from accumulated events

$$E_0 \cap E_n \neq \emptyset \textbf{ and } |E_0| = |E_n| \textbf{ and } E_0 = E_n$$

E_0 , E_n: the set of *qualitative* and *comparable* data recorded in *first* and *last* entry in *events batch*: side of an agent which is probed by ray and static object ID
$|E_0|$, $|E_n|$: the *cardinality* of *event data set* recorded in *first* and *last* entry in *events batch*
$E_0 = E_n \leftrightarrow \forall e[e \in E_0 \leftrightarrow e \in E_n]$

Algorithm structure (in pseudo code)

```
if events batch exists
    cast ray towards agent side
        if ray intersects object within walking alongside ray's length limit
            if events batch size ≥ sampling size
                for each event in events batch
                    if side in event is the same as in previous event
                        continue
                    else
                        clear events batch
                        break

                if side is the same in all events in events batch
                    if static object ID is the same in first and last event entry in events
batch
                        register a pattern
                        clear events batch
            else
                select the closest intersected static object
                if static object ID is equal to static object ID stored in last event in
events batch
                    record static object and agent data
                    add event to events batch
                else
                    clear events batch
```

Output Message: *"Agent ID moves alongside Static Object ID"*

Sources
Algorithm: Simulation.states / WorldAdvancedLogger.java *Event data bean*: Simulation.beans / RayEventBean.java

Table. A.2-4. "*Somebody climbs something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location, dimensions, shape	- Walking straight - Determining if Agent is above or below Static Object	- ***Agent data***: ID, location, walking direction - ***Static Object data***: ID, location, surface dimensions and shape - ***climbing ray's length limit***: used for delimiting ray's length and consequently defining a maximum distance at which collision may be detected - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***climbing interval***: how frequently events are being captured for this pattern

Description

Details of ray emission

$$r(x) = \vec{u}_p + x\vec{v}$$

$r(x)$: point laying on the ray surface located at x distance from its *emission point*
$\vec{u}_p$: the location of a *central node* of an agent defined in global coordinates
$\vec{v}$: a direction vector [0,1,0] for "up" and [0,-1,0] for "down"

Condition for registering event data for the pattern analysis

The events constituting a pattern of "climbing something" are registered when a ray of a specified length intersects an object located above or below an Agent.

$$p_1 \cdot \vec{n}_1 + (-p_0 \cdot \vec{n}_1) > 0$$

p_1 : point of intersection with an object geometry's triangle
$\vec{n}_1$: a normalized normal vector of additional triangle constructed during the operation
p_0 : point from which the ray has been emitted, e.g. agent's location

Criterion for recognising a pattern from accumulated events

Determination of whether an Agent climbs something up or down is dependent on the values of y-component of an Agent's location vector recorded in *first* and *last* entry in *events batch*.

Climbing up:

$$E_0 \cap E_n \neq \emptyset \textbf{ and } |E_0| = |E_n| \textbf{ and } E_0 = E_n \textbf{ and } \vec{u}_{0_y} < \vec{u}_{n_y}$$

Climbing down:

$$E_0 \cap E_n \neq \emptyset \textbf{ and } |E_0| = |E_n| \textbf{ and } E_0 = E_n \textbf{ and } \vec{u}_{0_y} > \vec{u}_{n_y}$$

E_0 , E_n: the set of *qualitative* and *comparable* data recorded in *first* and *last* entry in *events batch*: side of an agent which is probed by ray and static object ID
$|E_0|$, $|E_n|$: the *cardinality* of *event data set* recorded in *first* and *last* entry in *events batch*

$E_0 = E_n \leftrightarrow \forall e[e \in E_0 \leftrightarrow e \in E_n]$
$\vec{u}_0, \vec{u}_n$: the location of an agent in a world space recorded in *first* and *last* entry in *events batch*

Algorithm structure (in pseudo code)

```
if events batch exists
    cast ray towards agent side
        if ray intersects object within climbing ray's length limit
            if events batch size ≥ sampling size
                for each event in events batch
                    if side in event is the same as in previous event
                        continue
                    else
                        clear events batch
                        break

                if side is the same in all events in events batch
                    if Y-component of agent location in first event entry < Y-component of
agent location in last event entry
                        register climbing up pattern
                    else if Y-component of agent location in first event entry > Y-component
of agent location in last event entry
                        register climbing down pattern
                    clear events batch
            else
                select the closest intersected static object
                if static object ID is equal to static object ID stored in last event in
events batch
                     record static object and agent data
                     add event to events batch
                else
                     clear events batch
```

Output Message: *"Agent ID climbs up/down Static Object ID"*

Sources

Algorithm: Simulation.states / WorldAdvancedLogger.java
Event data bean: Simulation.beans / RayEventBean.java

Table. A.2-5. "*Somebody punches something / somebody*" behavioural pattern

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, limbs IDs (*hands*, *lower arms*, *chest*), limbs locations (*hands*, *lower arms*, *chest*) - ***Static Object***: ID, location, dimensions, shape	- Determining whether Agent stretches out an arm - Determining whether Agent punches Static Object / other Agent with a hand	- ***Agent data***: ID, location, walking direction, limbs IDs, limbs locations - ***Static Object data***: ID, location, surface dimensions and shape - ***minimum distance between hand and lower arm threshold***: minimum distance between *hand* and *low arm* that needs to be reached - ***minimum distance between hand and chest threshold***: minimum distance between *hand* and *chest* that needs to be reached - ***minimum distance between lower arm and chest threshold***: minimum distance between *low arm* and *chest* that needs to be reached - ***punching interval***: how frequently distances between limbs are being measured and collision detection tests carried out for this pattern - ***distance between hand and low arm***: distance between hand and low arm limbs locations - ***distance between hand and chest***: distance between hand and chest limbs locations - ***distance between low arm and chest***: distance between low arm and chest limbs locations
Description		

Condition for triggering event for the pattern analysis

A single event constituting a pattern of "punching something or somebody" is registered when a ghost sphere geometry surrounding a hand collides with a ghost sphere geometry surrounding Static Object or other Agent.

$$t \geq 0 \textbf{ and } t < 1$$

where

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_P + r_Q)^2]}}{B^2}$$

A : $P_1 - Q_1$ where P_1 and Q_1 are *initial centre position points* of two ghost spheres
B : difference of two velocity vectors calculated by $(P_2 - P_1) - (Q_2 - Q_1)$ where P_1 and Q_1 are *initial centre position points* and P_2 and Q_2 are *final centre position points* of two ghost spheres
r_P , r_Q: radii of the two spheres

Criterion for recognising a pattern

To capture a pattern when an Agent punches a Static Object or other Agent, the ghost sphere surrounding Agent's hand bone needs to collide with ghost sphere surrounding Static Object or other Agent. Furthermore, three distances between limbs need to be reached specified minimum thresholds.

$$|G_1 \cap G_2| > 0 \textbf{ and } \|\vec{u}_{ch} - \vec{u}_h\| \geq td_{chh} \textbf{ and } \|\vec{u}_e - \vec{u}_h\| \geq td_{eh} \textbf{ and } \|\vec{u}_{ch} - \vec{u}_e\| \geq td_{che}$$

$|G_1 \cap G_2|$: *cardinality* of a set resulting from intersection of two sets of points constituting ghost spheres surfaces surrounding Agent hand limb and other Agent or Static Object
$\vec{u}_{ch}$: location of Agent's chest limb
$\vec{u}_h$: location of Agent's hand limb
$\vec{u}_e$: location of Agent's elbow / lower arm limb
td_{chh}: minimum distance between hand and chest threshold
td_{eh}: minimum distance between elbow / lower arm and hand threshold
td_{che}: minimum distance between elbow / lower arm and chest threshold

Algorithm structure (in pseudo code)

```
if hand ghost sphere collides with other object ghost sphere
    if distance between hand and lower arm ≥ minimum distance between hand and lower arm threshold
        if distance between hand and chest ≥ minimum distance between hand and chest threshold
            if distance between lower arm and chest ≥ minimum distance between lower arm and chest threshold
                register pattern
```

Output Message: *"Agent ID punches Agent ID / Static Object ID"*

Sources

Algorithm: Simulation.states / WorldAdvancedLogger.java

Table. A.2-6. "*Somebody kicks something / somebody*" behavioural pattern

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, limbs IDs (*feet*, *lower leg*, *chest*), limbs locations (*feet*, *lower leg*, *chest*) - ***Static Object***: ID, location, dimensions, shape	- Determining whether Agent stretches out a leg - Determining whether Agent kicks Static Object / other Agent with a foot	- ***Agent data***: ID, location, walking direction, limbs IDs, limbs locations - ***Static Object data***: ID, location, surface dimensions and shape - ***minimum distance between foot and lower leg threshold***: minimum distance between *foot* and *lower leg* that needs to be reached - ***minimum distance between foot and chest threshold***: minimum distance between *foot* and *chest* that needs to be reached - ***minimum distance between lower leg and chest threshold***: minimum distance between *lower leg* and *chest* that needs to be reached - ***kicking interval***: how frequently distances between limbs are being measured and collision detection tests carried out for this pattern - ***distance between foot and lower leg***: distance between *foot* and *lower leg* limbs locations - ***distance between foot and chest***: distance between *foot* and *chest* limbs locations - ***distance between lower leg and chest***: distance between *lower leg* and *chest* limbs locations
Description		

Condition for triggering event for the pattern analysis

A single event constituting a pattern of "punching something or somebody" is registered when a ghost sphere geometry surrounding a hand collides with a ghost sphere geometry surrounding Static Object or other Agent.

$$t \geq 0 \textbf{ and } t < 1$$

where

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_P + r_Q)^2]}}{B^2}$$

A : $P_1 - Q_1$ where P_1 and Q_1 are *initial centre position points* of two ghost spheres
B : difference of two velocity vectors calculated by $(P_2 - P_1) - (Q_2 - Q_1)$ where P_1 and Q_1 are *initial centre position points* and P_2 and Q_2 are *final centre position points* of two ghost spheres
r_P ,r_Q: radii of the two spheres

Criterion for recognising a pattern

To capture a pattern when an Agent kicks a Static Object or other Agent, the ghost sphere surrounding Agent's foot bone needs to collide with ghost sphere surrounding Static Object or other Agent. Furthermore, three distances between limbs need to be reached specified minimum thresholds.

$$|G_1 \cap G_2| > 0 \textbf{ and } \|\vec{u}_{ch} - \vec{u}_f\| \geq td_{chf} \textbf{ and } \|\vec{u}_l - \vec{u}_f\| \geq td_{lf} \textbf{ and } \|\vec{u}_{ch} - \vec{u}_l\| \geq td_{chl}$$

$|G_1 \cap G_2|$: *cardinality* of a set resulting from intersection of two sets of points constituting ghost spheres surfaces surrounding Agent foot limb and other Agent or Static Object
$\vec{u}_{ch}$: location of Agent's chest limb
$\vec{u}_f$: location of Agent's foot limb
$\vec{u}_l$: location of Agent's lower leg
td_{chf}: minimum distance between foot and chest threshold
td_{lf}: minimum distance between lower leg and foot threshold
td_{chl}: minimum distance between lower leg and chest threshold

Algorithm structure (in pseudo code)

```
if foot ghost sphere collides with other object ghost sphere
    if distance between foot and lower leg ≥ minimum distance between foot and lower leg threshold
        if distance between foot and chest ≥ minimum distance between foot and chest threshold
            if distance between lower leg and chest ≥ minimum distance between lower leg and chest threshold
                register pattern
```

Output Message: *"Agent ID kicks Agent ID / Static Object ID"*

Sources

Algorithm: Simulation.states / WorldAdvancedLogger.java

Table. A.2-7. "*Somebody picks up / reaches out for / carry something*" behavioural pattern

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, limbs IDs (*hands*, *lower arms*, *chest*), limbs locations (*hands*, *lower arms*, *chest*) - ***Dynamic Object***: ID, location, dimensions, shape	- Determining whether Agent stretches out an arm - Determining whether Agent already carry Dynamic Object in hand - Determining whether Agent picks up Dynamic Object from the floor or higher levels, potentially a Static Object such as shelf	- ***Agent data***: ID, location, walking direction, limbs IDs, limbs locations - ***Dynamic Object data***: ID, location, surface dimensions and shape - ***minimum distance between hand and lower arm threshold***: minimum distance between *hand* and *low arm* that needs to be reached - ***minimum distance between hand and chest threshold***: minimum distance between *hand* and *chest* that needs to be reached - ***minimum distance between lower arm and chest threshold***: minimum distance between *low arm* and *chest* that needs to be reached - ***pick up interval***: how frequently distances between limbs are being measured and collision detection tests carried out for this pattern - ***distance between hand and lower arm***: distance between hand and lower arm limbs locations - ***distance between hand and chest***: distance between hand and chest limbs locations - ***distance between lower arm and chest***: distance between lower arm and chest limbs locations
Description		

Condition for triggering event for the pattern analysis

A single event that triggers the pattern evaluation of "picking something up" is registered when a ghost sphere geometry surrounding a hand collides with a ghost sphere geometry surrounding Dynamic Object.

$$t \geq 0 \textbf{ and } t < 1$$

where

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_P + r_Q)^2]}}{B^2}$$

A : $P_1 - Q_1$ where P_1 and Q_1 are *initial centre position points* of two ghost spheres
B : difference of two velocity vectors calculated by $(P_2 - P_1) - (Q_2 - Q_1)$ where P_1 and Q_1 are *initial centre position points* and P_2 and Q_2 are *final centre position points* of two ghost spheres
r_P ,r_Q: radii of the two spheres

Criterion for recognising a pattern

To capture a pattern when an Agent picks up a Dynamic Object, the ghost sphere surrounding Agent's hand bone needs to collide with ghost sphere surrounding Dynamic Object. The Dynamic Object ID cannot be registered as currently carried by an Agent, but if it is, it means that an Agent already carry it. Furthermore, if three distances between limbs reach specified minimum thresholds, one can conclude that Agent stretches out an arm to reach a Dynamic Object from higher levels, such as Static Object (shelf, counter etc.).

Reach out for:

$$e \notin O \textbf{ and } |G_1 \cap G_2| > 0 \textbf{ and } \|\vec{u}_{ch} - \vec{u}_h\| \geq td_{chh} \textbf{ and}$$
$$\|\vec{u}_e - \vec{u}_h\| \geq td_{eh} \textbf{ and } \|\vec{u}_{ch} - \vec{u}_e\| \geq td_{che}$$

Pick something up: $e \notin O$ **and** $|G_1 \cap G_2|$

Carry something up: $e \in O$ **and** $|G_1 \cap G_2|$

e : Dynamic Object currently being picked up by an Agent, i.e. Dynamic Object ghost sphere with whom an Agent hand ghost sphere collided with
O : a set of Dynamic Objects currently in possession of an Agent
$|G_1 \cap G_2|$: *cardinality* of a set resulting from intersection of two sets of points constituting ghost spheres surfaces surrounding Agent hand limb and Dynamic Object
$\vec{u}_{ch}$: location of Agent's chest limb
$\vec{u}_h$: location of Agent's hand limb
$\vec{u}_e$: location of Agent's elbow / lower arm limb
td_{chh}: minimum distance between hand and chest threshold
td_{eh}: minimum distance between elbow / lower arm and hand threshold
td_{che}: minimum distance between elbow / lower arm and chest threshold

Algorithm structure (in pseudo code)

```
if hand ghost sphere collides with static object ghost sphere
    if dynamic object ID is not in set of objects already carried by Agent
        if distance between hand and lower arm ≥ minimum distance between hand and lower arm
threshold
            if distance between hand and chest ≥ minimum distance between hand and chest
threshold
                if distance between lower arm and chest ≥ minimum distance between lower arm and
chest threshold
                    register reaches out for something pattern
                else
                    register picks something up pattern
```

``   add dynamic object ID to set of objects already carried by Agent   else   register dynamic object ID is carried by Agent pattern ```
Output Message: *"Agent ID picks up / reaches out for / carry Dynamic Object ID"* Note: the message when *"Agent carries a Dynamic Object ID"* is suppressed in current implementation for the purpose of reducing the noise in log file.
Sources
Algorithm: Simulation.states / WorldAdvancedLogger.java

Table. A.2-8. "*Somebody holds something over / places something on top of something*" behavioural pattern

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, view direction, walking direction - ***Dynamic Object***: ID, location, dimensions, shape - ***Static Object***: ID, location, dimensions, shape	- Determining whether Agent already carry Dynamic Object in hand - Determining whether Agent holds Dynamic Object over a Static Object - Determining whether Agent puts a Dynamic Object on top of Static Object	- ***Agent data***: ID, location - ***Dynamic Object data***: ID, location, surface dimensions and shape - ***Static Object data***: ID, location, surface dimensions and shape - ***placing interval***: how frequently collision detection tests are carried out for this pattern - ***carrying time sampling size***: the minimum number of timestamps that needs to be recorded for "*put something on top of something*" pattern evaluation - ***carrying interval***: how frequently timestamps are being recorded for "*put something on top of something*" pattern evaluation - ***timestamp time value threshold***: maximum time interval that timestamp needs to be recorded from current time for "*put something on top of something*" pattern evaluation - ***time passed from previous recording***: time interval passed from previous recording of time stamp
Description		

Details of ray emission

$$r(x) = \vec{u}_p + x\vec{v}$$

$r(x)$: point laying on the ray surface located at x distance from its *emission point*
$\vec{u}_p$: the location of a *Dynamic Object* currently carried by *Agent* defined in global coordinates
$\vec{v}$: a direction vector pointing down [0,-1,0]

Condition for triggering event for the pattern analysis

A single event that triggers the pattern evaluation of "holding something over something" is registered when a ray casted downwards from a Dynamic Object currently carried by an Agent intersects a Static Object. A single event that triggers the pattern evaluation of "put something on top of something" is registered when a ghost sphere surrounding Dynamic Object that was carried by an Agent collides with a ghost sphere surrounding a Static Object.

Ray casting collision detection:

$$p_1 \cdot \vec{n}_1 + (-p_0 \cdot \vec{n}_1) > 0$$

p_1 : point of intersection with an object geometry's triangle
$\vec{n}_1$: a normalized normal vector of additional triangle constructed during the operation
p_0 : point from which the ray has been emitted, i.e. Dynamic Object carried by Agent
Ghost spheres collision detection:

$$t \geq 0 \textbf{ and } t < 1$$

where

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_P + r_Q)^2]}}{B^2}$$

A : $P_1 - Q_1$ where P_1 and Q_1 are *initial centre position points* of two ghost spheres
B : difference of two velocity vectors calculated by $(P_2 - P_1) - (Q_2 - Q_1)$ where P_1 and Q_1 are *initial centre position points* and P_2 and Q_2 are *final centre position points* of two ghost spheres
r_P ,r_Q: radii of the two spheres

Criterion for recognising a pattern

To capture a pattern when "*Agent holds a Dynamic Object over a Static Object*", it needs to be in possession of an Agent and the ray casted downwards from that Dynamic Object needs to intersect a Static Object. To capture a pattern when "*Agent puts a Dynamic Object on top of a Static Object*", the following conditions need to be met:

- the Dynamic Object had to be in possession of an Agent. We do this by recording and evaluating timestamps when Agent carries a Dynamic Object and when Dynamic Object ghost sphere collides with Static Object ghost sphere.
- ghost sphere surrounding a Dynamic Object needs to collide with a Static Object

Holding over:

$$e \in O \textbf{ and } x \notin \emptyset$$

Put on:

$$e_t \in T \textbf{ and } \Delta t \leq td \textbf{ and } |G_1 \cap G_2| > 0$$

$e \in O$: Dynamic Object currently in possession of an Agent
x : the collision point detected by a ray casted downwards from Dynamic Object
$x \notin \emptyset$: the collision points detected by a ray casted downwards from Dynamic Object
e_t : Dynamic Object data that was in possession of an Agent recorded at time t as a *timestamp*
$e_t \in T$: a set of all timestamps T to which e_t belongs to
$|G_1 \cap G_2|$: *cardinality* of a set resulting from intersection of two sets of points constituting ghost spheres surfaces surrounding Dynamic Object and Static Object
Δt : a difference between *current time value* and *time value recorded in timestamp* e_t.
td : maximum time threshold value

Algorithm structure (in pseudo code)

```
if dynamic object ID is in set of objects already carried by Agent
    cast ray downwards from carried dynamic object
        if ray intersects static object
            register hold something over something pattern

    if dynamic object ghost sphere collides with static object ghost sphere
        if timestamps batch size ≥ carrying time sampling size
            for each timestamp in timestamps batch
                if (current time value - timestamp time value) ≤ timestamp time value
threshold
                    register put something on something pattern

   if time passed from previous recording < carrying interval
      if timestamps batch size ≥ carrying time sampling size
         clear timestamps batch

      record dynamic object and timestamp data
      add timestamp to set of timestamps
```

Output Message: *"Agent ID holds Dynamic Object ID over / puts Dynamic Object ID on Static Object ID"*

Sources

Algorithm: Simulation.states / WorldAdvancedLogger.java
Timestamp data bean: Simulation.beans / CarriedObjectBean.java

Table. A.2-9. "*Somebody passes over something to somebody*" behavioural pattern

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, limbs IDs (*hands*), limbs locations (*hands*) - ***Dynamic Object***: ID, location, dimensions, shape	- Determining whether Agent already carry Dynamic Object in hand	- ***Agent data***: ID, location, walking direction, limbs IDs, limbs locations - ***Dynamic Object data***: ID, location, surface dimensions and shape - ***pass over interval***: how frequently collision detection tests are carried out for this pattern - ***minimum distance between hands threshold***: minimum distance between two Agent's hands limbs - ***maximum distance between hands threshold***: maximum distance between two Agent's hands limbs - ***minimum distance between Agent hand and Dynamic Object threshold***: minimum distance between Agent hand limb and Dynamic Object - ***distance between hands***: distance between hand limb of *first* Agent and hand limb of *second* Agent - ***distance between first Agent hand limb and Dynamic Object***: distance between hand limb of *first* Agent and Dynamic Object - ***distance between second Agent hand limb and Dynamic Object***: distance between hand limb of *second* Agent and Dynamic Object
Description		

Condition for triggering event for the pattern analysis

A single event that triggers the pattern evaluation of *"passing something over to another Agent"* is registered when ghost sphere surrounding a Dynamic Object currently carried by an Agent collides with a ghost sphere surrounding a hand limb of another Agent.

$$t \geq 0 \textbf{ and } t < 1$$

where

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_P + r_Q)^2]}}{B^2}$$

A : $P_1 - Q_1$ where P_1 and Q_1 are *initial centre position points* of two ghost spheres
B : difference of two velocity vectors calculated by $(P_2 - P_1) - (Q_2 - Q_1)$ where P_1 and Q_1 are *initial centre position points* and P_2 and Q_2 are *final centre position points* of two ghost spheres
r_P ,r_Q: radii of the two spheres

Criterion for recognising a pattern

To capture a pattern when "*Agent passes Dynamic Object over to another Agent*", the following conditions need to be met:

- the Dynamic Object needs to be in possession of an Agent.
- ghost sphere surrounding carried Dynamic Object needs to collide with ghost sphere surrounding another Agent's hand limb
- the distances between hand limbs need to fall within certain thresholds
- the distances between hand limbs and Dynamic Object need to fall within certain thresholds

$$e \in O_1 \textbf{ and } e \notin O_2 \textbf{and } |G_1 \cap G_2| > 0 \textbf{ and}$$
$$\|\vec{u}_{h1} - \vec{u}_{h2}\| \geq htd_{min} \textbf{ and } \|\vec{u}_{h1} - \vec{u}_{h2}\| \leq htd_{max} \textbf{ and}$$
$$\|\vec{u}_{h1} - \vec{v}\| \geq td_{min} \textbf{ and } \|\vec{u}_{h2} - \vec{v}\| \geq td_{min}$$

$e \in O_1$: Dynamic Object currently in possession of *first* Agent
$e \notin O_2$: Dynamic Object cannot currently be in a possession of *second* Agent
$|G_1 \cap G_2|$: *cardinality* of a set resulting from intersection of two sets of points constituting ghost spheres surfaces surrounding Dynamic Object and hand limb of *second* Agent
$\vec{u}_{h1}$, $\vec{u}_{h2}$: location of a hand limb of *first* and *second* Agent
$\vec{v}$: location of a Dynamic Object
htd_{min}, htd_{max} : *minimum* and *maximum* distance thresholds between hands
td_{min}: *minimum* distance threshold between hands and Dynamic Object

Algorithm structure (in pseudo code)

```
if dynamic object ID is in set of objects already carried by first Agent
    if dynamic object ghost sphere collides with second Agent hand limb ghost sphere
        if dynamic object ID is not in set of objects already carried by second Agent
            if distance between hands ≥ minimum distance between hands threshold
                if distance between hands ≤ maximum distance between hands threshold
                    if distance between first Agent hand and dynamic object location ≥
minimum distance between Agent hand and Dynamic Object threshold
                        if distance between second Agent hand and dynamic object location ≥
minimum distance between Agent hand and Dynamic Object threshold
                            register pattern
                            remove dynamic object ID from set of objects already carried by
first Agent
                            add dynamic object ID to set of objects already carried by second
Agent
```

Output Message: *"Agent ID passes Dynamic Object ID over to Agent ID"*
Sources
Algorithm: Simulation.states / WorldAdvancedLogger.java

Table. A.2-10. "*Somebody drops down something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, limbs IDs (*hands*), limbs locations (*hands*) - ***Dynamic Object***: ID, location, dimensions, shape	- Determining whether Agent already carry Dynamic Object in hand - Determining how far is the carried Dynamic Object from Agent hand limb	- ***Agent data***: ID, location, walking direction, limbs IDs, limbs locations - ***Dynamic Object data***: ID, location, surface dimensions and shape - ***minimum distance between hand and Dynamic Object threshold***: minimum distance between Agent's hand and carried Dynamic Object at which pattern can be captured - ***distance between Agent hand limb and Dynamic Object***: distance between hand limb of Agent and Dynamic Object

Description

Criterion for recognising a pattern

To capture a pattern when "*Agent drops down a Dynamic Object*", the following conditions need to be met:

- the Dynamic Object needs to be in possession of an Agent.
- Dynamic Object needs to be within certain distance from Agent's hand limb

$$e \in O \textbf{ and } \|\vec{u}_h - \vec{v}\,\| \geq htd_{min}$$

$e \in O$: Dynamic Object currently in possession of Agent
$\vec{u}_h$: location of Agent hand limb
$\vec{v}$: location of carried Dynamic Object
htd_{min} : *minimum* distance threshold between hand and Dynamic Object

Algorithm structure (in pseudo code)

```
if dynamic object ID is in set of objects already carried by Agent
    if distance between hand and dynamic object ≥ minimum distance between Agent hand and
Dynamic Object threshold
        register pattern
        remove dynamic object ID from set of objects already carried by Agent
```

Output Message: *"Agent ID drops down Dynamic Object ID"*

Sources

Algorithm: Simulation.states / WorldAdvancedLogger.java

Table. A.2-11. "*Somebody looks up / down / right / left*" behavioural pattern

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, view direction, walking direction	- Agent looks up / down / left /right	- ***Agent data***: ID, view direction - ***minimum angle between viewing direction and Y-axis for looking up threshold***: minimum angle between viewing direction and Y-axis for registering *"looks up"* pattern - ***maximum angle between viewing direction and Y-axis for looking up threshold***: maximum angle between viewing direction and Y-axis for registering *"looks up"* pattern - ***minimum angle between viewing direction and Y-axis for looking down threshold***: minimum angle between viewing direction and Y-axis for registering *"looks down"* pattern - ***maximum angle between viewing direction and Y-axis for looking down threshold***: maximum angle between viewing direction and Y-axis for registering *"looks down"* pattern - ***minimum angle between viewing direction and X-axis for looking left threshold***: minimum angle between viewing direction and X-axis for registering *"looks left"* pattern - ***maximum angle between viewing direction and X-axis for looking left threshold***: maximum angle between viewing direction and X-axis for registering *"looks left"* pattern - ***minimum angle between viewing direction and X-axis for looking right threshold***: minimum angle between viewing direction and X-axis for registering *"looks right"* pattern - ***maximum angle between viewing direction and X-axis for looking right threshold***: maximum angle between viewing direction and X-axis for registering *"looks right"* pattern

Description

Criterion for recognising a pattern

To capture a pattern when "*Agent looks up / down / left / right*", the angle between viewing direction vector and Y-axis (up / down) or X-axis (left / right) need to fall within certain thresholds, e.g.:

Look up:

$$\measuredangle(\vec{v},\vec{u}\,) \geq td_{min} \textbf{ and } \measuredangle(\vec{v},\vec{u}\,) \leq td_{max}$$

$\vec{u}$: unit vector defining a standard basis of Y-axis of a local Cartesian coordinates space of an agent [0,1,0]
$\vec{v}$: unit vector defining a viewing direction of an Agent

$\measuredangle(\vec{v},\vec{u})$: angle between two unit vectors td_{min}, td_{max} : *minimum* and *maximum* distance thresholds between viewing direction and Y-axis for capturing "looking up" pattern *Look left:* $\measuredangle(\vec{v},\vec{u}) \geq td_{min}$ **and** $\measuredangle(\vec{v},\vec{u}) \leq td_{max}$ $\vec{u}$: unit vector defining a standard basis of X-axis of a local Cartesian coordinates space of an agent [1,0,0] $\vec{v}$: unit vector defining a viewing direction of an Agent $\measuredangle(\vec{v},\vec{u})$: angle between two unit vectors td_{min}, td_{max} : *minimum* and *maximum* distance thresholds between viewing direction and X-axis for capturing "looking left" pattern
Algorithm structure (in pseudo code)
if *angle between viewing direction and standard basis axis unit vector* ≥ ***minimum angle between viewing direction and standard basis axis unit vector for pattern threshold***   **if** *angle between viewing direction and standard basis axis unit vector* ≤ ***maximum angle between viewing direction and standard basis axis unit vector for pattern threshold***   **register** pattern Note: generic algorithm structure
Output Message: *"Agent ID looks up / down / right / left"*
Sources
Algorithm: Simulation.states / WorldAdvancedLogger.java

Patterns of group dynamic behaviour

Table. A.2-12. "*Somebody and somebody form a Pair*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location	- Estimating distance between one Agent and another Agent - Determine if Agent is a member of a Pair / Group	- ***Agent data***: ID, location - ***group formation interval***: how frequently distances between Agents are checked for this pattern - ***minimum distance between Agents to form a Pair threshold***: minimum distance between Agents to form a Pair - ***distance between Agents***: distance between two Agents
Description		

Criterion for recognising a pattern
To capture a pattern when "*Agent and another Agent forms a Pair*", neither of the Agents can already be a member of a Pair / Group. Furthermore, they need to find themselves within a close distance to each other that falls within a certain threshold. $$e_1 \notin G \textbf{ and } e_2 \notin G \textbf{ and } \|\vec{u}_1 - \vec{u}_2\| \leq td_{min}$$ $e_1 \notin G, e_2 \notin G$: *first* and *second* Agent does not belong to any Pair / Group $\vec{u}_1, \vec{u}_2$: location of *first* and *second* Agent defined in global coordinate space td_{min} : *minimum* distance threshold between two Agents that needs to be reached in order to form a Pair
Algorithm structure (in pseudo code)
`if distance between Agents ≤ minimum distance between Agents to form a Group threshold` `    if first Agent is not already in a Pair or Group` `        if second Agent is not already in a Pair or Group` `            register pattern` `            form Pair` `            add first and second Agent to Pair`
Output Message: *"Agent ID and Agent ID form a Pair ID"*
Sources: *Algorithm*: Simulation.states / GroupsManagerState.java

Table. A.2-13. "*Somebody joins a Group*" behavioural pattern and "*Two Pairs / Groups merge"* behavioural pattern.

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location	- Estimating distance between one Agent and another Agent - Determine if Agent is a member of a Pair / Group	- ***Agent data***: ID, location - ***group formation interval***: how frequently distances between Agents are checked for this pattern - ***minimum distance between Agents to join a Group threshold***: minimum distance between Agents to join a Pair / Group or for two different Pairs / Groups to merge - ***distance between Agents***: distance between two Agents
Description		

Criterion for recognising a pattern

To capture a pattern when "*Agent joins a Pair / Group*", an Agent cannot be already a member of a Pair or Group and needs to be within a certain distance from an Agent who is a member of a Pair or Group. To capture a pattern when "*Pair / Group merges with Pair / Group*", both Agents need to be members of two different Pairs / Groups and be within a certain distance from each other.

Joining a Pair / Group:

$$e_1 \notin G \textbf{ and } e_2 \in G \textbf{ and } \|\vec{u}_1 - \vec{u}_2\| \leq td_{min}$$

Merging of two Pairs:

$$e_1 \in G_1 \textbf{ and } e_2 \in G_2 \textbf{ and } |G_1| = 2 \textbf{ and } |G_2| = 2 \textbf{ and } \|\vec{u}_1 - \vec{u}_2\| \leq td_{min}$$

Merging of two Groups:

$$e_1 \in G_1 \textbf{ and } e_2 \in G_2 \textbf{ and } |G_1| > 2 \textbf{ and } |G_2| > 2 \textbf{ and } \|\vec{u}_1 - \vec{u}_2\| \leq td_{min}$$

e_1, e_2 : two different Agents being elements of Group sets
$e_1 \notin G, e_2 \in G$: *first* and *second* Agent does not belong to any Pair / Group
G_1, G_2: two different Group sets consisting of member Agents
$\vec{u}_1, \vec{u}_2$: location of first and second Agent defined in global coordinate space
td_{min} : *minimum* distance threshold between two Agents that needs to be reached in order for Agent to join a Pair / Group or two different Pairs / Groups to merge

Algorithm structure (in pseudo code)

```
if distance between Agents ≤ minimum distance between Agents to join a Group threshold
    if first Agent is not already in a Pair or Group
        if second Agent is already in a Group
            register first Agent joins second Agent Group pattern
            add first Agent to second Agent Group
    else if first Agent is already in a Pair or Group
        if second Agent is already in a Different Pair or Group
            register first Agent Group merges with second Agent Group pattern
            create new Group
            add first Agent Group members and second Agent Group members to new Group
```

Output Message: *"Agent ID joins a Pair ID / Group ID"*, *"Pair ID / Group ID merged with Pair ID / Group ID"*

Sources: *Algorithm*: Simulation.states / GroupsManagerState.java

Table. A.2-14. "*Somebody leaves a Group*" behavioural pattern and "*Group disbands*" behavioural pattern.

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location	- Estimating distance between one Agent and another Agent - Determine if Agent is a member of a Pair / Group	- ***Agent data***: ID, location - ***group formation interval***: how frequently distances between Agents are checked for this pattern - ***minimum distance between Agents to leave a Group threshold***: minimum distance between Agents that needs to be reached for an Agent to leave a Group - ***distance between Agents***: distance between two Agents
Description		
Criterion for recognising a pattern To capture a pattern when "*Agent leaves a Group*", an Agent needs to be a member of a Group and find itself within a certain minimum distance from one of the members of the Group. Since it is difficult to determine which of two Agents is leaving the Group on a basis of a distance between them, the developed, original yet simple approach presented in Formula A.2-1 is used. When the distance between two members of the Group reaches a certain threshold, one can calculate the distances between each of them and the rest of the group members. The sum of all distances is then compared against each other in order to determine which of two Agents is farther away from the rest of the Group members. $\mathrm{d}_1 = \sum_{i=1}^{n} \lVert \overrightarrow{ue}_1 - \overrightarrow{ue}_i \rVert \qquad \mathrm{d}_2 = \sum_{i=1}^{n} \lVert \overrightarrow{ue}_2 - \overrightarrow{ue}_i \rVert$ where n: number of members of a Group $\overrightarrow{ue}_1, \overrightarrow{ue}_2$: location of *first* and *second* Agent being members of the same Group defined in global coordinate space for whom $\lVert \overrightarrow{ue}_1 - \overrightarrow{ue}_2 \rVert > td_{min}$ condition was registered. $\overrightarrow{ue}_i$: location of a member of a Group $\overrightarrow{ue}_1 \neq \overrightarrow{ue}_i$ **and** $\overrightarrow{ue}_1 \neq \overrightarrow{ue}_2$ for d_1 $\overrightarrow{ue}_2 \neq \overrightarrow{ue}_i$ **and** $\overrightarrow{ue}_2 \neq \overrightarrow{ue}_1$ for d_2 First Agent leaves the group if and only if $\mathrm{d}_1 > \mathrm{d}_2$ Second Agent leaves the group if and only if $\mathrm{d}_2 > \mathrm{d}_1$ **Formula A.2-1.** Original equation for determining which of two Group members is leaving the Group based on their overall distance to other members of the Group. $e_1 \in G_1$ **and** $e_2 \in G_1$ **and** $\lvert G_1 \rvert > 2$ **and** $\lVert \vec{u}_1 - \vec{u}_2 \rVert > td_{min}$ e_1, e_2 : two different Agents being elements of a Group set G_1: a Group set containing all member Agents data $\vec{u}_1, \vec{u}_2$: location of *first* and *second* Agent being members of Group set G_1 defined in global coordinate space whose distance is greater than minimum distance threshold $\lVert \vec{u}_1 - \vec{u}_2 \rVert > td_{min}$ td_{min} : *minimum* distance threshold between two member Agents that needs to be reached in order for Agent to leave a Group		
Algorithm structure (in pseudo code)		

```
if distance between Agents > minimum distance between Agents to leave a Group threshold
    if first Agent is already in Group
        if second Agent is already in the same Group
            if first Agent overall distance to other members of Group > second Agent overall distance to other members of Group
                register first Agent leaves Group pattern
                remove first Agent from Group
            else if second Agent overall distance to other members of Group > first Agent overall distance to other members of Group
                register second Agent leaves Group pattern
                remove second Agent from Group

            if Group size = 2
                change Group type to Pair
            else if Group size = 1
                register Group disbands pattern
                remove Group
```

Output Message: *"Agent ID leaves a Group ID"*, *"Group ID disbands"*

Sources: *Algorithm*: Simulation.states / GroupsManagerState.java

Table. A.2-15. "*Somebody makes handshake with somebody*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, limbs IDs (*hands*), limbs locations (*hands*)	- Determining if two Agents are in a Group or Pair - Determining if two Agents hands are close to each other for a period of time	- ***Agent data***: ID, location, walking direction, limbs IDs, limbs locations - ***handshake gesture interval***: how frequently collision detection tests are carried out for this pattern - ***minimum distance between hands threshold***: minimum distance between two Agent's hands limbs - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***distance between first Agent hand limb and second Agent hand limb***: distance between hand limb of *first* Agent and hand limb of *second* Agent
Description		

Condition for triggering event for the pattern analysis

The events constituting a pattern of "*Agent making a handshake with other Agent* " are registered when ghost spheres surrounding each other hands collide. The *minimum distance threshold between hands* has been introduced in this case as a control measure for capturing events.

$$t \geq 0 \textbf{ and } t < 1 \textbf{ and } \|\vec{u}_1 - \vec{u}_2\| > td_{min}$$

where

$$t = \frac{-(A \cdot B) - \sqrt{(A \cdot B)^2 - B^2[A^2 - (r_P + r_Q)^2]}}{B^2}$$

A : $P_1 - Q_1$ where P_1 and Q_1 are *initial centre position points* of two ghost spheres
B : difference of two velocity vectors calculated by $(P_2 - P_1) - (Q_2 - Q_1)$ where P_1 and Q_1 are *initial centre position points* and P_2 and Q_2 are *final centre position points* of two ghost spheres
r_P ,r_Q: radii of the two spheres
$\vec{u}_1, \vec{u}_2$: distance between two ghost spheres at the time of collision
td_{min}: *minimum* distance threshold between a hand limb of *first* and *second* Agent participating in the genesis of behavioural pattern

Criterion for recognising a pattern

$$E_{1_0} = E_{2_0} \textbf{ and } E_{1_n} = E_{2_n} \textbf{ and}$$
$$|G_{1_0} \cap G_{2_0}| < 0 \textbf{ and } |G_{1_n} \cap G_{2_n}| < 0$$

$E_{1_0}, E_{2_0}, E_{1_n}, E_{2_n}$: The set of *qualitative* and *comparable* data recorded in *first* and *last* entry in *events batch* identifying *first* and *second* Agent involved in the genesis of behavioural pattern.
$|G_{1_0} \cap G_{2_0}|$, $|G_{1_n} \cap G_{2_n}|$: *The cardinality* of a set resulting from intersection of two sets of points constituting ghost spheres surfaces surrounding hand limb of *first* Agent and hand limb of *second* Agent recorded in *first* and *last* entry in *events batch*.

Algorithm structure (in pseudo code)

```
if events batch exists
    if first Agent hand limb ghost sphere collides with second Agent hand limb ghost sphere
        if distance between hands ≥ minimum distance between hands threshold
            if events batch size ≥ sampling size
                for each event in events batch
                    if first Agent ID in event is the same as in previous event
                        if second Agent ID in event is the same as in previous event
                            continue
                        else
                            clear events batch
                            break
                    else
                        clear events batch
                        break

                if first Agent ID in first event entry is the same as in last event entry
                    if second Agent ID in first event entry is the same as in last event
entry
                        register pattern
                        clear events batch
            else
                record agent data
                add event to events batch
```

Output Message: *"Agent ID makes handshake with Agent ID"*

Sources
Algorithm: Simulation.states / GroupsLoggerState.java *Event data bean*: Simulation.beans / GestureEventBean.java

Table. A.2-16. "*Group / Pair moves towards something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location	- Group walking straight - Determining distance between Group or Pair and Static Object - Determining average viewing direction - Determining average walking direction	- ***Group data***: location (midpoint), average viewing direction, average walking direction - ***Static Object data***: location, orientation (towards agent) - ***walking towards distance threshold***: used for determining if Agent is within a specific radius from Static Object - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***walking towards interval***: how frequently events are being captured for this pattern - ***overall direction proximity coefficient***: used for calculating the threshold against which the overall direction proximity value is checked against. The coefficient is restricted to the range of [0,1]. - ***overall direction proximity***: the calculated sum of dot products stored in events batch - ***distance***: between Group midpoint location and Static Object location

Description

Condition for registering event data for the pattern analysis

The events constituting a pattern of "Group / Pair walking towards" are registered when midpoint of a Group or Pair is located within a predefined distance threshold t_d from an object. In situations when several objects are located within this threshold, the closest one is selected.

$$\|\vec{v}_t - G_{Mt}\| \leq t_d$$

$\vec{v}_t$: the location of a nearby object in a world space at the time t that is involved in the event
G_{Mt} : the location of a Group or Pair in a world space at the time t when event occurred
t_d : distance threshold where $t_d \in \mathbb{N} : t_d \in [1, N]$

Criterion for recognising a pattern from accumulated events

$$\sum_{i=1}^{n} (\vec{v}_i - G_{Mi}\ ) \cdot \overrightarrow{Gdir_i} \geq n - (nC)$$

$\vec{v}_i$: the location of an object in a world space recorded in entry i in *events batch*
G_{Mi} : the *midpoint* location of a Group or Pair in a world space recorded in entry i in *events batch*
$\overrightarrow{Gdir_t}$: unit vector defining average viewing direction of a Group or Pair recorded in entry i in *events batch*
n : total number of registered events where $n > 0$

C : threshold coefficient where $C \in \mathbb{N} : 0 < C \leq 1$
Algorithm structure (in pseudo code)
<pre>if events batch exists   if distance ≤ walking towards distance threshold   if events batch size ≥ sampling size   for each event in events batch   calculate dot product of direction vector and Group or Pair view direction vector   add dot product value to overall direction proximity   if overall direction proximity ≥ (events batch size * 1.0) - overall direction proximity coefficient * (events batch size * 1.0)   register a pattern   clear events batch   else   select the closest static object   if static object ID is equal to static object ID stored in last event in events batch   record static object and Group or Pair data   add event to events batch   else   clear events batch</pre>
Output Message: *"Group ID / Pair ID moves towards Static Object ID"*
Sources
Algorithm: Simulation.states / GroupsLoggerState.java *Event data bean*: Simulation.beans / GroupEventBean.java

Table. A.2-17. "*Group / Pair moves away from something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location	- Group walking straight - Determining distance between Group or Pair and Static Object - Determining average viewing direction - Determining average walking direction	- ***Group data***: location (midpoint), average viewing direction, average walking direction - ***Static Object data***: location, orientation (towards agent) - ***walking away distance threshold***: used for determining if Agent is outside a specific radius of Static Object - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***walking away interval***: how frequently events are being captured for this pattern - ***distance***: between Group midpoint location and Static Object location
Description		

Condition for registering event data for the pattern analysis

The events constituting a pattern of "Group / Pair walking away" are registered when midpoint of a Group or Pair is located within a predefined distance threshold t_d from an object. In situations when several objects are located within this threshold, the closest one is selected.

$$\|\vec{v}_t - G_{Mt}\| \leq t_d$$

$\vec{v}_t$: the location of a nearby object in a world space at the time t that is involved in the event
G_{Mt} : the location of a Group or Pair in a world space at the time t when event occurred
t_d : distance threshold where $t_d \in \mathbb{N} : t_d \in [1, N]$

Criterion for recognising a pattern from accumulated events

$$(\vec{v}_0 - G_{M0}) \cdot \overrightarrow{Gd\imath r}_0 > 0 \textbf{ and } (\vec{v}_n - G_{Mn}) \cdot \overrightarrow{Gd\imath r}_n < 0$$

$\vec{v}_0$, $\vec{v}_n$: the location of an object in a world space recorded in *first* and *last* entry in *events batch*
G_{M0}, G_{Mn_n}: the midpoint location of a Group or Pair in a world space recorded in *first* and *last* entry in *events batch*
$\overrightarrow{Gd\imath r}_0, \overrightarrow{Gd\imath r}_n$: the walking direction of a Group or Pair recorded in *first* and *last* entry in *events batch*

Algorithm structure (in pseudo code)

```
if events batch exists
    if distance ≤ walking away distance threshold
        if events batch size ≥ sampling size
              if (dot product of first entry direction vector and first entry Group or Pair
view direction vector > 0) and (dot product of last entry direction vector and last entry
Group or Pair view direction vector < 0)
                  register a pattern
             clear events batch
        else
             select the closest static object
             if static object ID is equal to static object ID stored in last event in events
batch
                  record static object and Group or Pair data
                  add event to events batch
             else
                  clear events batch
```

Output Message: *" Group ID / Pair ID moves away from Static Object ID"*

Sources

Algorithm: Simulation.states / GroupsLoggerState.java
Event data bean: Simulation.beans / GroupEventBean.java

Table. A.2-18. "*Group / Pair moves alongside something*" behavioural pattern

Input	Primitive Operations	Parameters of the pattern
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location, dimensions, shape	- Group walking straight - Determining if a Static Object is by Group's left / right side	- ***Group data***: location (midpoint), average viewing direction, average walking direction - ***Static Object data***: ID, location, surface dimensions and shape - ***walking alongside ray's length limit***: used for delimiting ray's length and consequently defining a maximum distance at which collision may be detected - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***walking alongside interval***: how frequently events are being captured for this pattern

Description

Details of ray emission

$$r(x) = G_M + x(R_{side}\overrightarrow{Gd\imath r})$$

$r(x)$: point laying on the ray surface located at x distance from its *emission point*
G_M : the location of a *midpoint* of a Group defined in global coordinates
R_{side}: a rotation matrix declared in section 4.4.2.1., i.e. rotation about Y-axis 90°
$\overrightarrow{Gd\imath r}$: a normalized average viewing direction of a Group or Pair

Condition for registering event data for the pattern analysis

The events constituting a pattern of "Group / Pair walking alongside" are registered when a ray of a specified length intersects an object located to the left or right of the Group.

$$p_1 \cdot \vec{n}_1 + (-G_M \cdot \vec{n}_1) > 0$$

p_1 : point of intersection with an object geometry's triangle
$\vec{n}_1$: a normalized normal vector of additional triangle constructed during the operation
G_M : point from which the ray has been emitted, e.g. midpoint of a Group

Criterion for recognising a pattern from accumulated events

$$E_0 \cap E_n \neq \emptyset \textbf{ and } |E_0| = |E_n| \textbf{ and } E_0 = E_n$$

E_0 , E_n: the set of *qualitative* and *comparable* data recorded in *first* and *last* entry in *events batch*: side of a Group which is probed by ray and Static Object ID
$|E_0|$, $|E_n|$: the *cardinality* of *event data set* recorded in *first* and *last* entry in *events batch*
$E_0 = E_n \leftrightarrow \forall e[e \in E_0 \leftrightarrow e \in E_n]$

Algorithm structure (in pseudo code)

```
if events batch exists
    cast ray towards group side
        if ray intersects object within walking alongside ray's length limit
            if events batch size ≥ sampling size
                for each event in events batch
                    if side in event is the same as in previous event
                        continue
                    else
                        clear events batch
                        break

                if side is the same in all events in events batch
                    if static object ID is the same in first and last event entry in events
batch
                        register a pattern
                        clear events batch
            else
                select the closest intersected static object
                if static object ID is equal to static object ID stored in last event in
events batch
                    record static object and Group or Pair data
                    add event to events batch
                else
                    clear events batch
```

Output Message: *"Group ID / Pair ID moves alongside Static Object ID"*

Sources

Algorithm: Simulation.states / GroupsLoggerState.java
Event data bean: Simulation.beans / GroupRayEventBean.java

Table. A.2-19. "*Group / Pair climbs something***" behavioural pattern**

Input	**Primitive Operations**	**Parameters of the pattern**
- ***Individual***: ID, location, view direction, walking direction - ***Static Object***: ID, location, dimensions, shape	- Group walking straight - Determining if Group or Pair is above or below Static Object	- ***Group data***: location (midpoint), average viewing direction, average walking direction - ***Static Object data***: ID, location, surface dimensions and shape - ***climbing ray's length limit***: used for delimiting ray's length and consequently defining a maximum distance at which collision may be detected - ***sampling size***: the number of events that needs to be collected for pattern evaluation - ***climbing interval***: how frequently events are being captured for this pattern
Description		

Details of ray emission

$$r(x) = G_M + x\vec{v}$$

$r(x)$: point laying on the ray surface located at x distance from its *emission point*
G_M : the location of a *midpoint* of a Group defined in global coordinates
$\vec{v}$: a direction vector [0,1,0] for "up" and [0,-1,0] for "down"

Condition for registering event data for the pattern analysis

The events constituting a pattern of "Group / Pair climbing something" are registered when a ray of a specified length intersects an object located above or below a Group.

$$p_1 \cdot \vec{n}_1 + (-G_M \cdot \vec{n}_1) > 0$$

p_1 : point of intersection with an object geometry's triangle
$\vec{n}_1$: a normalized normal vector of additional triangle constructed during the operation
G_M : point from which the ray has been emitted, e.g. midpoint of a Group

Criterion for recognising a pattern from accumulated events

Determination of whether a Group / Pair climbs something up or down is dependent on the values of y-component of Group's midpoint location vector recorded in *first* and *last* entry in *events batch*.

Climbing up:

$$E_0 \cap E_n \neq \emptyset \textbf{ and } |E_0| = |E_n| \textbf{ and } E_0 = E_n \textbf{ and } G_{M_{0_y}} < G_{M_{n_y}}$$

Climbing down:

$$E_0 \cap E_n \neq \emptyset \textbf{ and } |E_0| = |E_n| \textbf{ and } E_0 = E_n \textbf{ and } G_{M_{0_y}} > G_{M_{n_y}}$$

E_0 , E_n: the set of *qualitative* and *comparable* data recorded in *first* and *last* entry in *events batch*: side of an agent which is probed by ray and static object ID
$|E_0|$, $|E_n|$: the *cardinality* of *event data set* recorded in *first* and *last* entry in *events batch*
$E_0 = E_n \leftrightarrow \forall e[e \in E_0 \leftrightarrow e \in E_n]$
G_{M_0}, G_{M_n}: the location of a *midpoint* of a Group in a world space recorded in *first* and *last* entry in *events batch*

Algorithm structure (in pseudo code)

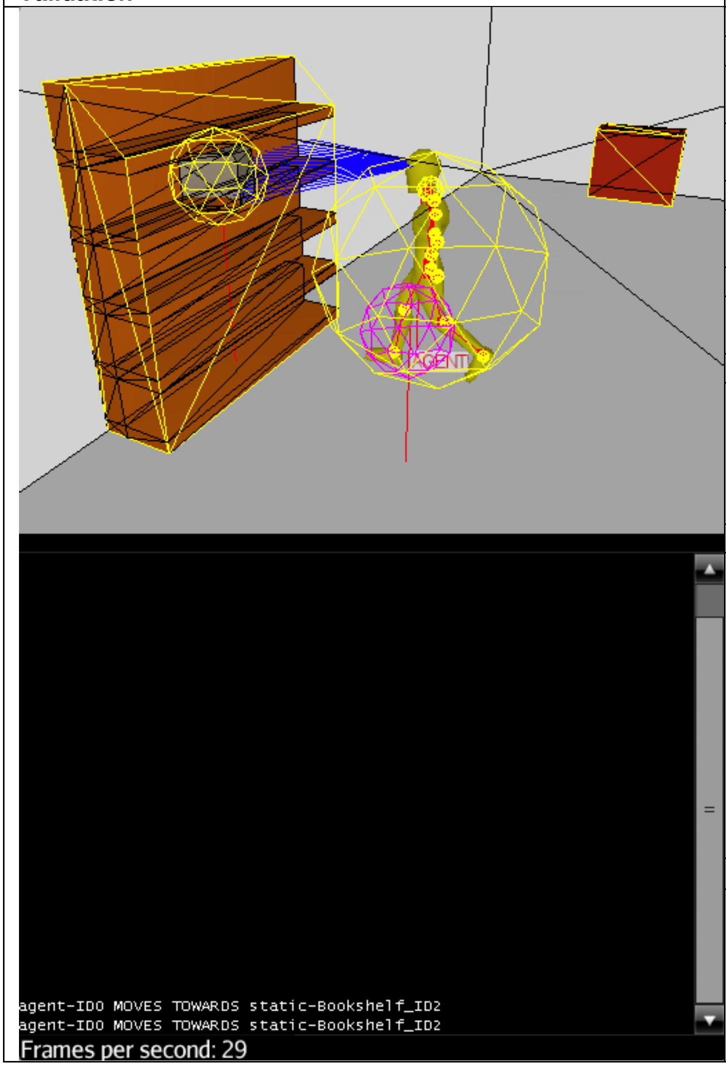
```
if events batch exists
    cast ray towards group side
        if ray intersects object within climbing ray's length limit
            if events batch size ≥ sampling size
                for each event in events batch
                    if side in event is the same as in previous event
                        continue
                    else
                        clear events batch
                        break

                if side is the same in all events in events batch
                    if Y-component of Group or Pair midpoint location in first event entry <
Y-component of group midpoint location in last event entry
                        register climbing up pattern
                    else if Y-component of Group or Pair midpoint location in first event
entry > Y-component of group midpoint location in last event entry
                        register climbing down pattern
                    clear events batch
            else
                select the closest intersected static object
                if static object ID is equal to static object ID stored in last event in
events batch
```

`record` *static object* and *Group or Pair data* `add` *event* to *events batch* `else` `clear` *events batch*
Output Message: *"Group ID / Pair ID climbs up/down Static Object ID"*
Sources
Algorithm: Simulation.states / GroupsLoggerState.java *Event data bean*: Simulation.beans / GroupRayEventBean.java

A.3. Experimental validation of the dynamic patterns recognition

The validation of behavioural patterns is made through comparing the expected outputs against their visual appearance in the console. When the message generated in the event console finally matches the expected observed behaviour in the 3D scene, it passes the overall evaluation. The default values of the parameters were selected after empirical optimisation (see Appendix A.4). All of them are estimated at standard 30 FPS frame rate of the video signal.

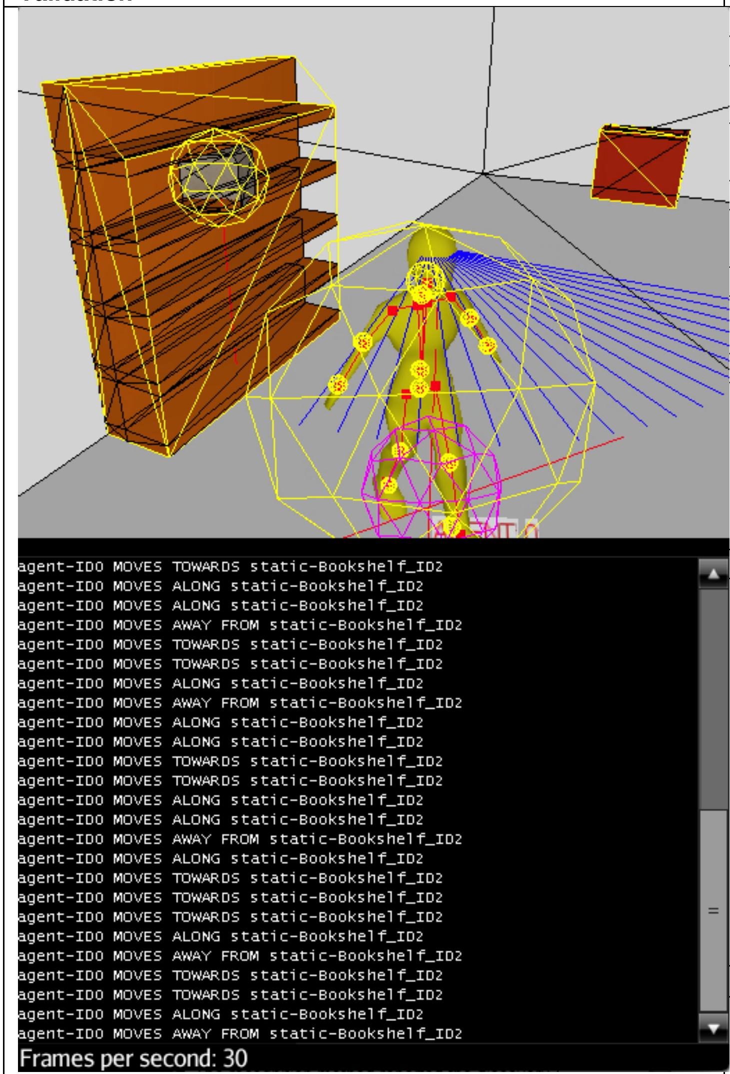
Table A.3-1. "*Somebody is walking towards something*" pattern validation

<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="14">
</td><td>No. Agents</td><td>1</td></tr>
<tr><td>Agent ID</td><td>agent-ID0</td></tr>
<tr><td>Location</td><td>[161.19113, 4.3999987, -0.8794838]</td></tr>
<tr><td>Viewing Direction</td><td>[-0.87773572, 0, 0.48221964]</td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2">agent-ID0 MOVES TOWARDS static-Bookshelf_ID2</td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2">agent-ID0 MOVES TOWARDS static-Bookshelf_ID2</td></tr>
<tr><td colspan="2">Observed behaviour</td></tr>
<tr><td colspan="2">- agent-ID0 walks towards a static-Bookshelf_ID2
- the flow of messages for this pattern has ceased when agent-ID0 moved away from Bookshelf_ID2</td></tr>
<tr><td colspan="2">Result analysis</td></tr>
<tr><td colspan="2">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was continuously walking towards static-Bookshelf_ID2 for 1 second according to time stamps recorded in the log file:

14/03/2017 10:08:50 :: 1st registration of pattern
14/03/2017 10:08:51 :: 2nd registration of pattern</td></tr>
<tr><td colspan="2">Validation status</td></tr>
<tr><td colspan="2">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-2. "*Somebody is walking away from something*" pattern validation

Validation



Initial data

No. Agents	`1`
Agent ID	`agent-ID0`
Location	`[161.19113, 4.3999987, -0.8794838]`
Viewing Direction	`[-0.87773572, 0, 0.48221964]`

Expected output

`agent-ID0 MOVES AWAY FROM static-Bookshelf_ID2`

Output

`agent-ID0 MOVES AWAY FROM static-Bookshelf_ID2`

Observed behaviour

- agent-ID0 walks away from a static-Bookshelf_ID2
- the message appeared when agent-ID0 was moving towards and along static-Bookshelf_ID2 and then moved away from it

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was continuously walking towards static-Bookshelf_ID2 for 1 second, alongside it for 1 second and moved away from it according to time stamps recorded in the log file:

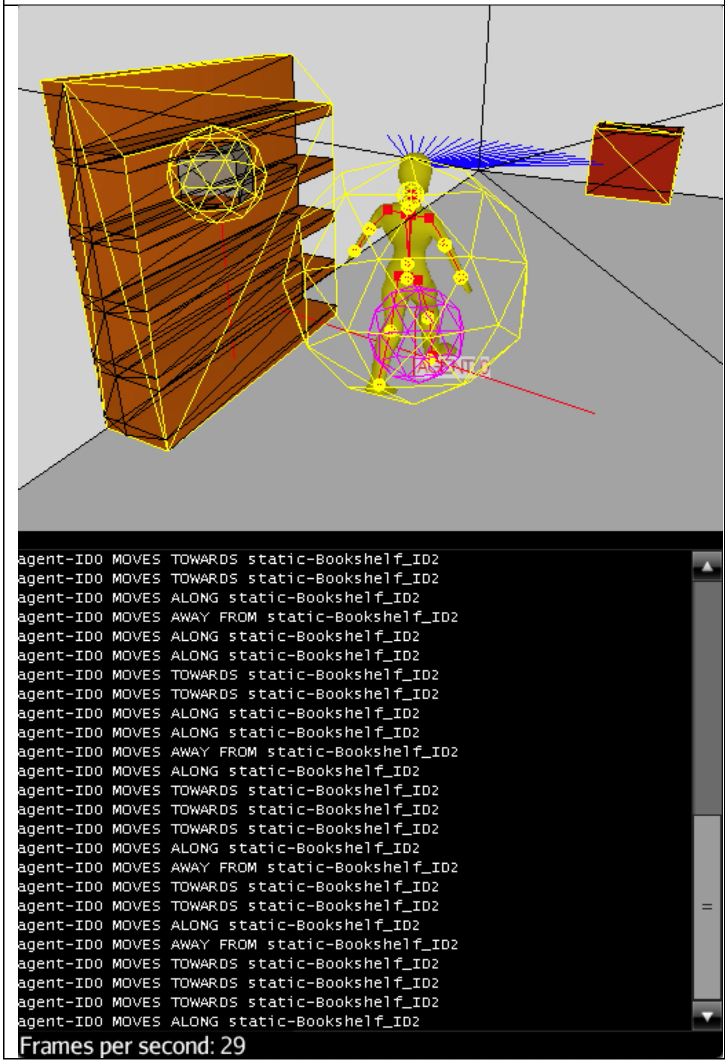
14/03/2017 10:08:51 :: moved towards
14/03/2017 10:08:52 :: moved towards
14/03/2017 10:08:53 :: moved alongside
14/03/2017 10:08:53 :: 1st registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-3. "*Somebody is walking alongside something"* pattern validation

Validation

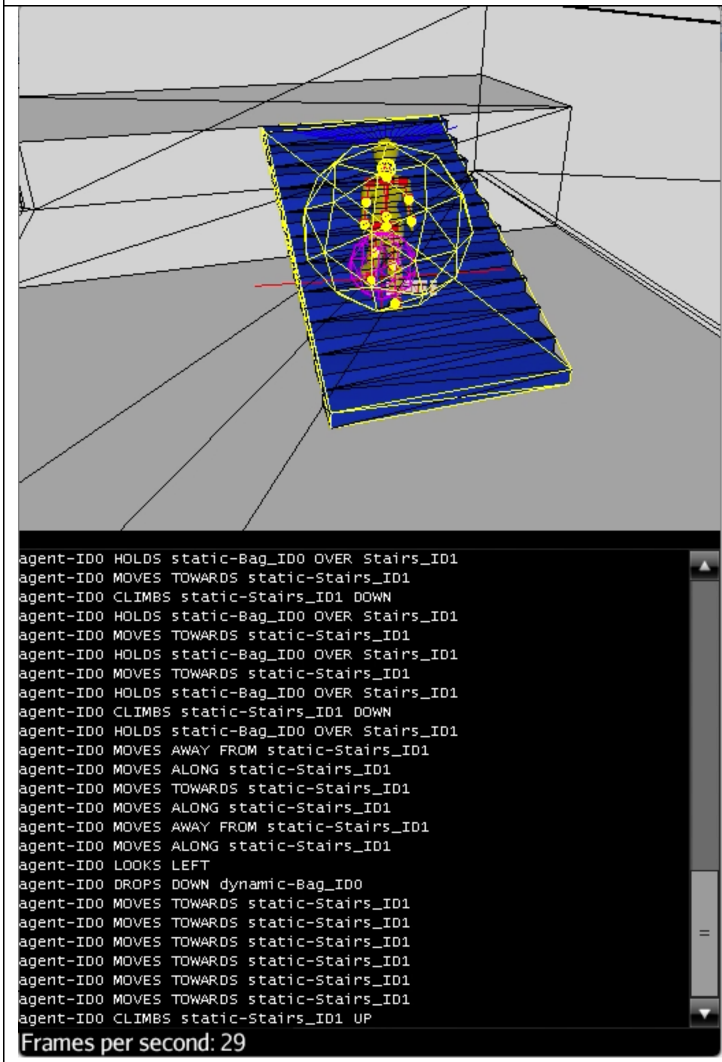


Initial data	
No. Agents	`1`
Agent ID	`agent-ID0`
Location	`[161.19113, 4.3999987, -0.8794838]`
Viewing Direction	`[-0.87773572, 0, 0.48221964]`

Expected output
`agent-ID0 MOVES ALONG static-Bookshelf_ID2`
Output
`agent-ID0 MOVES ALONG static-Bookshelf_ID2`
Observed behaviour
- agent-ID0 walks alongside static-Bookshelf_ID2 - the message appeared when agent-ID0 was moving towards and along static-Bookshelf_ID2 and then moved away from it
Result analysis
The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was continuously walking alongside static-Bookshelf_ID2 for 1 second according to time stamps recorded in the log file: 14/03/2017 10:09:11 :: 1st registration of the pattern 14/03/2017 10:09:12 :: 2nd registration of the pattern
Validation status
The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-4. "*Somebody climbs something up"* pattern validation

Validation



Initial data

No. Agents	1
Agent ID	agent-ID0
Location	[161.19113, 4.3999987, -0.8794838]
Viewing Direction	[-0.87773572, 0, 0.48221964]

Expected output

agent-ID0 CLIMBS static-Stairs_ID1 UP

Output

agent-ID0 CLIMBS static-Stairs_ID1 UP

Observed behaviour

- agent-ID0 is on the stairs (static-Stairs_ID1) and climbs them up
- the message appeared when agent-ID0 was above static-Stairs_ID1 and climbing them upwards to reach upper level

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was continuously climbing static-Stairs_ID1 upwards for 2 seconds according to time stamps recorded in the log file:
15/03/2017 10:48:41 :: 1st registration of the pattern
15/03/2017 10:48:43 :: 2nd registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-5. "*Somebody climbs something down"* pattern validation

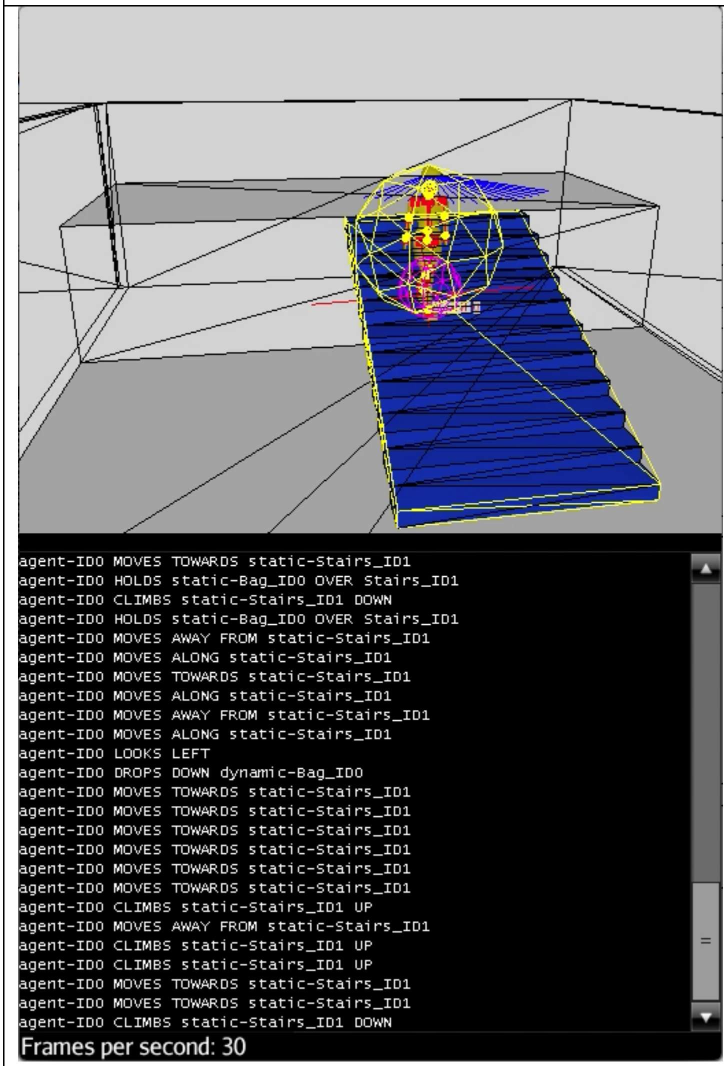
<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="14">
</td><td>No. Agents</td><td><code>1</code></td></tr>
<tr><td>Agent ID</td><td><code>agent-ID0</code></td></tr>
<tr><td>Location</td><td><code>[161.19113, 4.3999987, -0.8794838]</code></td></tr>
<tr><td>Viewing Direction</td><td><code>[-0.87773572, 0, 0.48221964]</code></td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2"><code>agent-ID0 CLIMBS static-Stairs_ID1 DOWN</code></td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2"><code>agent-ID0 CLIMBS static-Stairs_ID1 DOWN</code></td></tr>
<tr><td colspan="2">Observed behaviour</td></tr>
<tr><td colspan="2">- agent-ID0 is on the stairs (static-Stairs_ID1) and climbs them down
- the message appeared when agent-ID0 was above static-Stairs_ID1 and climbing them downwards to reach lower level</td></tr>
<tr><td colspan="2">Result analysis</td></tr>
<tr><td colspan="2">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was continuously climbing static-Stairs_ID1 downwards for 3 seconds according to time stamps recorded in the log file:
15/03/2017 10:48:59 :: 1st registration of the pattern
15/03/2017 10:49:02 :: 2nd registration of the pattern</td></tr>
<tr><td colspan="2">Validation status</td></tr>
<tr><td colspan="2">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-6. “*Somebody holds something over something*” pattern validation

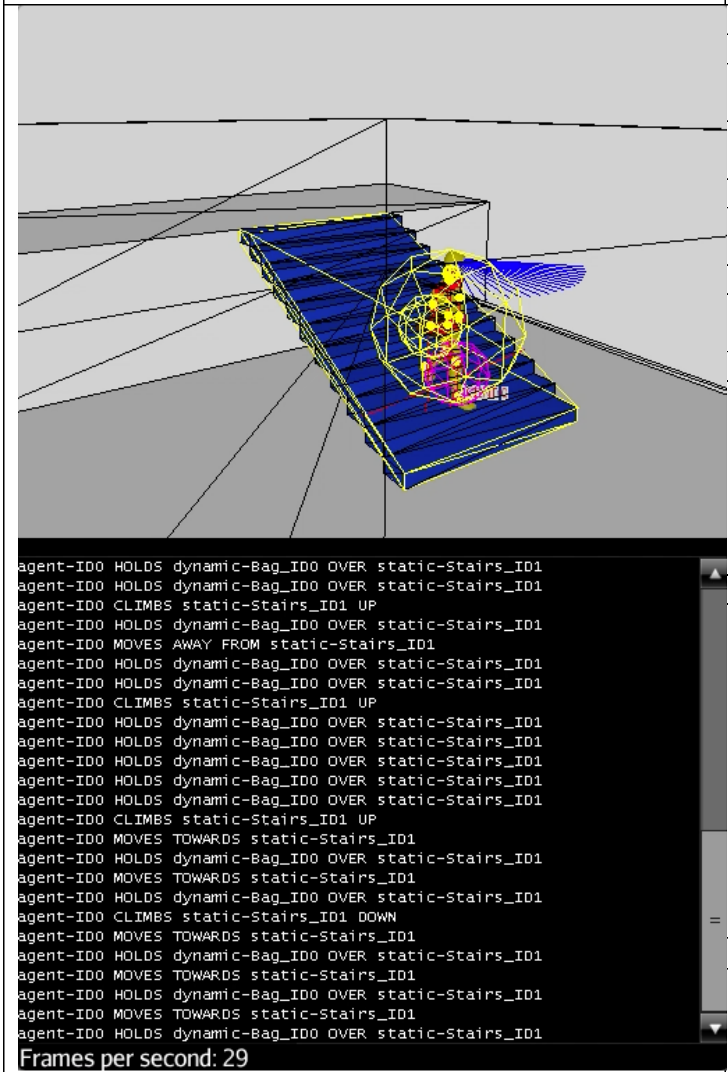
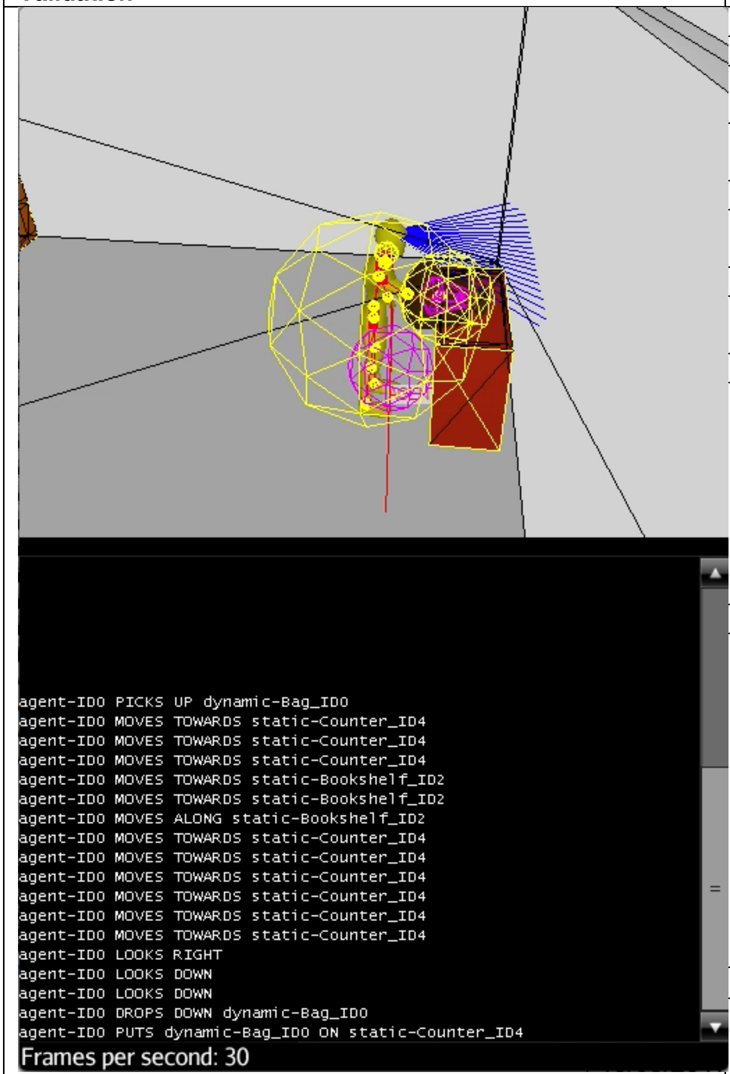
<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="10">
</td><td>No. Agents</td><td><code>1</code></td></tr>
<tr><td>Agent ID</td><td><code>agent-ID0</code></td></tr>
<tr><td>Location</td><td><code>[161.19113, 4.3999987, -0.8794838]</code></td></tr>
<tr><td>Viewing Direction</td><td><code>[-0.87773572, 0, 0.48221964]</code></td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2"><code>agent-ID0 HOLDS dynamic-Bag_ID0 OVER static-Stairs_ID1</code></td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2"><code>agent-ID0 HOLDS dynamic-Bag_ID0 OVER static-Stairs_ID1</code></td></tr>
<tr><td colspan="2">Observed behaviour</td></tr>
<tr><td colspan="2">- agent-ID0 is on the stairs (static-Stairs_ID1) while holding a bag (dynamic-Bag_ID0) in right hand
- the message appeared when agent-ID0 was holding dynamic-Bag_ID0 in right hand while standing on static-Stairs_ID1</td></tr>
<tr><td></td><td colspan="2">Result analysis</td></tr>
<tr><td></td><td colspan="2">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was holding a dynamic-Bag_ID0 in right hand while standing on (being above) static-Stairs_ID1 for 1 second according to time stamps recorded in the log file:
15/03/2017 12:08:03 :: 1st registration of the pattern
15/03/2017 12:08:04 :: 2nd registration of the pattern</td></tr>
<tr><td></td><td colspan="2">Validation status</td></tr>
<tr><td></td><td colspan="2">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-7. "*Somebody puts something on something*" pattern validation

Validation



Initial data

No. Agents	1
Agent ID	agent-ID0
Location	[161.19113, 4.3999987, -0.8794838]
Viewing Direction	[-0.87773572, 0, 0.48221964]

Expected output

agent-ID0 PUTS dynamic-Bag_ID0 ON static-Counter_ID4

Output

agent-ID0 PUTS dynamic-Bag_ID0 ON static-Counter_ID4

Observed behaviour

- agent-ID0 puts down a bag (dynamic-Bag_ID0) at the counter (static-Counter_ID4)
- the message appeared when agent-ID0 was holding dynamic-Bag_ID0 in right hand, hold it over a counter (static-Counter_ID4) and then put it on top of it

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message in the console appeared after agent-ID0 was holding a dynamic-Bag_ID0 in right hand, holding it over a counter (static-Counter_ID4) and put it on top of it. The message displayed only at the time when agent-ID0 dropped down the item on top of the counter.

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-8. "*Somebody picks up something*" pattern validation

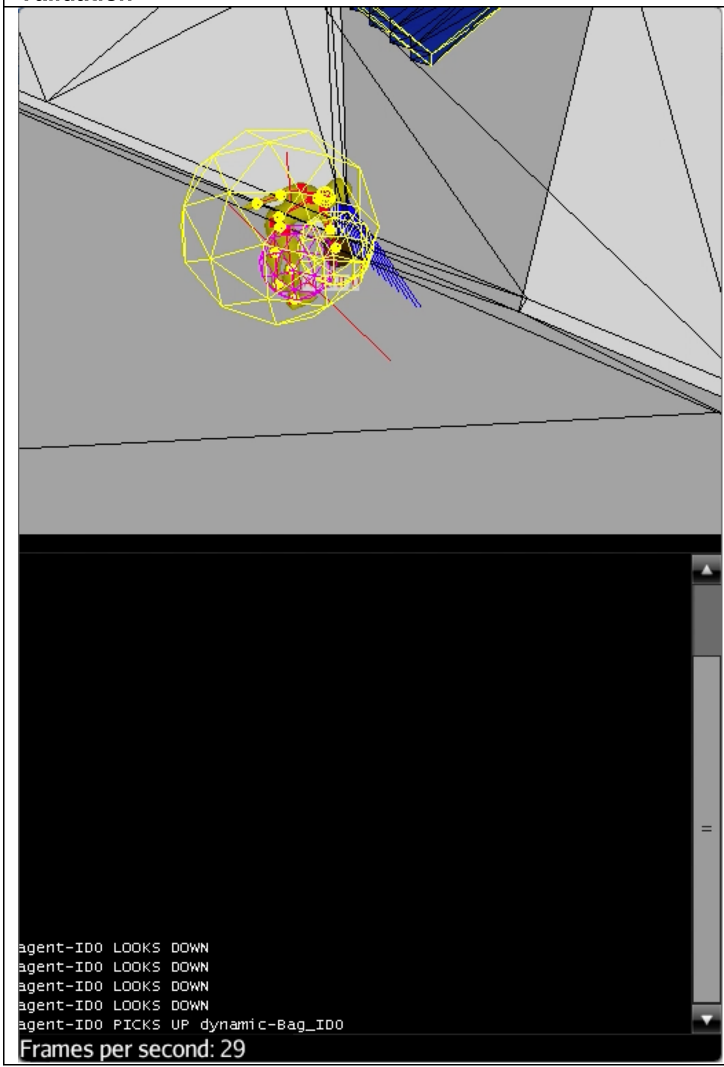
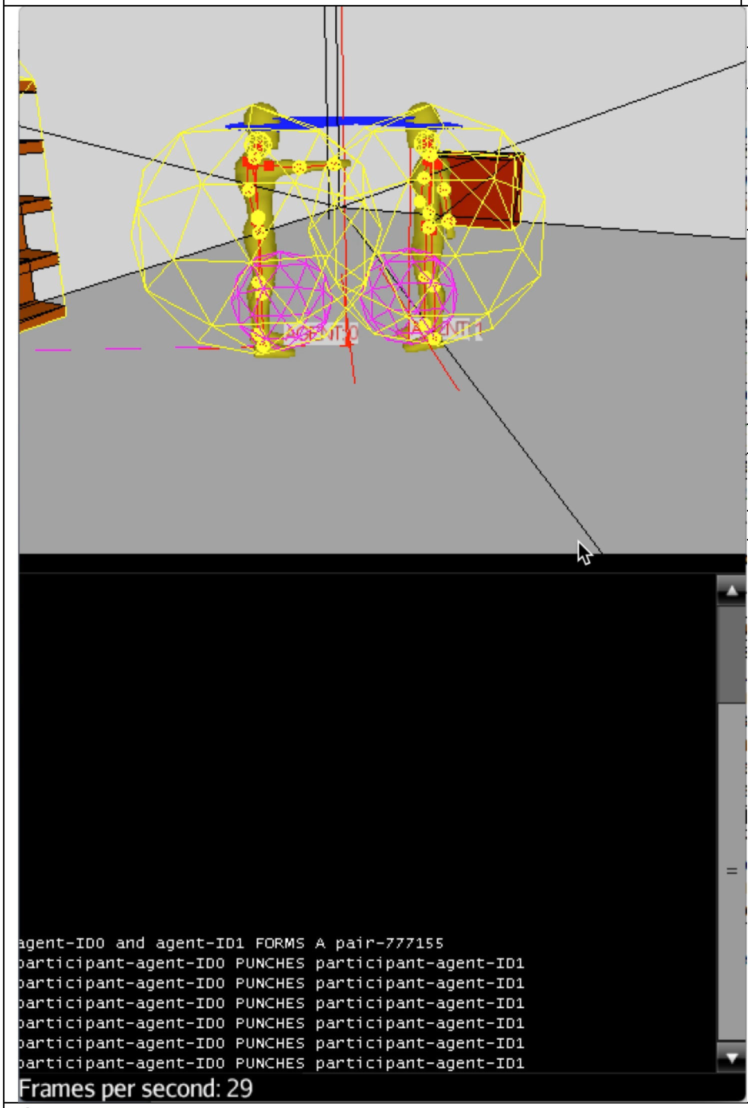
<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="14">
</td><td>No. Agents</td><td><code>1</code></td></tr>
<tr><td>Agent ID</td><td><code>agent-ID0</code></td></tr>
<tr><td>Location</td><td><code>[161.19113, 4.3999987, -0.8794838]</code></td></tr>
<tr><td>Viewing Direction</td><td><code>[-0.87773572, 0, 0.48221964]</code></td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2"><code>agent-ID0 PICKS UP dynamic-Bag_ID0</code></td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2"><code>agent-ID0 PICKS UP dynamic-Bag_ID0</code></td></tr>
<tr><td colspan="2">Observed behaviour</td></tr>
<tr><td colspan="2">- agent-ID0 picks up a bag (dynamic-Bag_ID0) from the floor
- the message appeared when agent-ID0 bent over a bag (dynamic-Bag_ID0) and picked it up from the floor.</td></tr>
<tr><td colspan="2">Result analysis</td></tr>
<tr><td colspan="2">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console right hand of agent-ID0 came into contact with a bag (dynamic-Bag_ID0) lying on the floor.</td></tr>
<tr><td colspan="2">Validation status</td></tr>
<tr><td colspan="2">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-9. "*Somebody punches somebody*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	`agent-ID0`, `agent-ID1`
Location	**agent-ID0:** [161.19113, 4.3999987, -0.8794838] **agent-ID1:** [121.19113, 4.3999987, 21.8794838]
Viewing Direction	**agent-ID0:** [-0.87773572, 0, 0.48221964] **agent-ID1:** [-0.87773572, 0, 0.48221964]

Expected output

`participant-agent-ID0 PUNCHES participant-agent-ID1`

Output

`participant-agent-ID0 PUNCHES participant-agent-ID1`

Observed behaviour

- agent-ID0 punches agent-ID1
- several messages appeared when agent-ID0 has reached out his right hand in the direction of agent-ID1

Result analysis

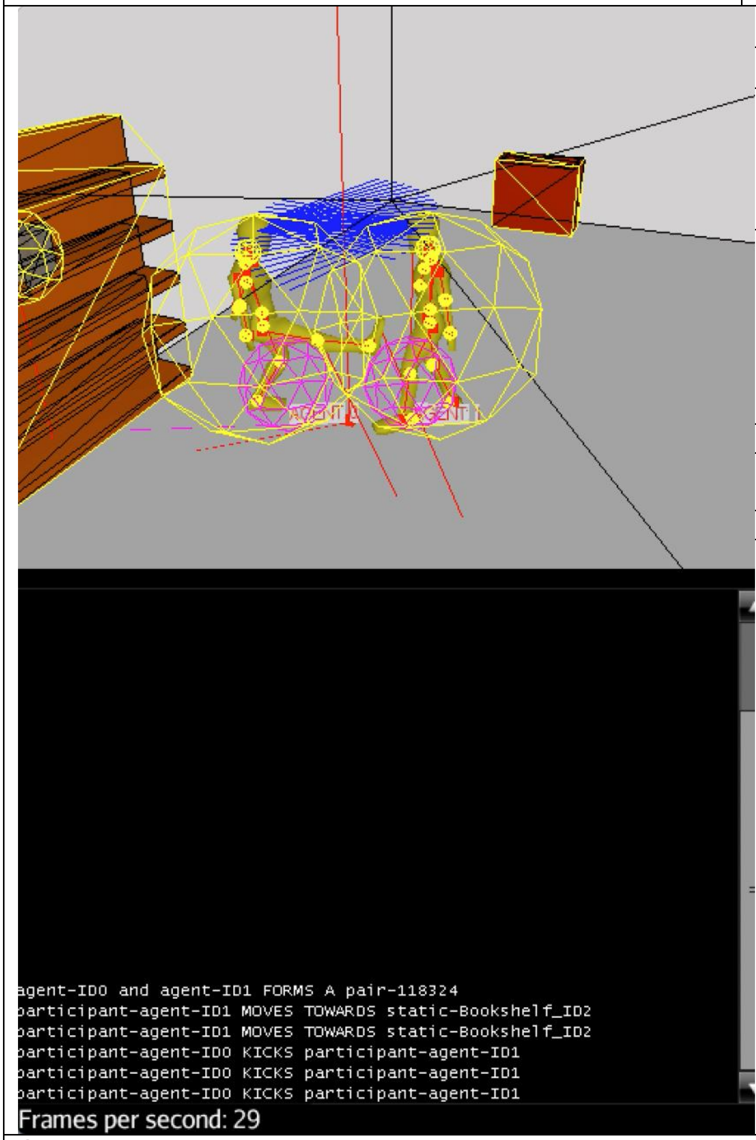
The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. Several messages appeared in the console when right hand of agent-ID0 came into contact with agent-ID1 body. No messages were generated when right hand was simply touching agent-ID1 body. No considerable time gaps between messages were found in the log:
15/03/2017 13:34:00 :: 1st registration of the message
15/03/2017 13:34:00 :: 2nd registration of the message

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-10. "*Somebody kicks somebody*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	`agent-ID0`, `agent-ID1`
Location	**`agent-ID0:`** `[161.19113, 4.3999987, -0.8794838]` **`agent-ID1:`** `[121.19113, 4.3999987, 21.8794838]`
Viewing Direction	**`agent-ID0:`** `[-0.87773572, 0, 0.48221964]` **`agent-ID1:`** `[-0.87773572, 0, 0.48221964]`

Expected output

`participant-agent-ID0 KICKS participant-agent-ID1`

Output

`participant-agent-ID0 KICKS participant-agent-ID1`

Observed behaviour

- agent-ID0 kicks agent-ID1
- several messages appeared when agent-ID0 has reached out his right leg in the direction of agent-ID1

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. Several messages appeared in the console when right leg of agent-ID0 came into contact with agent-ID1 body. No messages were generated when right leg was simply touching agent-ID1 body. No considerable time gaps between messages were found in the log:

15/03/2017 13:54:04 :: 1st registration of the message
15/03/2017 13:54:04 :: 2nd registration of the message

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-11. "*Somebody drops down something*" pattern validation

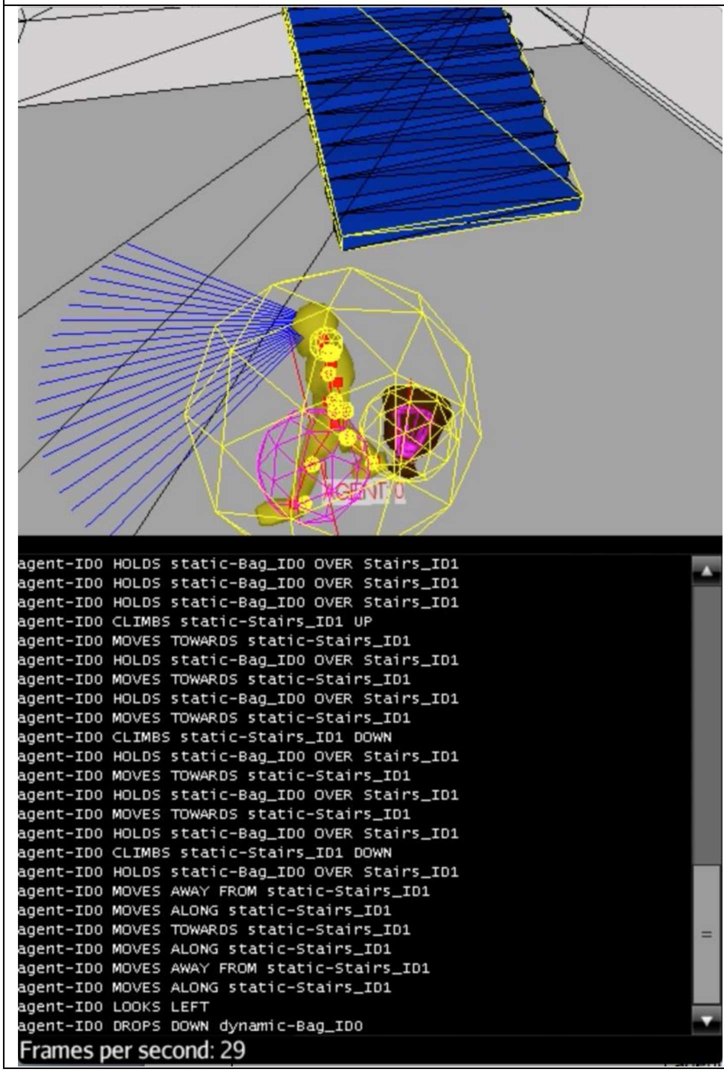
<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="14">
</td><td>No. Agents</td><td><code>1</code></td></tr>
<tr><td>Agent ID</td><td><code>agent-ID0</code></td></tr>
<tr><td>Location</td><td><code>[161.19113, 4.3999987, -0.8794838]</code></td></tr>
<tr><td>Viewing Direction</td><td><code>[-0.87773572, 0, 0.48221964]</code></td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2"><code>agent-ID0 DROPS DOWN dynamic-Bag_ID0</code></td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2"><code>agent-ID0 DROPS DOWN dynamic-Bag_ID0</code></td></tr>
<tr><td colspan="2">Observed behaviour</td></tr>
<tr><td colspan="2">- agent-ID0 drops down a bag (dynamic-Bag_ID0) on the floor
- the message appeared when agent-ID0 dropped a bag (dynamic-Bag_ID0) on the floor.</td></tr>
<tr><td colspan="2">Result analysis</td></tr>
<tr><td colspan="2">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console when a distance between right hand and a bag carried by agent-ID0 was clearly visible.</td></tr>
<tr><td colspan="2">Validation status</td></tr>
<tr><td colspan="2">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-12. "*Somebody passes something over to somebody"* pattern validation

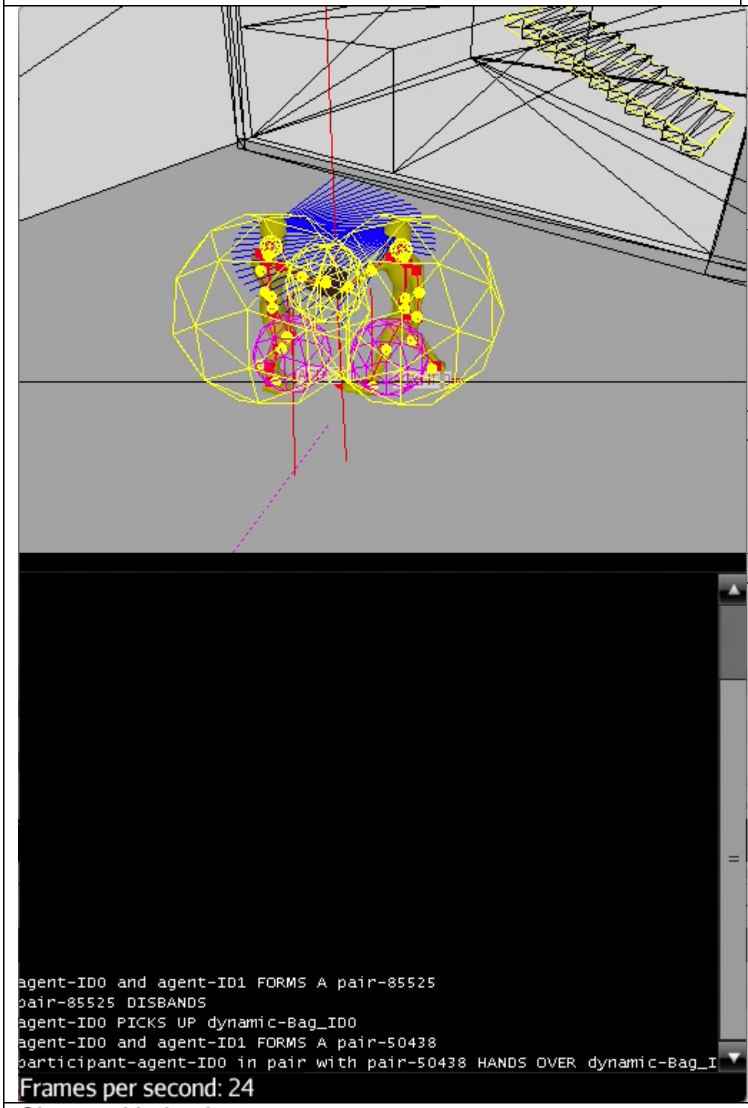
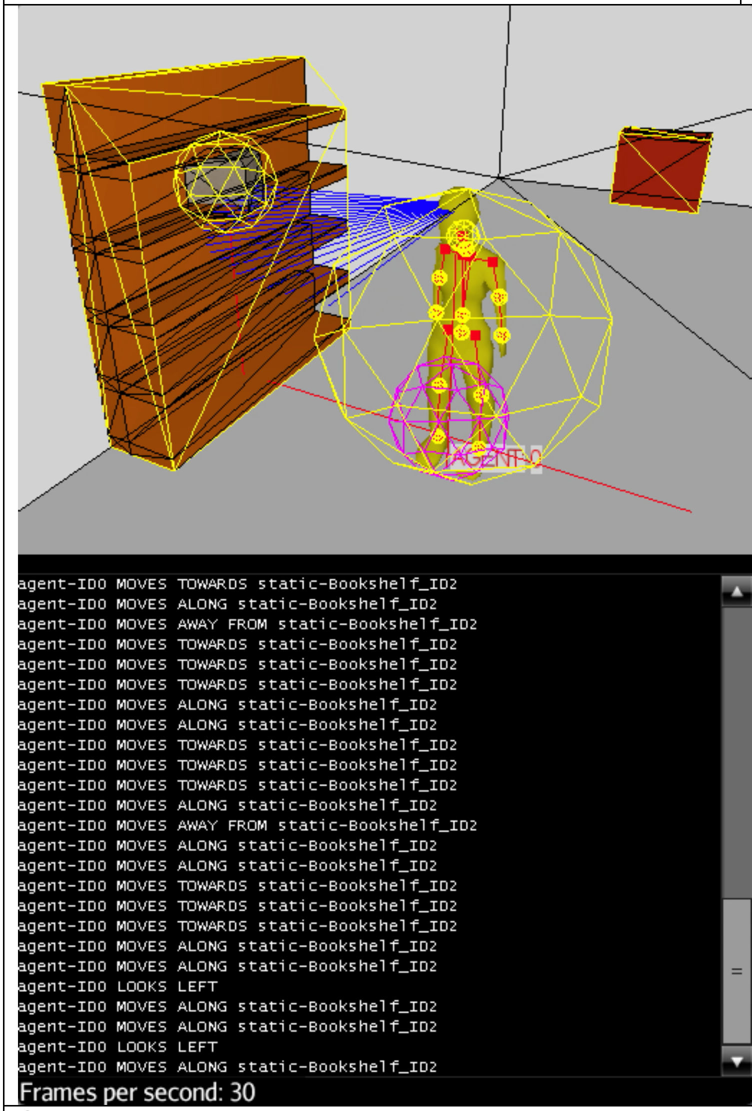
<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="9">
</td><td>No. Agents</td><td><code>2</code></td></tr>
<tr><td>Agent ID</td><td><code>agent-ID0</code>, <code>agent-ID1</code></td></tr>
<tr><td>Location</td><td><code>agent-ID0:</code> <code>[161.19113, 4.3999987, -0.8794838]</code>
<code>agent-ID1:</code>
<code>[121.19113, 4.3999987, 21.8794838]</code></td></tr>
<tr><td>Viewing Direction</td><td><code>agent-ID0:</code>
<code>[-0.87773572, 0, 0.48221964]</code>
<code>agent-ID1:</code> <code>[-0.87773572, 0, 0.48221964]</code></td></tr>
<tr><th colspan="2">Expected output</th></tr>
<tr><td colspan="2"><code>participant-agent-ID0 in pair with pair-50438 HANDS OVER dynamic-Bag_ID0 TO participant-agent-ID1 in pair with pair-50438</code></td></tr>
<tr><th colspan="2">Output</th></tr>
<tr><td colspan="2"><code>participant-agent-ID0 in pair with pair-50438 HANDS OVER dynamic-Bag_ID0 TO participant-agent-ID1 in pair with pair-50438</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><th colspan="3">Observed behaviour</th></tr>
<tr><td colspan="3">- agent-ID0 passed bag (dynamic-Bag_ID0) to agent-ID1
- a message appeared when agent-ID0 handed over a bag to agent-ID1</td></tr>
<tr><th colspan="3">Result analysis</th></tr>
<tr><td colspan="3">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console when a close distance between agent-ID0 right hand, a bag and agent-ID1 right hand was registered, i.e. when a bag came into contact with both hands of agents.</td></tr>
<tr><th colspan="3">Validation status</th></tr>
<tr><td colspan="3">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-13. "*Somebody looks left / right / up / down*" pattern validation

Validation	**Initial data**	
	*No. Agents*	`1`
	Agent ID	`agent-ID0`
	Location	`[161.19113, 4.3999987, -0.8794838]`
	Viewing Direction	`[-0.87773572, 0, 0.48221964]`
	Expected output	
	`agent-ID0 LOOKS DOWN` `agent-ID0 LOOKS UP` `agent-ID0 LOOKS LEFT` `agent-ID0 LOOKS RIGHT`	
	Output	
	`agent-ID0 LOOKS DOWN` `agent-ID0 LOOKS UP` `agent-ID0 LOOKS LEFT` `agent-ID0 LOOKS RIGHT`	

Observed behaviour

- agent-ID0 looked left / right / up / down
- a message appeared when agent-ID0 turned its head left / right / up / down

Result analysis

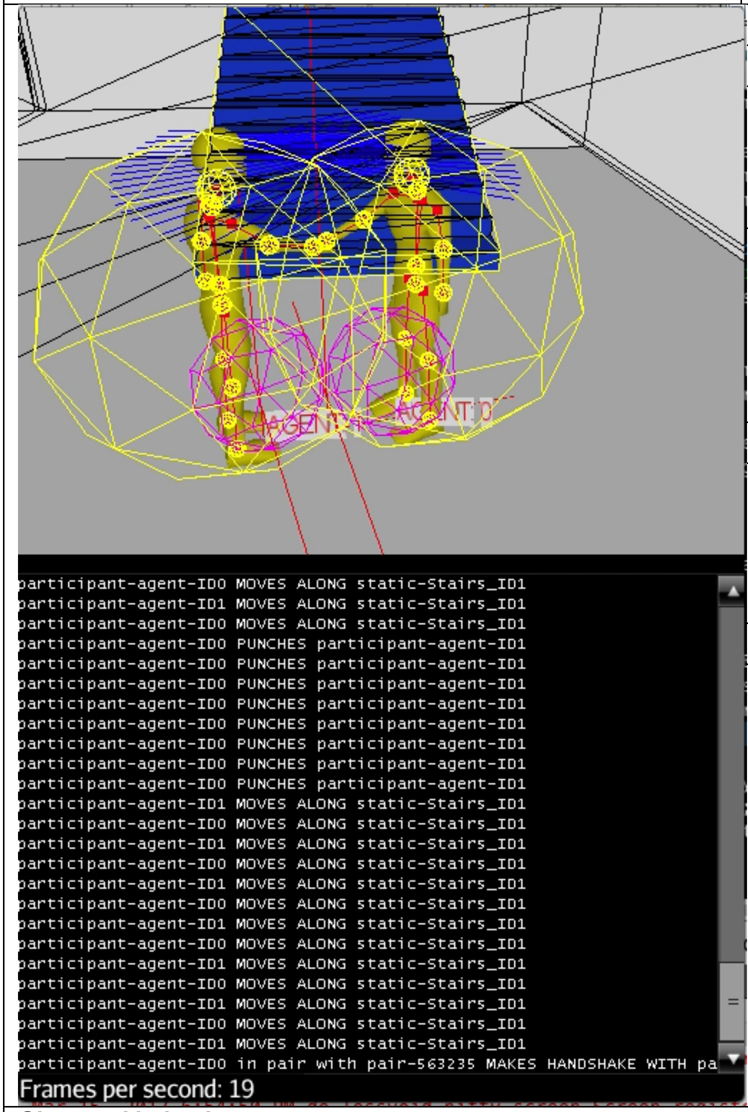
The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message that appeared in the console corresponds directly to the direction towards which agent-ID0 turned its head. The validation was made for all four directions, i.e. left, right, up, down. The pattern was discovered after 2 seconds according to time stamps recorded in the log file:

15/03/2017 10:47:12 :: agent starts turning its head left
15/03/2017 10:47:14 :: 1st registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

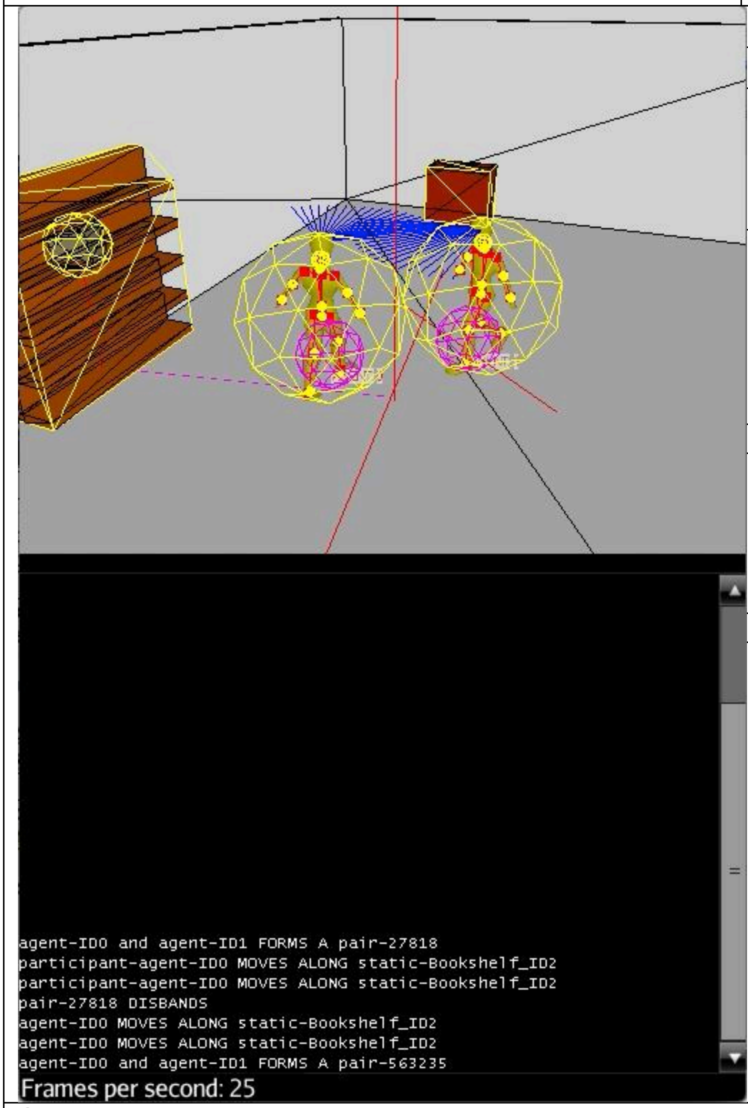
Table A.3-14. "*Somebody makes handshake with somebody*" pattern validation

<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="7">
</td><td>No. Agents</td><td><code>2</code></td></tr>
<tr><td>Agent ID</td><td><code>agent-ID0</code>, <code>agent-ID1</code></td></tr>
<tr><td>Location</td><td><code>agent-ID0:</code> <code>[161.19113, 4.3999987, -0.8794838]</code>
<code>agent-ID1:</code> <code>[121.19113, 4.3999987, 21.8794838]</code></td></tr>
<tr><td>Viewing Direction</td><td><code>agent-ID0:</code> <code>[-0.87773572, 0, 0.48221964]</code>
<code>agent-ID1:</code> <code>[-0.87773572, 0, 0.48221964]</code></td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2"><code>participant-agent-ID0 in pair with pair-<ID> MAKES HANDSHAKE WITH participant-agent-ID1 in pair with pair-<ID></code></td></tr>
<tr><td colspan="2">Output
<code>participant-agent-ID0 in pair with pair-563235 MAKES HANDSHAKE WITH participant-agent-ID1 in pair with pair-563235</code></td></tr>
<tr><td colspan="3">Observed behaviour</td></tr>
<tr><td colspan="3">- agent-ID0 and agent-ID1 made a handshake
- a message appeared when right hand of agent-ID0 and right hand of agent-ID1 came into contact and hold together for a brief moment</td></tr>
<tr><td colspan="3">Result analysis</td></tr>
<tr><td colspan="3">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console when right hands of both agents were hold together for a brief moment. The pattern was recognised after 4 seconds that elapsed from raising hands and holding them for a brief moment according to time stamps registered in the log file:

<code>15/03/2017 18:57:26</code> :: agents start raising their hands towards each other
<code>15/03/2017 18:57:30</code> :: 1st recognition of the pattern</td></tr>
<tr><td colspan="3">Validation status</td></tr>
<tr><td colspan="3">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-15. "*Somebody and somebody forms a pair*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	`agent-ID0`, `agent-ID1`
Location	**`agent-ID0:`** `[161.19113, 4.3999987, -0.8794838]` **`agent-ID1:`** `[121.19113, 4.3999987, 21.8794838]`
Viewing Direction	**`agent-ID0:`** `[-0.87773572, 0, 0.48221964]` **`agent-ID1:`** `[-0.87773572, 0, 0.48221964]`

Expected output

`agent-ID0 and agent-ID1 FORMS A pair-<ID>`

Output

`agent-ID0 and agent-ID1 FORMS A pair-170942`

Observed behaviour

- agent-ID1 approached agent-ID0 and found itself within its close proximity
- a message appeared when agent-ID1 approached agent-ID0

Result analysis

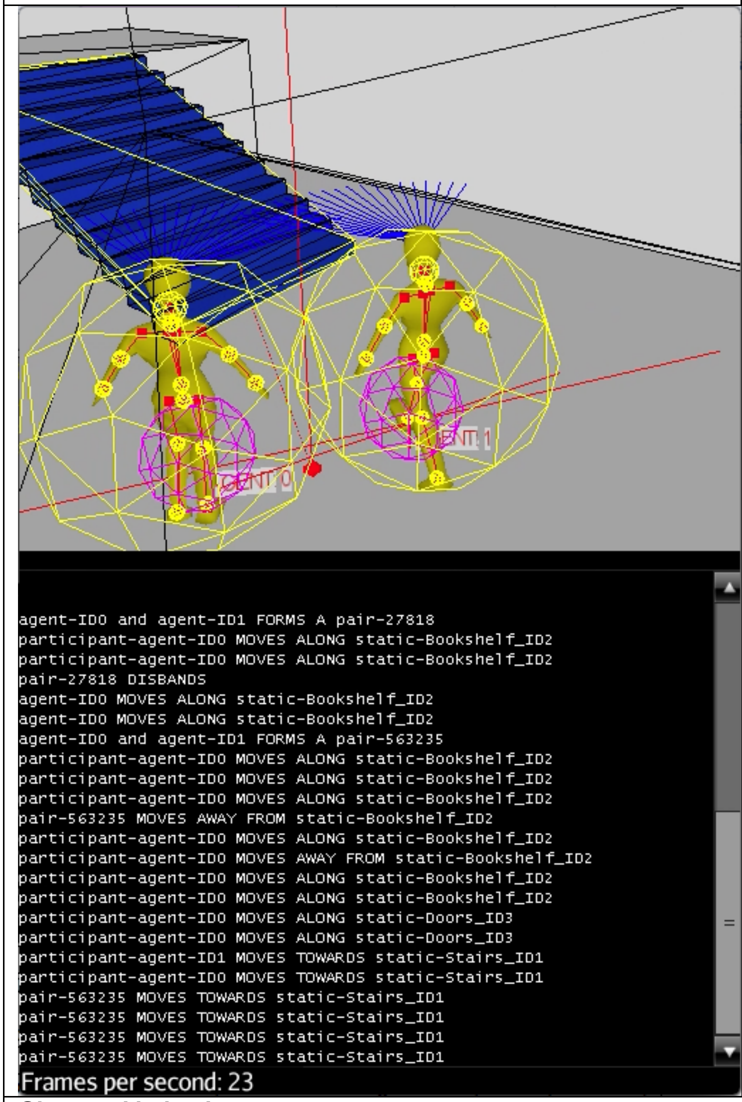
The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console when agent-ID1 approached agent-ID0 and found itself within its close proximity.

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-16. "*Group / Pair moves towards something*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	agent-ID0, agent-ID1
Location	**agent-ID0:** [161.19113, 4.3999987, -0.8794838] **agent-ID1:** [121.19113, 4.3999987, 21.8794838]
Viewing Direction	**agent-ID0:** [-0.87773572, 0, 0.48221964] **agent-ID1:** [-0.87773572, 0, 0.48221964]

Expected output

Pair-<ID> MOVES TOWARDS static-Stairs_ID1

Output

Pair-563235 MOVES TOWARDS static-Stairs_ID1

Observed behaviour

- both agents (agent-ID0 and agent-ID1) who formed a pair have been continuously walking towards stairs
- a message appeared when both agents continuously been walking towards stairs

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console after pair-563235 and all its members (agent-ID0 and agent-ID1) was continuously walking towards static-Stairs_ID1 according to time stamps recorded in the log file:

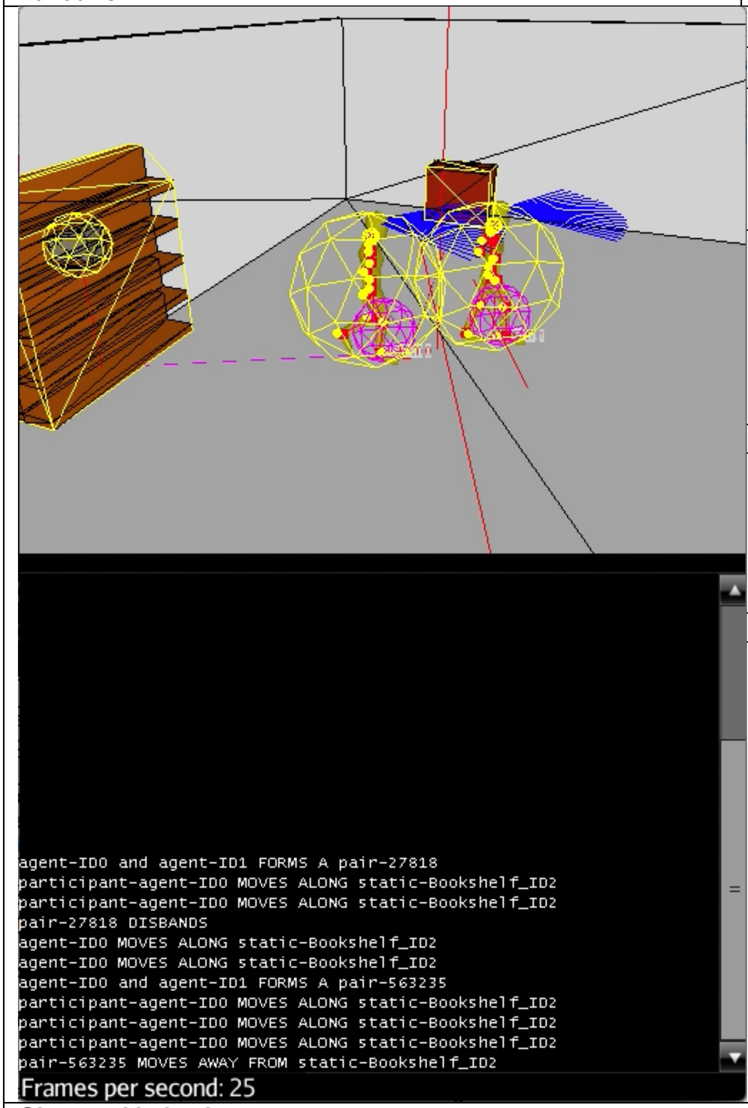
15/03/2017 18:56:33 :: 1st registration of pattern
15/03/2017 18:56:34 :: 2nd registration of pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-17. "*Group / Pair moves away from something*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	`agent-ID0`, `agent-ID1`
Location	**agent-ID0:** [161.19113, 4.3999987, -0.8794838] **agent-ID1:** [121.19113, 4.3999987, 21.8794838]
Viewing Direction	**agent-ID0:** [-0.87773572, 0, 0.48221964] **agent-ID1:** [-0.87773572, 0, 0.48221964]

Expected output

`Pair-<ID> MOVES AWAY FROM static-Bookshelf_ID2`

Output

`Pair-563235 MOVES AWAY FROM static-Bookshelf_ID2`

Observed behaviour

- both agents (agent-ID0 and agent-ID1) who formed a pair have moved away from Bookshelf_ID2
- a message appeared when both agents started to move away from the bookshelf

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console after pair-563235 and all its members (agent-ID0 and agent-ID1) was continuously walking towards static-Bookshelf_ID2 for 1 second, alongside it for 1 second and moved away from it according to time stamps recorded in the log file:

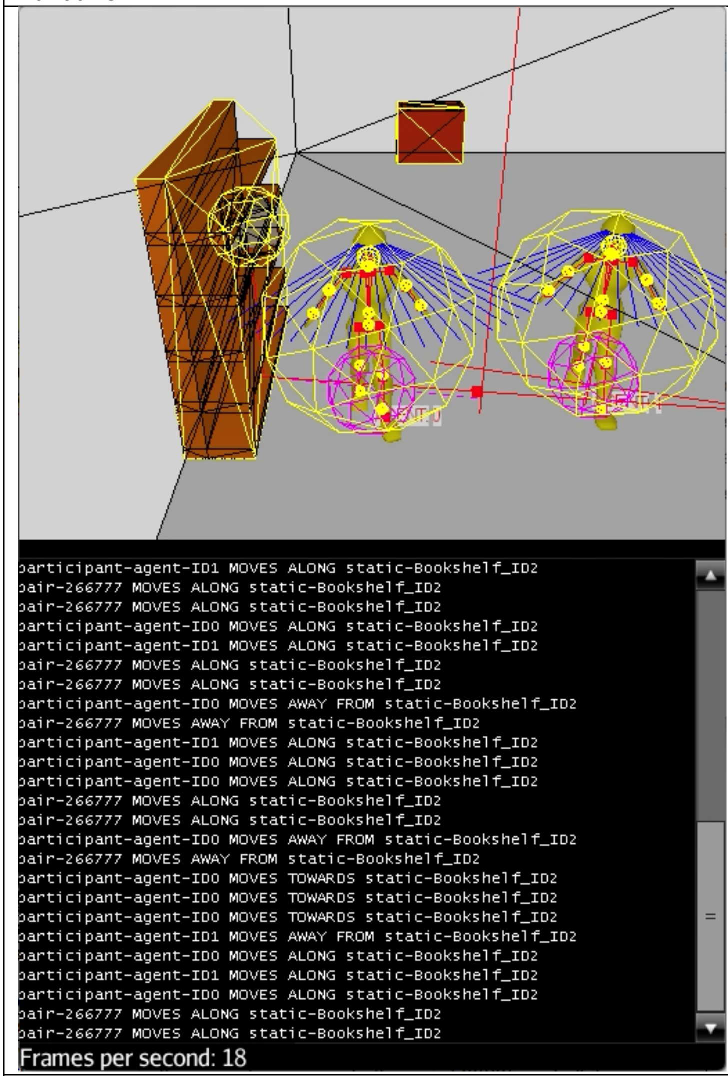
14/03/2017 20:08:35 :: pair moved towards
14/03/2017 20:08:36 :: pair moved towards
14/03/2017 20:08:37 :: pair moved alongside
14/03/2017 20:08:38 :: 1st registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-18. "*Group / Pair moves alongside something*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	`agent-ID0`, `agent-ID1`
Location	**agent-ID0:** [161.19113, 4.3999987, -0.8794838] **agent-ID1:** [121.19113, 4.3999987, 21.8794838]
Viewing Direction	**agent-ID0:** [-0.87773572, 0, 0.48221964] **agent-ID1:** [-0.87773572, 0, 0.48221964]

Expected output

`Pair-<ID> MOVES ALONG static-Bookshelf_ID2`

Output

`Pair-266777 MOVES ALONG static-Bookshelf_ID2`

Observed behaviour

- both agents (agent-ID0 and agent-ID1) who formed a pair are walking alongside Bookshelf_ID2
- a message appeared when both agents were moving alongside bookshelf

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console after pair-266777 and all its members (agent-ID0 and agent-ID1) was continuously walking alongside static-Bookshelf_ID2 for 1 second according to time stamps recorded in the log file:

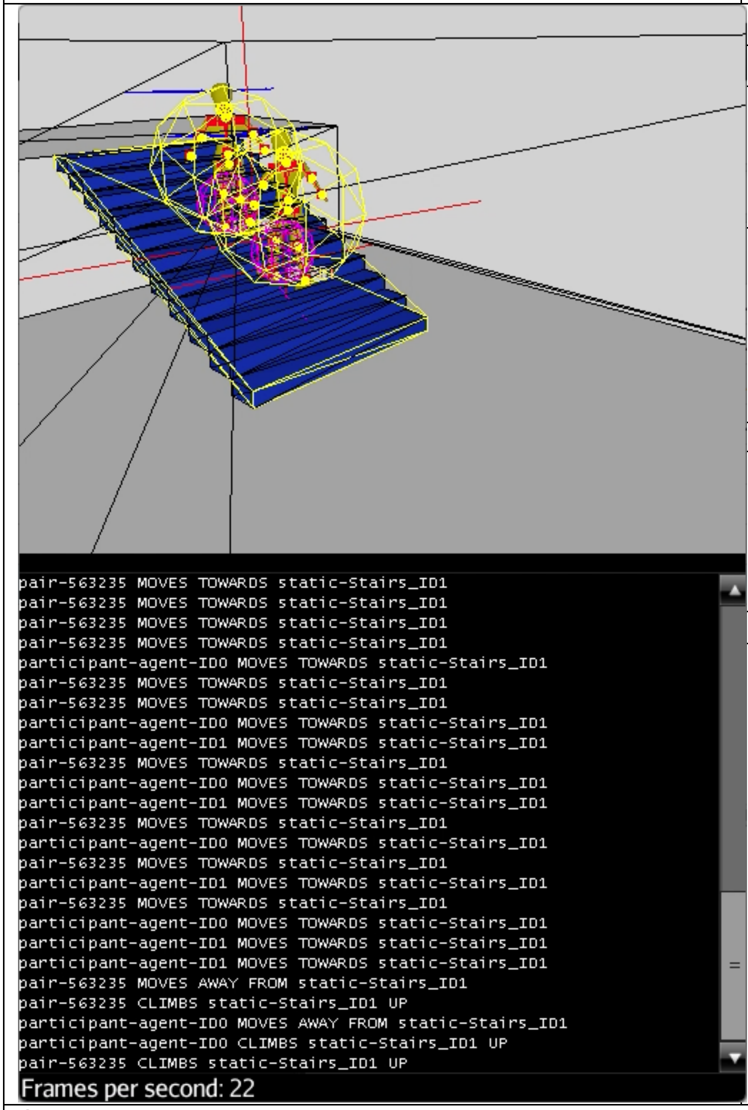
14/03/2017 20:51:24 :: 1st registration of the pattern
14/03/2017 20:51:25 :: 2nd registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-19 "*Group / Pair climbs something up*" pattern validation

Validation



Initial data

No. Agents	2
Agent ID	agent-ID0, agent-ID1
Location	**agent-ID0:** [161.19113, 4.3999987, -0.8794838] **agent-ID1:** [121.19113, 4.3999987, 21.8794838]
Viewing Direction	**agent-ID0:** [-0.87773572, 0, 0.48221964] **agent-ID1:** [-0.87773572, 0, 0.48221964]

Expected output

Pair-<ID> CLIMBS static-Stairs_ID1 UP

Output

Pair-563235 CLIMBS static-Stairs_ID1 UP

Observed behaviour

- both agents (agent-ID0 and agent-ID1) who formed a pair are on the stairs (static-Stairs_ID1) and climbing them up
- a message appeared when both agents were above static-Stairs_ID1 and climbing them upwards to reach upper level

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console after pair-563235 and all its members (agent-ID0 and agent-ID1) was continuously climbing up stairs for 1 second according to time stamps recorded in the log file:

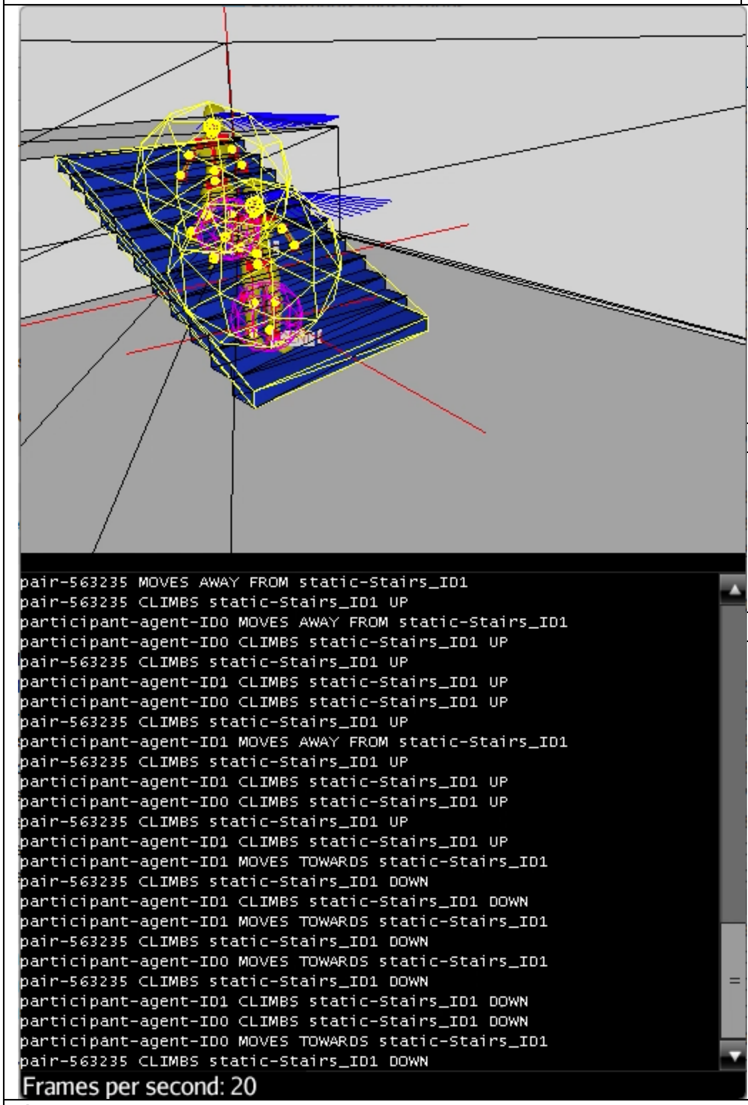
14/03/2017 22:11:12 :: 1st registration of the pattern
14/03/2017 22:11:13 :: 2nd registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-20. "*Group / Pair climbs something down"* pattern validation

Validation



Initial data

No. Agents	2
Agent ID	agent-ID0, agent-ID1
Location	**agent-ID0:** [161.19113, 4.3999987, -0.8794838] **agent-ID1:** [121.19113, 4.3999987, 21.8794838]
Viewing Direction	**agent-ID0:** [-0.87773572, 0, 0.48221964] **agent-ID1:** [-0.87773572, 0, 0.48221964]

Expected output

Pair-<ID> CLIMBS static-Stairs_ID1 DOWN

Output

Pair-563235 CLIMBS static-Stairs_ID1 DOWN

Observed behaviour

- both agents (agent-ID0 and agent-ID1) who formed a pair are on the stairs (static-Stairs_ID1) and climbing them up
- a message appeared when both agents were above static-Stairs_ID1 and climbing them downwards to reach lower level

Result analysis

The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console after pair-563235 and all its members (agent-ID0 and agent-ID1) was continuously climbing static-Stairs_ID1 downwards for 3 seconds according to time stamps recorded in the log file:

14/03/2017 23:39:22 :: 1st registration of the pattern
14/03/2017 23:39:25 :: 2nd registration of the pattern

Validation status

The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.

Table A.3-21. "*Somebody joins a Pair and form a Group*" pattern validation

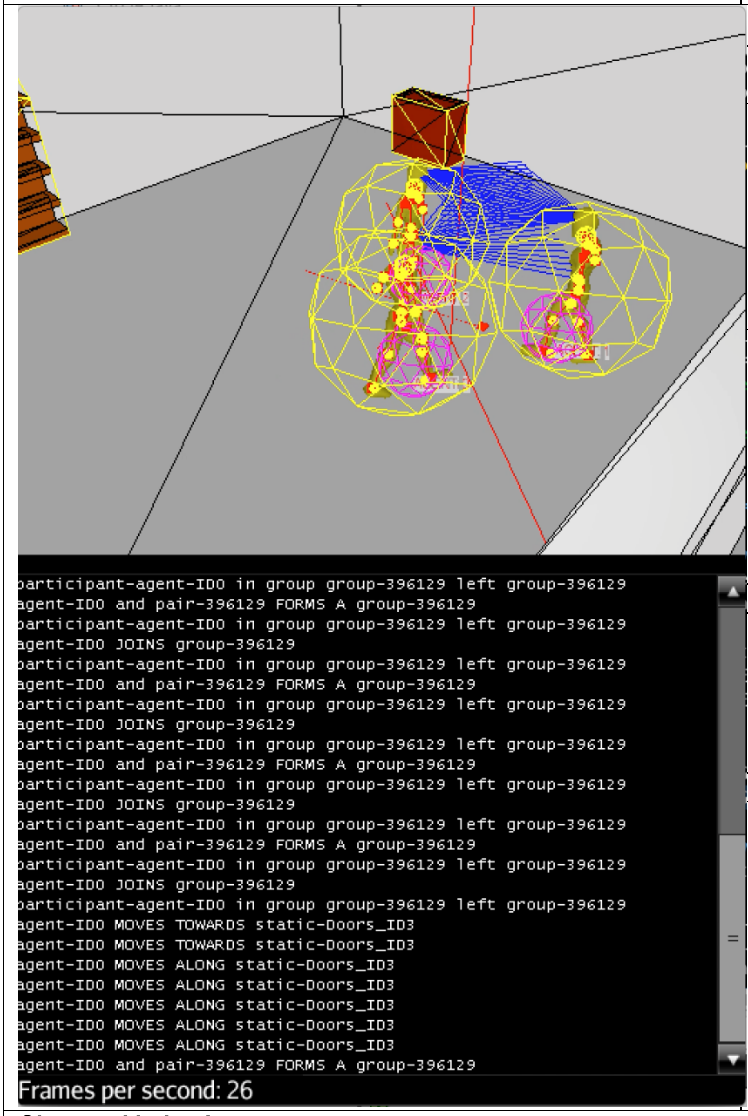
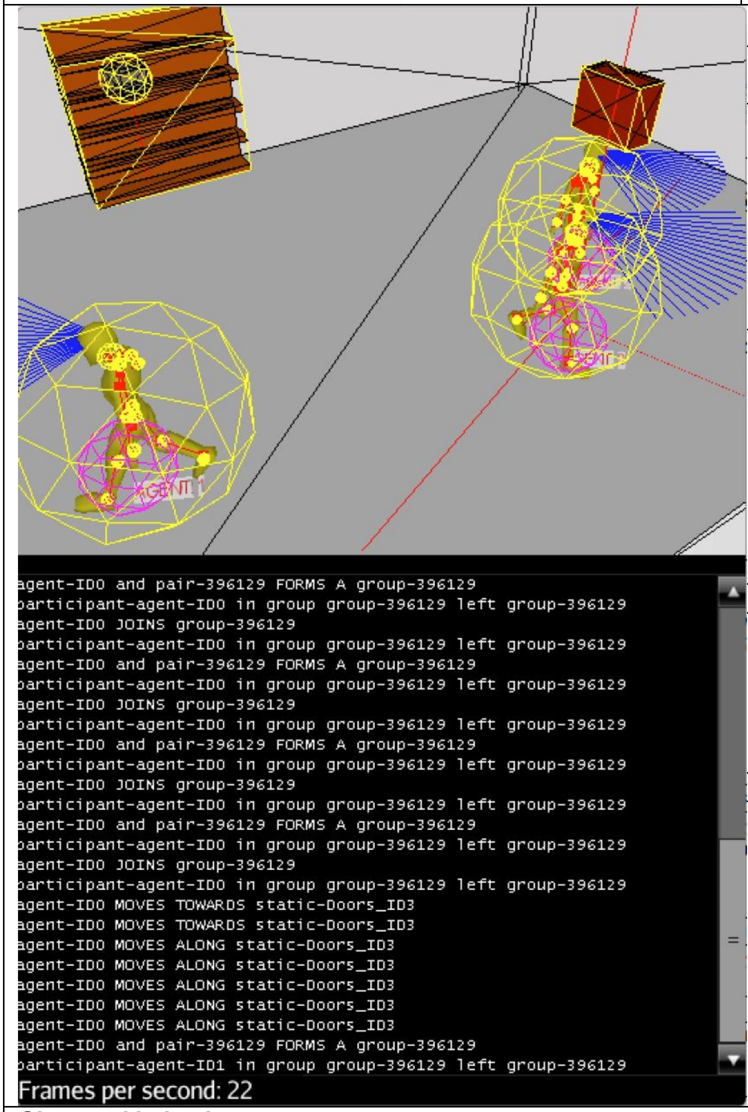
<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="8">
</td><td>No. Agents</td><td>3</td></tr>
<tr><td>Agent ID</td><td>agent-ID0, agent-ID1,agent-ID2</td></tr>
<tr><td>Location</td><td>agent-ID0: [161.19113, 4.3999987, -0.8794838]
agent-ID1: [121.19113, 4.3999987, 21.8794838]
agent-ID2: [82.19113, 4.3999987, 50.8794838]</td></tr>
<tr><td>Viewing Direction</td><td>agent-ID0: [-0.87773572, 0, 0.48221964]
agent-ID1: [-0.87773572, 0, 0.48221964]
agent-ID2: [-0.85773572, 0, 0.28221964]</td></tr>
<tr><td colspan="2">Expected output</td></tr>
<tr><td colspan="2">agent-ID0 and pair-<ID> FORMS A group-<ID></td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2">agent-ID0 and pair-396129 FORMS A group-396129</td></tr>
<tr><td colspan="3">Observed behaviour</td></tr>
<tr><td colspan="3">- agent-ID0 approached a pair formed out of two other agents (agent-ID1 and agent-ID2) and joined them
- a message appeared when agent-ID0 find itself within close proximity to a pair of agents</td></tr>
<tr><td colspan="3">Result analysis</td></tr>
<tr><td colspan="3">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console when agent-ID0 approached a pair-396129 formed out of two agents (agent-ID1 and agent-ID2) and was within a close proximity to both of them. The pair-396129 was then renamed to a group-396129.</td></tr>
<tr><td colspan="3">Validation status</td></tr>
<tr><td colspan="3">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

Table A.3-22. "*Somebody leaves a Group*" pattern validation

<table>
<tr><th>Validation</th><th colspan="2">Initial data</th></tr>
<tr><td rowspan="8">
</td>
<td>No. Agents</td><td>3</td></tr>
<tr><td>Agent ID</td><td>agent-ID0, agent-ID1,agent-ID2</td></tr>
<tr><td>Location</td><td>agent-ID0: [161.19113, 4.3999987, -0.8794838]
agent-ID1: [121.19113, 4.3999987, 21.8794838]
agent-ID2: [82.19113, 4.3999987, 50.8794838]</td></tr>
<tr><td>Viewing Direction</td><td>agent-ID0: [-0.87773572, 0, 0.48221964]
agent-ID1: [-0.87773572, 0, 0.48221964]
agent-ID2: [-0.85773572, 0, 0.28221964]</td></tr>
<tr><th colspan="2">Expected output</th></tr>
<tr><td colspan="2">participant-agent-ID1 in group group-<ID> left group-<ID></td></tr>
<tr><th colspan="2">Output</th></tr>
<tr><td colspan="2">participant-agent-ID1 in group group-396129 left group-396129</td></tr>
<tr><th colspan="3">Observed behaviour</th></tr>
<tr><td colspan="3">- agent-ID1 has walked away from the group
- a message appeared when agent-ID1 find itself within greater distance from other members of the group</td></tr>
<tr><th colspan="3">Result analysis</th></tr>
<tr><td colspan="3">The simulation was conducted with default parameters settings for this pattern (Appendix A) for validation. The message appeared in the console when agent-ID1 walked away from a group-396129 and found itself within greater distance from other members (agent-ID0 and agent-ID2).</td></tr>
<tr><th colspan="3">Validation status</th></tr>
<tr><td colspan="3">The recognition method reported the discovery of the pattern according to the projected outcome at adequate time during the simulation.</td></tr>
</table>

A.4. Performance analysis of the analyser

Experimentations were conducted from the point of view of accuracy and speed of reporting dynamic behavioural patterns from events occurring in three-dimensional space. We performed the evaluation of parameters influence on capturing events and patterns by measuring the discrepancies between console messages and our observations of dynamic movements in three-dimensional simulation. The methods related to the process are executed in a simulation loop.

Table A.4-1. Analysis of experimental results in simulating the pattern "*Somebody is walking towards something*"

<table>
<tr><td colspan="4">Log file output</td></tr>
<tr><td colspan="4"><code>18/03/2017 12:25:19 :: agent-ID0 MOVES TOWARDS static-Door_ID3</code></td></tr>
<tr><td>Range of parameters</td><td>Type</td><td colspan="2">Range (min, max)</td></tr>
<tr><td>walkTowardsInterval</td><td>seconds</td><td>0.2s</td><td>10s</td></tr>
<tr><td>walkTowardsSamplingSize</td><td>size number</td><td>2</td><td>90</td></tr>
<tr><td>walkTowardsDistanceTreshold</td><td>centimetres</td><td>10cm</td><td>200cm</td></tr>
<tr><td>walkTowardsOverallDirectionProximityTreshold</td><td>percentage</td><td>0.001</td><td>1.0</td></tr>
<tr><td colspan="4">Impact of the parameter variation</td></tr>
<tr><td colspan="4">
- The simulation was initiated with default parameter values.
- It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small.
- The distance threshold did not have an impact on the speed of reporting a pattern, but the increased values had made it difficult to visually distinguish objects that agent was actually moving towards to. It was caused by the fact that agent was too far away from the object. On the other hand, very small values had caused a situation where pattern was only registered when agent was in very close proximity of an object.
- The distance direction proximity threshold had not have a real impact on the pattern capturing. However, when this parameter was set to low values (floating-point number with three decimal places), the pattern was not registered at all.
</td></tr>
<tr><td colspan="4">Optimal configuration found</td></tr>
<tr><td colspan="3">walkTowardsInterval</td><td>0.2s</td></tr>
<tr><td colspan="3">walkTowardsSamplingSize</td><td>2</td></tr>
<tr><td colspan="3">walkTowardsDistanceTreshold</td><td>70cm</td></tr>
<tr><td colspan="3">walkTowardsOverallDirectionProximityTreshold</td><td>0.9</td></tr>
<tr><td colspan="4">Discussion</td></tr>
<tr><td colspan="4">The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern almost instantly according to timestamps recorded in the log. The distance threshold set to greater values can help identifying a pattern when agent is within greater distance to the object. However, it may cause visually vague results as it can become difficult to verify if agent is indeed walking towards an object.</td></tr>
</table>

Table A.4-2. Analysis of experimental results in simulating the pattern "*Somebody is walking away from something*"

Log file output			
`18/03/2017 13:56:46 :: agent-ID0 MOVES AWAY FROM static-Bookshelf_ID2`			
Range of parameters	**Type**	**Range (min, max)**	
walkAwayInterval	*seconds*	**0.2s**	**2s**
walkAwaySamplingSize	*size number*	**4**	**80**
walkAwayDistanceTreshold	*centimetres*	**20cm**	**90cm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing time intervals for collecting samples, the pattern was more difficult to capture. When the interval was set to more than 2 seconds, the pattern was not being recognised at all. The same observation was made when events batch size was set to bigger values. - The distance threshold did not have an impact on the speed of reporting a pattern. However, very low values resulted in lack of recognising a pattern. It was caused by the fact that events were recorded for a very short period of time and only when agent was in a very close proximity to an object. On the other hand, large values had caused a situation where pattern was registered when agent was relatively far away from the object.			
Optimal configuration found			
walkTowardsInterval			**0.5s**
walkTowardsSamplingSize			**4**
walkTowardsDistanceTreshold			**60cm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was noted that by setting time intervals, sampling sizes and distance thresholds to moderate values it is possible to recognise a pattern within 1-2 seconds according to timestamps recorded in the log. The distance threshold set to bigger values can help to identify a pattern when agent is within greater distance to the object. However, it may cause visually vague results as it can become difficult to verify if agent is walking away from a given object. **The experimental results indicate that lower distances are preferable in scenes with higher volumes of identified static objects, e.g. narrow corridors with multiple shopping shelves**. On the other hand, higher distance thresholds are more suitable to recognise this pattern in bigger scenes with less static objects.			

Table A.4-3. Analysis of experimental results in simulating the pattern "*Somebody is walking alongside something*"

Log file output			
`18/03/2017 14:22:06 :: agent-ID0 MOVES ALONG static-Bookshelf_ID2`			
Range of parameters	**Type**	**Range (min, max)**	
walkAlongInterval	*seconds*	**0.05s**	**6s**
walkAlongSamplingSize	*size number*	**10**	**50**
walkAlongDistanceToObject	*centimetres*	**20cm**	**200cm**

Impact of the parameter variation

- The simulation was initiated with default parameter values.
- It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small.
- The distance threshold did not have an impact on the speed of reporting a pattern, but the increased values caused registration of the pattern when agent was far away from the object. On the other hand, very small values had caused a situation where pattern was only registered when agent was in very close proximity of an object.

Optimal configuration found	
walkAlongInterval	**0.2s**
walkAlongSamplingSize	**2**
walkAlongDistanceToObject	**70cm**

Discussion

The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern almost instantly according to timestamps recorded in the log. The distance threshold set to bigger values can help to identify a pattern when agent is within greater distance to the object. **However, this implies that greater distance thresholds may produce more noisy data in the log if a scene has greater number of identified static objects on relatively small area. In such case, increasing sampling size and collection intervals may reduce the number of entries.** Higher distance thresholds are more suitable to recognise this pattern in more open spaces with less static objects.

Table A.4-4. Analysis of experimental results in simulating the pattern "*Somebody climbs something"*

Log file output			
`18/03/2017 16:21:07 :: agent-ID0 CLIMBS static-Stairs_ID1 UP`			
Range of parameters	**Type**	**Range (min, max)**	
climbInterval	*seconds*	**0.15s**	**2s**
climbSamplingSize	*size number*	**2**	**10**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small.			
Optimal configuration found			
climbInterval			**0.5s**
climbSamplingSize			**3**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern within 3-4 seconds according to timestamps recorded in the log. **During the experimentation it was observed that lower values of both parameters would be preferable in scenes in which stairs possess smaller number of steps.** This is due to the fact that higher values may prevent capturing a sufficient number of events to execute a sampling procedure in order to successfully recognise a pattern in time. **However, higher values are more suitable for scenes in which stairs possess greater number of steps as they prevent generating unnecessarily big number of entries in the log file and consequently reducing overall noise of produced analytic data.**			

Table A.4-5. Analysis of experimental results in simulating the pattern **"***Somebody holds something over / places something on top of something*"

<table>
<tr><td colspan="4">Log file output</td></tr>
<tr><td colspan="4">18/03/2017 16:51:34 :: agent-ID0 HOLDS dynamic-Bag_ID0 OVER static-Counter_ID4
18/03/2017 16:55:18 :: agent-ID0 PUTS dynamic-Bag_ID0 ON static-Counter_ID4</td></tr>
<tr><td>Range of parameters</td><td>Type</td><td colspan="2">Range (min, max)</td></tr>
<tr><td>carryInterval</td><td>seconds</td><td>0.1s</td><td>10s</td></tr>
<tr><td>carrySamplingSize</td><td>size number</td><td>2</td><td>20</td></tr>
<tr><td colspan="4">Impact of the parameter variation</td></tr>
<tr><td colspan="4">- The simulation was initiated with default parameter values.
- It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small.</td></tr>
<tr><td colspan="4">Optimal configuration found</td></tr>
<tr><td colspan="3">carryInterval</td><td>0.1s</td></tr>
<tr><td colspan="3">carrySamplingSize</td><td>2</td></tr>
<tr><td colspan="4">Discussion</td></tr>
<tr><td colspan="4">The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern almost instantly according to timestamps recorded in the log. During the experimentation it was observed that lower values of both parameters are more suitable for detecting a pattern in situations when agent is stretching an arm and holding object over another one. Higher values are suitable for recognising a pattern in situations when agent holds an item in hand while standing on top of a static object. In such case agent does not have to stretch an arm in order to hold an item over another one. In case of placing an item on top of a static object (static-Counter_ID4), the pattern was not recognised when higher values were set for both parameters or there was big discrepancy between their extremes, e.g. higher value set for one parameter and lower for the other.</td></tr>
</table>

Table A.4-6. Analysis of experimental results in simulating the pattern **"***Somebody picks up / reaches out for something"*

Log file output			
`18/03/2017 17:21:43 :: agent-ID0 PICKS UP dynamic-Bag_ID0` `18/03/2017 17:26:01 :: agent-ID0 REACHES OUT FOR dynamic-Parcel_ID1`			
Range of parameters	**Type**	**Range (min, max)**	
pickUpInterval	*seconds*	**0.1s**	**10s**
reachOutHandDistanceToRightLowArmMin	*decimetres*	**3dm**	**20dm**
reachOutHandDistanceToCenterPivotMin	*decimetres*	**4dm**	**20dm**
reachOutRightLowArmDistanceToCenterPivotMin	*decimetres*	**5dm**	**20dm**

Impact of the parameter variation

- The simulation was initiated with default parameter values.
- It was observed that by increasing time intervals, the pattern was more difficult to capture. When the interval was set to more than 2 seconds, the pattern was not being recognised at all.
- The distances between different body parts had to be directly proportional to their lengths at the time when agent was reaching out for an object situated on top of a static object (static-Bookshelf_ID2) in order to capture the pattern. Increasing distances between limbs resulted in pattern detection failure.

Optimal configuration found	
pickUpInterval	**0.1s**
reachOutHandDistanceToRightLowArmMin	**5dm**
reachOutHandDistanceToCenterPivotMin	**6dm**
reachOutRightLowArmDistanceToCenterPivotMin	**6dm**

Discussion

The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that by setting a time interval to lower values it was possible to detect a pattern when agent hand limb came into contact with dynamic object. During the experimentation it was observed that lower values of distances between limbs allowed the discovery of a pattern when agent reached out for a dynamic object situated on top of the static object by stretching its arm whereas higher values prevented it. **Very small distance thresholds in combination with low time intervals caused the generation of huge amount of repetitive entries in the log.** The distance values had no real impact on detecting a pattern when agent was picking up a dynamic object from the floor.

Table A.4-7. Analysis of experimental results in simulating the pattern **"***Somebody punches something / somebody*"

Log file output			
`18/03/2017 18:34:13 :: agent-ID0 PUNCHES static-Bookshelf_ID2`			
Range of parameters	**Type**	**Range (min, max)**	
limbsInterval	*seconds*	**0.1s**	**2s**
limbPunchHandDistanceToRightLowArmMin	*decimetres*	**4dm**	**30dm**
limbPunchHandDistanceToCenterPivotMin	*decimetres*	**6dm**	**30dm**
limbPunchHandRightLowArmDistanceToCenterPivotMin	*decimetres*	**6dm**	**30dm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing time intervals, the pattern was more difficult to capture. When the interval was set to more than 2 seconds, the pattern was not being recognised at all. - The distances between different body parts had to be directly proportional to their lengths at the time when agent was punching static object (static-Bookshelf_ID2) in order to capture the pattern. Increasing distances between limbs resulted in pattern detection failure.			
Optimal configuration found			
limbsInterval			**0.4s**
limbPunchHandDistanceToRightLowArmMin			**4dm**
limbPunchHandDistanceToCenterPivotMin			**6dm**
limbPunchHandRightLowArmDistanceToCenterPivotMin			**6dm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that by setting a time interval to lower values it was possible to detect a pattern when agent arm was stretched and its hand came into contact with a static object. During the experimentation it was observed that moderate values of distances between limbs allowed the discovery of a pattern when agent was punching a static object whereas higher values prevented it. **Lower time intervals caused the generation of huge amount of repetitive entries in the log.**			

Table A.4-8. Analysis of experimental results in simulating the pattern **"***Somebody kicks something / somebody*"

Log file output			
`18/03/2017 19:19:07 :: agent-ID0 KICKS static-Bookshelf_ID2`			
Range of parameters	**Type**	**Range (min, max)**	
limbsInterval	*seconds*	**0.1s**	**1s**
limbKickLegDistanceToRightLowLegMin	*decimetres*	**2dm**	**10dm**
limbKickLegDistanceToLeftFootLegMin	*decimetres*	**4dm**	**18dm**
limbKickLegDistanceToCenterPivotMin	*decimetres*	**4dm**	**18dm**
limbKickLegRightLowLegDistanceToCenterPivotMin	*decimetres*	**4dm**	**8dm**

Impact of the parameter variation

- The simulation was initiated with default parameter values.
- It was observed that by increasing time intervals, the pattern was more difficult to capture. When the interval was set to more than 1 second, the pattern was not being recognised at all.
- The distances between different body parts had to be directly proportional to their lengths at the time when agent was kicking static object (static-Bookshelf_ID2) in order to capture the pattern. Increasing distances between limbs resulted in pattern detection failure.

Optimal configuration found	
limbsInterval	**0.6s**
limbKickLegDistanceToRightLowLegMin	**3dm**
limbKickLegDistanceToLeftFootLegMin	**6dm**
limbKickLegDistanceToCenterPivotMin	**6dm**
limbKickLegRightLowLegDistanceToCenterPivotMin	**6dm**

Discussion

The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that by setting a time interval to lower values it was possible to detect a pattern when agent leg was stretched and its right foot came into contact with a static object. During the experimentation it was observed that moderate values of distances between limbs allowed the discovery of a pattern when agent was kicking a static object whereas higher values prevented it. **Lower time intervals caused the generation of huge amount of repetitive entries in the log.**

Table A.4-9. Analysis of experimental results in simulating the pattern **"***Somebody drops down something*"

Log file output			
`18/03/2017 19:41:14 :: agent-ID0 DROPS DOWN dynamic-Parcel_ID1`			
Range of parameters	**Type**	**Range (min, max)**	
dropDownCoolDownInterval	*seconds*	**0.1s**	**4s**
dropDownDistanceFromObjectMin	*decimetres*	**1dm**	**5dm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing cool down interval, the pattern was captured with a certain delay. When the cool down interval was set to more than 4 seconds, the dynamic object was already on the floor. - The distance between dynamic object (dynamic-Parcel_ID1) and agent right hand had to be set at least below 5dm, otherwise the pattern failed to be detected.			
Optimal configuration found			
dropDownCoolDownInterval			**1.5s**
dropDownDistanceFromObjectMin			**2.5dm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that by setting a cool down period to lower values it was possible to detect a pattern when agent had dropped carried dynamic object. During the experimentation it was observed that setting lower distances between a hand holding an object and dropped dynamic object had led to pattern recognition failure. **This implies that distance between hand and a dropped object cannot be greater than a distance from hand to the floor if agent remains in stationary position. It was also observed that greater cool down periods in combination with greater distances (but not exceeding the distance between hand and a floor) may cause a delay in reporting a pattern, i.e. an agent may be able to walk away from the dropped object before pattern can be recognised.**			

Table A.4-10. Analysis of experimental results in simulating the pattern "*Somebody looks left / right / up / down*"

Log file output			
`18/03/2017 20:21:50 :: agent-ID0 LOOKS DOWN` `18/03/2017 20:27:07 :: agent-ID0 LOOKS UP` `18/03/2017 20:28:42 :: agent-ID0 LOOKS RIGHT` `18/03/2017 20:29:06 :: agent-ID0 LOOKS LEFT`			
Range of parameters	**Type**	**Range (min, max)**	
headMoveInterval	*seconds*	**0.02s**	**0.2s**
headMoveCoolDownInterval	*degrees*	**0.02s**	**1s**
headMoveLeftMin	*degrees*	**30°**	**46°**
headMoveLeftMax	*degrees*	**37°**	**60°**
headMoveRightMin	*degrees*	**90°**	**130°**
headMoveRightMax	*degrees*	**125°**	**150°**
headMoveUpMin	*degrees*	**50°**	**66°**
headMoveUpMax	*degrees*	**67°**	**70°**
headMoveDownMin	*degrees*	**100°**	**123°**
headMoveDownMax	*degrees*	**114°**	**130°**

Impact of the parameter variation

- The simulation was initiated with default parameter values.
- It was observed that by increasing time interval, the pattern was more difficult to capture. When the interval was set to more than 0.5 seconds, the pattern was not being recognised at all. The same observation was made with an increased cool down periods.
- It was noted that the minimum value of angular threshold for detecting particular movement of the head had to always be lower than its maximum value in order to recognise a pattern.
- The difference between minimum and maximum values had always been within roughly 20-25 degrees on average in order to recognise a pattern.

Optimal configuration found	
headMoveInterval	**0.1s**
headMoveCoolDownInterval	**1s**
headMoveLeftMin	**40°**
headMoveLeftMax	**60°**
headMoveRightMin	**100°**
headMoveRightMax	**130°**
headMoveUpMin	**50°**
headMoveUpMax	**67°**
headMoveDownMin	**100°**
headMoveDownMax	**114°**

Discussion

The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that increasing the time interval as well as cool down period resulted in lack of recognising a pattern. During the experimentation it was observed that minimum values had to be always lower than the maximum ones in order to define adequate angular threshold range within which the head movement (rotation) could had been recognised, otherwise the pattern was not recognised at all. **It was noted that setting minimum value to 90 degrees for when "somebody looks right" had caused uncontrolled registration of entries in the log for this pattern even when agent was not turning its head at all. This indicates that setting lower values for defining angular thresholds could potentially cause erroneous output of analytic data.**

Table A.4-11. Analysis of experimental results in simulating the pattern **"***Somebody passes over something to somebody*"

<table>
<tr><td colspan="4">Log file output</td></tr>
<tr><td colspan="4"><code>18/03/2017 21:43:31 :: participant-agent-ID0 in pair with pair-108048 HANDS OVER dynamic-Parcel_ID1 TO participant-agent-ID1 in pair with pair-108048</code></td></tr>
<tr><td>Range of parameters</td><td>Type</td><td colspan="2">Range (min, max)</td></tr>
<tr><td>passOverInterval</td><td>seconds</td><td>0.20s</td><td>1s</td></tr>
<tr><td>passOverCoolDownInterval</td><td>seconds</td><td>0.2s</td><td>1.5s</td></tr>
<tr><td>passOverLimbsDistanceMinLimit</td><td>decimetres</td><td>1dm</td><td>6dm</td></tr>
<tr><td>passOverLimbsDistanceMaxLimit</td><td>decimetres</td><td>2dm</td><td>12dm</td></tr>
<tr><td colspan="4">Impact of the parameter variation</td></tr>
<tr><td colspan="4">- The simulation was initiated with default parameter values.
- It was observed that by increasing time intervals, the pattern was more difficult to capture. When the interval was set to more than 1 second, the pattern was not being recognised at all. The same observation was made with an increased cool down periods.
- The distances between each agent right hands at the time of passing the dynamic object could not had been less than 1dm and more than 11dm in order to capture the pattern. Increasing minimum and maximum distance values between limbs resulted in pattern detection failure.
-</td></tr>
<tr><td colspan="4">Optimal configuration found</td></tr>
<tr><td colspan="3">passOverInterval</td><td>0.3s</td></tr>
<tr><td colspan="3">passOverCoolDownInterval</td><td>0.6s</td></tr>
<tr><td colspan="3">passOverLimbsDistanceMinLimit</td><td>1dm</td></tr>
<tr><td colspan="3">passOverLimbsDistanceMaxLimit</td><td>5dm</td></tr>
<tr><td colspan="4">Discussion</td></tr>
<tr><td colspan="4">The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that by increasing time intervals and cool down periods it was difficult to capture a pattern. When cool down period was set below 1 second, but twice the amount of time set for time interval, the pattern had been captured almost every time agents handed over the object between each other. During the experimentation it was observed that the difference between minimum and maximum distance values between hands could not be greater than 10dm in order for a pattern to be recognised.</td></tr>
</table>

Table A.4-12. Analysis of experimental results in simulating the pattern **"***Somebody and somebody form a Pair / Somebody joins a Group*"

Log file output			
19/03/2017 11:42:43 :: agent-ID0 and agent-ID1 FORMS A pair-865450 **19/03/2017 11:52:35 :: participant-agent-ID0 in group group-662444 left group-662444** **19/03/2017 11:52:37 :: agent-ID0 and pair-662444 FORMS A group-662444**			
Range of parameters	**Type**	**Range (min, max)**	
groupFormSetInterval	*seconds*	**0.20s**	**6s**
groupFormProximityDistance	*centimetres*	**50cm**	**200cm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing time intervals, the pattern was being recognised with a delay. When the interval was set to more than 6 seconds, the pattern was recognised long after agent approached the other. - Setting greater distance threshold had caused reporting a pattern even when agents were too far away from each other. It was difficult to verify whether two agents actually formed a group from the visual state of the simulation at the time of pattern recognition in such case.			
Optimal configuration found			
groupFormSetInterval			**0.20s**
groupFormProximityDistance			**75cm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that greater values in time intervals had caused a delay in reporting a pattern despite setting greater distance thresholds. On the contrary, setting lower time intervals had caused almost immediate registration of the pattern when agent found itself within a specified distance from other agent. The same observations were made during simulation of one agent joining a pair and forming a group. **During the experimentation it was also noted that when a leaving distance threshold was smaller than group proximity distance threshold, two patterns had been interchangeably reported in the console in a relatively short amount of time: "Somebody and somebody form a Pair" and "Pair disbands" (i.e. "Somebody leaves a Group").**			

Table A.4-13. Analysis of experimental results in simulating the pattern "*Somebody leaves a Group*"

Log file output			
`19/03/2017 11:52:35 :: participant-agent-ID0 in group group-662444 left group-662444`			
Range of parameters	**Type**	**Range (min, max)**	
groupFormSetInterval	*seconds*	**0.20s**	**30s**
groupFormProximityLeavingDistance	*centimetres*	**50cm**	**200cm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing time intervals, the pattern was being recognised with a delay. When the interval was set to more than 30 seconds, the pattern was recognised long after agent has left the group. - Setting greater distance threshold had caused reporting a pattern even when agent was far away from the group. It was difficult to verify at what point the agent actually had left until it was within a greater distance from the group.			
Optimal configuration found			
groupFormSetInterval			**0.20s**
groupFormProximityLeavingDistance			**100cm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was observed that greater values in time intervals had caused a delay in reporting a pattern despite setting lower distance thresholds. On the contrary, setting lower time intervals had caused almost immediate registration of the pattern when agent found itself away from specified distance from the group. **During the experimentation it was also noted that when a leaving distance threshold was smaller than group proximity distance threshold, two patterns had been interchangeably reported in the console in a relatively short amount of time: "Somebody and somebody form a Pair" and "Pair disbands" (i.e. "Somebody leaves a Group").**			

Table A.4-14. Analysis of experimental results in simulating the pattern **"***Somebody makes handshake with somebody*"

Log file output			
`19/03/2017 12:18:11 :: participant-agent-ID1 in pair with pair-662444 MAKES HANDSHAKE WITH participant-agent-ID0 in pair with pair-662444`			
Range of parameters	**Type**	**Range (min, max)**	
groupGesturesInterval	*seconds*	**0.1s**	**2s**
groupGesturesSamplingSize	*size number*	**3**	**20**
groupGesturesHandShakeLimbsProximity	*decimetres*	**4dm**	**10dm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - It was observed that by increasing time intervals for collecting samples, the pattern was more difficult to capture. When the interval was set to more than 2 seconds, the pattern was not being recognised at all. The same observation was made when events batch size was set to larger values. - The distance threshold did not have an impact on the speed of reporting a pattern. However, greater values resulted in lack of recognising a pattern. It was caused by the fact that events could not have been captured for the events sampling procedure. On the other hand, lower values contributed to quick recognition of the pattern.			
Optimal configuration found			
groupGesturesInterval			**0.4s**
groupGesturesSamplingSize			**3**
groupGesturesHandShakeLimbsProximity			**4dm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and agent's movements. It was noted that by setting time intervals, sampling sizes and distance thresholds to lower values it is possible to recognise a pattern within 3-4 seconds according to timestamps recorded in the log. **The threshold, defining minimum distance between agent's hands, set to bigger values prevented recognising a pattern.** The experimental results indicate that events batch size cannot be set to greater values as it prevents recognising a pattern without a delay, despite setting lower values for time intervals. The same observations were made when batch size was set to lower values and time intervals to greater ones.			

Table A.4-15. Analysis of experimental results in simulating the pattern **"***Group / Pair moves towards something*"

Log file output			
`19/03/2017 12:35:28 :: pair-662444 MOVES TOWARDS static-Bookshelf_ID2`			
Range of parameters	**Type**	**Range (min, max)**	
groupWalkTowardsInterval	*seconds*	**0.2s**	**10s**
groupWalkTowardsSamplingSize	*size number*	**2**	**90**
groupWalkTowardsDistanceTreshold	*centimetres*	**10cm**	**200cm**
groupWalkTowardsOverallDirectionProximityTreshold	*percentage*	**0.001**	**1.0**
groupClosestDistanceTreshold	*centimetres*	**10cm**	**200cm**

Impact of the parameter variation

- The simulation was initiated with default parameter values.
- The simulation was conducted with the same parameter variations as in case of experimentation with individual pattern, i.e. "Somebody is walking towards something".
- It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small.
- The distance threshold did not have an impact on the speed of reporting a pattern, but the increased values had made it difficult to visually distinguish objects that a pair was actually moving towards to. It was caused by the fact that both agents (pair) were too far away from the object. On the other hand, very small values had caused a situation where pattern was only registered when agent was in very close proximity of an object. The threshold responsible for selecting the closest object had not have a greater impact on the recognition of the pattern due to presence of only few identified static objects located too far away from each other within the scene.
- The distance direction proximity threshold had not have a real impact on the pattern capturing. However, when this parameter was set to low values (floating-point number with three decimal places), the pattern was not registered at all.

Optimal configuration found	
groupWalkTowardsInterval	**0.2s**
groupWalkTowardsSamplingSize	**2**
groupWalkTowardsDistanceTreshold	**70cm**
groupWalkTowardsOverallDirectionProximityTreshold	**0.9**
groupClosestDistanceTreshold	**60cm**

Discussion

The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and pair movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern almost instantly according to timestamps recorded in the log. The distance threshold set to greater values can help identifying a pattern when a pair or group is within greater distance to the object. However, it may cause visually vague results as it can become difficult to verify if a pair or group is indeed walking towards an object. **The experimental results of this pattern are similar to the experimental results of simulating individual pattern, i.e. "Somebody is walking towards something".**

Table A.4-16. Analysis of experimental results in simulating the pattern "*Group / Pair moves away from something*"

Log file output			
`19/03/2017 13:03:45 :: pair-308010 MOVES AWAY FROM static-Bookshelf_ID2`			
Range of parameters	**Type**	**Range (min, max)**	
groupWalkAwayInterval	*seconds*	**0.2s**	**2s**
groupWalkAwaySamplingSize	*size number*	**4**	**80**
groupWalkAwayDistanceTreshold	*centimetres*	**20cm**	**90cm**
groupClosestDistanceTreshold	*centimetres*	**20cm**	**90cm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - The simulation was conducted with the same parameter variations as in case of experimentation with individual pattern, i.e. "Somebody is walking away from something". - It was observed that by increasing time intervals for collecting samples, the pattern was more difficult to capture. When the interval was set to more than 2 seconds, the pattern was not being recognised at all. The same observation was made when events batch size was set to bigger values. - The distance threshold did not have an impact on the speed of reporting a pattern. However, very low values resulted in lack of recognising a pattern. It was caused by the fact that events were recorded for a very short period of time and only when a pair was in a very close proximity to an object. On the other hand, large values had caused a situation where pattern was registered when agent was relatively far away from the object. - The threshold responsible for selecting the closest object had not have a greater impact on the recognition of the pattern due to presence of only few identified static objects located too far away from each other within the scene.			
Optimal configuration found			
groupWalkAwayInterval			**0.5s**
groupWalkAwaySamplingSize			**4**
groupWalkAwayDistanceTreshold			**60cm**
groupClosestDistanceTreshold			**50cm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and pair movements. It was noted that by setting time intervals, sampling sizes and distance thresholds to moderate values it is possible to recognise a pattern within 2-3 seconds according to timestamps recorded in the log. The distance threshold set to bigger values can help to identify a pattern when agent is within greater distance to the object. However, it may cause visually vague results as it can become difficult to verify if pair is walking away from a given object. **The experimental results are similar to simulation results of this pattern in case of individual agent**. They indicate that lower distances are preferable in scenes with higher volumes of identified static objects, e.g. narrow corridors with multiple shopping shelves. On the other hand, higher distance thresholds are more suitable to recognise this pattern in bigger scenes with less static objects.			

Table A.4-17. Analysis of experimental results in simulating the pattern "*Group / Pair moves alongside something*"

Log file output			
19/03/2017 13:08:42 :: pair-308010 MOVES ALONG static-Bookshelf_ID2			
Range of parameters	**Type**	**Range (min, max)**	
groupWalkAlongInterval	*seconds*	**0.05s**	**6s**
groupWalkAlongSamplingSize	*size number*	**10**	**50**
groupWalkAlongDistanceToObject	*centimetres*	**20cm**	**200cm**
groupClosestDistanceTreshold	*centimetres*	**20cm**	**200cm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - The simulation was conducted with the same parameter variations as in case of experimentation with individual pattern, i.e. "Somebody is walking alongside something". - It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small. - The distance threshold did not have an impact on the speed of reporting a pattern, but the increased values caused registration of the pattern when agent was far away from the object. On the other hand, very small values had caused a situation where pattern was only registered when agent was in very close proximity of an object. - The threshold responsible for selecting the closest object had not have a greater impact on the recognition of the pattern due to presence of only few identified static objects located too far away from each other within the scene.			
Optimal configuration found			
groupWalkAlongInterval			**0.2s**
groupWalkAlongSamplingSize			**2**
groupWalkAlongDistanceToObject			**70cm**
groupClosestDistanceTreshold			**60cm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and pair movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern almost instantly according to timestamps recorded in the log. The distance threshold set to bigger values can help to identify a pattern when agent is within greater distance to the object. **Similarly to simulation of pattern in case of individual agent, this implies that greater distance thresholds may produce more noisy data in the log if a scene has greater number of identified static objects on relatively small area.** In such case, increasing sampling size and collection intervals may reduce the number of entries. Higher distance thresholds are more suitable to recognise this pattern in more open spaces with less static objects.			

Table A.4-18. Analysis of experimental results in simulating the pattern **"***Group / Pair climbs something*"

Log file output			
19/03/2017 13:29:47 :: pair-185408 CLIMBS static-Stairs_ID1 UP **19/03/2017 13:30:00 :: pair-185408 CLIMBS static-Stairs_ID1 DOWN**			
Range of parameters	**Type**	**Range (min, max)**	
groupClimbInterval	*seconds*	**0.05s**	**6s**
groupClimbSamplingSize	*size number*	**10**	**50**
groupClosestDistanceTreshold	*centimetres*	**10cm**	**10cm**
Impact of the parameter variation			
- The simulation was initiated with default parameter values. - The simulation was conducted with the same parameter variations as in case of experimentation with individual pattern, i.e. "Somebody climbs something". - It was observed that by increasing time intervals for collecting samples, the time required for pattern recognition has extended, despite setting small event batch size (sampling size). The same observation was made when events batch size was set to bigger values while time intervals were small. - The threshold responsible for selecting the closest object had not have a greater impact on the recognition of the pattern. *It is caused by the fact that the incorporation of this parameter into event capturing method for this pattern was made on a basis of reusing the existing code. Hence it does not influence the process of pattern recognition in any way.*			
Optimal configuration found			
groupClimbInterval			**0.5s**
groupClimbSamplingSize			**3**
groupClosestDistanceTreshold			**10cm**
Discussion			
The optimal parameters were selected on a basis of the speed of reporting a pattern and low discrepancies between log entries and pair movements. It was noted that by setting time intervals and sampling sizes to low values it is possible to recognise a pattern within 2-3 seconds according to timestamps recorded in the log. **During the experimentation it was observed that lower values of both parameters would be preferable in scenes in which stairs possess smaller number of steps.** This is due to the fact that higher values may prevent capturing a sufficient number of events to execute a sampling procedure in order to successfully recognise a pattern in time. **However, higher values are more suitable for scenes in which stairs possess greater number of steps as they prevent generating unnecessarily big number of entries in the log file and consequently reducing overall noise of produced analytic data. The experimental results of this pattern are similar to the experimental results of simulating individual pattern, i.e. "Somebody climbs something".**			

A.5. Behavioural Pattern Language Grammar in EBNF notation

Object ::= Static-Object | Dynamic-Object | Actor
Static-Object ::= **'static-'** Identifier
Dynamic-Object ::= **'dynamic-'** Identifier

Actor ::= **'actor-'** Identifier
Participant ::= **'participant-'** Actor **'in group'** Group
| **'participant-'** Actor **'in pair with'** Pair

Group ::= **'group-'** Identifier
Pair ::= **'pair-'** Identifier

Action ::= Active-Action | Dynamic-Action | Passive-Action | Group-Action

Active-Action ::= **(** Actor | Participant **)** Motion-Action
| **(** Actor | Participant **)** Interactive-Action
| **(** Actor | Participant **) (FALLS DOWN I STANDS UP)**
| **(** Actor | Participant **) LOOKS (** Vertical-Direction | Horizontal-Direction **)**

Dynamic-Action ::= **(** Static-Object | Dynamic-Object **)** Passive-Action-Descriptor Object

Passive-Action ::= Static-Object Passive-Action-Descriptor Object

Group-Action ::= **(** Group | Pair **)** Motion-Action
| **(** Group | Pair **)** Interactive-Action

System-Action ::= System-Action-Descriptor System-Unit-Descriptor

Motion-Action ::= Motion-Action-Descriptor Relative-Motion-Descriptor Object
| Motion-Action-Descriptor **(** Vector-Direction | Vertical-Direction **)**
| **CLIMBS** Static-Object Vertical-Direction
| **CLIMBS ON** Static-Object
I TURNS Horizontal-Direction

Interactive-Action ::= **(** Interactive-Action-Descriptor | Passive-Action-Descriptor **) (** Static-Object | Dynamic-Object **)**

Motion-Action-Descriptor ::= **MOVES** | **WALKS** | **RUNS**
Relative-Motion-Descriptor ::= **ALONG** | **TOWARDS** | **AWAY FROM**
Passive-Action-Descriptor ::= **PUSHES** | **PULLS I KNOCKS DOWN** | **PICKS UP**
| **DROPS DOWN**

Interactive-Action-Descriptor ::= **I REACHES OUT FOR** | **LOOKS AT**
| **KICKS I PUNCHES I THROWS I HITS I HEADBUTTS**

System-Action-Descriptor ::= **CALL** | **SEND I REQUEST I RECEIVE**
System-Unit-Descriptor ::= **STAFF I POLICE I FIRE BRIGADE I BOMB SQUAD**

Follow-Up-Action ::= Dynamic-Object Interactive-Action-Descriptor Object **'following' (** Action | Event **)**

Reaction ::= **(** Actor | Participant **)** Interactive-Action **'in response to' (** Action | Event **)**

Direction ::= Vector-Direction | Vertical-Direction | Horizontal-Direction
Vector-Direction ::= **FORWARDS** | **BACKWARDS**
Vertical-Direction ::= **UP** | **DOWN**
Horizontal-Direction ::= **LEFT** | **RIGHT**

Event ::= Asynchronous-Event | Change-Event | Cause-Event | Sequence-Event
| Change-Event | Result-Group-Event

Asynchronous-Event ::= Static-Object Asynchronous-Event-Descriptor

Change-Event ::= **(** Static-Object | Dynamic-Object **) 'has'** Change-Event-Descriptor

Result-Event ::= **(** Active-Action | Group-Action **) 'results in'** Change-Event

Cause-Event ::= Event **‘causes’** Event

Sequence-Event ::= Event **‘follows’** Event

Result-Group-Event ::= Actor **‘and’** Actor **‘formed a’** Pair
| Actor **‘and’** Pair **‘formed a’** Group
| Pair **‘and’** Group **‘formed a’** Group
| Actor **‘joined’** Group
| Participant **‘left’** Group
| Group **‘merged with’** Group
| **(** Pair | Group **)** **‘disbands’**

Asynchronous-Event-Descriptor ::= **GOES OFF** | **FALLS DOWN** | **APPEARS**
| **DISAPPEARS** | **TURNS ON** | **TURNS OFF** | **OPENS** |
| **CLOSES** | **BLOCKS** | **UNBLOCKS**

Change-Event-Descriptor ::= **OPENED** | **CLOSED** | **MOVED** | **TURNED ON**
| **TURNED OFF** | **FELL DOWN**

State ::= **(** Actor | Participant | Pair | Group **)** **‘is’** State-Actor-Descriptor
| **(** Static-Object | Dynamic-Object **)** **(** **‘is’** | **‘are’** **)** State-Object-Descriptor

Configuration ::= State
| State **;** Configuration
| **‘at current time’** **(** Actor | Participant | Pair | Group **)** **‘is’** State-Actor-Descriptor
| ‘at current time’ **(** Static-Object | Dynamic-Object **)** **‘is’** State-Object-Descriptor
| **‘for the past’** digit **(** **‘minutes’** | **‘hour’** **)** **(** Actor | Participant | Pair | Group **)** **‘was’** **(** State-Motion-Descriptor | State-Actor-Descriptor **)**
| **‘for the past’** digit **(** **‘minutes’** | **‘hour’** **)** **(** Static-Object | Dynamic-Object **)** **‘was’** **(** State-Motion-Descriptor | State-Object-Descriptor **)**
| **‘from’** Time-Descriptor **‘to’** Time-Descriptor **(** Actor | Participant | Pair | Group **)** **‘was’** **(** State-Motion-Descriptor | State-Actor-Descriptor **)**
| **‘from’** Time-Descriptor **‘to’** Time-Descriptor **(** Static-Object | Dynamic-Object **)** **‘was’** **(** State-Motion-Descriptor | State-Object-Descriptor **)**

Situation ::= **‘at current time the situation is’** Situation-Descriptor
| **‘for the past’** digit **(** **‘minutes’** | **‘hours’** **)** **‘the situation was’** Situation-Descriptor
| **‘from’** Time-Descriptor **‘to’** Time-Descriptor **‘the situation was’** Situation-Descriptor

State-Motion-Descriptor ::= **STILL** | **MOVING I FALLING I FLYING**

State-Actor-Descriptor ::= **‘acting’ (VIOLENTLY | HARMFULLY | ERRATICALLY**
I SUSPICIOUSLY I RISKY I UNCONCIOUSLY I NORMALLY)
| **LAYING ON THE (FLOOR I** Static-Object **)**

State-Object-Descriptor ::= **OPEN I CLOSED I BLOCKED**
I UNBLOCKED I ON I OFF
I LAYING ON THE (FLOOR I Static-Object **)**

Situation-Descriptor ::= **NORMAL** | **SUSPICIOUS** | **HAZARDOUS I DANGEROUS**

Time-Descriptor ::= **‘[‘** digit **‘:’** digit **‘]’**

Expressions ::= Action | Action **;** Expression
| Event | Event **;** Expression
| Configuration | Configuration **;** Expression
| Situation | Situation **;** Expression
| **‘if’** … **‘else’** …

A.6. XML schema of the configuration file

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="config">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="parameterSet" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="parameter" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="name"/>
                    <xs:element type="xs:float" name="value"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="class" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.7. XML schema of the streamlined data files

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="streamInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="parameterSet">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="parameter">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="name"/>
                    <xs:element name="value">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="agents">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="agent" maxOccurs="unbounded" minOccurs="0">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="location">
                                        <xs:complexType>
                                          <xs:sequence>
                                            <xs:element type="xs:float" name="vectorX"/>
                                            <xs:element type="xs:float" name="vectorY"/>
                                            <xs:element type="xs:float" name="vectorZ"/>
                                          </xs:sequence>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element name="viewDirection">
                                        <xs:complexType>
                                          <xs:sequence>
```

```
                                              <xs:element type="xs:float" name="vectorX"/>
                                              <xs:element type="xs:float" name="vectorY"/>
                                              <xs:element type="xs:float" name="vectorZ"/>
                                            </xs:sequence>
                                          </xs:complexType>
                                        </xs:element>
                                      </xs:sequence>
                                      <xs:attribute type="xs:string" name="id" use="optional"/>
                                    </xs:complexType>
                                  </xs:element>
                                </xs:sequence>
                                <xs:attribute type="xs:byte" name="number"/>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
              <xs:attribute type="xs:string" name="class"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

A.8. A detailed list of default parameters used by the simulator.

Pattern name	Parameter name	Value Type	Unit of measurement	Default Value
"Somebody is walking towards something"	walkTowardsInterval	Floating-point	seconds	0.10
	walkTowardsSamplingSize	Integer	size number	5
	walkTowardsDistanceTreshold	Floating-point	centimetres	80
	walkTowardsOverallDirectionProximityTreshold	Floating-point	percentage	0.3
"Somebody is walking away from something"	walkAwayInterval	Floating-point	seconds	0.15
	walkAwaySamplingSize	Integer	size number	15
	walkAwayDistanceTreshold	Floating-point	centimetres	80
"Somebody walks alongside something"	walkAlongInterval	Floating-point	seconds	0.05
	walkAlongSamplingSize	Integer	size number	10
	walkAlongDistanceToObject	Floating-point	centimetres	100
"Somebody climbs something"	climbInterval	Floating-point	seconds	0.1
	climbSamplingSize	Integer	size number	15
"Somebody holds something over / places something on top of something"	carryInterval	Floating-point	seconds	0.9
	carrySamplingSize	Integer	size number	10
"Somebody picks up / reaches out for / carry something"	pickUpInterval	Floating-point	seconds	0.25
	reachOutHandDistanceToRightLowArmMin	Floating-point	decimetres	12
	reachOutHandDistanceToCenterPivotMin	Floating-point	decimetres	9
	reachOutRightLowArmDistanceToCenterPivotMin	Floating-point	decimetres	6

Pattern name	Parameter name	Value Type	Unit of measurement	Default Value
"*Somebody punches something / somebody*"	limbsInterval	Floating-point	seconds	0.2
	limbPunchHandDistanceToRightLowArmMin	Floating-point	decimetres	5.2
	limbPunchHandDistanceToCenterPivotMin	Floating-point	decimetres	10
	limbPunchHandRightLowArmDistanceToCenterPivotMin	Floating-point	decimetres	9
"*Somebody kicks something / somebody*"	limbsInterval	Floating-point	seconds	0.2
	limbKickLegDistanceToRightLowLegMin	Floating-point	decimetres	8
	limbKickLegDistanceToLeftFootLegMin	Floating-point	decimetres	12
	limbKickLegDistanceToCenterPivotMin	Floating-point	decimetres	12.5
	limbKickLegRightLowLegDistanceToCenterPivotMin	Floating-point	decimetres	10
"*Somebody passes over something to somebody*"	passOverInterval	Floating-point	seconds	0.25
	passOverCoolDownInterval	Floating-point	seconds	1.5
	passOverLimbsDistanceMinLimit	Floating-point	decimetres	1
	passOverLimbsDistanceMaxLimit	Floating-point	decimetres	11
"*Somebody drops down something*"	dropDownCoolDownInterval	Floating-point	seconds	1.5
	dropDownDistanceFromObjectMin	Floating-point	decimetres	5
"*Somebody looks left*"	headMoveInterval	Floating-point	seconds	0.02
	headMoveCoolDownInterval	Floating-point	seconds	0.8
	headMoveLeftMin	Floating-point	decimetres	46
	headMoveLeftMax	Floating-point	decimetres	52

Pattern name	Parameter name	Value Type	Unit of measurement	Default Value
"*Somebody looks right*"	headMoveInterval	Floating-point	seconds	0.02
	headMoveCoolDownInterval	Floating-point	seconds	0.8
	headMoveRightMin	Floating-point	angle degrees	130
	headMoveRightMax	Floating-point	angle degrees	132
"*Somebody looks up*"	headMoveInterval	Floating-point	seconds	0.02
	headMoveCoolDownInterval	Floating-point	seconds	0.8
	headMoveUpMin	Floating-point	angle degrees	66
	headMoveUpMax	Floating-point	angle degrees	67
"*Somebody looks down*"	headMoveInterval	Floating-point	seconds	0.02
	headMoveCoolDownInterval	Floating-point	seconds	0.8
	headMoveDownMin	Floating-point	angle degrees	123
	headMoveDownMax	Floating-point	angle degrees	125
"*Somebody and somebody form a Pair*"	groupFormSetInterval	Floating-point	seconds	1.5
	groupFormProximityDistance	Floating-point	centimetres	50
"*Somebody joins a Group*"	groupFormSetInterval	Floating-point	seconds	1.5
	groupFormProximityDistance	Floating-point	centimetres	50
"*Somebody leaves a Group*"	groupFormSetInterval	Floating-point	seconds	1.5
	groupFormProximityLeavingDistance	Floating-point	centimetres	70

Pattern name	Parameter name	Value Type	Unit of measurement	Default Value
“*Somebody makes handshake with somebody*”	groupGesturesInterval	Floating-point	seconds	0.1
	groupGesturesSamplingSize	Integer	size number	10
	groupGesturesHandShakeLimbsProximity	Floating-point	decimetres	5
“*Group / Pair moves towards something*”	groupWalkTowardsInterval	Floating-point	seconds	0.1
	groupWalkTowardsSamplingSize	Integer	size number	5
	groupWalkTowardsDistanceTreshold	Floating-point	centimetres	80
	groupWalkTowardsOverallDirectionProximityTreshold	Floating-point	percentage	0.3
	groupClosestDistanceTreshold	Floating-point	centimetres	80
“*Group / Pair moves away from something*”	groupWalkAwayInterval	Floating-point	seconds	0.15
	groupWalkAwaySamplingSize	Integer	size number	15
	groupWalkAwayDistanceTreshold	Floating-point	centimetres	80
	groupClosestDistanceTreshold	Floating-point	centimetres	80
“*Group / Pair moves alongside something*”	groupWalkAlongInterval	Floating-point	seconds	0.05
	groupWalkAlongSamplingSize	Integer	size number	10
	groupWalkAlongDistanceToObject	Floating-point	centimetres	100
	groupClosestDistanceTreshold	Floating-point	centimetres	80
“*Group / Pair climbs something*”	groupClimbInterval	Floating-point	seconds	0.1
	groupClimbSamplingSize	Integer	size number	8
	groupClosestDistanceTreshold	Floating-point	centimetres	80

A.9 Sample XML data file generated in interactive mode


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<video>
  <movementSet class="VideoState">
    <agents number="1">
      <agent id="agent-ID0">
        <movement counter="1" counterID="agent-ID0-1" frameRate="0.5066449" time="15-10-2017-15-
11-52" tpf="0.0" tpfSinceLast="1.973769">
          <globalPosition>
            <vectorX>161.19113</vectorX>
            <vectorY>4.3999987</vectorY>
            <vectorZ>51.879482</vectorZ>
          </globalPosition>
          <walkDirection direction="">
            <vectorX>0.0</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.0</vectorZ>
          </walkDirection>
          <viewDirection direction="none" directionValue="0.0">
            <vectorX>-0.87773573</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.48221964</vectorZ>
          </viewDirection>
        </movement>
        <movement counter="2" counterID="agent-ID0-2" frameRate="2.146515" time="15-10-2017-15-11-
52" tpf="1.973769" tpfSinceLast="0.46587142">
          <globalPosition>
            <vectorX>161.19113</vectorX>
            <vectorY>4.3951845</vectorY>
            <vectorZ>51.879482</vectorZ>
          </globalPosition>
          <walkDirection direction="">
            <vectorX>0.0</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.0</vectorZ>
          </walkDirection>
          <viewDirection direction="none" directionValue="0.0">
            <vectorX>-0.87773573</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.48221964</vectorZ>
          </viewDirection>
        </movement>
 […]
```

```
      <movement counter="280" counterID="agent-ID0-280" frameRate="30.299955" time="15-10-2017-15-11-54" tpf="12.3829565" tpfSinceLast="0.03300335">
          <globalPosition>
            <vectorX>136.80289</vectorX>
            <vectorY>4.398747</vectorY>
            <vectorZ>88.1191</vectorZ>
          </globalPosition>
          <walkDirection direction="forward">
            <vectorX>0.0</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.0</vectorZ>
          </walkDirection>
          <viewDirection direction="none" directionValue="-3.1415927">
            <vectorX>-0.09739578</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.9967296</vectorZ>
          </viewDirection>
        </movement>
      <movement counter="281" counterID="agent-ID0-281" frameRate="30.20945" time="15-10-2017-15-11-54" tpf="12.41596" tpfSinceLast="0.033102226">
          <globalPosition>
            <vectorX>136.80289</vectorX>
            <vectorY>4.398989</vectorY>
            <vectorZ>88.1191</vectorZ>
          </globalPosition>
          <walkDirection direction="forward">
            <vectorX>0.0</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.0</vectorZ>
          </walkDirection>
          <viewDirection direction="none" directionValue="-3.1415927">
            <vectorX>-0.09739578</vectorX>
            <vectorY>0.0</vectorY>
            <vectorZ>0.9967296</vectorZ>
          </viewDirection>
        </movement>
      </agent>
    </agents>
  </movementSet>
</video>
```