

Data Integration and Visualization of Air Quality in Urban Environment

Viktor Sowinski-Mydlarz, Vassil Vassilev, Pawel Gasiorowski

Cyber Security Research Centre – London Metropolitan University, London, UK

Emails: {w.sowinski-mydlarz,v.vassilev p.gasiorowski}@londonmet.ac.uk

Abstract – Air quality visualization importance cannot be overestimated for several reasons, ranging from public health and environmental awareness to policy-making and urban planning. The tool described here is meant to bridge the gap between scientific data and general audience, promoting greater understanding and prevention of air pollution through taking the hourly readings from 23 Develiot stations in the city of Sofia and overlaying them onto terrain map and 3d buildings layer (coming from Open Street Maps).

Various open source / non-commercial tools are used for data preprocessing, analysis, and integration. The languages and data formats are explained. Implementation of the platform, transformation, integration, and visualisation of sensor, historical, and geolocation data are discussed in detail. Integrating 3d building reconstruction and triangulation of measurements for arbitrary points in the map are explained. Finally, the data flow from the sensors through the API, MongoDB database, Flask framework to the frontend web application is shown.

Users can interact with the data, zoom in/out, rotate the view, and access additional information about specific data points. All elements of the map are interactive – clicking on the air quality stations domes gives the station details and measurements, selecting an OSM 3d building shows the semantic context (address, type of building, name in Bulgarian and English, owner, etc.). Double clicking on any point in map creates a triangulated marker.

Keywords – Geolocation Data, Multi-dimensional Data Visualisation, Web Visualization, City Infrastructure, Air Pollution

1. Introduction

Monitoring air pollution in cities is a proactive and preventive measure safeguarding public health, the environment, and overall urban well-being. It helps with informed decision-making and empowers communities to work together toward cleaner, healthier, and more sustainable cities.

Analysing the air pollution in the city involves taking measurements from air quality sensors, transferring data from the stations through an API to a NoSQL database (MongoDB), reading the data in JSON format, parsing it, and finally visualizing using JavaScript in Cesium Ion platform. This work describes integration, analysis, and visualization aspect of the Urban Data Space (Sofia, Bulgaria) project.

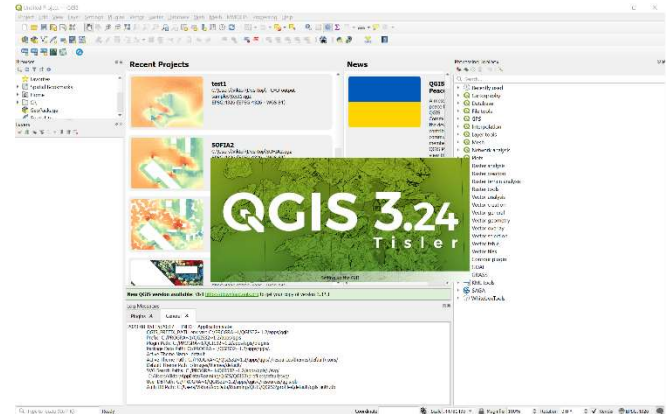


FIGURE 1. QGIS VERSION 3.24.2-TISLER

Initially CityGML 3.0 Conceptual Model and QGIS 3.24.2-Tisler have been used for pre-processing and editing GeoData. For visualization 3D City Database and Cesium Ion platform for 3D geospatial data have been containerized using Kubernetes and Docker. Cesium allows efficient processing, managing, and displaying large-scale geospatial data in a highly interactive and visually appealing manner.

Open Street Maps 3d buildings mappings with semantic context have been overlayed over ESRI World Imagery terrain layer as well as air quality station's locations and pollution data. The simulation of airflow around the buildings is a work in progress and will be described in due course.

The live data comes through an API from 23 active air pollution stations located in the city of Sofia. Historical data are stored in Mongo DB. Automatic hourly updating of measurements is shown in Cesium Ion. KML is a file format used to display geographic data in an Earth browser such as Google Earth. KML uses a tag-based structure with nested elements and attributes and is based on the XML standard.

2. Tools and Resources

All the tools used in the project are open source and/or non-commercial.

2.1. Applications and Tools

QGIS version 3.24.2-Tisler has been employed for pre-processing and editing GeoData. QGIS is a user-friendly Open-Source Geographic Information System (GIS) licensed under the GNU General Public License.

QGIS is an official project of the Open-Source Geospatial Foundation (OSGeo). It runs on various platforms (Linux, Unix, Mac OSX, Windows, and Android). It includes various vector, raster, and database formats and functionalities. [1]

3D City Database and Cesium Ion platform have been used for 3D geospatial data visualization. 3D City Database is a free geodatabase for storage, visualization, and management of digital twin 3D city models. It is a relational database. Schema applies the CityGML standard. Besides visualization, semantic urban objects are incorporated in advanced analysis. 3DCityDB has been used in numerous research projects related to 3D city models for more than 14 years. It incorporates tools for data exchange and works with cloud services. The formats supported are KML, COLLADA, and glTF. 3DCityDB is compatible with Google Earth, ArcGIS, and the WebGL-based Cesium Virtual Globe. [2] Cesium Ion is a platform for 3D geospatial data. It is used for optimization, cloud hosting and streaming 3D Tiles. It facilitates curated global 3D content together with Cesium World Terrain, Bing Maps imagery, and Cesium OSM Buildings. The results can be shared over web browser. [3] PyCharm is an integrated development environment (IDE) for programming in Python. It is developed by the Czech company JetBrains.

2.2. Languages and Formats

Constructing Cesium Ion visualization and updating measurement updates have been originally done through CityGML 3.0 Conceptual Model and KML (Keyhole Markup Language) layers. CityGML Conceptual Model provides definitions for basic entities, attributes, and relations of a semantically rich 3D city model. It supports maintenance of 3D city models for reuse of the same data in different application fields. CityGML 3.0 gives a platform of integrated 3D built environment data and digital twins. 3D city models can be used for urban planning, disaster management, 3D cadastre, energy modelling, environmental analysis and simulation and tourism amongst others. Version 3.0 expands version 2.0 making it more versatile. [5]

KML is an XML language for geographic visualization and annotation of maps and images. The visualization allows control of the navigation. KML is compatible with OGC standards including GML (Geography Markup Language), WFS (Web Feature Service) and WMS (Web Map Service). Version 2.2 employs geometry features from GML 2.1.2. including points, line strings, linear rings, and polygons. [6]

The measurement data is transferred and represented by JSON (JavaScript Object Notation). JSON is a lightweight data-interchange format. It is human friendly, easily parsed and generated. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. The format is language independent, using concepts typical to C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and others. JSON is a data-interchange language. [7]

2.3. Utilities

Containerization of services was achieved using Docker and Kubernetes. Docker is an open-source platform and format for portable, self-reliant containers running in the cloud or on premises. Kubernetes and Docker are basically different technologies that work together for building, delivering, and scaling containerised apps. Kubernetes is open-source orchestration software that provides an API to control containers deployment. It allows running Docker containers and workloads, helping to manage functional complexities in scaling multiple containers, deployed across multiple servers. [8] Docker-compose is a tool for defining and running multi-container Docker applications.

YAML human-readable data-serialization language is used to configure applications' services. The port 5432 is used for internal communication. A Docker image file (3dcitydb/3dcitydb-pg:latest) was used to execute the code in a container. Docker image is a template for the container and in this case relies on PostgreSQL (relational database management system) metadata. Additional code opens HTTP port 8000 for the web browser access and installs some server folders as the container volumes.

3. Tasks and Implementation

3.1. Transformation, Integration and Visualisation of sensor, historical, and geolocation data

There are 23 air quality stations located in the city of Sofia. The static data about stations includes: the site name and type, data of activation, coordinates (also in WGS84 format), data owner and manager, site web link, local authority code and name. The station domes are colour coded – from most polluted red through amber and least polluted green.

Stations provide following measurements:

- NO₂ - Nitrogen dioxide - a gas that is mainly produced during the combustion of fossil fuels.
- CO value - Carbon monoxide - a colourless, odourless, tasteless, and toxic air pollutant.
- O₃ - "ground-level" ozone, this gas is harmful to human health and the environment.
- PM₁₀ - Particulate Matter with a diameter of 10 microns or less.
- PM₂ - Fine particulate matter (PM_{2.5}) air pollutant.
- SO₂ - Sulphur dioxide - a colourless, bad-smelling, toxic gas.
- Nitric Oxide - air pollutant generated by automotive engines and thermal power plants, etc.

Sensor measurements are automatically updated hourly. The program takes measurements in JSON format either from MongoDB database or directly from Air Things API. JSON data is ingested and parsed for each station using a self-adjusting algorithm. Next, the KML file is loaded and parsed, its current values are substituted with new values and KML layer is passed to 3d City Database.

The code is enhanced in comparison to its static predecessor. All the values are taken from API and MongoDB database and dynamically assigned based on site code. The categories of measurements and all details of air quality stations are not hardcoded and KML layer is generated on the fly, adjusting to new stations introduced into the grid and old stations being deactivated. The CESIUM ION visualization platform has a full 3D view of city of Sofia. The map can be scrolled, zoomed, panned and rotated 360°. There are additional options such as changing the terrain layer.

Available terrain layers include:

- Bing Maps Aerial
- Bing Maps Roads
- ESRI World Imagery
- ESRI World Street Map
- ESRI National Geographic
- Open Street Map
- Stamen Watercolour
- Stamen Toner

3.2. Migration to JavaScript and Flask

Changing Keyhole Markup Language layers to pure JavaScript in Cesium visualization is an optimal solution, making the code more transparent and improving the loading times. The variables are adapted from being hardcoded to being created on the flight from MongoDB content. MongoDB is an open-source multiplatform document-oriented database. Classed as a NoSQL database, MongoDB can use JSON format documents with optional schemas.

Flask interface between Python code updating the measurements and frontend web application is implemented. 3d spheres (changed to domes in due course) are created as colour coded objects signifying the air quality stations measurements. Flask is a web application framework written in Python and bridges the Python and JavaScript code. The 3d domes are generated by `viewer.entities.add` call in JavaScript and added to the scene. Height reference is clamped to ground (scenery level). The colour ramp range is between green (less polluted) and red (more polluted).

```
ellipsoid: {
  radii: new Cesium.Cartesian3(300.0, 300.0,
300.0),
  maximumCone: Cesium.Math.PI_OVER_TWO,
  material: Cesium.Color.fromBytes(color, 255-
color, 0, 70),
  outline: true,
  outlineColor: Cesium.Color.BLACK,
  heightReference:
Cesium.HeightReference.CLAMP_TO_GROUND,
},
});
```

A "while" loop is coded for creating the 3d domes continuously from the MongoDB content transferred through Flask. This is an improvement over hardcoded record of Develiot air quality stations. The station objects are now created on the fly based on availability of measurements data. They are added and automatically removed thanks to adaptable algorithm

checking for presence of data output from MongoDB.

4. Integrating 3d building reconstruction

The 3d model of a kindergarten building is loaded from OBJ file into Cesium assets. The upload options chosen determine which asset type is created. Cesium Ion optimizes each asset type for typical usage in 3D geospatial applications. OBJ format is compatible with Cesium platform, it needs only minor adjustments. The OBJ file format is a simple and open format with wide export and import support.

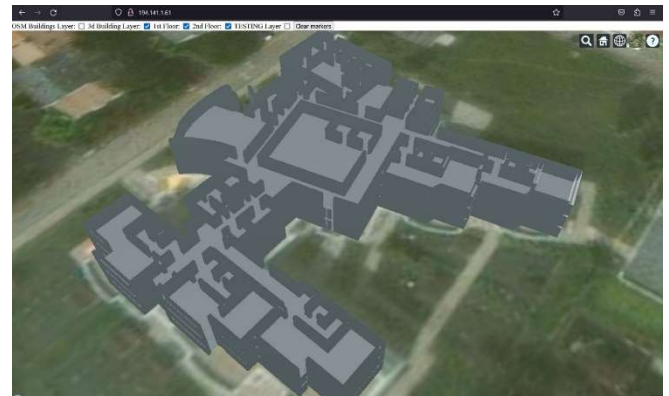


FIGURE 3. 3D BUILDING RECONSTRUCTION

Coordinates and scaling of the 3d model are adjusted. When file is uploaded to Cesium ion, it is tiled into 3D Tiles for 3D content. This is an open specification for streaming massive heterogeneous 3D geospatial datasets. The new asset is integrated into Cesium Ion and treated as a new Cesium resource. The viewer scene camera and canvas are focused on the resource showing it in the centre of the screen. The last step is adding checkboxes to switch floor layers on and off. The toolbar at the top of the screen contains:

- OSM Buildings Layer – shows 3d buildings from Open Street Maps with semantic information across the visualization.
- 3d Building Layer – ground floor of the kindergarten building.
- 1st floor of the kindergarten building
- 2nd floor of the kindergarten building
- Testing layer (for airflow simulation)
- Clear Markers button – removes all active markers on the map.

5. Triangulation

A screen space event handler is implemented. It handles user input events and is used for creating a marker on mouse click at any point in map. First, we get cartographic/cartesian3 coordinates on mouse position and then we add the marker by double clicking. Next stage is calculating the cartesian distance between a marker and all measurement stations. Based on cartesian coordinates, we check all active stations positions, and sort them ascending.

Three stations nearest to the marker are automatically selected. The three stations with shortest distances to

the marker are tagged. Next we calculate an average for three nearest results. Each out of three has the values of carbon monoxide, nitrogen dioxide, ozone, particulate matter 10, particulate matter 2 and sulphur dioxide parsed, and results are averaged between them. A pop-up message with distances and results is shown. The alert with 3 nearest stations, their distances and triangulated values (average from 3 locations) is produced. Finally the label is assigned to the marker. A floating red label is produced showing the results corresponding to the location selected on the map.

6. Data flow and components

We start with taking data from air quality stations (Develiot) distributed across the city of Sofia. Live API sensors stream (AirThings API) is ingested by MongoDB NoSQL database. The data format is JSON (JavaScript Object Notation) and it is parsed and pre-processed by self-adjusting Python code, then transferred through Flask into JavaScript to be visualized in Cesium Ion platform. Cesium is accessible as a web application which is the last step in the development.

Besides the measurements and air quality stations details we incorporate additional layers such as Open Street Maps 3d buildings overlay with the semantic context, 3d building reconstruction from architectural plans (for future internal air quality implementation), testing layer with simulation of airflow around the buildings. Also, there is a functionality to triangulate measurements for any point in map by taking readings from three closest stations and creating a marker.

6.1. Air quality measurements

Air quality stations use sensors for collecting data on pollutants and other environmental parameters such as temperature, humidity, and atmospheric pressure.

These stations typically monitor pollutants such as:

- Carbon Monoxide (CO): A colourless, odourless gas produced by incomplete combustion of carbon-containing fuels.
- Nitrogen Dioxide (NO₂): A gas produced from combustion processes, often emitted from vehicles and industrial sources.
- Ozone (O₃): A secondary pollutant formed from chemical reactions involving pollutants from vehicles and industrial processes.
- Particulate Matter (PM_{2.5} and PM₁₀): Fine particles suspended in the air that can have adverse effects on human health when inhaled.
- Sulphur Dioxide (SO₂): A gas produced by burning fossil fuels, particularly in power plants and industrial processes.

6.2. AirThings API

AirThings is a company that specializes in air quality monitoring solutions. They offer a range of products that help individuals, businesses, and organizations monitor and analyse the air quality. These products typically measure parameters such as radon levels, carbon dioxide (CO₂), humidity, temperature, volatile organic compounds (VOCs), and more.

An API, or Application Programming Interface, is a set of protocols, rules, and tools allowing software applications to communicate with each other. It defines the methods and data structures that developers can use to interact with a specific software component, service, or platform, without needing to understand the inner workings of that component.

The AirThings API contains all commands and methods needed for ingesting sensor and device information and measurements into MongoDB database, either for real-time readings or historical information from the AirThings Cloud. A secure connection to the AirThings Cloud through the API ensures that data is only available to authorized users. [10] Device and sensor data is easily accessed through the REST API. Data is never locked away in proprietary protocols or closed systems.

6.3. MongoDB

MongoDB is an open-source, NoSQL database management system. Unlike traditional relational databases, which store data in structured tables with predefined schemas, MongoDB uses a document-oriented model to store data in a format called BSON (Binary JSON), which is a binary representation of JSON (JavaScript Object Notation) documents with optional schemas.

It is used in a wide range of applications such as web and mobile applications, content management systems, real-time analytics and Internet of Things (IoT) platforms. MongoDB allows storing different types of data in the same collection and scales horizontally, allowing distribution of data across multiple servers to handle large amounts of it. The architecture features in-memory storage and automatic sharding (the dataset is automatically divided into parts where each part, called a shard, holds only a subset of rows, and is stored on a separate machine). [11]

MongoDB supports a powerful query language for retrieving and manipulating data and various types of indexes. An index is a pointer to data in a table and indexing is a powerful tool used in the background of a database to speed up querying, at the cost of additional writes and storage space.

6.4. JSON (JavaScript Object Notation)

JSON is a lightweight and user-friendly data interchange format that is easy to parse and generate. JSON is based on JavaScript programming language, but it is language agnostic. JSON data is represented in a simple text format consisting of key-value pairs. The keys are always strings, the values can be strings, numbers, booleans, arrays, or nested objects.

JSON is used for data exchange between a web server and a web client (browser) in our Cesium visualization. Fetching data from an API, the response is in JSON format. It is used as a storage format for NoSQL databases, including MongoDB.

6.5. Flask

Flask is a micro web framework for building web applications in Python. It is lightweight, flexible, and easy to use. Flask provides the essentials for building web applications, such as routing URLs to functions, handling HTTP requests and responses, and managing templates for rendering dynamic content. In this project, Flask is an intermediary between Python code on the server-side and JavaScript on the client-side. This approach allows to have a front-end (JavaScript, HTML, Cesium) interacting with a back-end (server-side code, MongoDB, Airthings API). Flask, being a web framework for Python, handles the server-side logic and communication, while JavaScript takes care of the client-side interactivity.

Python code in Flask retrieves data from MongoDB database, parses it, and returns it as JSON response. JavaScript consumes this data on the client-side and updates the measurements accordingly. A template is rendered containing placeholders for dynamic content. JavaScript fetches the required data from Flask and inserts it into the HTML templates to update the page.

6.6. Cesium Ion

This is the last stage of data integration and visualization. All types of information are displayed in the frontend web application. The geospatial visualization is shared in a two and three-dimensional context. Pollutant concentrations at different geographical locations are updated hourly and shown as colour coded domes. Clicking on the station gives it's details and current measurements. The code generating the domes is dynamic and adjusts the number of stations according to the contents of MongoDB database, adding new and removing inactive stations.

7. Summary

Various aspects of frontend and backend of Cesium Ion visualization of air pollution have been discussed along with integrating different sources and formats of data.

We have shown the journey of the data from the air quality sensors through various stages of pre-processing, processing, transformation, and correlation to the last phase of visualisation. For each step of this process, the corresponding applications, tools, languages, formats, and utilities have been described.

Main stages of the process have been defined, including georeferenced data acquisition, storing in MongoDB database, consumption by Cesium Ion, visualization of colour gradients or heatmaps of pollutant concentrations, adjusting parameters based on readings values, incorporating additional geospatial and semantic context from Open Street Maps. Users can interact with the data, zoom in/out, rotate the view, and access additional information about specific data points.

The main challenges faced in this project involved creating dynamic JavaScript and maintaining the Python code which updates the readings. This means parsing JSON data, bridging updating hourly measurements and sending them to Cesium Ion platform for visualization. The benefit of this design is

analysis of air quality measurements combined with 3d buildings semantic context in OSM layer. Also, triangulation of readings for any point on the map (marker) is available. This is the last step in the Urban Data Space platform operation where we can see the results on the map.

8. Future work

The CFD (Computational Fluid Dynamics) simulation of the airflow around the buildings is a work in progress. The wind and movement of air masses is directly related to air pollution levels in urban environments. The models are being integrated into testing layer of Cesium visualization and will be overlayed onto OSM 3d buildings layer. The goal is to understand flow conditions around buildings in urban area and answer questions such as heat and pollution transport or urban wind energy exploitation.

CFD simulations provide valuable insights into the behavior of airflow around buildings under different conditions, also helping architects to make informed decisions about creating energy-efficient, comfortable, and sustainable structures. These simulations play a crucial role in architectural design, urban planning, HVAC (Heating, Ventilation, and Air Conditioning) system design, and environmental impact assessment.

The computed wind flow within the context of 3D city models is going to be shown as polygons in a KML layer uploaded as a Cesium Ion asset. [12].

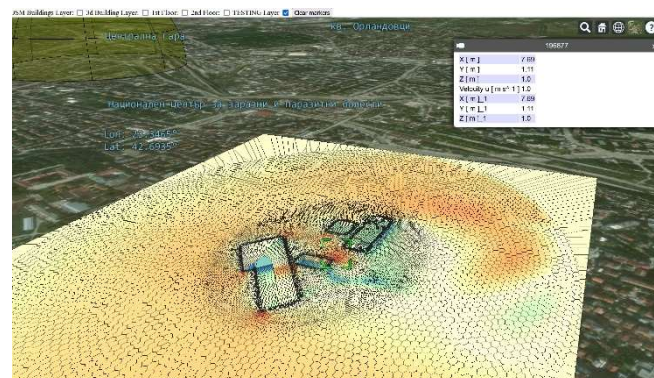


FIGURE 4. INTEGRATING CFD SIMULATION.

Acknowledgements

The work reported here has been conducted with the financial support from UK HEIF Fund and the Bulgarian OPNOIR Programme Grant Agreement No. BG05M2OP001-1.003-0002-C01).

References

- [1] QGIS. (2023). "QGIS - The Leading Open Source Desktop GIS". QGIS Website. [online] Available at: <https://qgis.org/en/site/about/index.html> [Accessed 16 Aug. 2023].
- [2] Technische Universität München. (2023) "The CityGML Database". 3D City DB Website. [online] Available at: <https://www.3dcitydb.org/3dcitydb/> [Accessed 16 Aug. 2023].
- [3] Cesium GS, Inc. (2023). "Cesium ion". CESIUM Website. [online] Available at:

- <https://cesium.com/platform/cesium-ion/> [Accessed 16 Aug. 2023].
- [4] Blender Foundation (2023). "The Freedom to Create" Blender Website. [online] Available at: <https://www.blender.org/about/> [Accessed 16 Aug. 2023].
- [5] Open Geospatial Consortium. (2021) "Public comment requested: CityGML 3.0 Conceptual Model". OGC Website. [online] Available at: <https://www.ogc.org/standards/requests/219> [Accessed 16 Aug. 2023].
- [6] Open Geospatial Consortium. (2023) "KML". OGC Website. [online] Available at: <https://www.ogc.org/standards/kml> [Accessed 17 Aug. 2023].
- [7] JSON.org. (2023). "Introducing JSON". JSON Website. [online] Available at: <https://www.json.org/json-en.html> [Accessed 17 Aug. 2023].
- [8] Microsoft. (2023). "Kubernetes vs. Docker". Microsoft Azure Website. [online] Available at: <https://azure.microsoft.com/en-in/topic/kubernetes-vs-docker/> [Accessed 17 Aug. 2023].
- [9] GeoSLAM (2023). "What is LiDAR and how does it work?". GeoSLAM Website. [online] Available at: <https://geoslam.com/what-is-lidar/> [Accessed 17 Aug. 2023].
- [10] Airthings. (2023) "All of your data in one secure place". Airthings Website. [online] Available at: <https://www.airthings.com/en-gb/business/api#:~:text=The%20Airthings%20API%20contains%20all,information%20from%20the%20Airthings%20Cloud.> [Accessed 17 Aug. 2023].
- [11] Drake, Mark. (2022) "Understanding Database Sharding" DigitalOcean, LLC. [online] Available at: <https://www.digitalocean.com/community/tutorials/understanding-database-sharding> [Accessed 17 Aug. 2023].
- [12] Deininger, Martina E., Maximilian von der Grün, Raul Piepereit, Sven Schneider, Thunyathep Santhanavanich, Volker Coors, and Ursula Voß. 2020. "A Continuous, Semi-Automated Workflow: From 3D City Models with Geometric Optimization and CFD Simulations to Visualization of Wind in an Urban Environment" ISPRS International Journal of Geo-Information 9, no. 11: 657. <https://doi.org/10.3390/ijgi9110657>