OGC® DOCUMENT: 24-067R1

External identifier of this OGC® document: http://www.opengis.net/doc/PER/UDTIP

Open Geospatial Consortium

URBAN DIGITAL TWIN INTEROPERABILITY PILOT REPORT

ENGINEERING REPORT

DRAFT

Submission Date: 2025-04-23
Approval Date: 2025-03-06
Publication Date: 2025-07-10
Editor: Soheil Sabri, Sina Taghavikish

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I OVERVIEW

Current Urban Digital Twins (UDTs) face several challenges. For instance, scaling Digital Twin systems to manage complex urban environments is difficult. It requires integrating multiple systems and data sources into a cohesive model, which is both complex and resource-intensive. Additionally, data interoperability and system integration remain areas that need further exploration. Managing diverse data sources, achieving real-time data processing, and ensuring system scalability and security are all challenging due to the lack of standardized protocols.

The Urban Digital Twin Interoperability Pilot Project (UDTIP) aimed to create a functional UDT ecosystem by developing and integrating various modules. The project focused on addressing challenges related to spatial information interoperability and promoting the portability and reuse of UDT modules. The pilot was divided into several technical objectives, each with specific deliverables, to demonstrate the value of interoperability to external stakeholders. Key components of the project and their deliverables include:

1. Noise Modelling Interoperability (D100): Created a prototype API and workflow for noise simulation within a UDT, integrating IoT sensor data and 3D built environment models. It used CityGML building models and GML street models, and included the conversion of BIM/CAD data to CityGML 2.0, traffic profiles to synthetic noise data, and the direct integration of noise sensor data using the SensorThings API standard. The OpeNoise tool within QGIS was used to evaluate noise levels at different times of day and at various altitudes.

2. Camera Imagery Interoperability (D101): Enabled interoperability of camera imagery for machine learning (ML) workflows. This involved preparing geo-referenced camera images from multiple sensor types, maintaining essential metadata including GeoPose 1.0, and trajectories documented. FFMPEG was used for video frame sampling, and INS metadata was synchronized with camera imagery. INS data was then converted into GeoPose format.

3. Geo-AI Analysis Interoperability (D102): Designed a prototype API and workflow for ML-driven image-based object detection within a UDT. It used input imagery formats and metadata adopted by D101 and utilized the TrainingDML standard for training data and metadata outputs. This included the creation of a TDML-AI pipeline for processing GeoPose data, exploring methods for labeling and annotating images using manual annotation and OSM datasets, and developing a system for classifying road surface types. Machine learning models were applied to the RTK dataset, and the deliverable also designed an API to label and classify road types using OGC API standards.

4. Inter-Module Interoperability (D103): Designed API and OGC standards-enabled data flows between UDT modules. A prototype for data exchange between different UDT modules was developed. OGC API-Features was used for training data and Geo-AI inference results, OGC API-Tiles provided a raster tile data access interface, and OGC API-3D GeoVolumes provided interfaces for accessing 3D data. The use of OGC API Collections was expanded to include

noise simulation and TrainDML imagery. The data server was implemented using Node.js and Express.js.

5. Visualization (D104): Produced a visualization of the project's results to show the value of interoperability. The Cesium JS framework was used to integrate urban noise data from OGC API services. The visualization client combines various geospatial data formats, including CityGML building models, GeoTiff DEM models, 3D noise data, and ground noise data. The client uses a static website architecture with HTML, JavaScript, and CSS.

6. Stakeholder Engagement (D105): Engagemed the community to ensure project outcomes were fit for purpose and aligned with real user needs4. Stakeholders included the Land and Housing Agency of Korea (LH) as sponsors, the UN as a user, and various teams responsible for developing the functionalities as participants. The Open Geospatial Consortium (OGC) coordinated both the sponsors and the participants.

The UDTIP's overall architecture is divided into two sections: one supporting noise analysis and the other supporting object detection/classification using Geo-AI. The project used OGC standards to test and evaluate interoperability between modules, with interfaces for both the noise analysis and Geo-AI systems based on OGC APIs. This project's findings, including the challenges and solutions, contribute to the advancement of urban systems towards the digital twin ideal. The use of OGC standards highlighted their role in enabling interoperability and data sharing across diverse systems.

II EXECUTIVE SUMMARY

This initiative addresses the challenges of data interoperability and multi-system integration within Urban Digital Twins (UDTs). It explores the development and interoperability of various modules within an Urban Digital Twin (UDT) framework, leveraging OGC standards to enable seamless data exchange and communication.

The effort focuses on two key applications: noise modeling and Geo-AI analysis.

For noise modeling, the program utilizes 3D city models in CityGML format, traffic profiles, and noise sensor readings. This data is used to simulate and visualize noise levels in urban environments, supporting urban planning and management decisions aimed at mitigating noise pollution.

For Geo-AI analysis, the undertaking processes camera imagery, INS metadata, and labeled training data. This data is converted into formats such as GeoPose and TrainingDML to support machine learning tasks like object detection (e.g., identifying obstacles or illegal dumping) and road surface classification.

The initiative employs various OGC APIs for data access and processing, ensuring interoperability across modules. Key outcomes include the development of prototype APIs,

workflows, and visualization tools that demonstrate the value of OGC standards in building a robust and interoperable UDT ecosystem.

This project also emphasizes the importance of stakeholder engagement, involving sponsors such as the Land and Housing Agency of Korea and user groups like the United Nations, to ensure the practical relevance of the UDT applications.

Although the current focus is on two applications, the framework is designed to be extensible, allowing for additional or alternative applications in the future and aims to serve as a reference framework for building UDTs using OGC standards.

III KEYWORDS

The following are keywords to be used by search engines and document catalogues.

Urban Digital Twin, Interoperability, OGC Standards, CityGML, GeoPose, TrainDML, OGC APIs, Noise Modeling, Geo-AI, 3D Visualization, Urban Analytics

IV FUTURE OUTLOOK

The Urban Digital Twin Interoperability Pilot Project (UDTIP), with its focus on noise modeling and Geo-AI analysis using OGC standards, has a promising future with broad applications across diverse fields. The successful integration of various modules within the UDT framework lays a strong foundation for expansion into areas critical to smart city development and beyond.

UDTs can provide dynamic insights into urban challenges, including—but not limited to—climate change, urban mobility, and critical infrastructure development. For example, integrating ground-based sensors with satellite imagery can enhance our understanding of the urban heat island (UHI) effects caused by urban intensification. Additionally, real-time modeling, simulation, and predictive analytics can be incorporated into UDTs to improve urban traffic flow and overall mobility within cities and regions. The integration of multi-dimensional data (2D, 3D, and real-time) within UDTs also enables utility providers to better manage both underground and above-ground assets, ultimately improving service delivery to communities.

Urban health applications can also benefit significantly from UDT capabilities. Noise pollution —a key focus of the project—is a known contributor to health issues such as stress, sleep disturbances, and cardiovascular problems. By modeling and visualizing noise levels across urban landscapes, city planners can identify hotspots and implement mitigation strategies, ultimately improving public health outcomes. Furthermore, the project's emphasis on Geo-AI analysis of camera imagery can be extended to monitor environmental factors like air quality, detect potential health hazards such as illegal dumping sites, and assist in crowd management during public health emergencies.

The UDT framework also holds considerable potential for natural disaster preparedness and response. By integrating real-time sensor data from various sources, UDTs can provide dynamic insights into evolving situations during floods, earthquakes, or other disasters. This includes monitoring rising water levels, identifying structurally compromised areas, and tracking the movement of people and emergency response teams. The visualization tools developed through the project can be instrumental in communicating these insights to decision-makers and first responders, enabling more effective and timely interventions.

Moreover, the interoperability aspect of UDTs—facilitated by OGC standards—is particularly relevant for the advancement of autonomous vehicles. Accurate and up-to-date information about the urban environment is essential for safe and efficient navigation. UDTs can provide this information by integrating data from traffic cameras, road sensors, and weather stations. The project's work on Geo-AI analysis, particularly in road surface classification, can further enhance the perception capabilities of autonomous vehicles, allowing them to adapt to varying road conditions and potential hazards.

With its emphasis on open standards and interoperability, the UDTIP sets the stage for future innovations that can transform how we understand, manage, and interact with our increasingly complex urban environments.

V VALUE PROPOSITION

The value proposition of the UDTIP lies in its ability to integrate diverse urban data and functionalities to create a comprehensive and interoperable digital twin ecosystem. The project focuses on developing mechanisms for inter-module interoperability using OGC standards, which allow for the seamless exchange of data, metadata, and code between different modules, promoting portability and reuse across various urban applications. By applying OGC standards, the project enables the integration of various data, including 3D city models and AI-driven analysis. This integration facilitates a holistic understanding of urban environments and supports the development of a unified platform for accessing and managing diverse geospatial data, eliminating the need to retrieve data from multiple sources. This interoperability is crucial for addressing challenges in areas such as noise pollution and object detection in smart cities, providing a robust platform for urban planning, management, and analysis.

1 INTRODUCTION

1 INTRODUCTION

A digital twin is, in essence, a 6D (three spatial axes, one phenomenon time axis, one valid time axis, one "what-if" axis) geospatial model of a portion and aspect of the biophysical-social world, combined with one or more workflows that set or update the objects and attributes of the model. A digital twin is a digital representation of an aspect of the real world that mirrors its counterpart's changes in reality. An Urban Digital Twin (UDT) is thus an approach to understanding the characteristics and processes of a built environment at the scale of a city.

The real-world interfaces (sensors, surveys) may or may not themselves form part of the model. The available output of the digital twin is any information contained within the model itself.

As a technological ideal, digital twins present a number of challenges, particularly in terms of spatial information interoperability. Two challenges have been addressed but not yet solved through OGC initiatives and standardization:

A digital twin requires the integration of persistent information—such as digital models of buildings and urban infrastructure—with dynamic information, such as the trajectories of people and vehicles or environmental properties like noise or air quality. The 3D-IoT Pilot conducted by OGC with support from LH explored several approaches to accomplishing this integration through the use of OGC standards and interoperability architectures. The persistent and dynamic information elements typically derive from different sources, managed by different communities using different systems. Timely integration of these elements is both a technical and semantic challenge—for example, ensuring that a given sensor provides accurate and timely estimates for a particular property of a specific persistent feature, such as a street intersection or building hallway.

It is not feasible for a single digital twin to represent the entire appearance and behavior of the real world. To provide a broader and more accurate representation, it is necessary for individual digital twin models and systems to interoperate—to exchange information with each other as well as with systems that provide sensor observations and/or analytical processing. In order to coordinate these interchanges, the coordination of digital twin systems needs to be based on a common spatial-temporal framework. This framework must align the persistent elements of each twin as well as the sensors and other sources of dynamic information. Issues addressed in OGC Testbeds include APIs for system-to-system interchange of dynamic information, services for discovery across distributed systems, and standardization of training data across multiple machine learning models.

Existing OGC APIs do not directly address all the requirements posed by digital twin interoperability needs. However, the formulation of OGC API elements as "building blocks" presents the opportunity to assemble existing API building blocks into a more appropriate "Digital Twin API" interface specifically directed at those needs. The specification and prototyping of such an interface would be a valuable contribution to the advancement of urban systems toward the digital twin ideal.

1.1. Aims

The aim of the Urban Digital Twin Interoperability Pilot Project (UDTIP) is to demonstrate and improve the interoperability of different modules within an Urban Digital Twin (UDT) ecosystem using OGC standards.

1.2. Objectives

1. **Technical Objective 1: Urban traffic noise modeling to support urban planning and management** The ability to generate predictive noise models based on traffic patterns and/or historical traffic and noise data will enable the use of digital twins in various aspects of urban planning and city management. Relevant applications include the design and management of road networks, placement of traffic control mechanisms, e.g., lights and detours, management of traffic volume by vehicle type, e.g., passenger cars, buses, and trucks, and road pavement typ, e.g., asphalt and concrete, and the planning of locations of buildings and siting of public facilities where noise levels should be considered. Participants working on this deliverable (D100) will produce a design of a prototype API, OGC standards-enabled workflow to facilitate reusable and reproducible execution of a noise simulation integrating IoT sensor data, and 3D built environment models within an Urban Digital Twin. (D100) This deliverable will include documentation of the workflow carried out for the following tasks:

    a) Production of a parametric urban 3D noise model and supporting workflow

        i) The noise model and workflow will incorporate 3D city models, provided by the sponsors, typical of those used in the planning phase of a Smart City

        ii) The noise modeling workflow will include the conversion of estimated traffic profiles to synthetic noise data, providing a mechanism for use of traffic profile data as an input. Samples of data were provided by the sponsors.

    b) Integration of the noise model and 3D city model within a Digital Twin Platform capable of:

        i) Receiving updates to the 3D-built environment model

        ii) Receiving noise levels and additional data at given points

        iii) Communicating with the urban noise analysis module

iv) Providing analysis results to the visualization module described in Technical Objective 4 (D104)

v) Exchanging data and interoperating with the UDT described in Technical Objective 2 (D101/D102) using the mechanism described in Technical Objective 3 (D103)

2. **Technical Objective 2: Detection and identification of unwanted objects (obstacles and unauthorized dumping) and road surface classification in SmartCity contexts** (D101 and D102) The ability to use sensors and cameras to support the management of built-up environments is critical to the ongoing operations of Smart Cities. In the context of cities, it is essential to detect obstacles to mobility. While systems to alert people to the presence of alterations to road vehicle mobility, including the cause, are operational, the detection of unwanted objects acting as obstacles to active transport (walking, jogging, cycling) is not well developed. Equally, the capability to detect unwanted objects, including illegal trash dumps and abandoned objects, is under-developed. The participants on these deliverables will collaborate to develop a prototype system and supporting workflow to:

   a) Enable **Camera Imagery Interoperability (D101)** for training / testing / validation for ML workflows (D101). The prototype system and workflow must allow Geo-Referenced Camera Images produced by multiple sensor types (those typically used to capture images or video of moving vehicles) to be prepared for use in ML feature detection and scene understanding workflows. The multi-source image sets are expected to maintain essential metadata to enable traceability. Essential metadata for this deliverable includes the camera position and Field of View (FoV), as well as the trajectories of the camera, properly documented and aligned with the required spatial-temporal framework.

   b) Enable **Geo-AI Analysis Interoperability (D102)**. Design a prototype API and OGC standards-enabled workflow to enable reusable and reproducible execution of an ML-driven, image-based object detection within an Urban Digital Twin. The focus of the ML training should support the overall technical objective. The ML-driven system should, wherever possible, use the input imagery formats and metadata formulations developed by the D101 participants. The training data and metadata outputs should be provided following the TDML: Training DML for AI Standard.

   c) The outputs of D101 and D102 should be interoperable with the Urban Digital Twin described in Technical Objective 1 (D100) using the mechanism described in Technical Objective 3 (D103)

3. **Technical Objective 3: Inter-module Interoperability** to support portability and reuse for diverse UDT applications **(D103)**. Enabling the creation of Urban Digital Twins for a range of specific applications and promoting portability and reuse of their modules requires development of explicit mechanisms for UDT module

interoperability, enabled by standards. The participants working on this technical objective will collaborate with those on D100, D101 and D102 to deliver:

a) **Inter-module Interoperability (D103)**. Design of a prototype for API- and OGC standards-enabled data flows between Digital Twin modules built for different applications, using the two applications in this pilot as example use cases.

b) The participants working on this deliverable are responsible for developing the functionality to coordinate the exchange of data, metadata, and code as required between the data, analytic, and visualization modules produced for the UDTIP.

c) Participants working on this deliverable are expected to report on any limits of OGC standards for supporting this application within their Report contribution. Planning for generalized UDT interoperability beyond the specific use cases in this Pilot is encouraged.

4. **Technical Objective 4: Communication and Engagement** with the community of practice involved in designing, developing, operating, and using urban digital twins is essential to ensuring the outcomes of the Pilot are fit for purpose, well aligned with real user needs, and have a path to take. The participants working on this technical objective will collaborate with all other participants to:

a) **Produce a Visualization (D104)** of results to convey the value of interoperability to external stakeholders.

b) **Lead Stakeholder Engagement (D105)**. Active engagement throughout the Pilot process is expected with the Land and Housing Agency of Korea, the United Nations Global Service Centre, and the wider stakeholder community. The Participant organizations are expected to gather information on their priorities and requirements, as users, and engage with them through evaluation of prototypes. Communities engaged may include a range of organizations interested in digital technologies for urban planning and management, organizations using urban digital twins, and organizations developing digital twins in other contexts. This engagement could take the form of user needs assessments, paper-based design workshops, prototype reviews, or other activities.

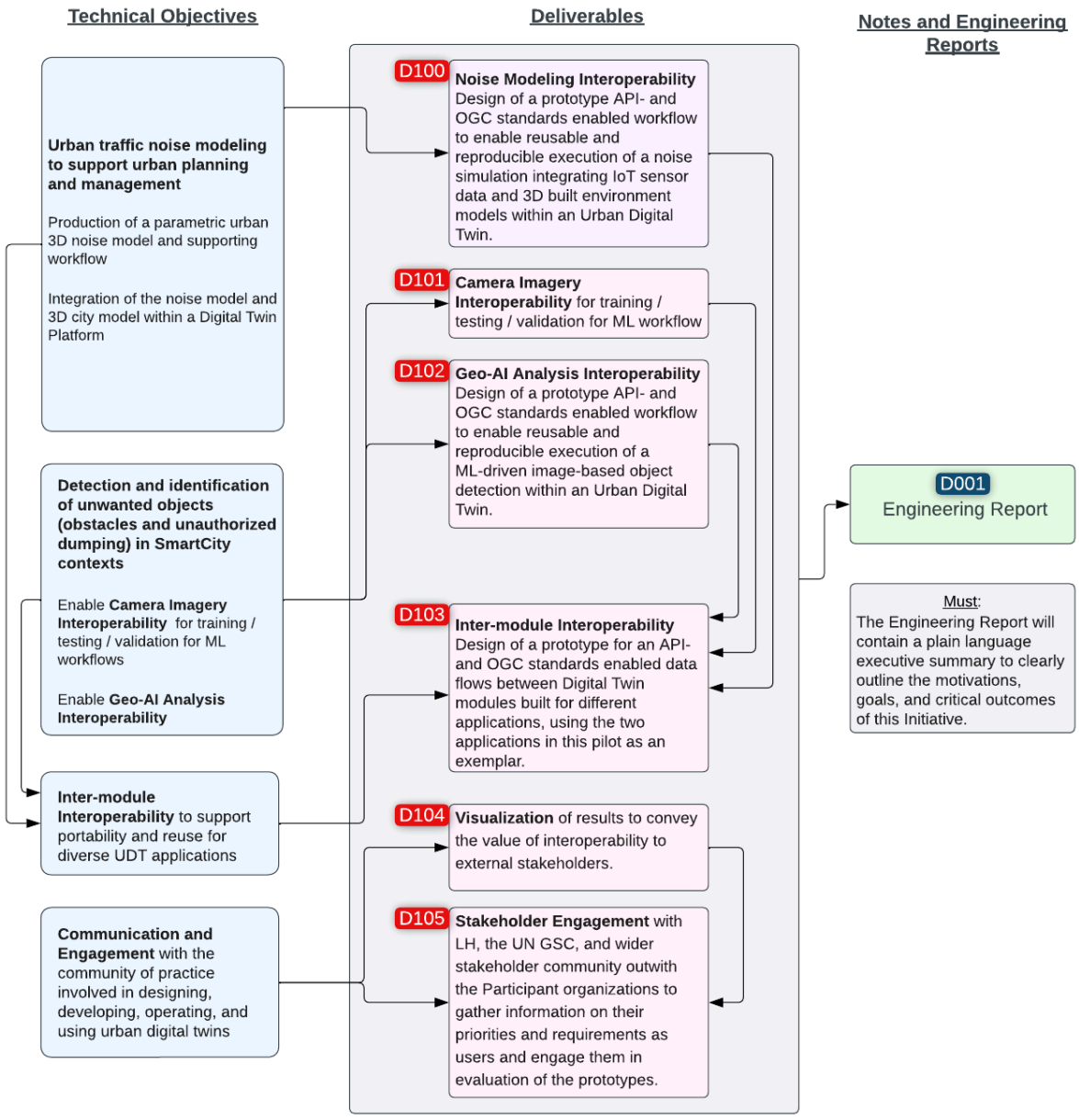

Figure 1 — Technical Objectives and Deliverables

2 TOPICS

2 TOPICS

2.1. Noise-Modeling Interoperability

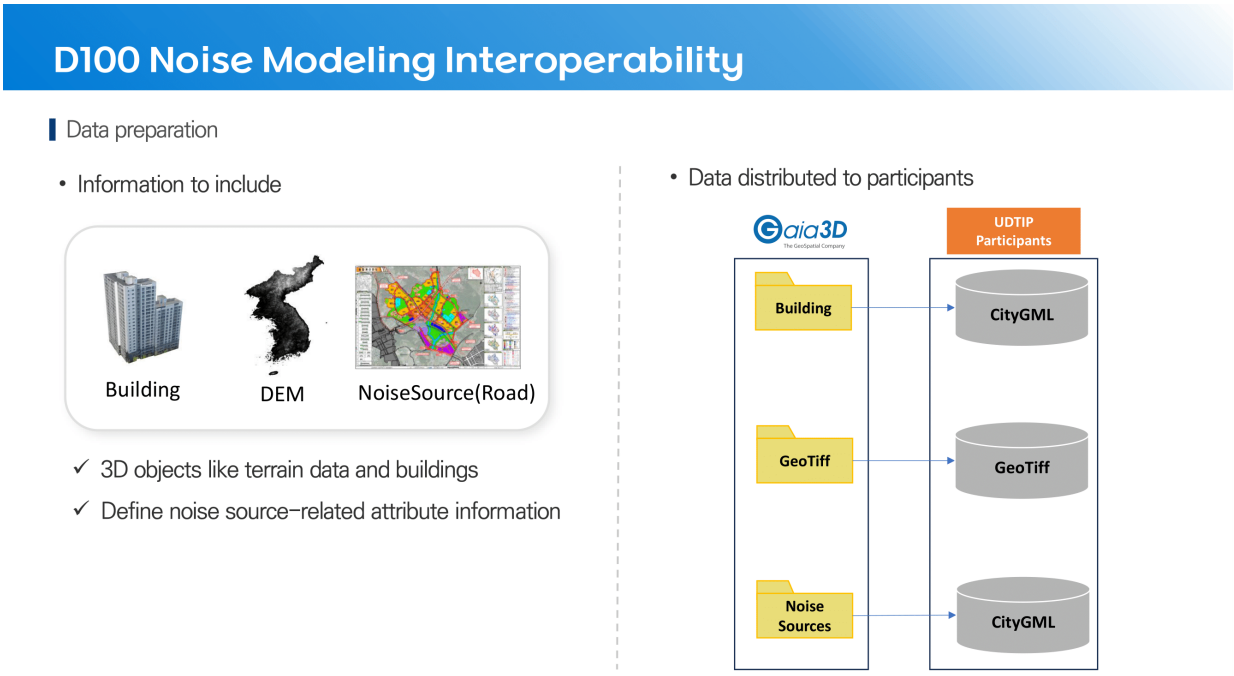

Figure 2 — Workflow for Noise Modeling Data Integration with 3D City Models

The goal of D100 is to integrate information related to noise modeling and define a data structure and format applicable in digital twins. This project provides interoperability through 3D Tiles, enabling bidirectional use of CityGMLdata (including urban planning/design information), DEM, and noise prediction modeler outputs.

The noise modeling interoperability implementation is comprised of three key components:

The advancement of digital twin technology is moving beyond simple 3D visualization of buildings and terrain, requiring the integration of diverse sensor data and analysis results. This report outlines the interoperability framework for applicable OGC standards in noise modeling, preparing CityGML Dataset for noise visualization and illustrating the noise impact on 3D Buildings. Focusing on key components, this section highlights the data inputs and outputs, visualization techniques, and analysis outcomes. This initiative highlights the value of digital twins for urban planning by integrating urban traffic noise prediction analysis with 3D noise impact on buildings.

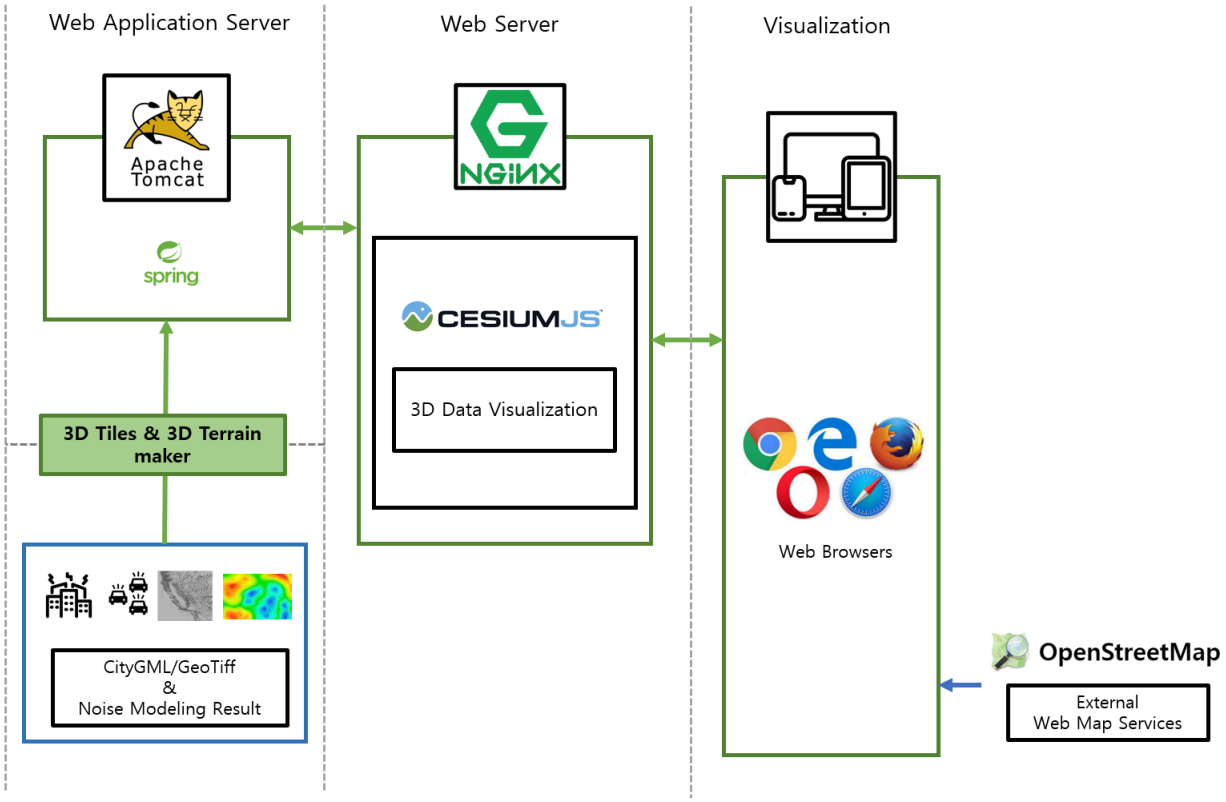

Figure 3 — Software Configuration

2.1.1. CityGML building model

Sample data corresponding to road noise sources, including 3D objects such as buildings and their engineering properties necessary for noise prediction, are provided in CityGML, which features a Level of Detail 1 (LOD1) representation with accurate information on individual building heights. The sample data was generated based on urban planning drawings for the planning phase for Wangsuk District 2 in Namyangju, Gyeonggi-do, South Korea, a government-approved urban development project. Each building is modeled to its correct absolute height, referencing a Digital Elevation Model (DEM) also produced by Gaia3D. Before its use, the CityGML building model undergoes validation for geometrical accuracy using CityDoctor, a free software tool designed for quality checks of 3D city models in CityGML format.

2.1.2. GML street model

The GML street model consists of polyline geometries representing road segments. Along with the road segment geometries, the model includes relevant properties necessary for calculating noise levels for each segment using the OpeNoise Map plugin in QGIS. These properties include: the number of lanes, number of small vehicles per hour at day, number of large vehicles per hour at day, number of small vehicles per hour at night, number of large vehicles per hour at night, speed of small vehicles at day, speed of large vehicles at day, speed of small vehicles at night,

speed of large vehicles at night, surface status code, applied additional noise reduction method and noise reduction effect by the method.

2.1.3. Noise Modeling

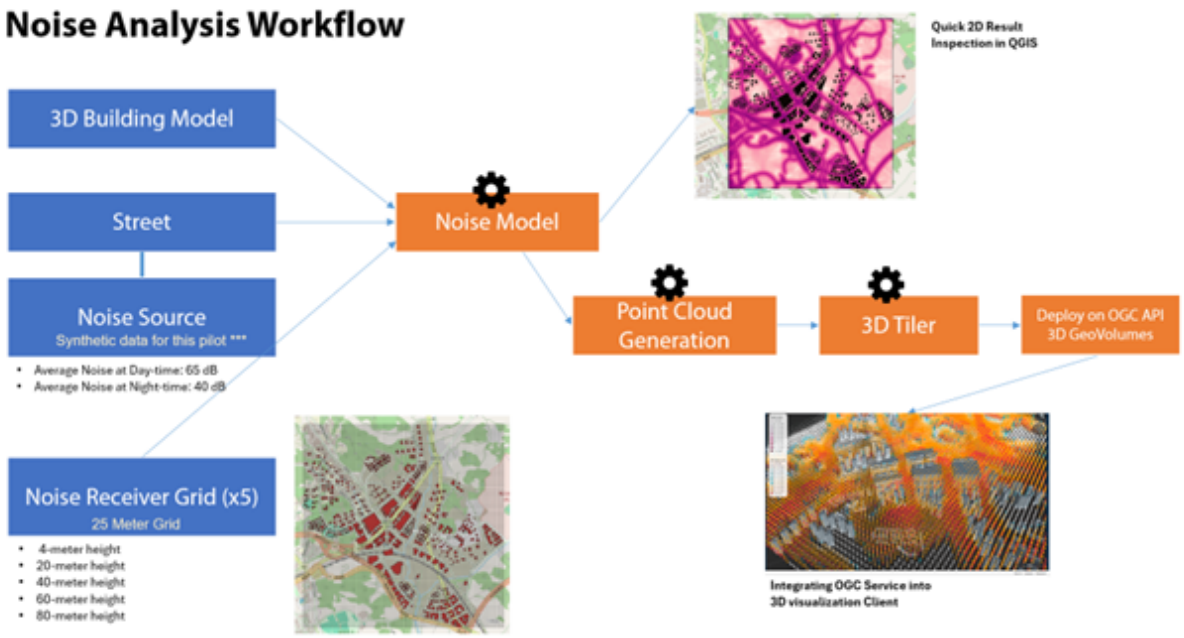

Figure 4 — Noise Analysis Workflow

The noise modeling was primarily conducted using data from the CityGML building model and the GML street model. These datasets provided the essential spatial information to develop a foundational noise propagation model using the OpeNoise tool. The OpeNoise model as integrated in QGIS was employed to evaluate noise levels across two times of day: day and night. By leveraging the CityGML city model for accurate spatial representation and the road network data, the OpeNoise tool was able to compute noise levels effectively for both day and night conditions. In this model, synthetic data was used with the assumption that the noise source from all road surfaces is the same. In this model, synthetic data was used with the assumption that the noise source from all road surfaces is uniform. We approximated a noise level of 65 dB during the day and 40 dB at night. This estimation was used as a stand-in for actual sensor data, which will be integrated in future work. It is indeed possible to incorporate real noise sensor data by feeding actual measurements into the model and interpolating them from the street polygons (or other noise source geometry). In the current work, we had no real sensor data available, so we consulted literature to apply uniform noise source values—one for daytime and one for nighttime. However, the noise model is capable of much more detailed simulations by integrating accurate traffic data, varying noise source intensities, and real-time sensor inputs, among other possible refinements. The noise levels were computed by incorporating environmental factors into the model. Key inputs for the noise model included synthetic noise levels for roads, assuming uniform noise sources. The tool utilized these inputs to simulate the propagation of noise from roads to the surrounding urban environment, factoring in reflections, diffractions, and absorption by building facades as represented in the CityGML model.

The noise modeling used data from the CityGML and GML models to simulate noise propagation. The OpeNoise tool in QGIS was employed to calculate noise levels, incorporating environmental factors like reflections, diffractions, and absorptions. Noise was assessed at ground level and elevations of 20m, 40m, 60m, and 80m above the terrain.

It is possible to perform noise calculations per floor level by leveraging additional attributes available in the CityGML dataset, such as storeysAboveGround and measuredHeight. These attributes make it feasible to calculate the height of each floor and generate a noise receiver grid for every level of a building. However, because this process increases the complexity and duration of data preparation, a simplified approach was used by calculating noise at specific height levels rather than per floor.

Noise levels for roads were approximated at 65 dB during the day and 40 dB at night. Synthetic data was used as a placeholder for actual sensor readings. Future plans include integrating real sensor data for improved accuracy. It is indeed possible to incorporate real noise sensor data by feeding actual measurements into the model and interpolating them from the street polygons (or other noise source geometry).

2.1.4. Visualization

Visualization plays a critical role in understanding noise distribution and supporting urban planning. The following techniques and tools were used:

1. Cesium.js An open-source rendering engine was employed to visualize:
    - 3D Tiles for building data
    - Quantized Mesh for terrain data, chosen for its high-resolution capability
    - Noise analysis results as independent 3D objects
2. Data Conversion Custom modules were developed for converting:
    - CityGML to 3D Tiles
    - GeoTiff terrain data to Quantized Mesh
    - Noise Modeler outputs to 3D Tiles

Figure 5 — Quantized Mesh Visualized in Cesium

2.1.5. Analysis Results

The analysis results were visualized as independent layers, enhancing clarity and accessibility. Key features include:

1. Noise Propagation
   Noise meshes were generated to represent spatial and temporal variations. These meshes were developed to ensure interoperability across platforms, leveraging the 3D Tiles format for seamless integration with digital twin environments.

2. Legend Configuration
   The legend categorizes noise levels for intuitive understanding, configured based on criteria detailed in the project documentation to represent various noise thresholds effectively.

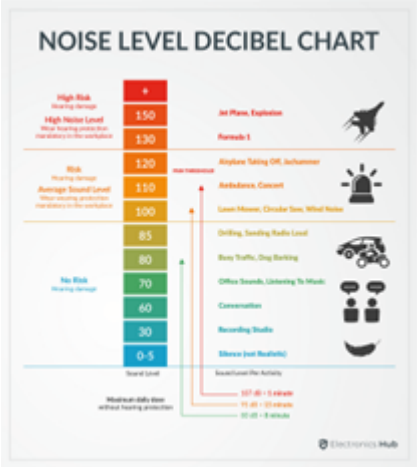

Figure 6 — Noise Analysis Legend

2.1.6. Conclusion

This holistic approach integrates advanced visualization and modeling techniques, supporting urban planning and design by providing meaningful analysis into noise propagation in urban environments.

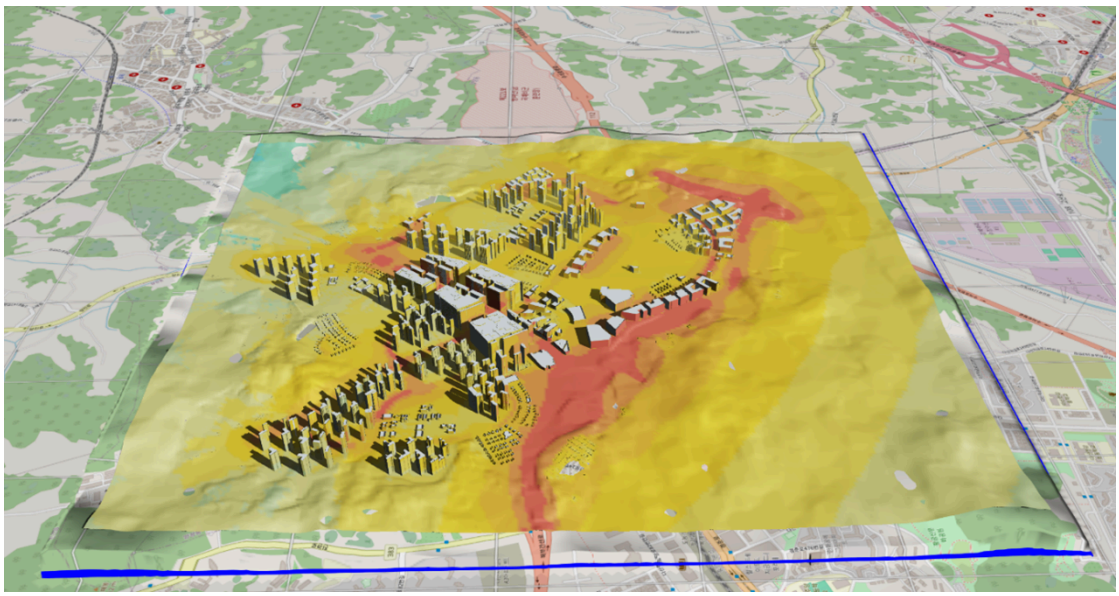

Figure 7 — Noise Analysis Results on OSM in Cesium

2.2. Camera Imagery Interoperability

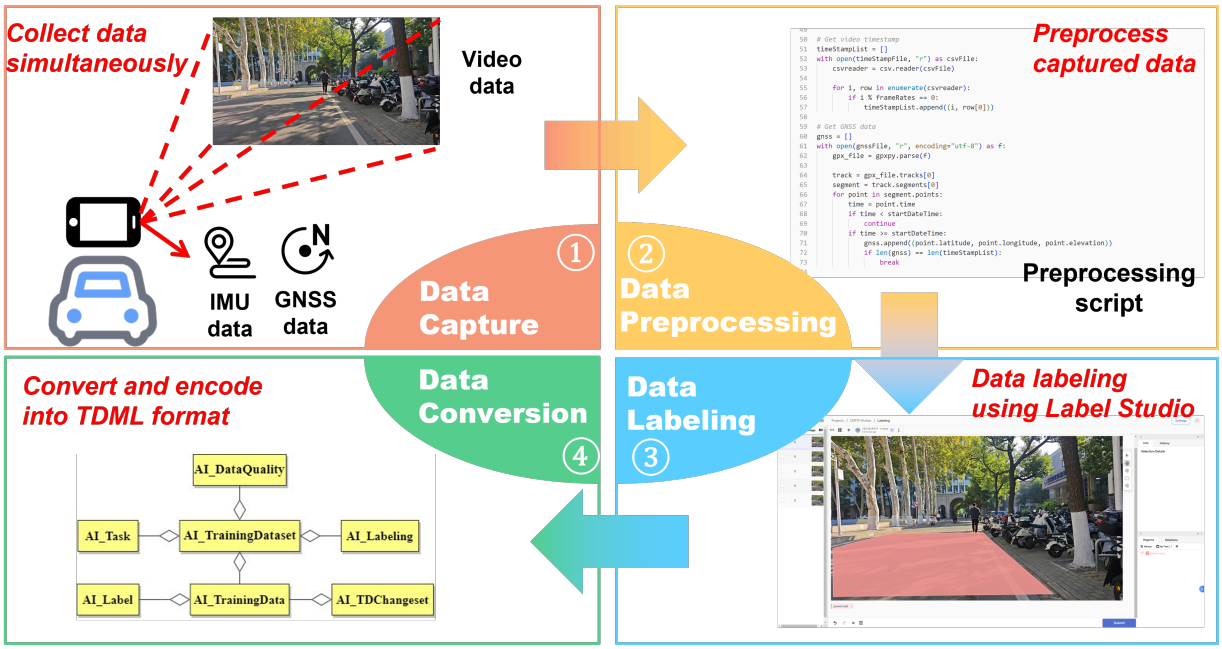

Figure 8 — Data Collection and Preprocessing Workflow

Open-source applications are utilized on an in-vehicle mobile phone to synchronously collect video, IMU, and GNSS data. The collected data, after preprocessing and annotation, can be converted into TDML format to enhance interoperability with other modules.

Camera imagery interoperability is centered on establishing interoperability between camera imagery and Inertial Navigation System (INS) metadata, which includes position and orientation information in relation to a specific frame of reference. This integration is pivotal for merging visual data with spatial positioning, thereby enhancing the accuracy and utility of urban digital twin systems. This deliverable seeks to bridge the gap between disparate geographical data sources and create a cohesive and actionable dataset. The resultant integration of camera imagery and INS metadata as GeoPose facilitates a higher degree of interoperability. The structured approach includes comprehensive data collection, precise sampling, meticulous processing, synchronization, and conversion, which culminates in a thorough validation of results. This then supports the D102 deliverable, which aims to leverage the processed data for advanced Geo-AI analysis.

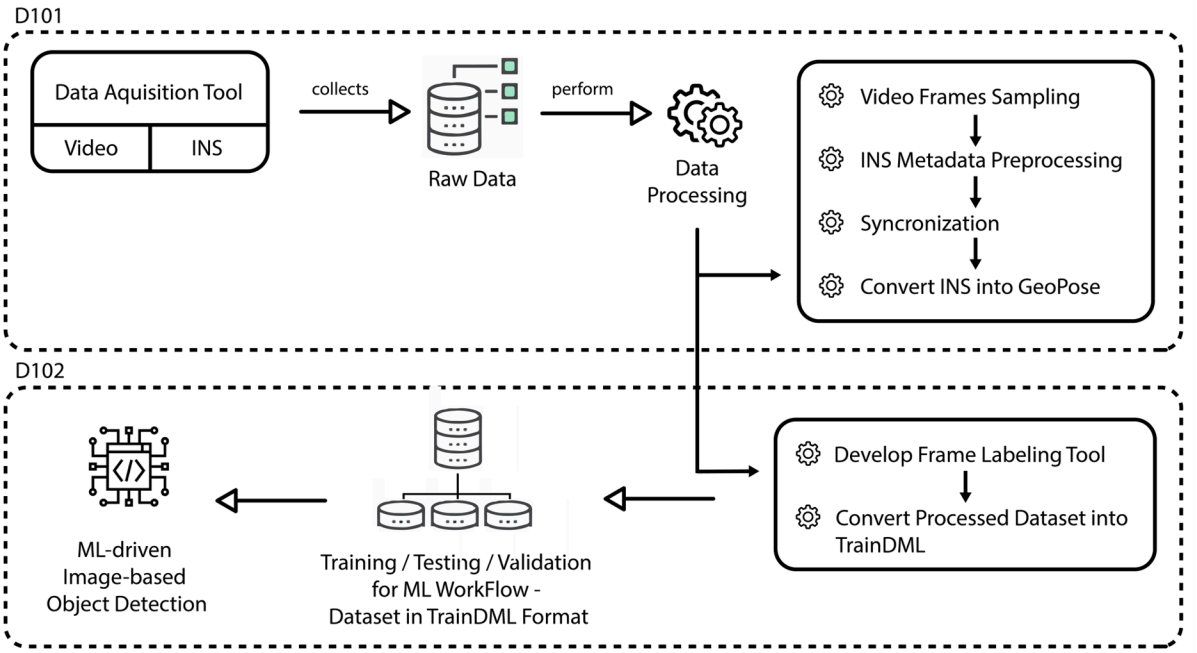

Figure 9 — D101 to D102 Workflow

Figure 9 is a strategic outline detailing the integration of Camera Imagery Interoperability (D101) and Geo-AI Analysis Interoperability (D102) within Urban Digital Twin systems, enabling seamless data processing, synchronization, and machine learning-based object detection for smart city applications.

2.2.1. Data Acquisition

The goal of data acquisition is to capture high-quality imagery and sensor data with GeoPose information to generate geo-referenced training datasets. This process was implemented using an on-board mobile phone and open-source software, ensuring transparency and flexibility.

Imagery was collected using smartphone cameras in Red-Green-Blue (RGB) format at varying resolutions—1080×1080 pixels at 1.0 Hz and 1920×1080 pixels at 30 Hz—ensuring both detailed accuracy and real-time capture. To enhance spatial and positional precision, additional sensor data was collected, including:

- **Global Navigation Satellite System (GNSS) Data:** Provided latitude, longitude, and altitude for precise georeferencing.
- **Inertial Measurement Unit (IMU) Data:** Included gravimeter, magnetometer, and accelerometer readings, supporting both Yaw-Pitch-Roll (YPR) and quaternion encoding schemes for orientation tracking.
- **Mobile Phone Recording Data:** Utilized for timestamp synchronization through screen recordings.

Data acquisition was conducted using Sensor Logger, an open-source cross-platform application available for both Android and iOS platforms, ensuring transparency and flexibility for customization. Detailed measurement units and their interpretations can be found here: Sensor Logger Units Reference.

Pre-existing datasets were also integrated to enhance the analysis, including:

- **HillyFields Bubble Dataset:** Offers front and rear video footage from StreetDrone (car) with precise INS metadata for environmental and spatial analysis.
- **KITTI-360 Dataset:** A widely utilized dataset in autonomous driving research, comprising over 320,000 images and sensor data for tasks such as object detection and 3D tracking.

Field tests were conducted on open roads in Wuhan to validate the workflow and data acquisition methods. For detailed description and specification of used tools and dataset, refer to the annexes of UCF and WHU.

2.2.2. Data Processing

The FFMPEG, a popular open-source command-line toolbox, was employed as a cross-platform toolbox for frame extraction to sample frames at specific intervals to balance detail and processing load.

2.2.3. Video Frames Sampling

FFMPEG was leveraged to manipulate, convert, and stream multimedia content. FFMPEG was specifically employed for video frame sampling, extracting frames from a video at regular intervals or a predefined frame rate.

This extracted video frames at a specific rate to create training datasets, analyze video content, and synchronize with data sources such as INS metadata.

2.2.4. INS Metadata Preprocessing

Data cleaning and filtering eliminates noise and irrelevant information, ensuring compatibility with GeoPose conversion. Key steps include:

- Performing feature extraction by selecting and transforming columns as needed
- Addressing missing data by implementing strategies to manage gaps and inconsistencies in the metadata
- Converting epoch time to date-time format
- Calibrating the frequencies of location and orientation data
- Down sampling to eliminate redundant data

This enhances the processing efficiency and prevents overfitting during the training phase of deliverable D102.

Location and orientation data were synchronized with camera imagery using timestamps, and GPS time has been incorporated in the generation of the resultant GeoPose.

2.2.5. Synchronization

Camera imagery was aligned with the INS metadata by referencing timestamps to ensure precise synchronization between video and metadata. Each frame instance was represented as a georeferenced image, linking the image with its corresponding positional and orientation data from the INS system. For a detailed explanation with visual representation, refer to the technical specifications.

2.2.6. Convert INS data into GeoPose

The refined INS metadata was converted via script into the GeoPose standard in JSON format, specifically adhering to the GeoPose.Composite.Sequence.Series.Regular standardization target. This conversion employs Latitude (deg), Longitude (deg), and Altitude (m) for positional encoding and supports both Yaw, Pitch, and Roll (YPR) and quaternions encoding scheme for measuring orientation.

2.2.6.1. Data Preprocessing

The collected data need to be preprocessed before being fed into the subsequent process. The purpose of data preprocessing is to temporally align the collected data, extract frames from videos, and convert raw data into the required format.

The workflow of data preprocessing is shown in Figure 10. It contains three main stages: frame extraction, IMU conversion and frame match. The result of the preprocessing is that the data collected by different software will be sampled to a frequency of 1 Hz and provide an unlabeled dataset with geopose data in the TrainingDML-AI format.

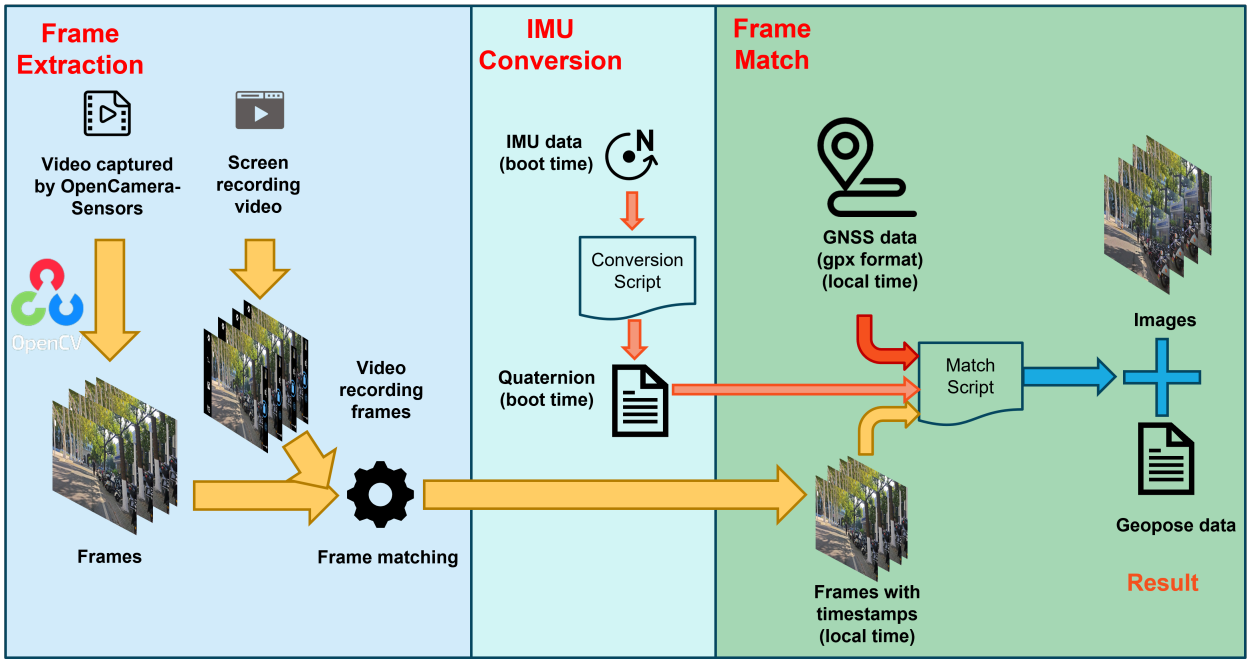

Figure 10 — Workflow of Data Preprocessing

2.2.6.1.1. Frame Extraction

The temporal synchronization pipeline addresses timestamp absence in primary video streams captured by OpenCameraSensors through auxiliary screen recordings. The workflow implements cross-modal frame alignment as follows:

Synchronization Protocol

1. Frame Extraction
    - Decode primary (sensor) and auxiliary (screen) videos via OpenCV
    - Maintain native resolution: Primary @ 30Hz (33ms/frame), Auxiliary @ system refresh rate
2. Timestamp Anchoring
    - Manual visual matching: Identify identical frames across video sources
    - Apply temporal transfer: Assign auxiliary frame's system-clock timestamp to corresponding primary frame
    - Calculate sub-frame precision: Leverage 30Hz sampling to resolve timestamps with ±16ms accuracy
3. Output Preparation

- Preserve traceability: Log frame-source relationships in metadata

2.2.6.1.2. IMU Conversion

The purpose of IMU conversion is to convert the collected IMU data into geopose. The IMU data collected by the OpenCameraSensors includes the accelerometer data, gyroscope data, and magnetometer data, each with a timestamp. Note that the three data don't have the same frequency and start time, so they should be pre-processed using algorithms such as the Kalman filter, converted to 30Hz geopose data for subsequent processing. As it is not the main research focus of this pilot, the algorithm for converting IMU data to geopose will not be elaborated here. The converted data is encoded in xml format, using x, y, z, w to represent the pose of the camera, each pose records a timestamp of the boot time since the mobile phone was booted.

2.2.6.1.3. Frame Match

This phase focuses on temporal frame synchronization and dataset preparation for subsequent annotation. The primary deliverables include georeferenced image sequences with synchronized GeoPose data. The workflow comprises the following key operations:

1. Time Synchronization
    - Manual calibration of the time offset between device boot time and local time
    - Automatic timestamp adjustment for IMU data using application-generated frame timestamps
2. Data Sampling
    - Downsampling all sensor streams to 1Hz through temporal alignment
    - Selection criteria: Frames and GeoPose data points temporally closest to GNSS sampling points
3. Output Configuration
    - Final dataset frequency locked to 1Hz (determined by GNSS sensor's sampling rate)
    - Modular GNSS hardware support enabling potential upgrades for higher-frequency collection

The preprocessed output follows the TrainingDML-AI specification, structured as:

- Frame sequences organized chronologically

- JSON-formatted metadata containing:
  - System identification parameters
  - Asset descriptors (name/quantity)
  - GeoPose coordinates (Basic-Quaternion format)
- Extensible TDML architecture allowing label integration during annotation

2.2.6.2. Data Labeling

In annotation stage, Label Studio, an open-source labeling tool, is used to enable collaborative labeling of data. The annotation can be exported in JSON format, so that it can be converted into TDML format.

2.2.7. TDML Format Conversion

Training data must be converted into the TDML-AI format to ensure interoperability between modules. In accordance with D101 specifications, GeoPose data has been integrated as an extension within the TDML encoding framework. A dedicated Python script has been developed to systematically transform metadata, GeoPose data, and label information into the standardized TDML format. The conversion process requires:

1. Metadata and label data to be converted into JSON format in compliance with the OGC TDML-AI Part 1 standard
2. GeoPose data to be formatted using the Basic-Quaternion representation, aligned with the OGC GeoPose 1.0 Data Exchange Standard
3. All converted GeoPose data to be stored alongside corresponding image metadata

2.3. Geo-AI Analysis Interoperability

2.3.1. TDML-AI Pipeline Creation and GeoPose Standard Enhancement

GeoPose data, based on video and sensor data, provides a structured representation of the position and orientation of each video frame in space. However, this data must often be converted into formats such as TDML-AI for broader use in machine learning (ML) models and AI training pipelines. The project aimed to streamline the process of converting and labeling GeoPose data to enhance the accuracy of geospatial data labeling and metadata creation.

Various methods were explored for labeling and annotating images that were extracted from video component of the source dataset. This was broken into two parts:

1. Create a proof-of-concept for labeling images
2. Manually annotate road surfaces within images

The produced proof-of-concept demonstrates that it may not be feasible to rely on OSM APIs for frequent, consecutive data retrieval due to rate limits. Although downloaded OSM datasets were a valuable alternative, the process to use them could benefit from automated acquisition and processing.

For detailed steps for the methodology, see the pipeline methodology in the technical annex.

2.3.1.1. TDML-AI File Creation

GeoPose data is converted to the TDML-AI format by reviewing OGC documentation, aggregating internal and exported annotation data, synthesizing it into a TDML-AI dataset, and validating compliance with OGC requirements.

2.3.1.2. Working with GeoPose Data

Latitude and longitude information had been extracted GeoPose data previously, but this process is cumbersome due to the data's encoding structure.

2.3.1.2.1. Manual Image Annotation

Bounding boxes were drawn around specific objects such as roads and grass, and annotations were exported in COCO format, which supported both bounding boxes and precise segmentation.

2.3.1.2.2. Performance Testing on GeoPose Metadata Extraction

Both array-based encoding (strategy 1) and object encoding (strategy 2) significantly outperformed the current method of encoding parameters as string in the GeoPose standard. They were roughly 4-7 times faster in extracting latitude and longitude data across various test configurations.

2.3.2. Image Machine Learning to Classify Road Surface Types

The automatic classification of road types and conditions is a critical task due to the limitations of human-involved solutions, which are not only costly but also time-intensive. Image-based machine learning approaches present a promising alternative by automating the identification of road types efficiently. For any given image of a road surface, a classification model can be

designed to predict the surface type, offering a streamlined and scalable solution for road condition analysis.

This classification task represents a typical supervised learning problem. Due to an initial lack of a pre-prepared, labeled dataset tailored to the project's need, as an interim measure, a public dataset was used to facilitate the development of the classification model until project-specific data could be collected. To evaluate potential solutions, various machine learning models were considered, with ResNet50 and InceptionV3 ultimately selected for their widespread use and demonstrated effectiveness in similar tasks.

2.3.2.1. Working with Public Datasets

The project began with the Road Traversing Knowledge (RTK) dataset, a public resource designed for road surface classification using images captured by low-cost cameras. The dataset includes approximately 43 minutes of video footage, equivalent to 77,547 image frames, collected in a developing country context. These images represent three road surface types: asphalt, paved, and unpaved, each of which is further subdivided into two or three condition-specific classes.

To prepare the RTK dataset for modeling, images were divided into training (80%, 29,283 images), validation (10%, 6,066 images), and test (10%, 6,066 images) sets. Preprocessing steps included cropping the bottom two-thirds of each image to focus on road surfaces and resizing images to meet the input requirements for the selected models (224×224 pixels for ResNet50 and 299×299 pixels for InceptionV3).

Experimental results highlighted the strengths and limitations of the models. ResNet50 demonstrated near-perfect accuracy in identifying the asphalt class and high accuracy for the paved class. However, it struggled with distinguishing between paved and unpaved surfaces, an issue that stemmed from the visual similarity between these classes in the RTK dataset. InceptionV3 exhibited similar trends, reinforcing the difficulty of differentiating paved and unpaved surfaces using the given dataset.

2.3.2.2. Observations from Initial Work

The classification of Asphalt surfaces was highly accurate, but significant errors occurred in detecting the Unpaved class due to the overlap in appearance between Paved and Unpaved images within the RTK dataset. These challenges underscored the need for careful preparation of datasets and clear definitions of road surface types to improve model performance. For practical applications, such as those supporting United Nations missions, datasets had to be personally collected to better reflect real-world conditions.

The datasets from the UN Global Service Center revealed notable differences compared to the RTK images, particularly in the Unpaved class, which encompassed a broader range of surfaces such as gravel, rock, dirt, and sand. Consequently, the project adopted a three-class definition—Asphalt, Paved (e.g., cement), and Unpaved (including all other surfaces)—to better align with the variability observed in real-world road conditions.

2.3.3. Interoperability Framework Design

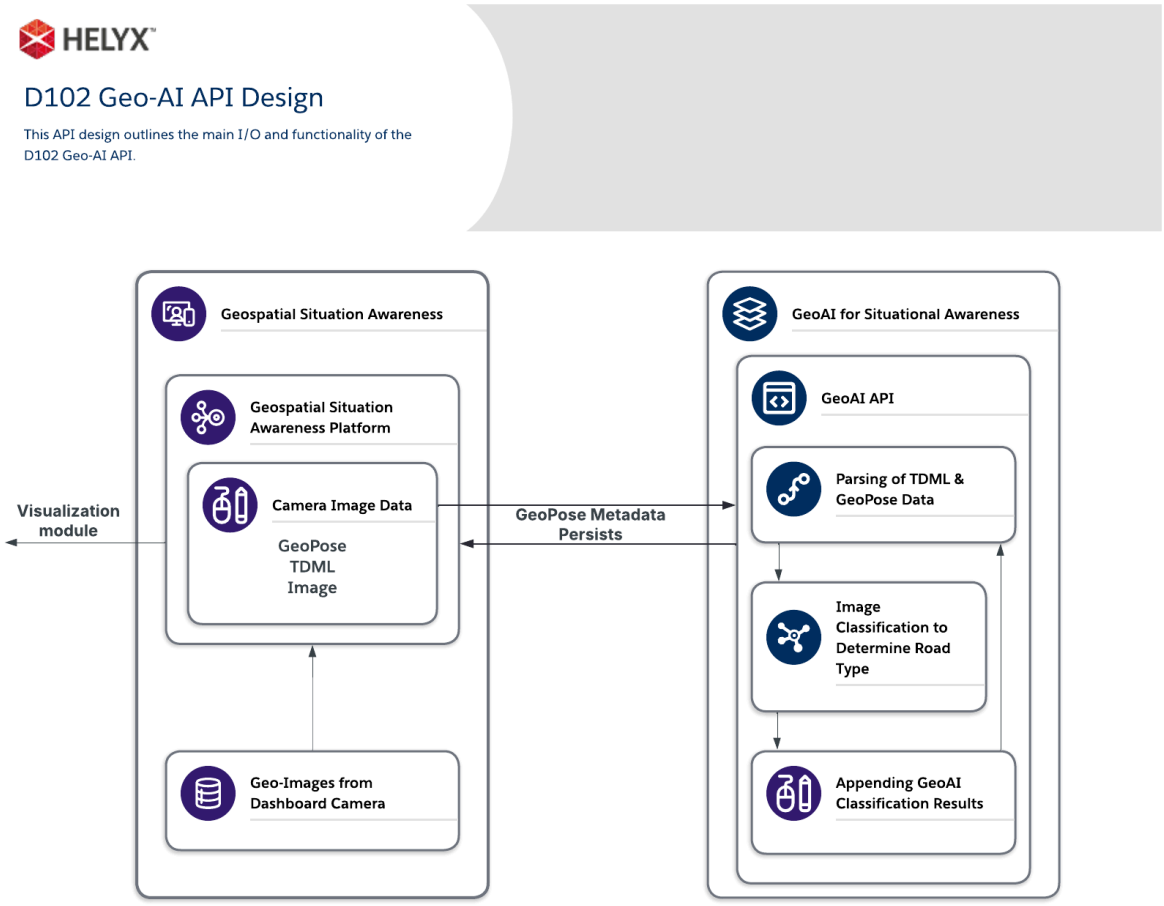

Figure 11 — Geo-AI API Design

The Geospatial Situation Awareness Platform begins with dashboard camera images that capture the surrounding environment, incorporating GeoPose, TDML, and image metadata. The GeoAI API pipeline processes this data by extracting spatial information, classifying road types using machine learning models, and appending GeoPose metadata for persistence. Finally, the processed data is integrated into the visualization module, enhancing geospatial awareness and supporting informed decision-making.

To extend the current road type classification system and address existing challenges, the development of a Geo-AI analysis interoperability framework takes priority. This effort, see the Helyx API structure, involves designing a prototype API and enabling workflows compliant with OGC standards. The framework creates a reusable and reproducible machine learning (ML)-driven object detection system integrated within an Urban Digital Twin. By aligning ML training objectives with overarching technical goals, this system leverages input imagery formats and metadata developed by D101 participants. Additionally, the output training data and metadata follow the Training DML standard, ensuring consistency and compatibility across applications.

2.3.3.1. Requirements Analysis

The API focuses on providing flexible and reusable capabilities within the OGC API Standards framework. It enables the labeling and classification of road types and conditions based on supplied input images. Leveraging OGC API Processes, the API streamlines workflows for pre-processing, image classification, and post-processing of data. Comprehensive documentation of steps and parameters forms an integral part of the API design, ensuring transparency and reproducibility in its operation. To minimize discrepancies, a stable process ensures consistent results for identical data inputs across multiple executions. Furthermore, the integration of GeoPose and Training DML data collection and transmission ensures continuity and alignment throughout the classification process.

2.3.3.2. Technical Design

To promote reproducibility and cross-platform functionality, development takes place within Docker containers. These containers host the required environment, with dependencies such as Torchvision, Flask, and Swagger installed and managed through Dockerfiles. This approach facilitates an automated and repeatable deployment process suitable for both testing and production environments.

The API supports pre-trained models to ensure seamless classification of input data without requiring real-time model training. Collected training data, formatted in TDML, converts into PyTorch datasets to support model training and evaluation. This architecture emphasizes modularity, enabling the integration of new datasets or models without disrupting the system's foundational design.

2.3.3.3. Deployment Considerations

The API deploys in an Azure-hosted environment with exposed endpoints accessible to other components within the Urban Digital Twin project. These endpoints facilitate seamless interaction between systems, enhancing collaboration and interoperability. To optimize data handling and minimize bandwidth demands, the system explores using Amazon S3 buckets for storing temporary or returned data and images. This approach balances efficiency with scalability, ensuring robust and accessible system performance across diverse user scenarios.

2.3.3.4. API Design/Endpoint Overview

Geospatial Situation Awareness Platform:

- Begins with Images from dashboard cameras that capture the surrounding environment.
- This data, including GeoPose, TDML, and image metadata, flows into the GeoAI API pipeline.

GeoAI API Pipeline:

- **Parsing TDML & GeoPose Data:** Extracts spatial information and associated image metadata necessary for further processing.
- **Image Classification:** Determines the road type (e.g., asphalt, concrete, unpaved) using machine learning models, as shown in the API's image classification capabilities.
- **Appending GeoAI Classification Results:** The classified data, along with GeoPose metadata, is appended and persists for further use.

Output & Visualization:

- The processed GeoPose and road classification data are made available to the visualization module, enhancing geospatial awareness and decision-making.

2.4. Inter-module Interoperability

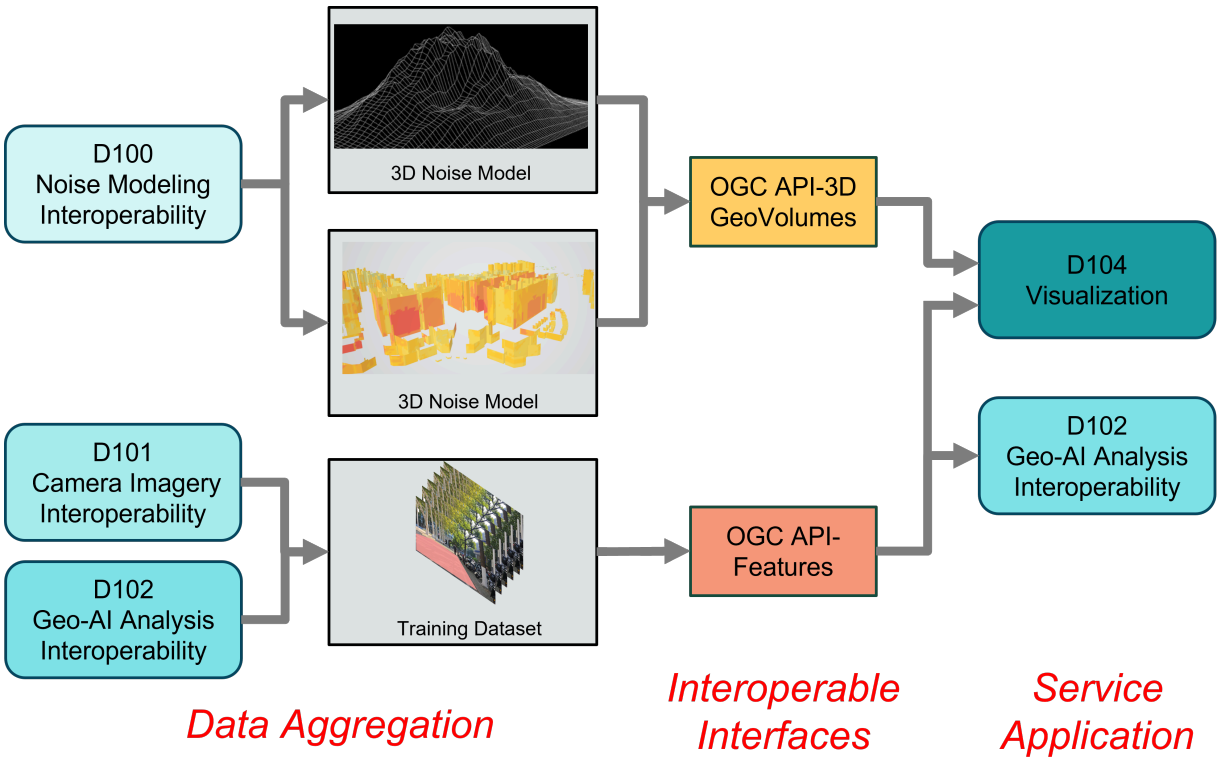

Figure 12 — Inter-module Interoperability Workflow

Multiple OGC API-compliant data interfaces based on the deliverables' data provide interactive services for the deliverables.

2.4.1. OGC API-Features

Deliverable D103 provides both training data/datasets interfaces and Geo-AI inference result interfaces conforming to the OGC API-Features, which provides access to the training dataset through network for D102 model training and testing, and D104 visualization.

Specifically, the endpoint of OGC API-Features provides web access to training data and labels. The format of the training dataset is encoded according to the encoding schema of the TDML standard, and the GeoPose data is expanded into the meta-information of the image data. At the same time, two data management methods are provided: data bypass and storing data in local server.

2.4.2. OGC API-Tiles

A raster tile data access interface conforming to the OGC API-Tiles is provided to D104 for ground noise data visualization.

2.4.3. OGC API-3D GeoVolumes

The D103 provides interfaces to access 3D data conforming to OGC API-3D GeoVolumes for D104 visualization. Formats of accessible 3D data include 3D Terrain, 3D Building Model and GLB format data with geo references.

2.4.4. Interoperability Implementation

The inter-module interoperability implementation utilizes OGC API standards, specifically the GeoVolumes API. This development builds upon previous use cases that are implemented by WiTech, including the OGC Container and Tiles Pilot (2020), the OGC Interoperable Simulation and Gaming Sprint (2020-2021), and OGC Testbed-18: Building Energy Spatial Data Interoperability. These initiatives have enabled the delivery of various 3D geospatial data formats across multiple client implementations.

The CityGML building model, as described in Clause 2.1, is converted to 3D Tiles using Feature Manipulation Engine software and hosted on WiTech's GeoVolumes server. Additionally, the Digital Elevation Model (DEM) from Gaia3D, provided in GeoTIFF format, is converted to a quantized mesh format using the Cesium Terrain Builder developed by TUMunich and is also hosted on the GeoVolumes server. The noise modeling results from the OpeNoise Map plugin for QGIS is converted to a point cloud and, subsequently, to 3DTiles (point cloud format) before hosting it on OGC APIs collection services.

In this pilot, the use of the OGC API Collections was expanded to include not only 3D geospatial data but also noise simulation data and Training DML imagery data within the same service platform. By doing so, the versatility of the OGC API standards in handling diverse data types is demonstrated. The noise simulation results from the OpeNoise Map plugin for QGIS were integrated alongside the 3D collections, enabling comprehensive environmental analyses.

Additionally, the Training DML imagery data was incorporated into the platform, facilitating enhanced visualizations and interoperability across different client implementations. This unified approach underscores the scalability of the GeoVolumes server and the effectiveness of OGC APIs in delivering a wide range of geospatial data formats.

The service is designed to comply with the OGC API-Common foundation resources, which include a landing page, conformance declaration, and collections. As part of the OGC API family of standards, it facilitates easy sharing, consumption, and filtering of 3D geospatial resources via the web using defined resource-centric APIs. Instead of accessing geo-data from various vendors, users can leverage the API to manage data heterogeneity and access resources from a single source. An overview of the OGC API resource path is provided in the Annex.

The data server has been implemented using Node.js and the Express.js web framework and is publicly available. This implementation adheres to the development and design guidelines of the OGC API architecture. The services provided by WiTech and WHU have been deployed and are publicly accessible.

2.5. Visualization

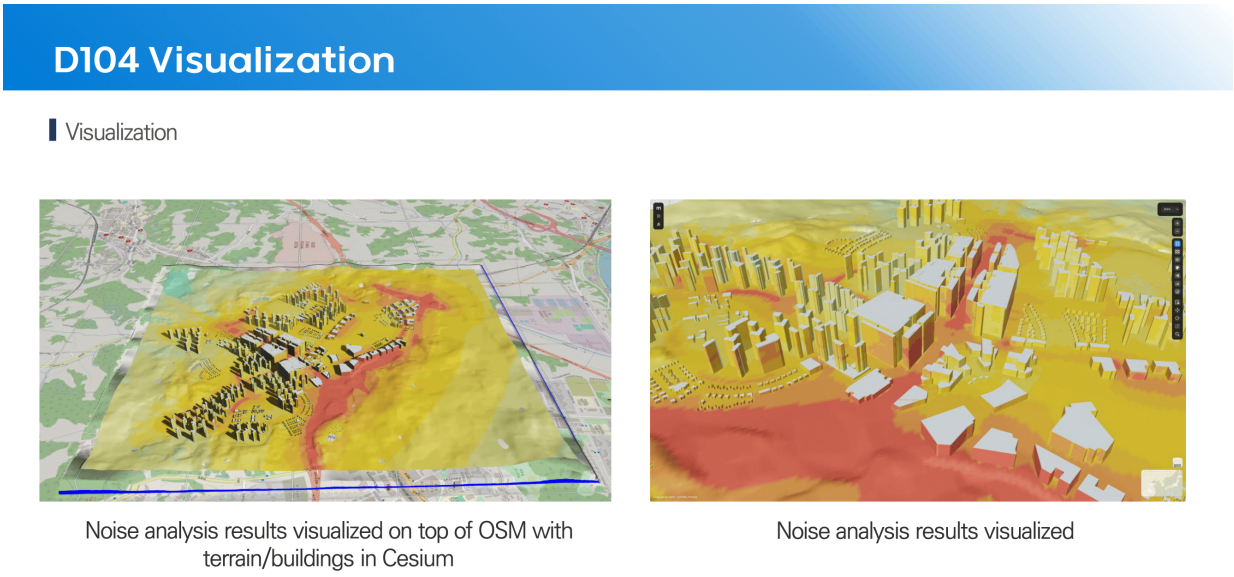

Figure 13 — Visualizing Noise Modeling Results in A Digital Twin Environment

Visualizing the noise modeling results in a digital twin environment plays a crucial role in maximizing the understanding and utilization of noise analysis data. Visualization provides an intuitive grasp of spatial distribution and temporal changes in noise, supporting important decision-making processes in urban planning and design. This project uses Cesium.js, an

open-source tool, for rendering. Building data is visualized in 3D Tiles format, while terrain is visualized in Quantized Mesh, supported by Cesium.

The 3D visualization client developed by WiTech effectively leverages the Cesium JS framework to integrate urban noise data from OGC API services, particularly derived from Deliverable D103. This client seamlessly combines various geospatial data formats, enhancing its capability to provide detailed visualizations. The key components include:

1. CityGML Building Model: Utilizes the Gaia3D data in 3DTiles format for rendering detailed 3D representations of urban structures.
2. GeoTiff DEM Model: Incorporates a Digital Elevation Model (DEM) from Gaia3D in Quantised Mesh format, enabling realistic terrain visualization.
3. 3D Noise Data: Integrates noise data from the OpeNoise Map plugin of QGIS, presented in 3DTiles (Point Cloud) format for an accurate portrayal of noise dispersion in three dimensions.
4. Ground Noise Data: Utilizes raster tiles from the OpeNoise Map plugin, providing a comprehensive view of ground-level noise across the urban landscape.

This combination allows for an immersive exploration of urban environments, facilitating analysis and decision-making related to noise pollution and its impacts on urban planning.

The client adopts a straightforward static website architecture, utilizing HTML, JavaScript, and CSS. This lightweight design approach ensures several advantages:

1. Accessibility: The static structure allows for easy access across different devices and platforms, enabling users to visualize data without complex setups.
2. Ease of Use: The simplicity of the design facilitates intuitive navigation, making it user-friendly for both technical and non-technical users.
3. Performance: By leveraging a lightweight framework, the client ensures quick loading times and efficient rendering of 3D models, enhancing the overall user experience.

The combination of CesiumJS with a clean web architecture promotes effective visualization of urban noise data while prioritizing accessibility and usability. The 3D visualization client is available online. This implementation not only visualizes urban noise analysis within the digital twin but also showcases the value of OGC service interoperability to external stakeholders. By leveraging integrated geospatial systems, the client provides actionable insights through intuitive and effective visual representation. This work highlights the potential of interoperable OGC standards to advance urban planning and analysis through enhanced data visualization capabilities.

2.6. Stakeholder Engagement

2.6.1. Overview

The purpose of this project is to integrate various functionalities into an ecosystem of Urban Digital Twin. When building an urban digital twin, the integration of multiple modules is essential and the interoperability between each module, therefore plays a crucial role. To test and evaluate this interoperability, OGC standards are applied through the Urban Digital Twin Interoperability Pilot Project (UDTIP). Multiple stakeholders are participating in this UDTIP as shown in Figure 14.

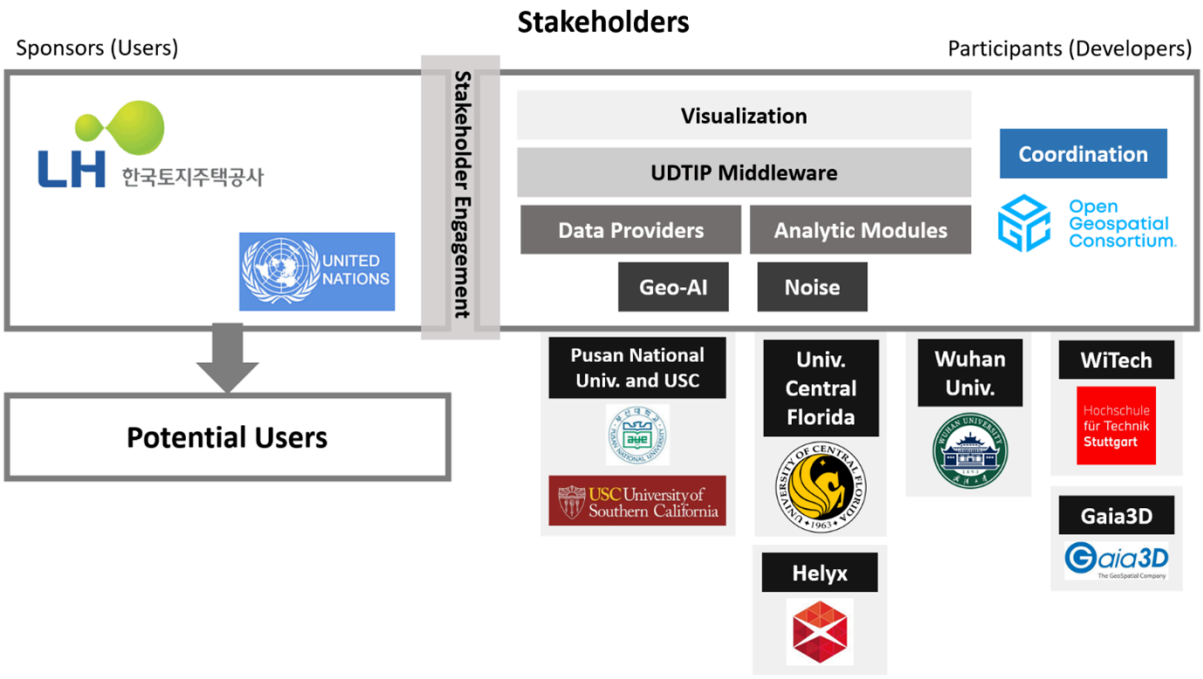

Figure 14 — Stakeholders

Stakeholders are broadly divided into three categories: Sponsors or User Groups, Participants or Developer Groups including OGC coordinating both groups, and Potential Stakeholders. The roles of each group can be summarized as follows:

- Sponsor: This group sponsors the project, specifies deliverables and requirements, and is responsible for reviewing the outcomes of the UDTIP. LH has been a central organization in adopting and utilizing urban digital twins in Korea. Although not a sponsor, the UN is included among the users. The UN participates as a user in this pilot project with a particular need for urban digital twin functionalities centered on its UN Camps.
- Participants: This group consists of teams responsible for developing all the functionalities of the UDTIP and submitting the final deliverables. It is highly recommended to provide

the deliverables as open source software. The table below summaries the teams for each deliverable.

Table 1 — UDTIP Participants and Respective Deliverables

	DELIVERABLES	TEAM
D001	Engineering Report	UCF (University of Central Florida, USA), HSR+ (Health Service Research, USA)
D100	Noise Modeling Interoperability	WiTech (Germany), Gaia3D (Korea)
D101	Camera Imagery Interoperability	UCF, WHU(Wuhan University, China)
D102	Geo-AI Analysis Interoperability	Helyx (UK), PNU (Pusan National Univ. Korea)
D103	Inter-Module Interoperability	WHU, WiTech
D104	Visualization	Gaia3D, WiTech
D105	Stakeholder Engagement	PNU

- Potential stakeholders: The outcomes of UDTIP can be used as a reference for other Urban Digital Twin projects. Any stakeholders involved in future projects aiming to build Urban Digital Twins based on OGC standards can potentially benefit from these outcomes. In particular, since UDTIP has been developed primarily using open-source technology, it offers high scalability. Organizations with broad user bases, such as the UN, are supposed as key potential stakeholders for this project.

2.6.2. System Architecture and Interfaces between Components

The overall structure of the system is depicted by Figure 15, where it is divided into two sections: one supporting Noise Analysis and the other supporting object detection/classification by Geo-AI.

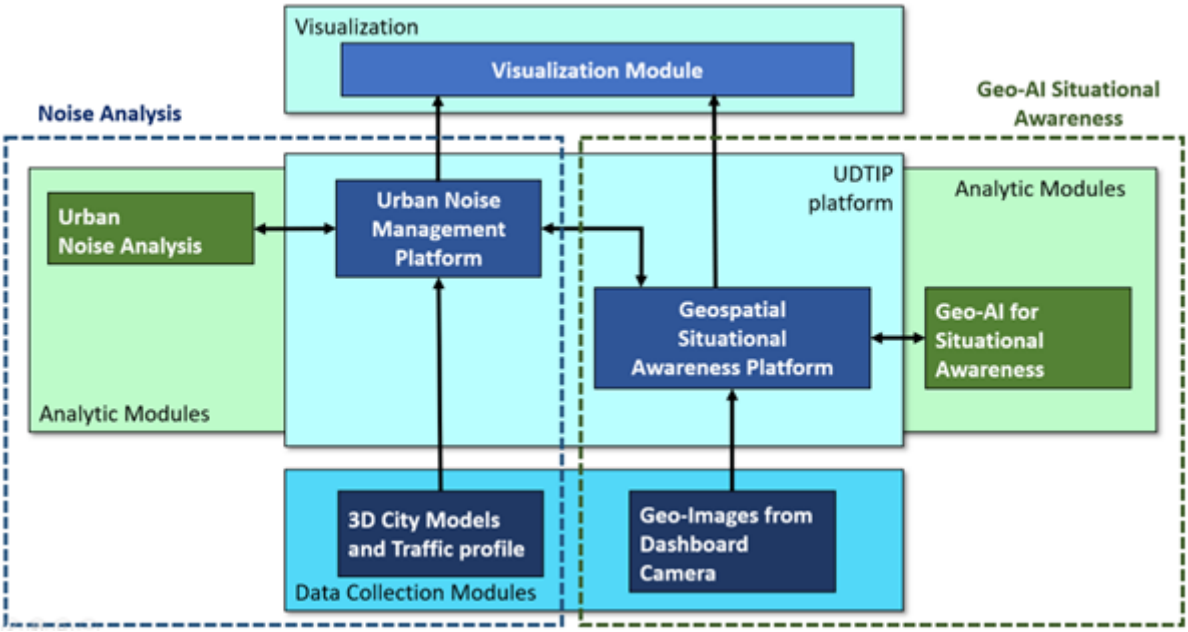

Figure 15 — Architecture

The Noise Analysis system, as shown in Figure 16, is primarily composed of the following components: a Data Collection Module, an Urban Noise Analysis Module, a Visualization Module, and the UDTIP Platform, which manages the interfaces between these modules. The pipeline and corresponding interfaces are summarized as follows:

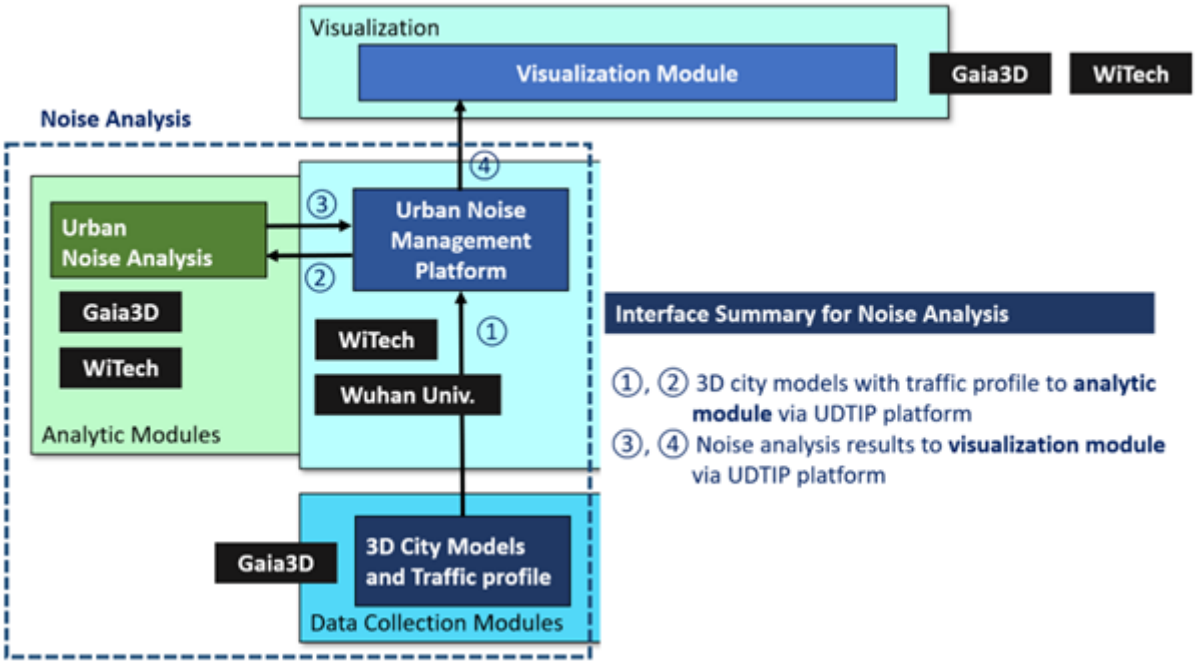

Figure 16 — Interfaces for Noise Modeling Subsystem

1. The Data Collection Module provides a 3D data model of the test site and a traffic profile in CityGML 2.0 data, then transmits it to the UDTIP Platform. The interface for this transmission uses OGC API to transfer the CityGML data.

2. The UDTIP Platform forwards the received data to the Noise Analysis Module. The interface for this step also uses the OGC API.

3. The Noise Analysis Module converts the analysis results into a format suitable for visualization (such as 3D tiles) and sends them back to the UDTIP Platform.

4. The UDTIP Platform then bypasses the visualization information received from the Noise Analysis Module to the Visualization Module. For detailed information on the interfaces required in the above process, please refer to Clause 2.4.

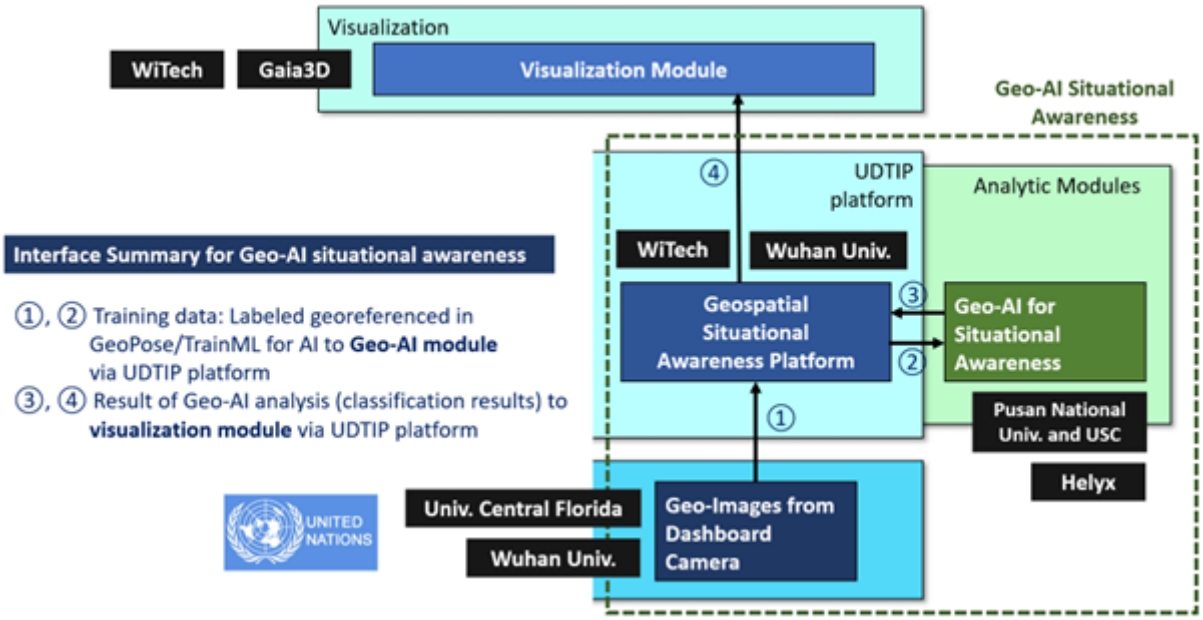

Figure 17 — Interfaces for GeoAI

The Geo-AI functionalities, as shown in Figure 17, consist of a Data Collection Module for gathering labeled training data from camera images, a Geo-AI Analysis Module that performs deep learning to detect objects, a Visualization Module to display analyzed results, and the UDTIP Platform, which manages the interfaces among these components. The operational pipeline is similar to that of Noise Analysis and can be summarized as following:

1. Data Collection Module: Images are first captured from on-site cameras using smartphones mounted on vehicles. The images are labeled, and data including GeoPose (containing image data, geo-references, location, and FoV) along with Training DML data (for labeling) are sent to the UDTIP Platform.

2. The UDTIP Platform forwards the data received from the Data Collection Module to the Geo-AI Analysis Module.

3. The Geo-AI Module detects or classifies objects and sends the detected object information with geo-referenced data back to the UDTIP Platform.

4. Finally, the UDTIP Platform transmits the visualization data from the Geo-AI Module to the Visualization Module. The interfaces required in this process are based on OGC API, with further details available in the description of D103.

2.6.3. Requirements and Inputs from Sponsors

The sponsor of this pilot project is LH (Korea Land and Housing Corporation), a national corporation responsible for urban planning and new town construction. Since urban planning is its primary business, a Digital Twin is a crucial system for LH. With recent advancements in IT technology, creating a virtual city, especially an Urban Digital Twin as well as physical cities has become a critical task for LH. Therefore, this pilot project to build an Urban Digital Twin based on OGC standards holds significant importance for LH.

The main expectation of this pilot project from LH is to create a reference model for building an Urban Digital Twin based on OGC standards. Two major aspects were considered in this regard. The first aspect is to provide a geospatial framework for the Urban Digital Twin by developing a noise analysis function based on a 3D city model. The second aspect is to incorporate recent progress of Geo-AI technologies and specifically a function for the classification and analysis of road conditions. The goal is to build a Digital Twin based on OGC standards and validate the interoperability supporting these two applications. The specific objectives are outlined as follows;

- Develop a Pilot plan to implement digital twin interoperability scenarios and a supporting standard digital twin API for Urban noise analysis and Situational analysis of geo-referenced still and moving imagery for use cases
- Prototyping open standard interfaces for:
    1. Data collection from the real world to an Urban Digital Twin
    2. Data exchanges between UDTIP and analytic processes
    3. Presentation of analytic processing results
- Collaboratively exploring best tools and representations through co-design with scientists, technology developers, and decision makers to develop, test, and validate advanced interoperability and integration methods for Urban Digital Twins.

2.6.3.1. Functional Requirements and Inputs for Noise Analysis

As noise is a critical factor in urban planning, the sponsor requests the function to analyze noise level in urban area as below;

- Estimate noise level at any given point considering the traffic conditions such
    1. noise level at any point at surface

2. noise level at any point of given height
3. noise level on building wall surface

- Visualization of noise level via web portal in 3D

In order to carry out the functions above, LH provided the following data for input;

- 3D City Model: CityGML with LoD 1 and 2 including mainly buildings, generated from urban planning profile
- Terrain Data: OGC GeoTIFF
- Traffic Profile

These data sets were generated by Gaia3D from the raw data and urban planning profile.

2.6.3.2. Functional Requirements and Inputs for Geo-AI

The required functions for Geo-AI applications involve the classification and analysis of road surfaces. Specifically, this includes detecting objects such as road surface materials, damage conditions, and dumping objects on roads. The classification of road surface material follows the standards of OpenStreetMap, which has been heavily used as base map by the UN. The detail requirements and road surface classification are explained in Annex C. The application site for this pilot project has been designated as the UN VMC (Verification Mission in Colombia) camp. Due to constraints of resources, the project has initially categorized road surfaces into the following three types.

- Asphalt Road
- Paved Road
- Unpaved Road

A deep learning model is trained using a set of labeled image data. When camera images or videos collected from mission vehicle are provided with geo-reference data, the pre-trained model is applied to classify the data in real-time into the three categories mentioned above. The results are then visualized on a map. For these functionalities, input data was provided with the cooperation of the UN VMC and the UNGSC (UN Global Service Center). Specifically, they supplied training data with labeled geo-referenced image data. Although the classification of road surface for this pilot project is limited, it may be easily extended to cover more detail categories if the corresponding training data set would be provided.

3 OUTLOOK

3 OUTLOOK

The application of noise modeling using CityGML building and road network data has provided foundational insights into noise propagation in urban environments. Leveraging the OpenNoise tool integrated with QGIS, simulations for day and night scenarios offer a baseline understanding of noise impacts across time and elevation. Future advancements should focus on integrating real-world sensor data to refine noise source estimations and incorporating additional factors such as road surface characteristics, traffic volume, and vehicle speed. Expanding the model's capacity to simulate diverse environmental conditions and employing adaptive methodologies for multi-elevation analysis will further enhance its utility for urban planning. This iterative approach supports more precise noise assessments, enabling effective mitigation strategies and fostering sustainable urban development.

The successful implementation of D101 has established a robust framework for capturing and preprocessing geopose-referenced imagery datasets. This interoperability supports downstream modules such as Geo-AI model training (D102) and visualization (D104). Future efforts should focus on enhancing data acquisition through higher-resolution sensors and more precise timestamp synchronization techniques. Expanding the adaptability of the TDML-AI format to incorporate evolving standards will streamline module interoperability and improve cross-platform collaboration. Continued automation of labeling and processing workflows will be critical to maintaining scalability and efficiency.

A key priority moving forward is to establish clear and definitive criteria for classifying road surface types, ensuring team consensus on the precise definitions of "Paved" and "Unpaved." This alignment is essential for consistency in data interpretation and application. While awaiting a labeled dataset from the UN, the team will focus on developing models using public datasets and generating synthetic data samples based on the agreed-upon class definitions. Additionally, the input and output modules of the classification model will be designed to comply with interoperability standards, enabling seamless integration and data exchange with broader systems. These steps will enhance the model's applicability and readiness for deployment in real-world scenarios.

The adoption of OGC API standards across modules highlights a scalable approach to inter-module data exchange, particularly in the distribution and visualization of training datasets. By integrating OGC API-Features, API-Tiles, and API-3D GeoVolumes, D103 effectively bridges the gap between training data and inference visualization, ensuring compatibility and seamless access to diverse data types. Future efforts should focus on enhancing data encoding schemas and implementing real-time data management techniques—such as dynamic bypassing and distributed server networks—to strengthen system resilience and expand interoperability across varied operational contexts.

The integration of visualization tools into digital twin environments has proven instrumental in enhancing the interpretability and utility of noise modeling data for urban planning. By leveraging 3D Tiles and Quantized Mesh formats, the project ensures efficient, high-resolution rendering of complex urban and terrain data while maintaining compliance with interoperability standards. Future advancements should focus on refining real-time data handling capabilities to address dynamic urban scenarios. Developing sophisticated visualization techniques—such as adaptive rendering based on user interaction or advanced noise analysis legends—will further

enhance decision-making processes. Expanding support for additional data formats and open-source visualization tools will also contribute to the robustness and scalability of the system.

Stakeholder engagement remains pivotal in the development and implementation of Urban Digital Twin Interoperability Pilot Projects (UDTIP). By categorizing stakeholders into Sponsors, Users, and Participants, this initiative fosters collaborative innovation, aligning diverse expertise to achieve module interoperability. Sponsors—such as LH and the UN—provide critical user insights and strategic requirements, while developer groups ensure the technical realization of deliverables, often contributing as open-source projects. Sustaining this engagement model is essential for refining interoperability pipelines, enhancing real-world testing environments, and adapting to evolving user needs. Expanding participation to include additional user scenarios —such as emergency response and urban resilience—and fostering stronger feedback loops between Sponsors and Participants will strengthen the project's applicability. This collaborative ecosystem offers a robust foundation for advancing urban digital twin applications globally.

This series of Urban Digital Twin (UDT) pilots demonstrates the transformative potential of this technology for cities and their residents. We recommend incorporating healthcare use cases in future pilots to highlight UDT's ability to revolutionize urban healthcare. This is especially relevant in Korea, where health expenditure surged by 208,900% between 1970 and 2022, with healthcare accounting for 13.6% of general government spending in 2022.

The healthcare industry stands to benefit from enhanced operational efficiency, improved health outcomes, and reduced costs through real-time visibility and what-if analysis of resources, hospital bed occupancy, device availability, and patient flow. Key applications include remote patient monitoring, personalized medicine, and chronic disease management—enabling more proactive, data-driven healthcare decisions.

Furthermore, integrating healthcare with broader UDT applications—such as urban noise monitoring and situational imagery analysis—can help mitigate health risks while maximizing existing investments. This approach also engages a broader community within the digital twin ecosystem, unlocking greater value from UDT initiatives.

We encourage the Land & Housing Corporation to explore healthcare use cases and extract health-related insights from non-health pilots, while also surveying the Korean healthcare sector for potential industry applications. More details are provided in Annex B.

3.1. Collaboration between OGC, OSGeo, and UN

The outcomes of UDTIP can be used as a reference for other Urban Digital Twin projects. Therefore, any stakeholders involved in future projects aiming to build Urban Digital Twins based on OGC standards can potentially benefit from these outcomes. In particular, since UDTIP has been developed primarily using open-source technology, it offers high extensibility. The availability of the open source software used or developed by the project is summarized in Annex D. Organizations with broad user bases, such as the UN, are supposed as key potential stakeholders for this project.

A strategy to implement international cooperation is presented in this section. A simplified representation of these strategies is shown in the following Figure 18. The international

organizations mentioned in this figure are all multilateral in nature and possess significant recognition and influence. First, the UN Open GIS Initiative plays a key role in connecting the large-scale demand of the United Nations with open GIS software technology. OGC and OSGeo (Open Source Geospatial Foundation) are the most important international organizations responsible for the development of open spatial information technology serving as key suppliers. Facilitating cooperation among these three international organizations is a crucial strategy for international collaboration in spatial information. This aligns with the comprehensive cooperation strategy between demand and supply entities.

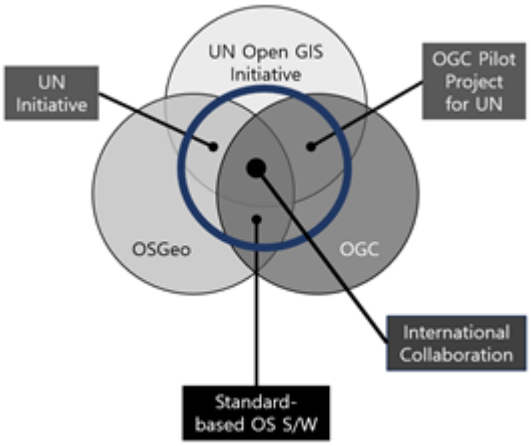

Figure 18 — Collaboration between OGC, OSGeo, and UN

The relationships and roles of these organizations can be summarized as shown in the figure below. First, the OGC provides the foundation of open spatial information technology for standards. Based on this foundation, OSGeo develops open-source spatial information software. The developed software is then applied to various UN projects and related agencies through the UN Open GIS Initiative. The role of the UN Open GIS Initiative is extensive. Its primary role is to supply open-source software required by the UN, but ultimately, it aims to build and manage a community encompassing UN-affiliated organizations and international activities connected to the UN. This community is closely related with the development of human capacity for the effective and sustainable use of open-source software.

Figure 19 — Roles of OGC, OSGeo, and UN

Providing open geospatial information solutions to the UN or developing countries is an important strategy. Either existing open spatial information software can be provided or new solutions can be developed through international cooperation when necessary. An international cooperation in open spatial information aims to support the installation and deployment of these solutions for the use by the UN or developing countries and to establish its sustainability. The most crucial point to consider is that the goal should not merely be to deliver a single solution to the target UN organization or country, but rather to provide a comprehensive ecosystem. Offering just a single solution makes it difficult to ensure sustainability. It is desirable to provide a solution that also includes human capacity-building and institutional support. While previously developed solutions have used traditional technologies, additional work is required to apply emerging technologies as open solutions. For instance, there is a high demand for technologies such as Geo-AI spatial information, digital twins for urban environments, and IoT-based situational awareness technologies. Although individual technologies have been developed, there is still a challenge of geospatial information solutions that can fully apply them to the fields. These gaps can be addressed through an international cooperation between three organizations – OGC, OSGeo, and UN Open GIS initiative as described earlier. A detailed strategy for development is illustrated in the figure below;

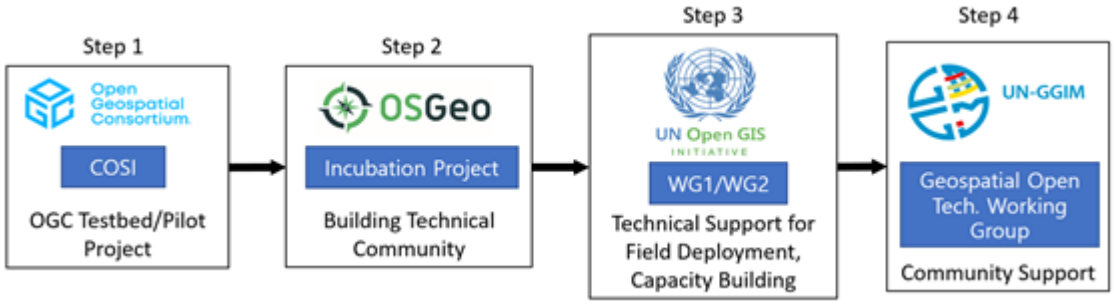

Figure 20 — Roadmap of Collaboration

- Step 1 — Standard-Based System Integration Development: Typically, solutions are developed by integrating multiple components where OGC standards play a crucial role. Integration development can be facilitated through COSI (Collaborative Solutions and Innovation Program) under OGC. This program collaborates on the development of integrated systems using OGC geospatial standards.

- Step 2 – Open-Source Project Organization: While the results developed through OGC's COSI are often one-time outcomes, we need to transform them to sustainable structures. The Incubation project under OSGeo may serve this purpose. This project helps to establish a technical community for ensuring the sustainability of open-source solutions. Once a solution has graduated from OSGeo's Incubation project, it is considered to have an established sustainable international community capable of managing it.

- Step 3 — Field Application: Once a technically sustainable solution is created through OSGeo, the role of building an ecosystem for field application may be handled by the UN Open GIS Initiative. This initiative's Working Groups, such as WG 1 for hybrid architecture, WG 2 for capacity building, and WG 5 for Geo-AI may be responsible for providing technical support and capacity-building training for field implementation.

- Step 4 — Building and Operating an International Community: For the sustainable and expansible applications of open geospatial solutions deployed in the field, the

organizational resources such as UN-GGIM can be utilized. Since this organization involves the largest number of stakeholders related to geospatial information, it is the most suitable body to support the expanded application of developed solutions.

This process is primarily designed for newly developed open spatial information solutions but can also be applied to existing ones. For instance, if a solution has already been developed and is being used within the UN, it can be enhanced by establishing a robust technical community through Step 2 and expanding its reach through Step 4.

3.1.1. Extensibility

The Geo-AI application implemented in this project is limited to the classification of road surfaces into three categories: asphalt, paved, and unpaved, due to the constraints in collecting training data within the given timeframe. However, in real-world applications, road surfaces can be classified into more types. Additionally, the Geo-AI function can be expanded to not only classify different types of road surfaces but also to identify the types and severity of road surface damage, as well as detect objects on the road such as dumped objects along the roadside. Extending the capabilities of Geo-AI can be easily achieved by TrainDML for AI, an OGC standard used in this pilot project. The method on how to extend TrainDML for AI is explained in TrainDML for AI.

4

SECURITY, PRIVACY AND ETHICAL CONSIDERATIONS

4 SECURITY, PRIVACY AND ETHICAL CONSIDERATIONS

During the course of this project, a thorough review was conducted to identify any potential security, privacy, and ethical concerns. After careful evaluation, it was determined that none of these considerations were relevant to the scope and nature of this project. Therefore, no specific measures or actions were required in these areas.

A

ANNEX A (NORMATIVE) TECHNICAL DETAILS

A ANNEX A (NORMATIVE) TECHNICAL DETAILS

A.1. Annex for D100

A.1.1. D100 — WiTech Noise Analysis Workflow

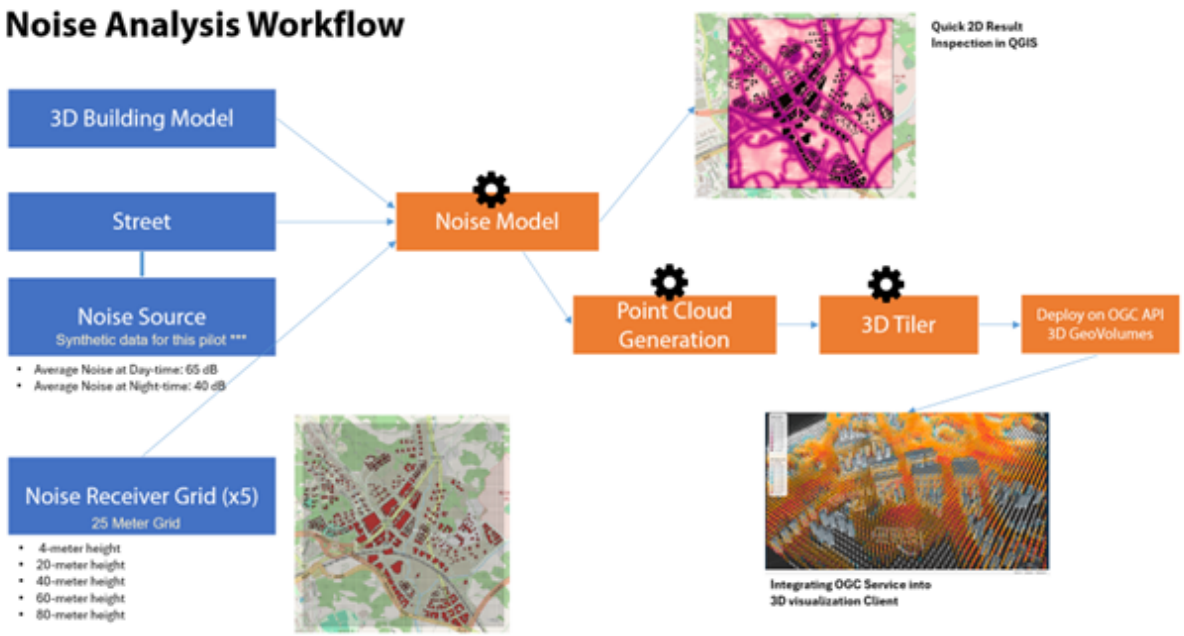

Figure A.1 — Noise Analysis Workflow

Noise modeling was conducted using CityGML building and GML street models to develop a foundational noise propagation model with the OpeNoise tool integrated into QGIS. The model simulated day and night noise levels using synthetic data, assuming uniform noise sources from road surfaces, while factoring in environmental interactions such as reflections and diffractions. To enhance accuracy, future work will integrate real-world sensor data and consider variables like traffic volume, speed, and surface characteristics for more precise noise assessments across different urban elevations.

A.1.2. D100 — Gaia3D

The source data of CityGML data is the urban planning data of Wangsuk District 2, Namyangju, Korea, which is the result of LH's past urban planning work and was provided by the sponsor, LH. The data consisted of building layout plans, plan elevations (future terrain), and detailed drawings, which were provided in SHP, GeoTIFF (DEM), and DXF formats for the building layout plans and detailed drawings. The SHP files were first processed into GeoJSON with only the height (distance between top and bottom planes) as an attribute, which is necessary for converting them into three-dimensional objects, and the DXF files were first processed into GeoJSON by extracting the geometry of the roads and combining it with the attributes related to road noise sources recorded in the environmental impact assessment report of the city plan, all of which were done manually. The GeoJSON files from the first round of processing were converted to CityGML using FME software. The building layout plan GeoJSON file was combined with the DEM and converted to CityGML with absolute heights, and Createch, a company specializing in noise analysis provided feedback on the results of combining the road noise source GeoJSON and DEM, and based on this, a second GeoJSON with three-dimensional information was created and converted to CityGML using FME. (The building layout plan and road noise source GeoJSON and DEM were provided to Createch as inputs for noise analysis).

1. Work with FME (ETL)

   a) Creating Building.gml

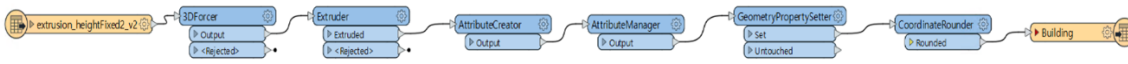

Figure A.2

FME data conversion process

   b) Creating noiseSources.gml

Figure A.3

FME data conversion process

2. CityGML Properties

   a) Building

      i) Description

         - Format: CityGML

- EPSG:5186
- Destination: Wangsuk 2 District, Namyangju, South Korea

b) Noise Sources

i) Description

- Format: CityGML
- EPSG:5186
- Destination: Wangsuk 2 District, Namyangju, South Korea
- The noise sources used in this study are limited to road noise ources. This is because the target area, Wangsuk 2 District, Namyangju, is a new city, and the analysis was based on data generated only during the planning stage. In fact, there are various types of noise sources such as airplanes and subways, but only road noise data was processed in this project.
- Road noise sources were defined and deployed at a low level compared to the existing CityGML-based structure. This was chosen to increase data usability and interoperability by clearly and simply defining the necessary information

ii) Road Width

- Property name: width
- Property type: double
- Description and units: Represents the width of the road in meters.
- Example value: 10

iii) Number of Lanes

- Property name: number of lanes
- Property type: int
- Description and units: Indicates the number of lanes that exist on the road.
- Example value: 3

iv) Number of Small Vehicles per Hour at Day

- Property name: number of small vehicles per hour at day

- Property type: int
- Description and units: The number of small vehicles that pass the road per hour during daylight hours (vehicles/hour).
- Example value: 181

v) Number of Large Vehicles per Hour at Day

- Property name: number of large vehicles per hour at day
- Property type: int
- Description: Represents the number of heavy vehicles passing the road per hour during the daytime.
- Example value: 9

vi) Number of Small Vehicles per Hour at Night

- Property name: number of small vehicles per hour at night
- Property type: int
- Description: Represents the number of small vehicles passing through the road per hour at night.
- Example value: 60

vii) Number of Large Vehicles per Hour at Night

- Property name: number of large vehicles per hour at night
- Property type: int
- Description: Represents the number of heavy vehicles passing through the road per hour at night.
- Example value: 3

viii) Speed of Small Vehicles at Day

- Property name: speed of small vehicles at day
- Property type: double
- Description: Represents the average speed of small vehicles in km/h during the daytime.
- Example value: 30

ix) Speed of Large Vehicles at Day

- Property name: speed of large vehicles at day
- Property type: double
- Description: Represents the average speed of heavy vehicles in km/h during the daytime.
- Example value: 30

x) Speed of Small Vehicles at Night

- Property name: speed of small vehicles at night
- Property type: double
- Description and units: The average speed of small vehicles at night, in km/h.
- Example value: 30

xi) Speed of Large Vehicles at Night

- Property name: speed of large vehicles at night
- Property type: double
- Description and units: The average speed of heavy vehicles at night, in km/h.
- Example value: 30

xii) Surface status code

- Property name: surface status code
- Property type: int
- Description and units: A code indicating the pavement condition of the road. (No units)
- 0: Flawless sandy asphalt, asphalt concrete, mastic asphalt
- 1: Concrete, sandy asphalt with imperfections
- 2: Packaging with a flat surface
- 3: Other Packaging
- Example values: one of 0, 1, 2, or 3

xiii) Applied Additional Noise Reduction Method

- Property name: applied additional noise reduction method
- Property type: string
- Description: Description of noise reduction methods, if any, that are not applied in the noise modeler.
- Example value: Apply a road paving method developed by company YYY in 202X.

xiv) Noise Reduction Effect by the Method

- Property Name: Noise reduction effect by the method
- Property type: double
- Description and units: The noise reduction effect in decibels (dB) due to the applied noise reduction method.
- Example value: 15

c) DEM (merged_dem_trimmed in GeoTiFF)

i) Description

- Format: GeoTiFF
- Terrain data was originally intended to be provided in
- CityGML format, but due to the large size of the data, the
- data was provided in GeoTiF at 1m, 5m, and 10m resolution.
- Source data (GeoTiff) from TINRelief5.

A.1.2.1. Building Data

Building data from CityGML was converted into 3D Tiles format using a custom-built conversion module. Below is an example of the converted and visualized building data.

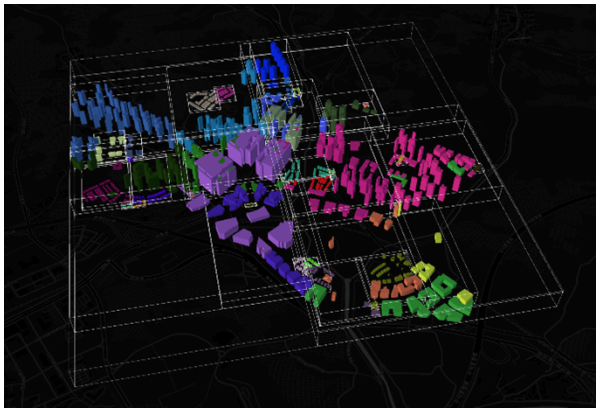

Figure A.4 — Visualization of the Generated 3D Tiles in Cesium

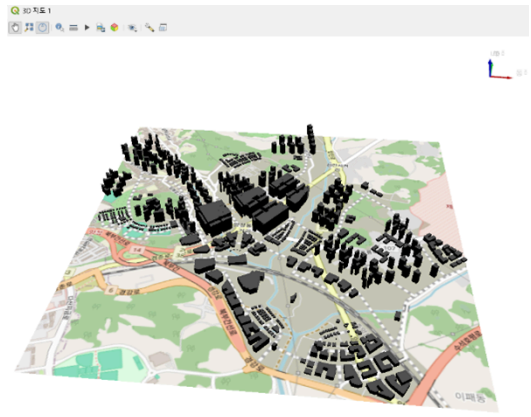

Figure A.5 — Visualization of the Generated 3D Tiles in Cesium

A.1.2.2. Terrain Data

A custom-built module was developed to convert the GeoTiff terrain data into Quantized Mesh format, and the results are as follows.

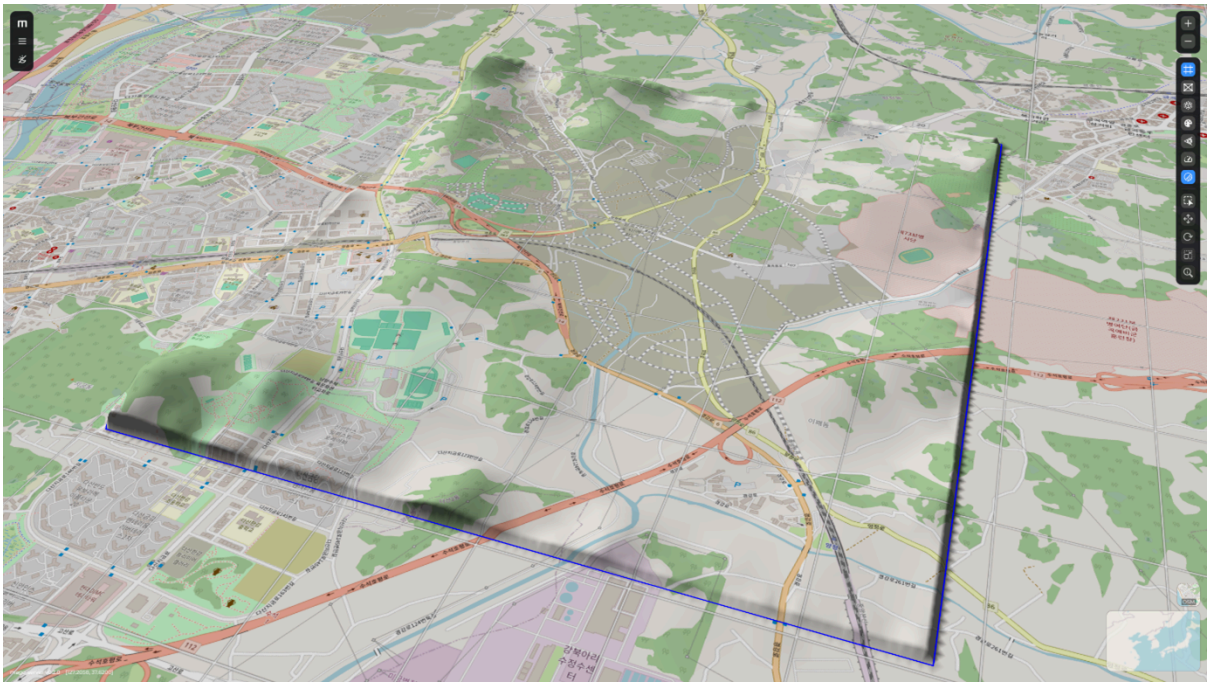

Figure A.6 — Visualization of the Generated 3D Tiles in Cesium

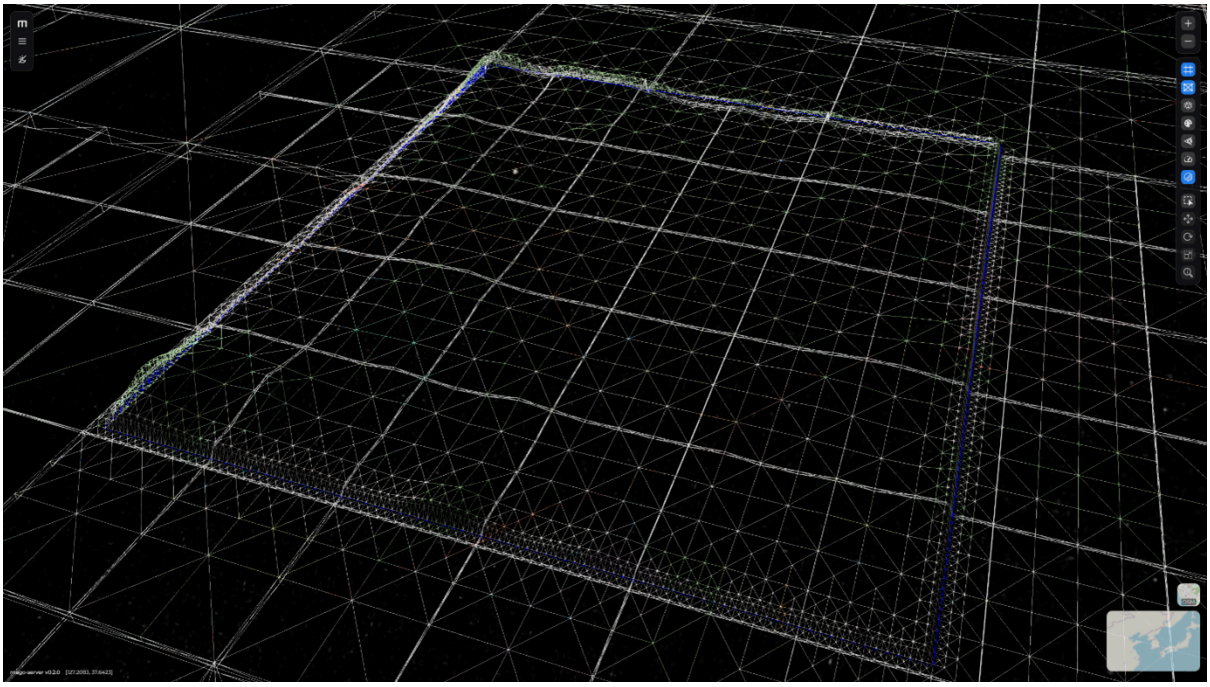

Figure A.7 — Visualization of the Generated 3D Tiles in Cesium

A.2. Annex for D101

A.2.1. D101 — UCF Roadmap

A.2.1.1. Development framework

This framework is developed to achieve camera imagery and Geo-AI analysis interoperability within the context of Smart Cities. The workflow diagram is divided into two major components:

- D101 — Camera Imagery Interoperability: This phase focuses on data acquisition and processing using cameras and INS (Inertial Navigation System) capture. The raw data collected undergoes systematic processing, including video frame sampling, preprocessing of INS metadata, synchronization, and conversion into the GeoPose standard in JSON format. This ensures that the imagery is geo-referenced and ready for machine learning workflows.
- D102 — Geo-AI Analysis Interoperability: The processed data from D101 serves as the input for this phase. Key tasks include developing a frame labeling tool and converting the processed dataset into the TrainDML standard. The dataset is then used for training, testing, and validation in machine learning workflows aimed at image-based object detection. This phase supports the development of ML-driven solutions for detecting unwanted objects in urban environments.

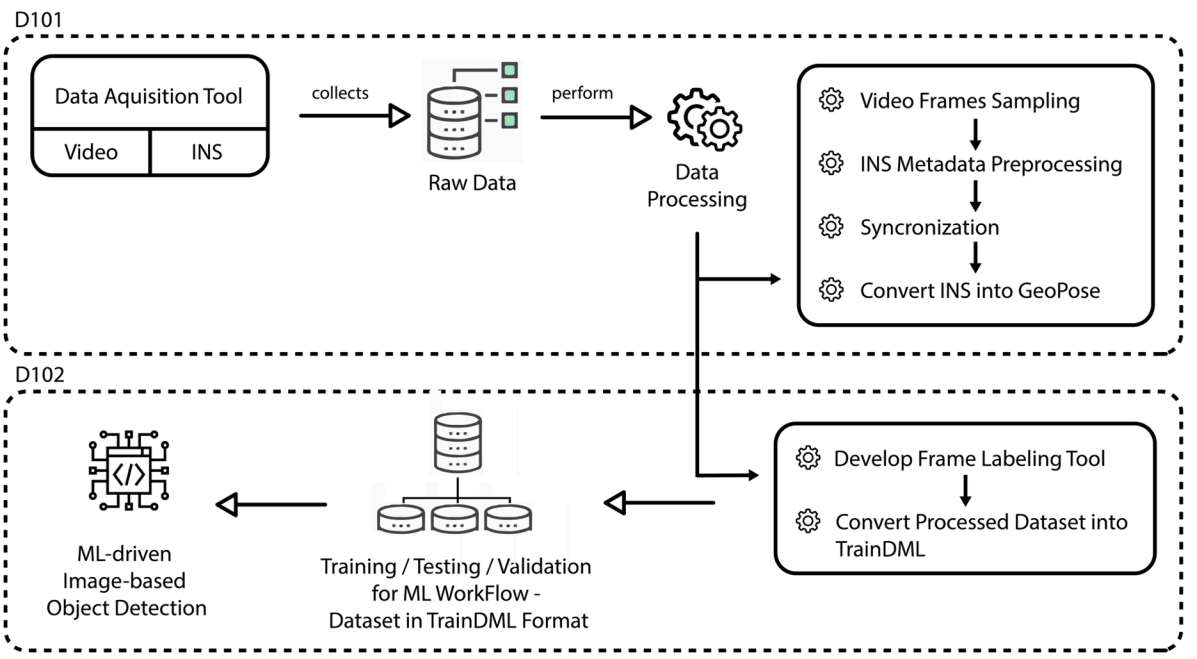

Figure A.8 — D101 to D102 Workflow

A.2.2. D101 — UCF Technical Specifications

A.2.2.1. Data Acquisition Methods

Investigated various geospatial camera imagery applications and datasets, including G-Meter, IP Webcam Pro, NUS Global Streetscapes and the Road Damage Dataset, to assess their suitability for our requirements. The custom build acquisition uses Sensor Logger application that captures images in `.jpg` format, and positioning and orientation data in terms of latitude, longitude, altitude, radians (`pitch`, `roll`, `yaw`), and quaternions (`qx`, `qy`, `qz`, and `qw`).

Where,

- `altitude`, is in meters as a height above the WGS84 ellipsoid.
- `latitude`, in degrees. Positive values are north of the equator (-90 to 90)
- `longitude`, in degrees. Positive values are east of the meridian line (-180 to 180)

Table A.1 — Technical Specifications

ACQUISITION METHOD	DESCRIPTION	FREQUENCY/FRAME RATE
Custom Build	Images: 1798 samples INS: 1805 samples	1 Hz 1 Hz
Hillyfields Bubble: Run 3	Video: 05:35 mm:ss INS: 33601 samples	4.91 fps 100 fps
Kitti-360	Images: 320k INS: 4×83,000	10 fps 10 fps

A.2.2.2. Video Frame Sampling

A.2.2.2.1. Command Line Instructions for Extracting Video Frames

The command used in FFMPEG to extract frames is as follows:

```
ffmpeg -i input_video.avi -vf fps=<NO. of images>/<per no. of seconds> img%0
<padding No. of digits>d.jpg
```

Listing A.1

OR

```
ffmpeg -i <input_video> -vf fps=<frames_per_second> <output_pattern>
```

Listing A.2

- `<input_video>`: The path to the input video file.
- `<frames_per_second>`: The rate at which frames should be extracted. This can be specified as a ratio (e.g., `1/1` for one frame per second).
- `<output_pattern>`: The pattern for naming the output image files. The `%0<padding No. of digits>d` is used to define the format and padding for the output image filenames.

A.2.2.2.2. FFMPEG Command-line Interface (CLI)

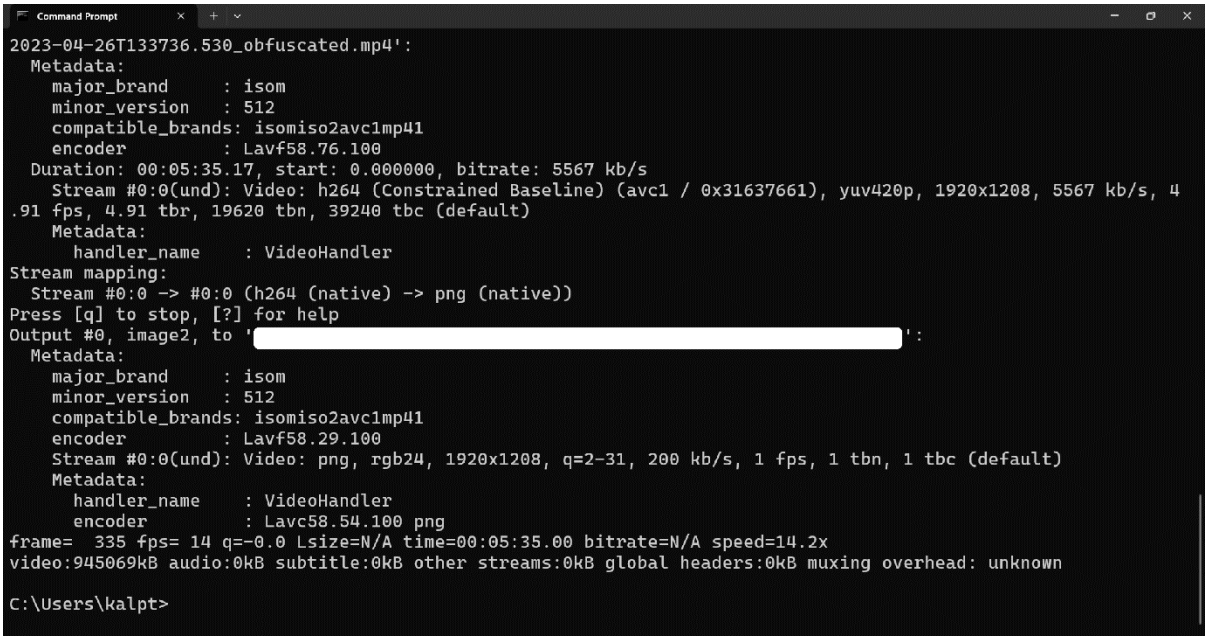

Figure A.9 — FFMPEG Command-line Interface

A.2.2.3. Synchronization

A.2.2.3.1. Video-INS Spatial Synchronization

The image depicts the process of synchronizing video frames and INS metadata, where each has a different frame rate (FPS). A synchronization technique aligns the time-stamped INS metadata with the corresponding video frames. The result is a georeferenced training sample or frame instance, combining visual imagery with accurate spatial data for use in urban digital twin models or AI-based analysis.

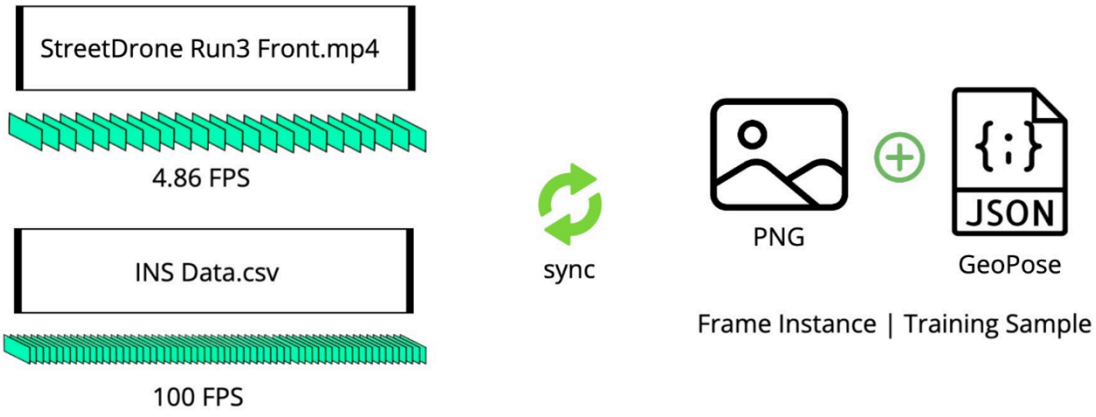

Figure A.10 — Synchronization of Video and INS Data for Accurate Spatial Analysis

A.2.2.4. Applicable OGC Standard — GeoPose 1.0

A.2.2.4.1. Standardization Targets

Used GeoPose.Composite.Sequence.Series.Regular standardization target to ensure consistent encoding of positional and orientation data for interoperability.

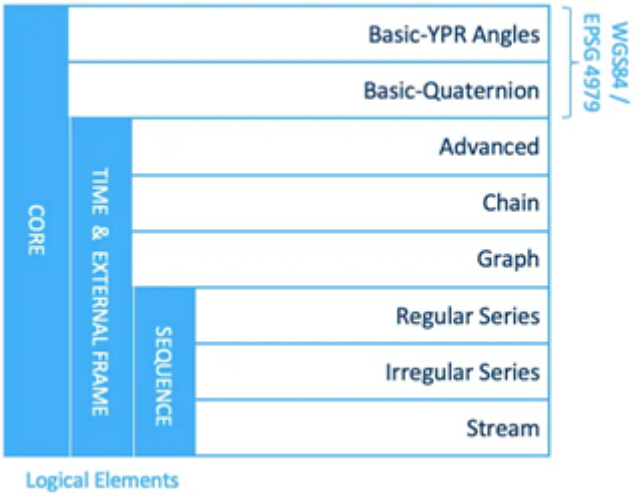

Figure A.11 — GeoPose Standardization Targets

A.2.2.5. GeoPose Standard Conversion

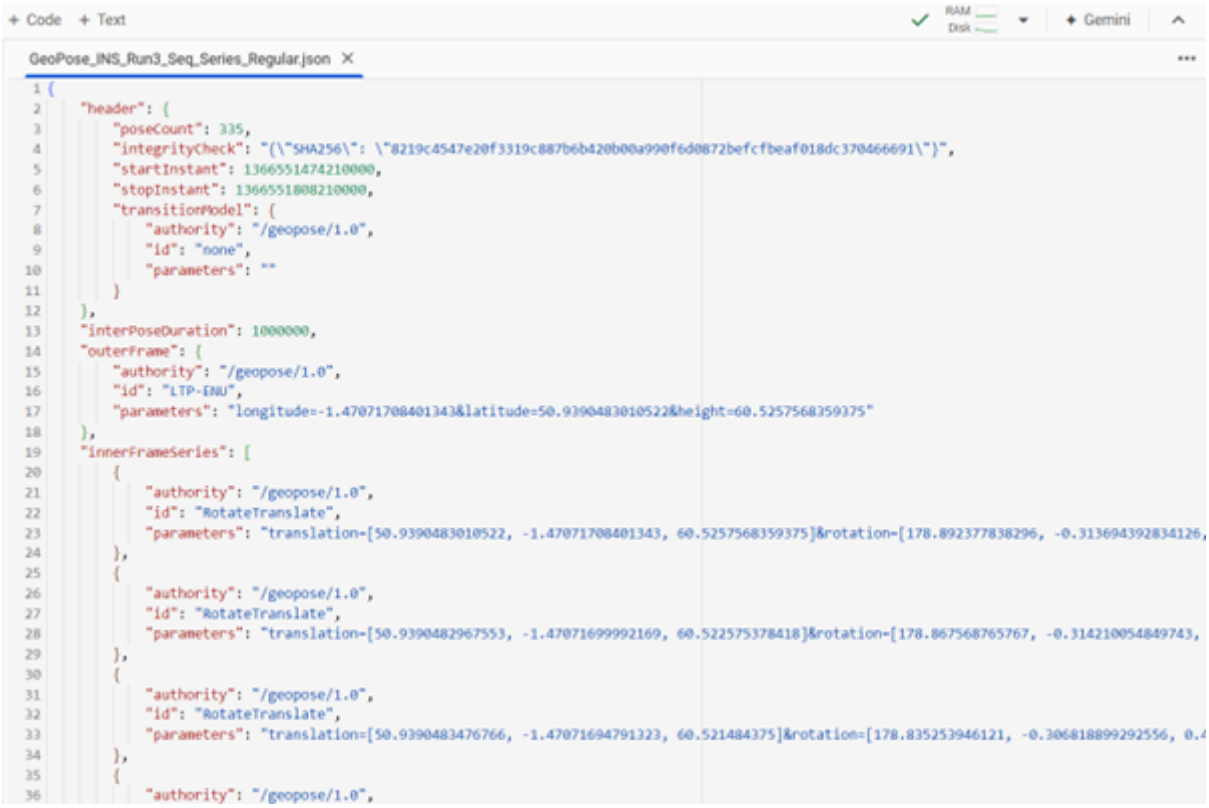

Figure A.12 — GeoPose of Hillyfields Dataset for Run-3

A.2.3. D101 — Wuhan Camera Interoperability

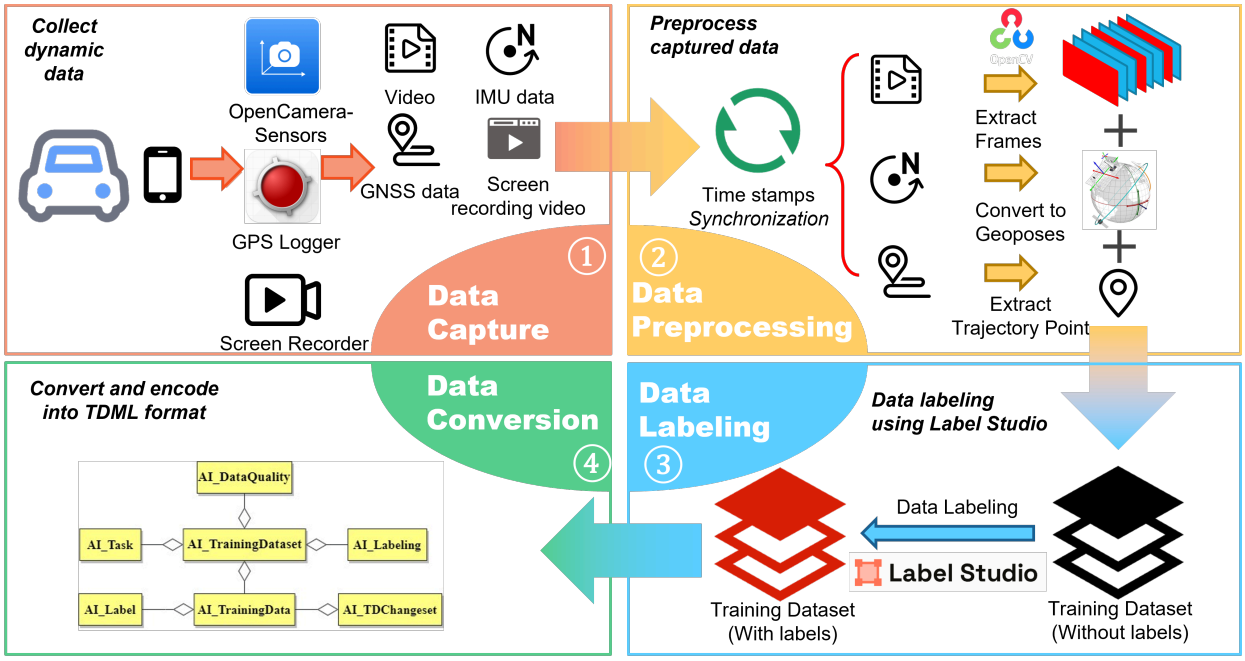

Figure A.13 — Workflow for D101

D101 is responsible for collecting imagery data with geopose information, which is constructed as a training dataset after data preprocessing and data labeling. The training dataset can be used for D102 Geo-AI Analysis Interoperability and D104 Visualization. The workflow of D101 is shown in figure Workflow for D101.

A.2.3.1. Data Capturing Tool Usage Guide

This chapter will guide you through the process of using data capturing tools to collect data for UDTIP.

A.2.3.1.1. Applications Installation

Two applications are used in this chapter. The first one is the GPS Logger. It is used to capture GNSS data, and can be installed through Google Play. Make sure this app can run in the background. The second app is OpenCamera-Sensors. It is an open-source app and the latest apk can be downloaded from GitHub releases.

For both apps, the appropriate permissions should be granted.

A.2.3.1.2. Application Settings

For OpenCamera-Sensors, accelerpmeter, gyroscope and magnetometer should be enabled and the video resolution should be set to Full HD (1920×1080).

A.2.3.1.3. Data Capturing

1. Start recording GNSS Data with GPS Logger.
2. Use the screen recording function that comes with your phone to record your screen. It need to be set to 30 fps before recording.
3. Start recording video and IMU data using the OpenCamera-Sensors.
4. Show a clock in front of the camera and stay at least five seconds. This will help align the timestamps during data preprocessing.

Figure A.14

A.2.3.1.4. Data Export

- GPS Logger: You can check the recorded trajectories in the trajectory list. And it can be exported in kml and gpx format.

Figure A.15

- Screen Record: The screen recorded video needs to be exported.
- OpenCamera-Sensors: Connect the phone to the PC and export the data from the phone. Get data from “DCIM/OpenCamera”.

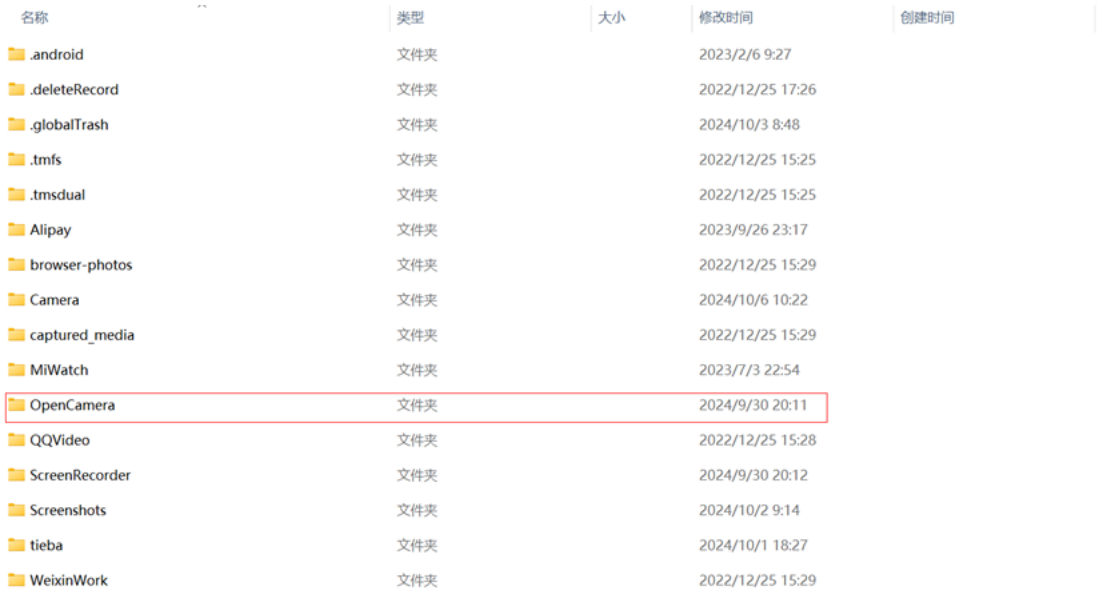

Figure A.16

And in this dictionary, export both the video and IMU data.

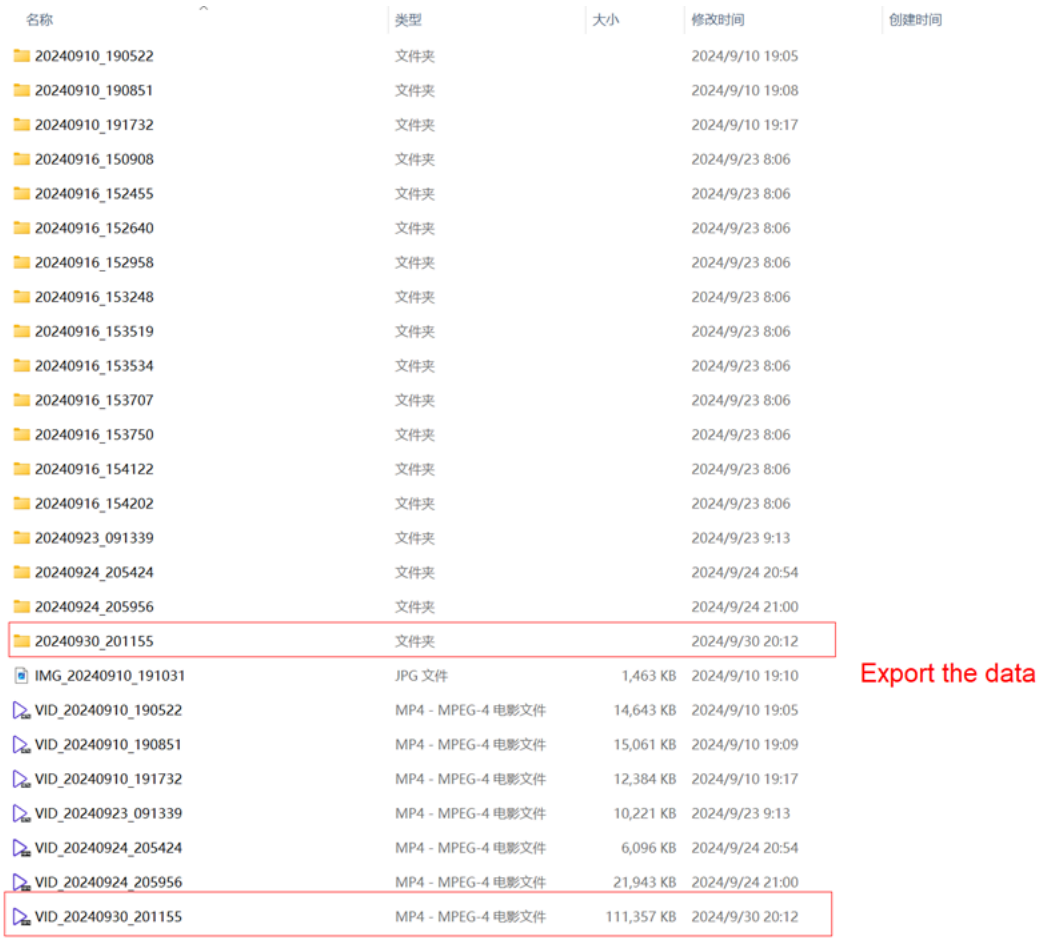

Figure A.17

A.2.3.2. Data Preprocessing

The collected data need to be preprocessed before being fed into the subsequent process. The purpose of data preprocessing is to temporally align the collected data, extract frames from videos, and convert raw data into the required format.

The workflow of data preprocessing is shown below. It contains three main stages: frame extraction, IMU conversion and frame match. The result of the preprocessing is that the data collected by different software will be sampled to a frequency of 1 Hz and provide an unlabeled dataset with geopose data in the TrainingDML-AI format.

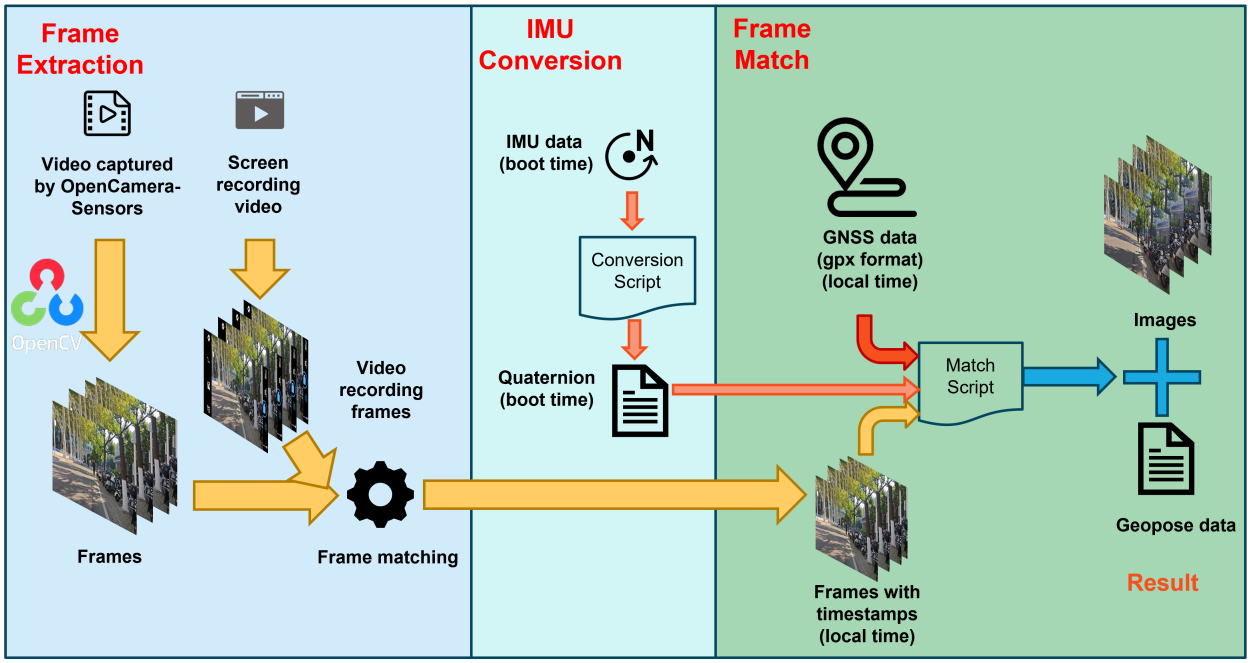

Figure A.18 — Workflow of Data Preprocessing

A.2.3.2.1. Frame Extraction

Video captured by OpenCameraSensors (main video) doesn't have any timestamps. To solve this problem, screen recording video (support video) is saved to calculate the timestamp for them. Specifically, firstly, videos would be cut into frames (main frames and support frames) using OpenCV. Then the same frame could be found manually. The timestamp on the support frame could be referred as the timestamp for same frame in main frames. Since the main video are collected in 30Hz, precise timestamps could be calculated for main frames. It should be noted that the timestamp in this stage is the local time. Main frames with timestamp could be used in subsequent processes.

A.2.3.2.2. IMU Conversion

The purpose of IMU conversion is to convert the collected IMU data into geopose. The IMU data collected by the OpenCameraSensors includes the accelerometer data, gyroscope data, and magnetometer data, each with a timestamp. Note that the three data don't have the same frequency and start time, so they should be pre-processed using algorithms such as the Kalman filter, converted to 30Hz geopose data for subsequent processing. As it is not the main research focus of this pilot, the algorithm for converting IMU data to geopose will not be elaborated here. The converted data is encoded in xml format, using x, y, z, w to represent the pose of the camera, each pose records a timestamp of the boot time since the mobile phone was booted.

A.2.3.2.3. Frame Match

This stage aims to match frames, and prepare ready-to-label dataset. The result of this stage is images with geopose data. The manual work required at this stage is to calculate the difference between the boot time and the local time. This can be done by the frames, which has boot timestamp, provided by the app. Then the time in IMU data according to this interpolated value. The next step is to sample all the data to a frequency of 1Hz. To simplify the process, the frames and geopose data closest to the sampling time of the GNSS data are selected. Therefore, the output data has a frequency of 1Hz, which is the same as the lowest of all the data sampled, i.e., the GNSS data. The GNSS data collector can be changed if the sampling frequency needs to be increased to obtain higher accuracy data. The result of preprocessing will be provided as an unlabeled dataset in TrainingDML-AI format. Frames will be organized into frame sequences. Metadata including identification, name, amount and geopose data will be recorded in JSON format, and because of the extensibility of TDML, labels can be extended into it after the annotation.

A.2.3.3. Data Labeling

In annotation stage, Label Studio, an open-source labeling tool, is used to enable collaborative labeling of data. The annotation can be exported in JSON format, so that it can be converted into TDML format.

A.2.3.4. TrainDML Conversion

The collected data and annotations need to be converted into the TDML format to enable interoperability. Specifically, the TDML-format dataset markup files are encoded in JSON format, which include:

- Dataset metadata: The dataset metadata includes information such as the id, name, dataset size, data provider, created time, and bands.
- Task information: The task information records the tasks for which the dataset can be used. In UDTIP, based on the requirements of Geo-AI tasks and the format of data annotation, tasks can be either scene classification tasks (scene-level) or semantic segmentation tasks (pixel-level).
- Per-image metadata: The metadata for each image includes the image's id, URL, collection time, and geopose. According to the needs of the pilot, geopose is added as an attribute to the metadata. Since it is not defined in the TDML standard, it is incorporated into the image's metadata in accordance with the Basic-Quaternion in Geopose 1.0 Data Exchange Standard encoding.
- Annotation details: The annotation information includes two aspects: the dataset's category codelist and the annotation results for each image. The codelist is encoded as an attribute of the dataset and can be extended according to the task requirements. For this pilot, possible categories include: Paved Road, Asphalt Road, Concrete Road, Gravel Road,

Dirt Road, and Unpaved Road.The labels defined in the codelist can be referenced in the data labels and serve as the annotations for the images.

A demonstration of the TDML conversion result is as follows:


```
{
    "type": "AI_EOTrainingDataset",
    "id": "D101_example",
    "name": "D101_example",
    "description": "Example dataset for UDTIP D101.",
    "version": "1.0",
    "amountOfTrainingData": 47,
    "createdTime": "2025-01-01",
    "providers": [
        "UDTIP"
    ],
    "bands": [
        {
            "name": [
                {
                    "code": "red"
                }
            ]
        },
        {
            "name": [
                {
                    "code": "green"
                }
            ]
        },
        {
            "name": [
                {
                    "code": "blue"
                }
            ]
        }
    ],

    "classes": [
        {
            "key": "Paved Road",
            "value": null
        },
        {
            "key": "Unpaved Road",
            "value": null
        },
        {
            "key": "Asphalt Road",
            "value": null
        }
    ],
    "numberOfClasses": 3,

    "tasks": [
        {
            "type": "AI_EOTask",
            "id": "whu_example",
            "taskType": "Scene Classification"
```

```
        }
    ],
    "data": [
        {
            "type": "AI_EOTrainingData",
            "id": "road_0000",
            "dataURL": [
                "dataset/frame_0000.jpg"
            ],
            "dataTime": "2024-12-01",
            "geopose": {
                "position": {
                    "lon": 114.35413057,
                    "lat": 30.53005352,
                    "h": 19.744
                },
                "quaternion": {
                    "x": -0.6948143326140601,
                    "y": -0.17674656664355012,
                    "z": 0.6955324418081127,
                    "w": -0.04720505021965216
                }
            },
            "labels": [
                {
                    "type": "AI_SceneLabel",
                    "class": "Asphalt Road"
                }
            ]
        },
        ...
    ]
}
```


Listing A.3 — Example - TDML Conversion Result

A.3. Annex for D102

A.3.1. D102 — UCF TDML-AI Pipeline Methodology

The proof-of-concept labeling followed these steps:

1. Evaluate the retrieval of OpenStreetMap (OSM) data via APIs, such as Overpass and Nominatim, were evaluated.
    a) Position data from GeoPose data could be used as query parameters to perform reverse geocoding.
    b) The returned data contained OSM data about that specific location, such as road classification.

c) The retrieved classification could be used as a label for the image the position data corresponded to.

2. Use downloaded OSM datasets from Geofabrik for deeper analysis.

a) The datasets contained shape files were specific to a region and organized by map feature.

b) Use dataset on region's roads to cross-reference GeoPose location data. This process was similar to reverse geocoding but returned data specific to roads within the dataset's respective region.

c) Use road classification from returned data to label images.

Manually annotating images followed these steps: . Evaluate multiple labeling platforms: Label Studio, SuperAnnotate, Computer Vision Annotation Tool (CVAT) . Label images using CVAT's magic wand tool. This tool automatically generated bounding boxes around visually similar objects. . Export annotation data of labeled images in COCO 1.0 dataset format.

Facilitate the conversion of GeoPose data into the TDML-AI format followed these steps: . Reviewed OGC's documentation, which provided foundational understanding and examples for encoding data into JSON formats, based on the TDML-AI standard. . Aggregated all data that would be used inputs. GeoPose data was created internally, annotation data was exported from CVAT and dataset metadata was written manually. . Created TDML-AI dataset by synthesizing data from all inputs. . Validated the created TDML-AI dataset to ensure it met all requirements outlined by OGC.

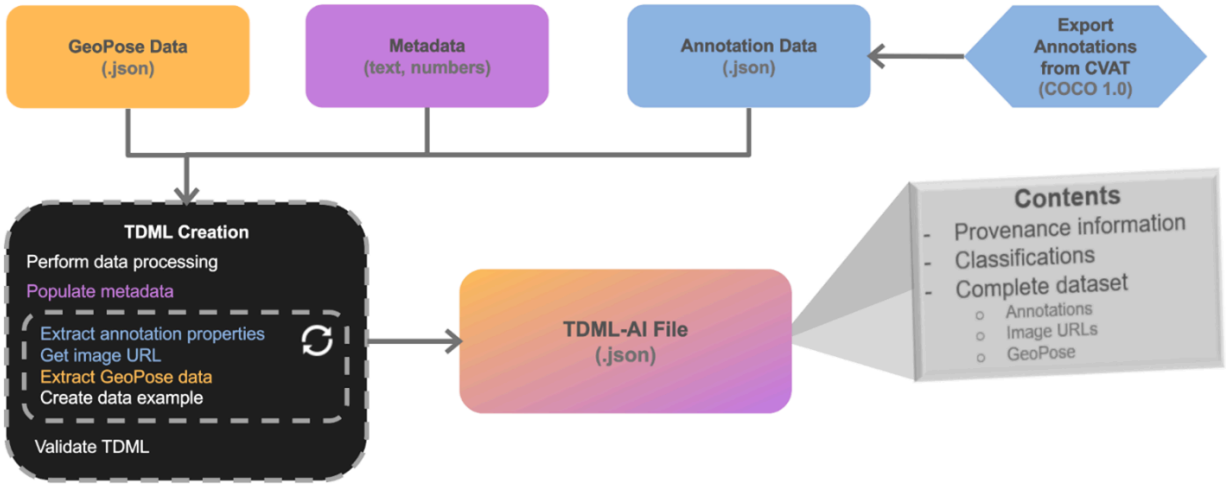

Figure A.19 — Geopose to TDML-AI Pipeline

Testing Specifications for GeoPose Data To evaluate performance improvements that could be realized through revising the encoding format for more efficient data extraction, extensive testing was performed:

Testing Environment

OS: Linux (Google Colab)
Processor: x86_64
Python version: 3.10.12

Test Parameters

num_runs: Number of rounds of testing.
num_iterations: Number of executions within each round.

Test Strategies (i.e. methods for structuring GeoPose frame parameters)

- String (current way of encoding parameters)
- Arrays (abstracting the translation and rotation parameters into arrays of values)
- Nested Object (abstracting every parameter value into its own key)

Test Steps

1. Encode a single set of parameters in each strategy
2. Under each strategy, execute code that extracts the latitude and longitude parameters a number of times equal to num_iterations.
3. Repeat the prior step a number of times equal to num_runs
4. Adjust test parameters and repeat steps 2 and 3

Table A.2 — Test Results

ITERATIONS	RUNS	STRATEGY 1 SPEED INCREASE (TIMES FASTER)	STRATEGY 2 SPEED INCREASE (TIMES FASTER)
1,000	5	6.98	4.92
10,000	5	7.18	6.31
10,000	10	7.64	6.76
100,000	5	4.02	3.81

Figure A.20 — Execution Times JSON

A.3.2. D102 — Wuhan Data Labeling Guide

1. Install and start the Label Studio (If there is already an instance, skip this step)

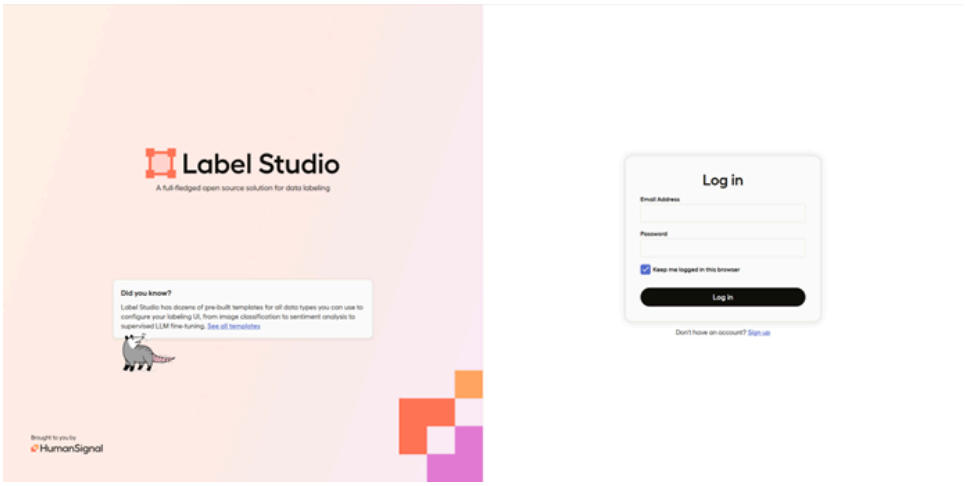

Figure A.21

a) Install: `pip install label-studio`

b) Start: `label-studio start`

c) Open Label Studio at http://localhost:8080.

2. Sign up with an email address and password that you create.

3. Create a project (Skip this step if the project has been created)

a) Name the project.

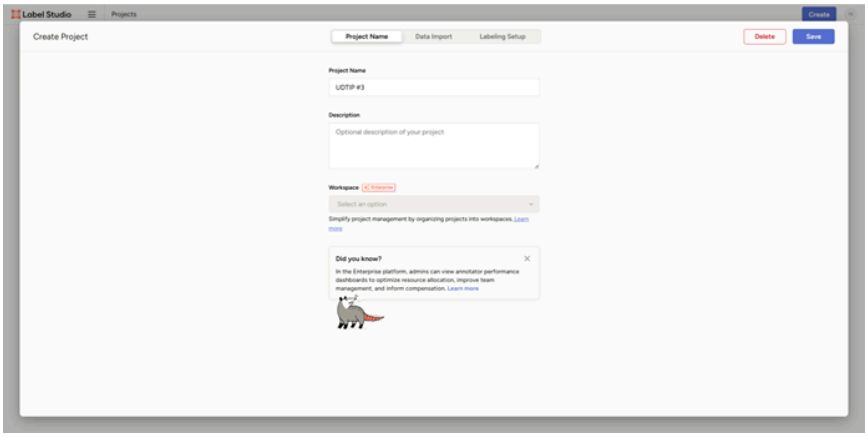

Figure A.22

b) Import your imagery data.

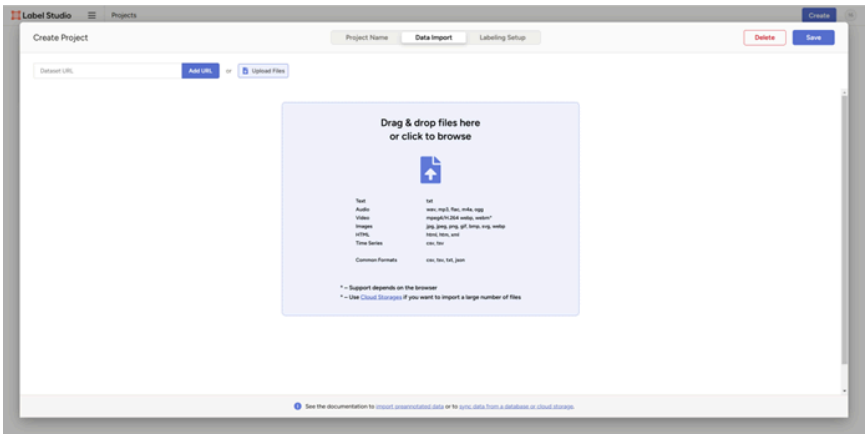

Figure A.23

c) Choose a label format (Classification task is chosen here as an example).

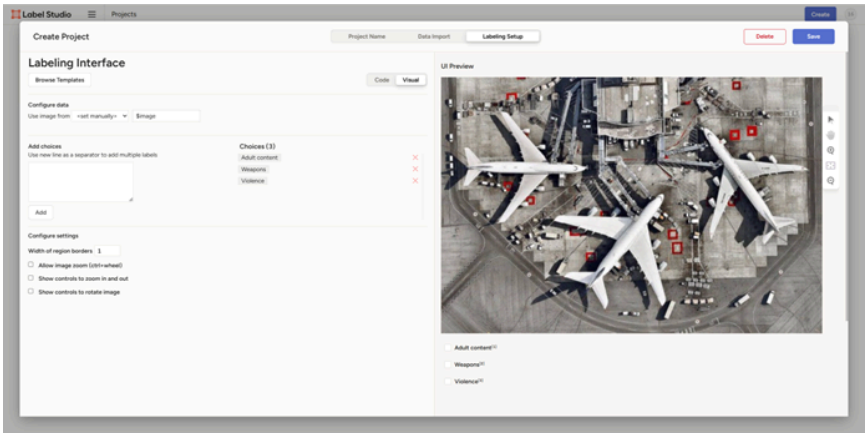

Figure A.24

d) Add label choices for the project, and save it.

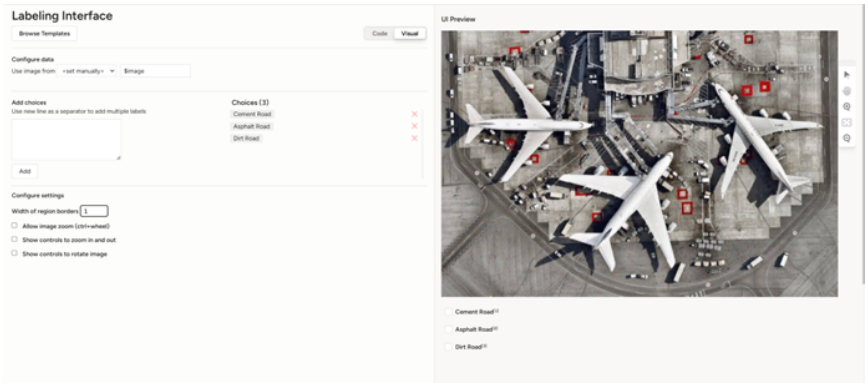

Figure A.25

4. Start labeling

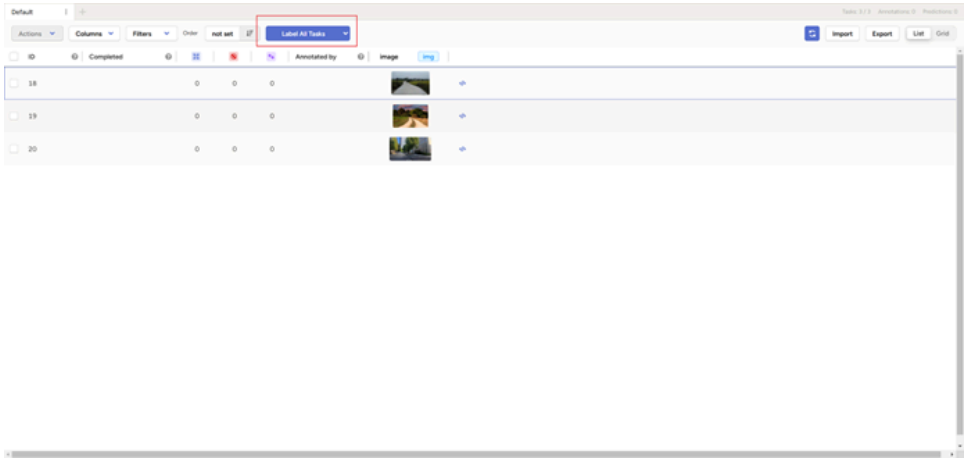

Figure A.26

a) Select a label for each image.

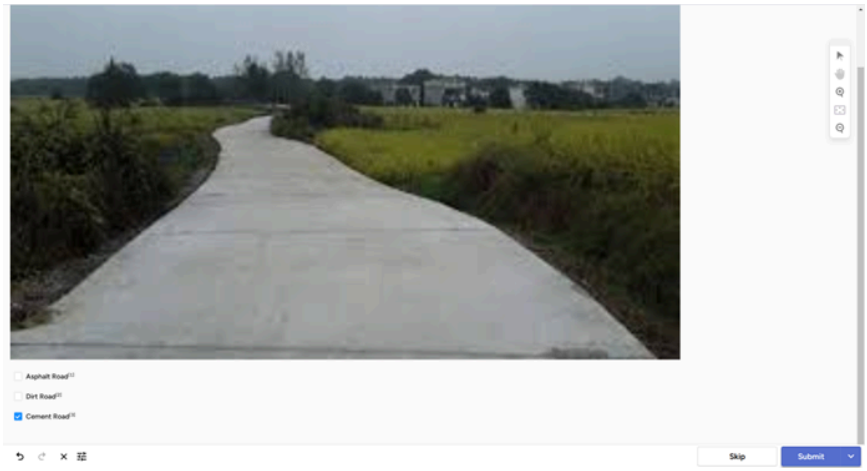

Figure A.27

b) Export the result

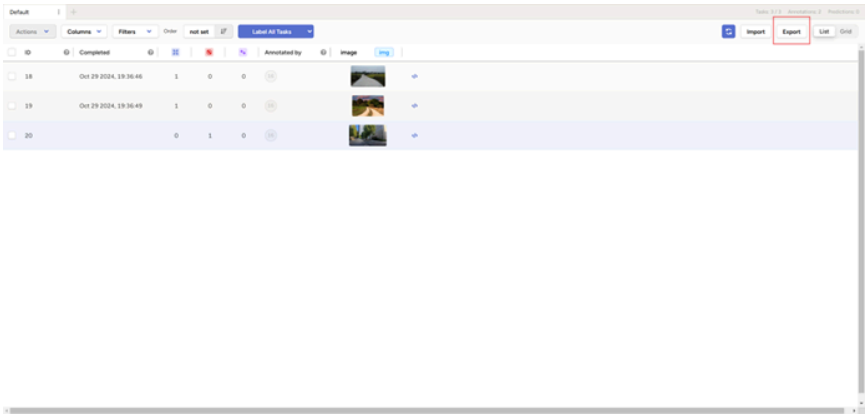

Figure A.28

c) After all of the images are labeled, you can export your label data from the Label Studio.

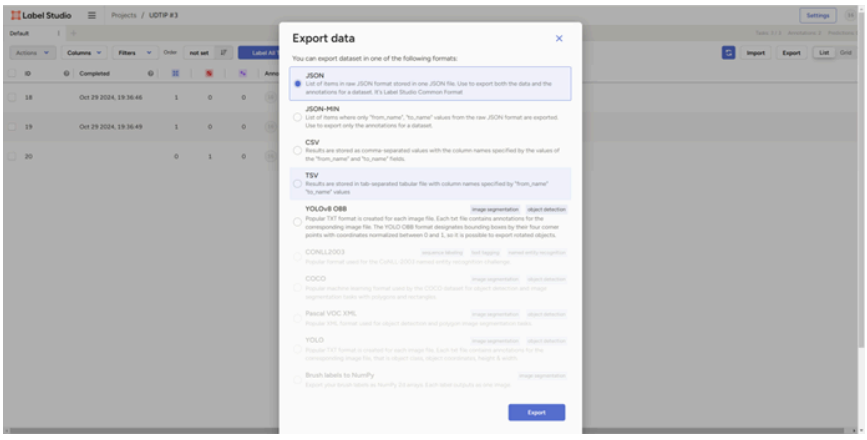

Figure A.29

d) It is recommended to be exported in JSON format, and we can convert the labels from JSON format to TDML format.

A.3.3. D102 — USC

A.3.3.1. Purpose of Geo-AI for Classification of Road Surface Types

As an example of Geo-AI analysis in D102, we developed an image machine learning model to automatically classify road types. For a given geo-referenced image dataset, the classification model generates a prediction of road surface types in three classes — Asphalt, Paved, and Unpaved.

A.3.3.2. Geo-AI Pipeline

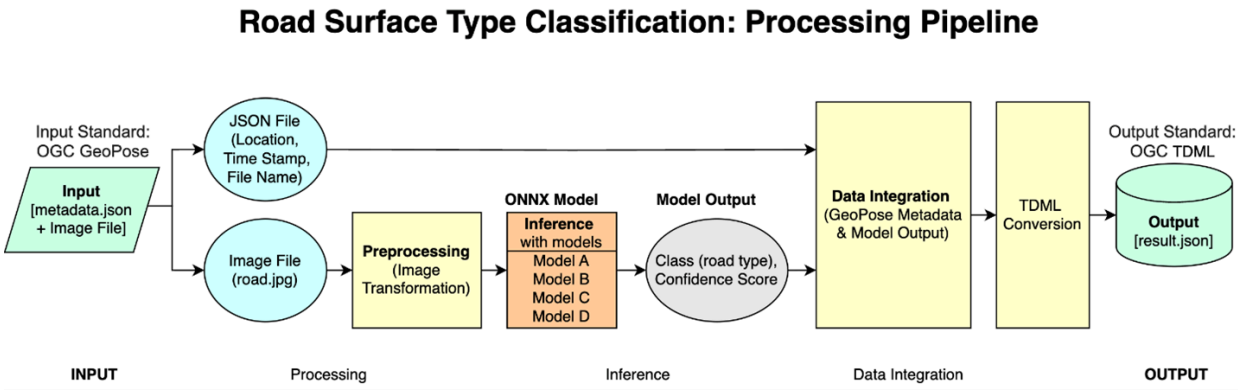

Figure A.30 — Geo-AI Pipeline in D102

Geo-AI analysis follows input/output standard as defined in the OGC GeoPose and TDML. Geo-AI module reads an image with corresponding geo-coordinates in the standard form. Then, the image is analyzed by a classification model (e.g., road type classifier in this report but any other analysis module can be implemented and applied). Analysis result is stored with the corresponding geo-coordinates in TDML format.

We used a public dataset, the Road Traversing Knowledge (RTK) Dataset, for road type classification modeling, which include labels as the same way we target; Asphalt, Paved, and Unpaved. Additionally, we utilized a set of images collected from Internet to augment the dataset. The image dataset were captured from low-cost cameras such as smartphones. The labelled image dataset was divided into training, evaluation, and test dataset for supervised learning.

Table A.3 — Example of Road Surface Types

<Asphalt> <Paved> <Unpaved>

Multiple deep learning models were applied and evaluated: ResNet50, ResNet1-v2, InceptionV3, DenseNet121, and EfficientNet_b0. For evaluation, a larger and diverse road surface test dataset was used with varied backgrounds (2506 images per each class). Our experimental results demonstrated that ResNet-based models performed strongly across all metrics. ResNet-based models are designed to handle deep networks using residual connections to avoid problems such as vanishing gradients. This allows them to effectively learn both simple and complex patterns. In the road surface type classification dataset, not only color but also texture-based features (cracks, granularity, wear, etc.) are important factors and ResNet-based models work well with those factors.

Table A.4 — Experimental Results of Various Models

InceptionV3	Precision	Recall	F1-score
asphalt	0.8780	0.9625	0.9183
paved	0.9305	0.8180	0.8707
unpaved	0.9287	0.9517	0.9401

Resnet50	Precision	Recall	F1-score
asphalt	0.9539	0.9992	0.9760
paved	1	0.9757	0.9877
unpaved	0.9992	0.9761	0.9875

Resnet152-v2	Precision	Recall	F1-score
asphalt	0.9672	1	**0.9833**
paved	1	0.9844	**0.9922**
unpaved	0.9976	0.9792	**0.9883**

Densenet121	Precision	Recall	F1-score
asphalt	0.9396	0.9992	0.9685
paved	1	0.9621	0.9807
unpaved	0.9992	0.9737	0.9863

Efficientnet_b0	Precision	Recall	F1-score
asphalt	0.9063	1	0.9509
paved	1	0.9753	0.9875
unpaved	1	0.9214	0.9591

A.3.3.3. Geo-AI Result Visualization

Geo-AI analysis follows standard input/output as defined in OGC GeoPose and TDML. The actual locations of the images in RTK dataset are not known but the overall route where the images were captured is reported in[1]. So, we generated synthetic locations of test images. To match the number of images in the test dataset, corresponding points along the road network were sampled. Each sampled point was assigned a location based on the class label of the corresponding image in the dataset. Classification results are shown using colored dots: Blue — Paved, Yellow — Unpaved, and Red — Asphalt.

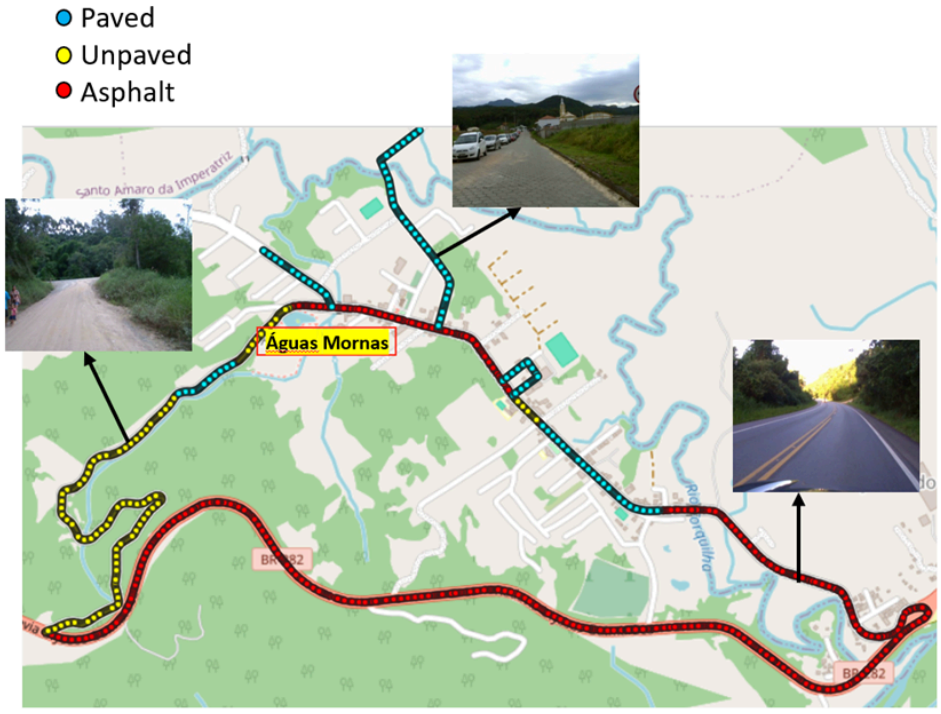

Figure A.31 — Visualization of Road Type Classification Results

[1]Rateke, T., Justen, K. A., & Von Wangenheim, A. (2019). Road surface classification with images captured from low-cost camera-road traversing knowledge (rtk) dataset. Revista de Informática Teórica e Aplicada, 26(3), 50-64.

A.3.4. D102 — Helyx

A.3.4.1. Helyx Road Classification API Overview

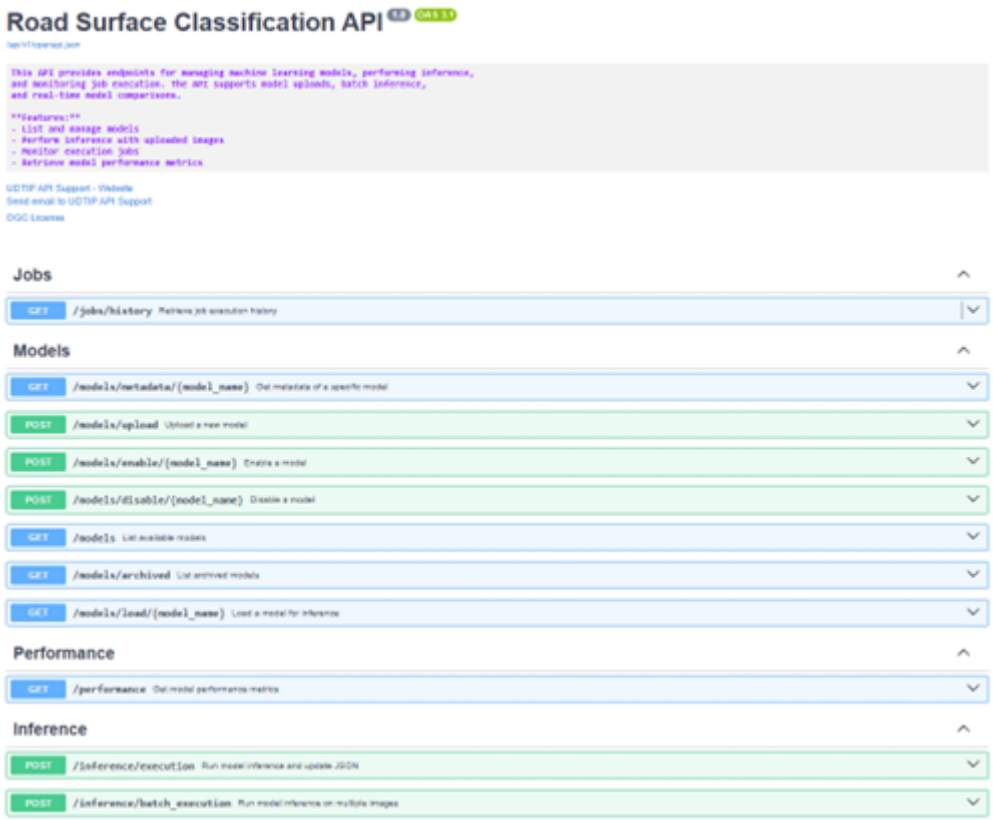

Figure A.32 — Road Classification API Endpoint Overview

The API developed for this part of D102 enables the management of trained Machine Learning models used to classify road surfaces through the use of Image Classification.

Users can load models, interact with uploaded models, and conduct inference on images supplemented with TDML and GeoPose data to retrieve a prediction and confidence of the road surface type.

A.3.4.2. Model Training

At first, a model with custom weights was trained. The training process involved two distinct architectures: SimpleCNN and EnhancedCNN (ResNet50). Each model was designed to classify road surface types such as asphalt, concrete, and unpaved roads.

SimpleCNN Architecture

The SimpleCNN model is a lightweight convolutional neural network (CNN) designed for efficient road surface classification. It consists of:

- Three convolutional layers with increasing filter sizes (16, 32, and 64 filters) to extract spatial features from the input images.
- ReLU activation functions after each convolutional layer to introduce non-linearity.
- Max pooling layers to reduce the spatial dimensions and retain essential features.
- A fully connected classifier with two linear layers:
    - The first linear layer maps extracted features to a 128-unit hidden layer with ReLU activation and dropout (to prevent overfitting).
    - The final layer maps to three output classes (asphalt, concrete, and unpaved), producing logits for classification.
    - This model was trained using the cross-entropy loss function and optimized with Adam to adjust the model's weights. Due to its simple structure, SimpleCNN is efficient but may struggle with complex variations in road textures.

EnhancedCNN Architecture (ResNet50-based)

To improve classification accuracy, a more advanced architecture based on ResNet50 was also trained. EnhancedCNN is a fine-tuned version of ResNet50, leveraging pre-trained weights on ImageNet to capture richer feature representations. The ResNet50 backbone consists of 50 layers, including residual blocks that enable deeper training without vanishing gradients.

The original fully connected layer was replaced with:

- A 128-unit dense layer with batch normalization, ReLU activation, and dropout to improve generalization.
- A final output layer that maps features to the three surface types.

This model benefits from ResNet50's deep feature extraction, allowing it to distinguish fine details in road textures. However, it requires more computational resources compared to SimpleCNN.

Training Process

Both models were trained on a dataset of road surface images with corresponding labels. The training process included:

Data Preprocessing

- Images were resized to 224×224 pixels to match model input requirements.
- Data augmentation (random rotations, flips, and brightness adjustments) was applied to improve model robustness.

- Images were normalized to a [0,1] range for better convergence.

Hyperparameters

- Batch size: 32
- Learning rate: 0.001 (adjusted with decay)
- Optimizer: Adam
- Loss function: Cross-entropy loss
- Number of epochs: 20 (with early stopping based on validation loss)

Evaluation Metrics

- Accuracy: Measures how often the model correctly classifies a road surface type.
- Precision & Recall: Helps assess the model's ability to distinguish between similar surfaces.
- Confusion Matrix: Visualises classification performance across categories.

Results & Observations

SimpleCNN performed well on clear, well-lit road images but struggled with complex surfaces and shadows.

EnhancedCNN achieved higher accuracy due to deep feature extraction but required more memory and computation time.

Both models were evaluated on a separate test set, and their performance was logged to track improvements over time.

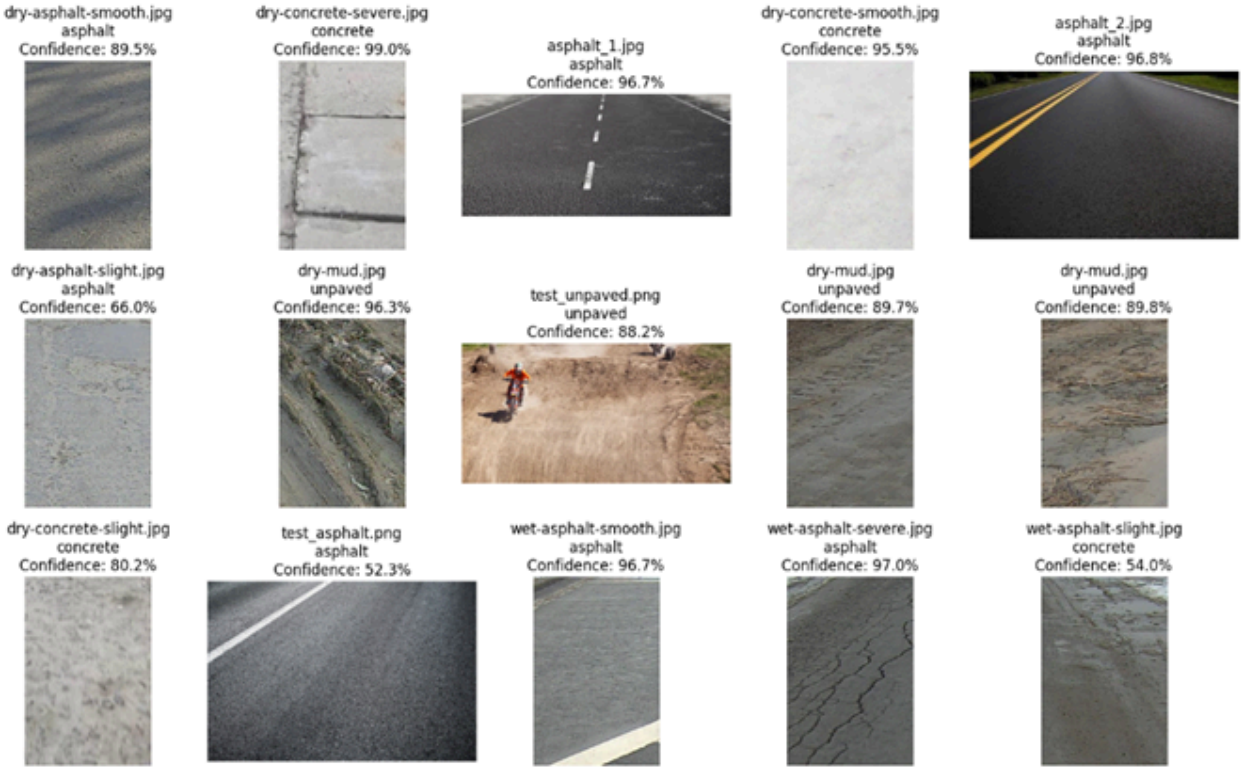

Figure A.33 — Predictions run on 15 Road Surface Images

A.3.4.3. API Endpoints

The Road Surface Classification API provides a set of RESTful endpoints designed for model management, inference execution, and performance monitoring. These endpoints allow users to upload, enable, and disable machine learning models, as well as run predictions on road images. Below is a detailed explanation of each endpoint, its purpose, and how it functions.

A.3.4.3.1. Model Management Endpoints

These endpoints allow users to manage machine learning models within the system, including uploading new models, enabling or disabling them, and retrieving model information.

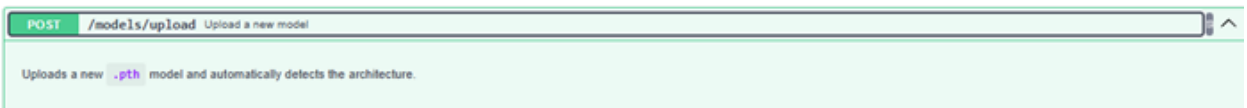

Figure A.34 — Upload a Model

- Endpoint:
  - POST /models/upload

- Description:
  - Allows users to upload a .pth model file to the system. The API automatically detects the model architecture and stores the model in the archive directory.
- Request:
  - uploaded_file: The .pth file containing the trained model weights.
- Response:
  - A confirmation message along with the detected model architecture.
- Example Usage:
  - A user uploads a model file called simple_model.pth, and the API identifies it as a SimpleCNN model.

Figure A.35 — Enable a Model

- Endpoint:
  - POST /models/enable/{model_name}
- Description:
  - Moves a model from the archive to the active models directory, making it available for inference.
- Request:
  - model_name:
    - The name of the model to enable (without .pth extension).
- Response:
  - A message confirming that the model is now active.
- Example Usage:
  - A user enables simple_model, allowing it to be used for inference.

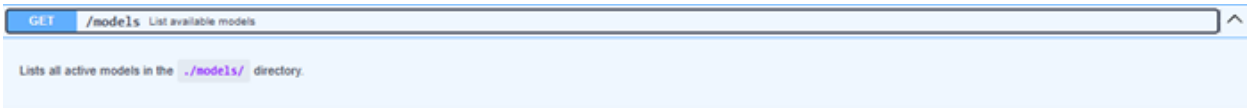

Figure A.36 — List Models

- Endpoint:
  - GET /models
- Description:
  - Retrieves a list of all active models available for inference.
- Response:
  - A JSON object containing the names of active models.
- Example Usage:
  - A user checks which models are currently available for inference.

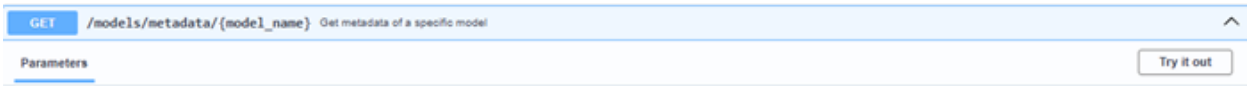

Figure A.37 — Get Model Metadata

- Endpoint:
  - GET /models/metadata/{model_name}
- Description:
  - Retrieves detailed information about a specific model, including its architecture, number of parameters, and file size.
- Request:
  - model_name: The name of the model.
- Response:
  - A JSON object containing metadata about the model.
- Example Usage:

- A user queries simple_model and receives its architecture (SimpleCNN), the number of trainable parameters, and file size.

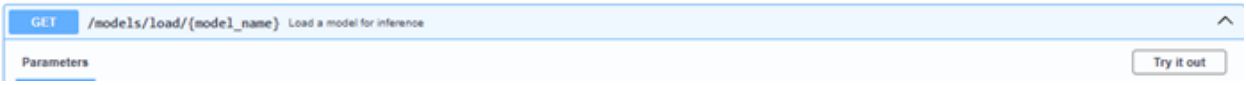

Figure A.38 — Load a Model

- Endpoint:
  - GET /models/load/{model_name}
- Description:
  - Loads a model into memory for inference.
- Request:
  - model_name: The name of the model to load.
- Response:
  - A message confirming that the model has been loaded.
- Example Usage:
  - Before running inference, a user loads enhanced_model into memory.

A.3.4.3.2. Inference Endpoints

These endpoints allow users to perform image classification using the active model.

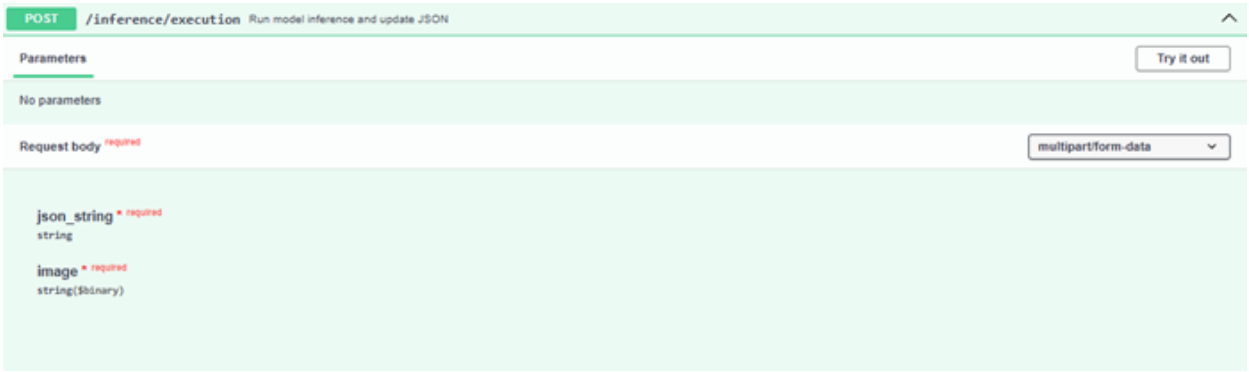

Figure A.39 — Run Inference on a Single Image

- Endpoint:

    - POST /inference/execution

- Description:
    - Runs inference on a single image, predicting the type of road surface.

- Request:
    - json_string: A JSON-formatted string containing metadata about the image.
    - image: The uploaded image file.

- Response:
    - The original JSON data with an appended prediction and confidence score.

- Example Usage:
    - A user uploads asphalt_road.jpg, and the API classifies it as “asphalt” with a confidence of 95%.

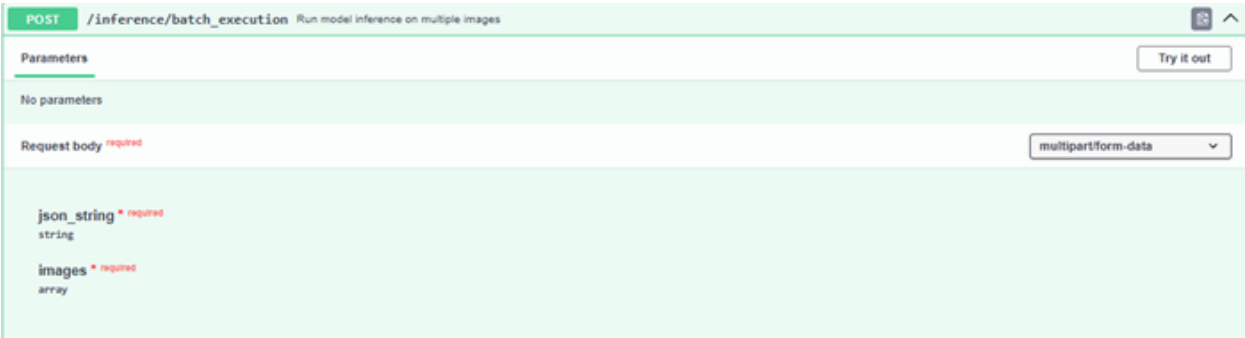

Figure A.40 — Run Batch Inference on Multiple Images

- Endpoint:
    - POST /inference/batch_execution

- Description:
    - Runs inference on multiple images at once, allowing efficient classification of large datasets.

- Request:
    - json_string: JSON metadata for multiple images.

- images: A list of uploaded image files.

- Response:
  - A JSON object containing predictions and confidence scores for each image.

- Example Usage:
  - A user submits 10 images of different roads, and the API returns a classification for each.

A.3.4.3.3. Performance & Logging Endpoints

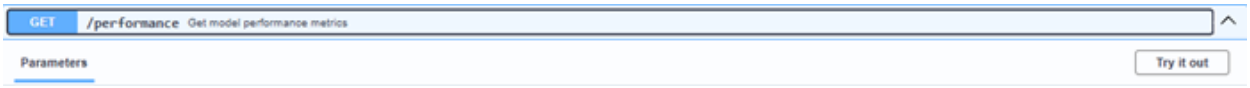

Figure A.41 — Retrieve Performance Metrics

- Endpoint:
  - GET /performance

- Description:
  - Returns performance statistics such as inference time and memory usage.

- Response:
  - JSON object with last inference time and memory usage.

- Example Usage:
  - A user checks how fast the model processes images and how much memory it consumes.

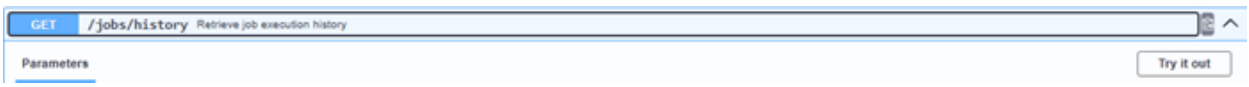

Figure A.42 — Retrieve Job History

- Endpoint:
  - GET /jobs/history

- Description:

    - Retrieves a log of previous API requests, including timestamps, endpoints called, and responses.

- Optional Parameter:
    - limit: (Optional) The number of recent job history entries to retrieve.

- Response:
    - A JSON object containing past API calls.

- Example Usage:
    - A user reviews past inference requests to track which images were classified.

A.4. Annex for D103

A.4.1. D103 — WiTech Inter-module Interoperability

An overview of the OGC API resource path is provided:

Table A.5 — The path and result from the OGC API resources.

Resource	Path	Result
Landing Page	/	Landing page in JSON or HTML
Conformance declaration	/conformance	Conformance in JSON
API definition	/api	API definition in JSON
Collection	/collections	All Collections in JSON/HTML
Collection (with bbox)	/collections?bbox=[bbox]	Filtered collections in JSON/HTML matching the bounding box [bbox]
Container	/collections/[container_id]	Collections of [container_id] in JSON
Container (with bbox)	/collections/[container_id]?bbox=[bbox]	Filtered Collections of [container_id] in JSON matching the bounding box [bbox]
Resources	/collections/[container_id]/[resource_format]	Data resources of [container_id] in the requested formats

The data server has been implemented using Node.js and the Express.js web framework and is publicly available. This implementation adheres to the development and design guidelines of the OGC API architecture. The service is deployed and can be accessed publicly.

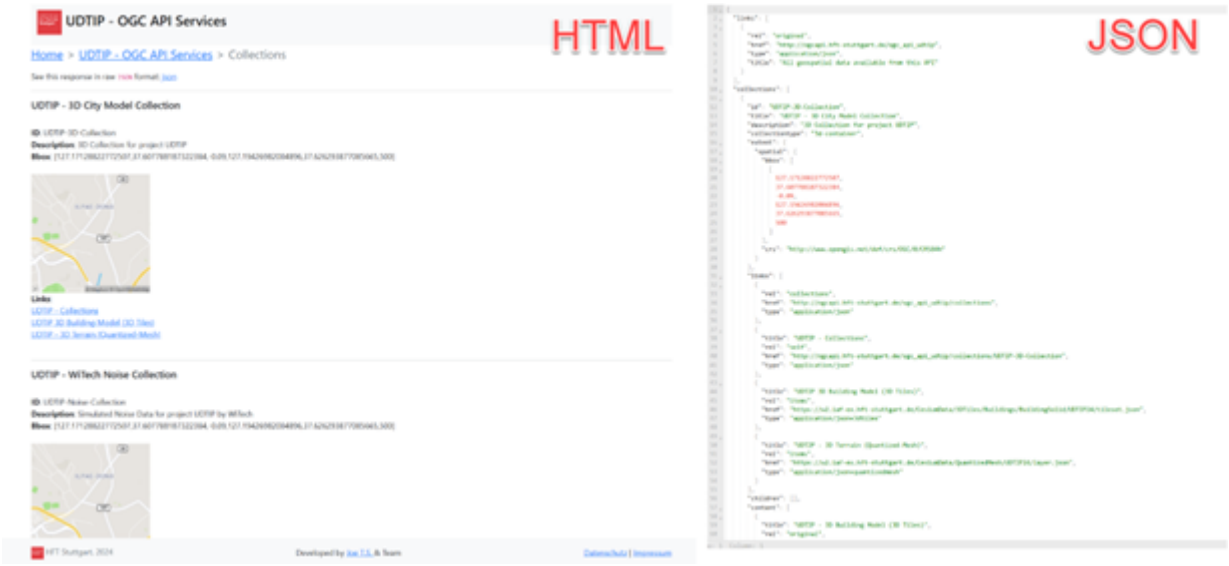

Figure A.43 — OGC API

A.4.2. D103 — Wuhan University Inter-module Interoperability

Based on the output data and the required input data of deliverables, interfaces following OGC API standards were implemented. The workflow of D103 is shown below.

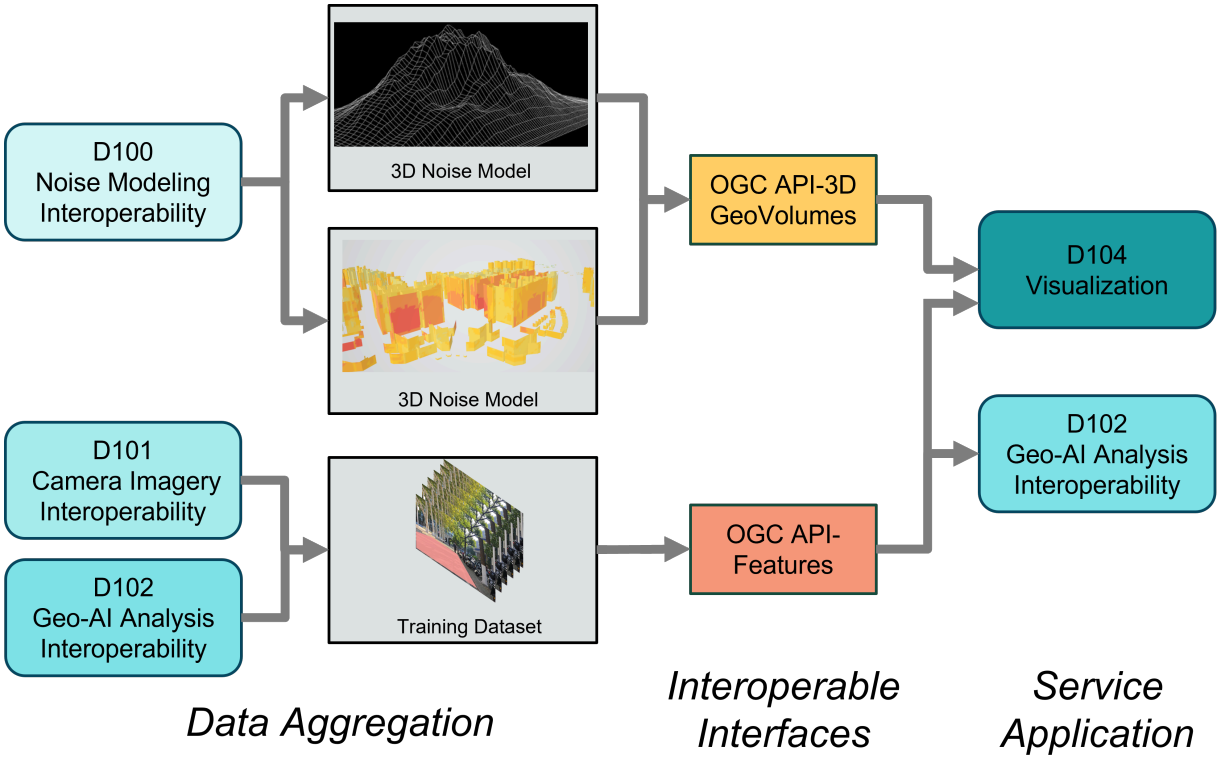

Figure A.44 — Workflow for D103

The back end was implemented using Python with Flask. The front end was secondary developmed from open-source project. The service is deployed locally and can be accessed publicly.

A.5. Annex for D104

A.5.1. D104 — Gaia3D

Visualizing the noise modeling results in a digital twin environment plays a crucial role in maximizing the understanding and utilization of noise analysis data. Visualization provides an intuitive grasp of spatial distribution and temporal changes in noise, supporting important decision-making processes in urban planning and design.

This project uses Cesium.js, an open-source tool, for rendering. Building data is visualized in 3D Tiles format, while terrain is visualized in Quantized Mesh, supported by Cesium.

This visualization approach efficiently handles large-scale data and supports the representation of real-time changes in noise levels. 3D Tiles facilitate rapid rendering of complex urban models, and Quantized Mesh is suitable for high-resolution terrain data visualization.

A.5.1.1. Software Configuration

The visualization part of this project was implemented as a simple web service system. The data and software used are as follows (3D Tiles & 3D Terrain maker was developed in-house)

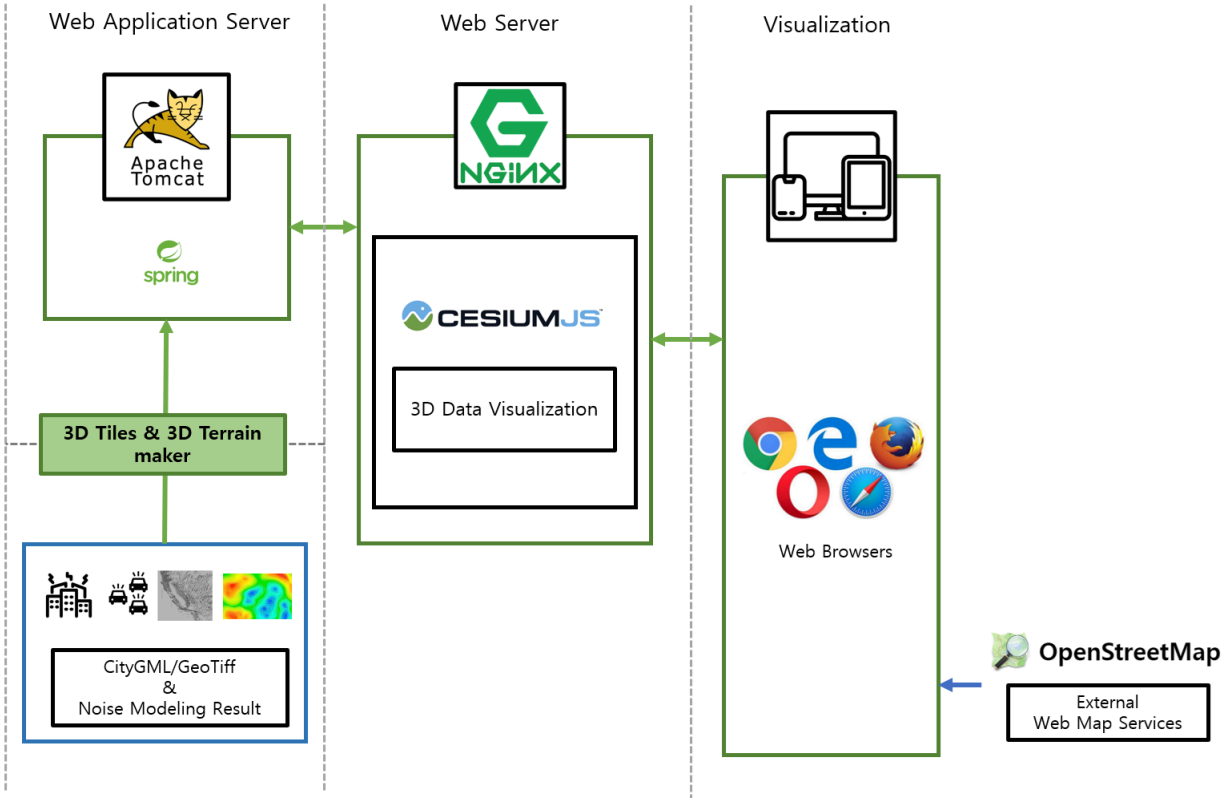

Figure A.45 — D104 Software Configuration

A.5.1.2. Data Conversion and Visualization Results

The building data and noise prediction analysis results entered in CityGML were converted into 3D Tiles, and the terrain data, GeoTiff, was converted into a Quantized Mesh supported by Cesium to enable visualization in Cesium, and the detailed results of each data visualization are as follows.

A.5.1.2.1. Building Data

Building data from CityGML was converted into 3D Tiles format using a custom-built conversion module. Below is an example of the converted and visualized building data.

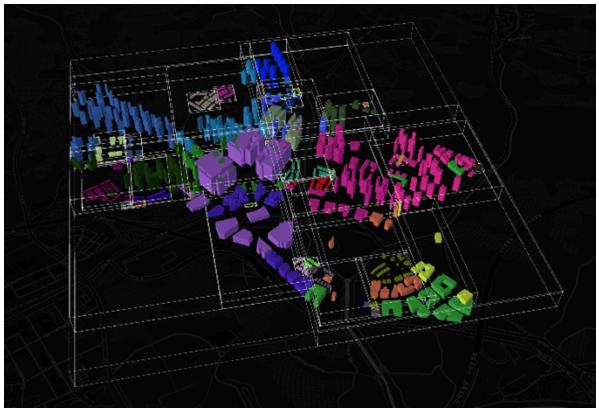

Figure A.46 — The visualization of the generated 3D Tiles in Cesium

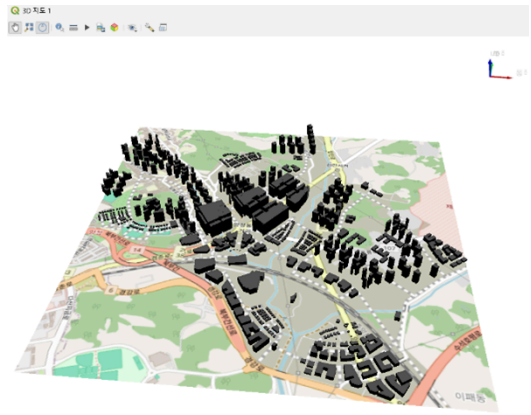

Figure A.47 — The visualization of the results in QGIS along with OSM

A.5.1.3. Terrain Data

A custom-built module was developed to convert the GeoTiff terrain data into Quantized Mesh format, and the results are as follows.

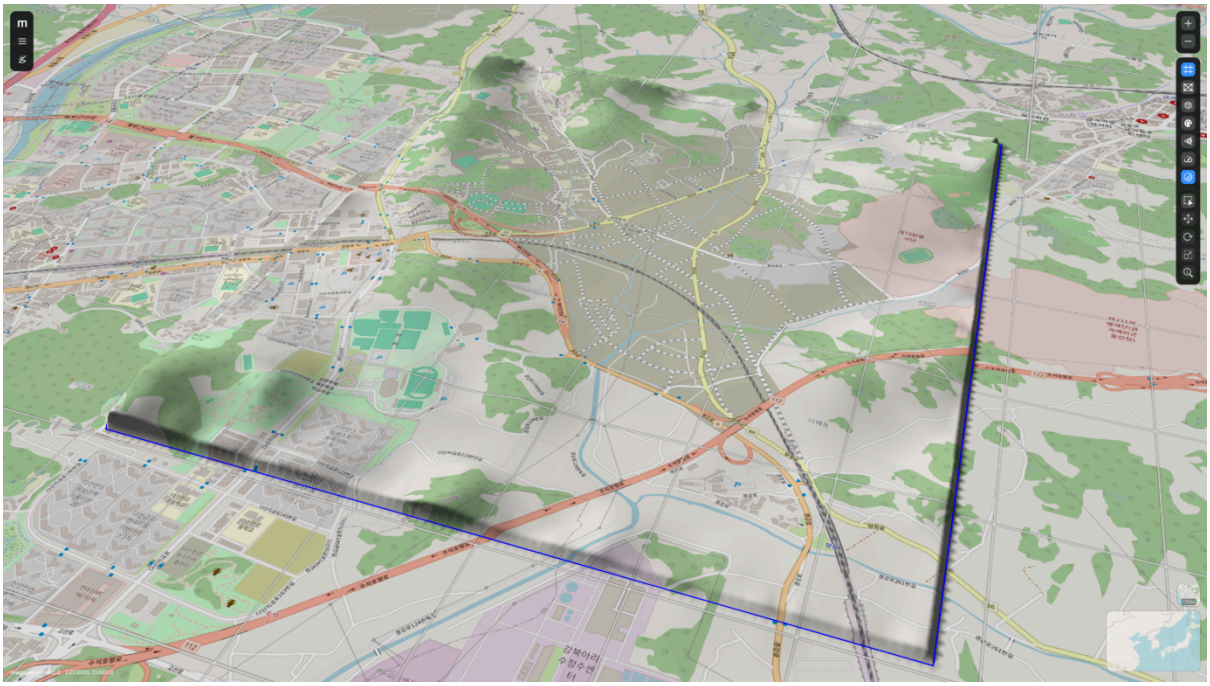

Figure A.48

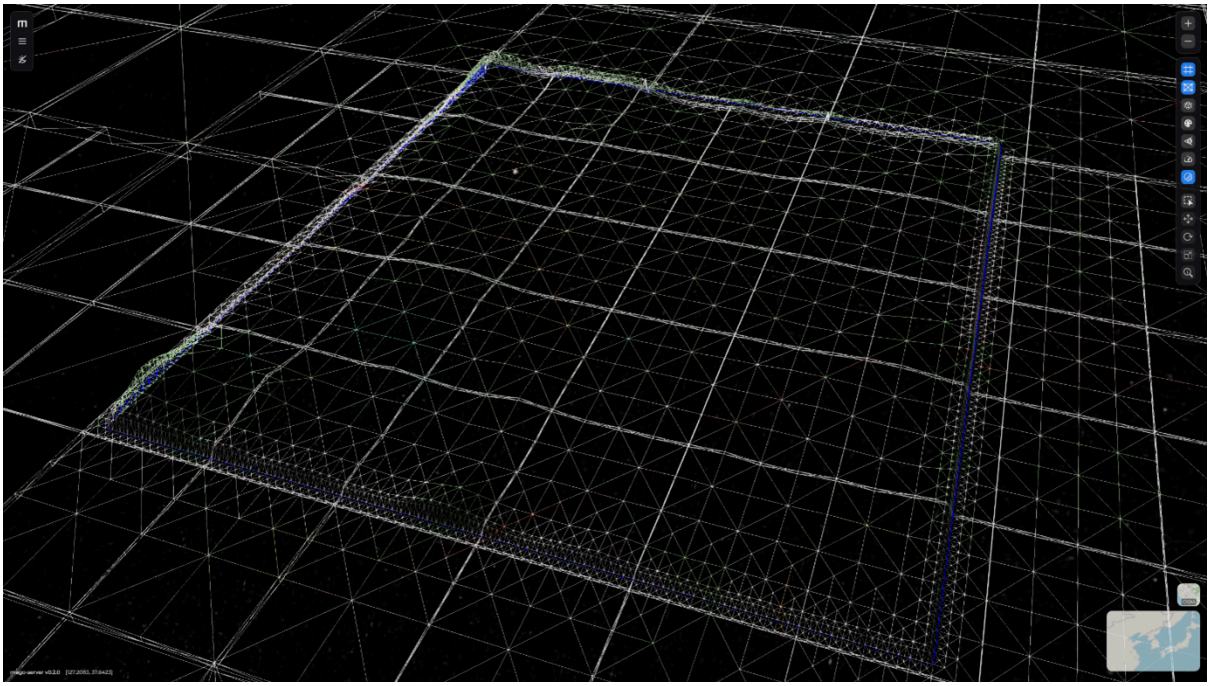

Figure A.49

Figure A.50

Quantized mesh visualized in Cesium (from top to bottom: terrain visualized with OSM, terrain visualized with wireframe, terrain visualized with building 3D Tiles)

A.5.1.4. Noise Analysis Results

A custom-built module was developed and applied to parse the output of the Noise Modeler used in this project and convert it into GLB format. The generated GLB was then repacked into 3D Tiles using the converter developed in section 2.3.2. The visualization of the result in Cesium is as follows.

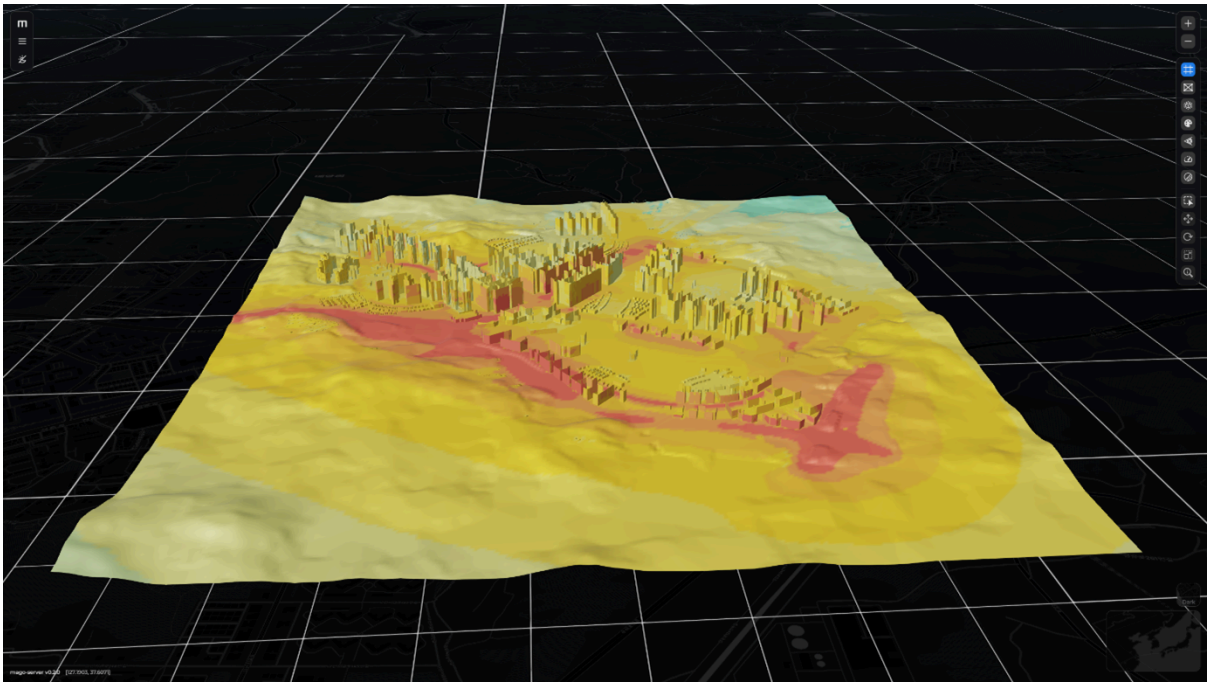

Figure A.51 — Noise analysis results converted to 3D Tiles and visualized in Cesium

A.5.1.5. Visualization

The open-source Cesium.js was used as the front-end visualization engine, and web-based visualization data formats such as 3D Tiles and Cesium Quantized Mesh were utilized.

The noise prediction results were visualized as separate, independent 3D objects, rather than applying them as texture materials to building data.

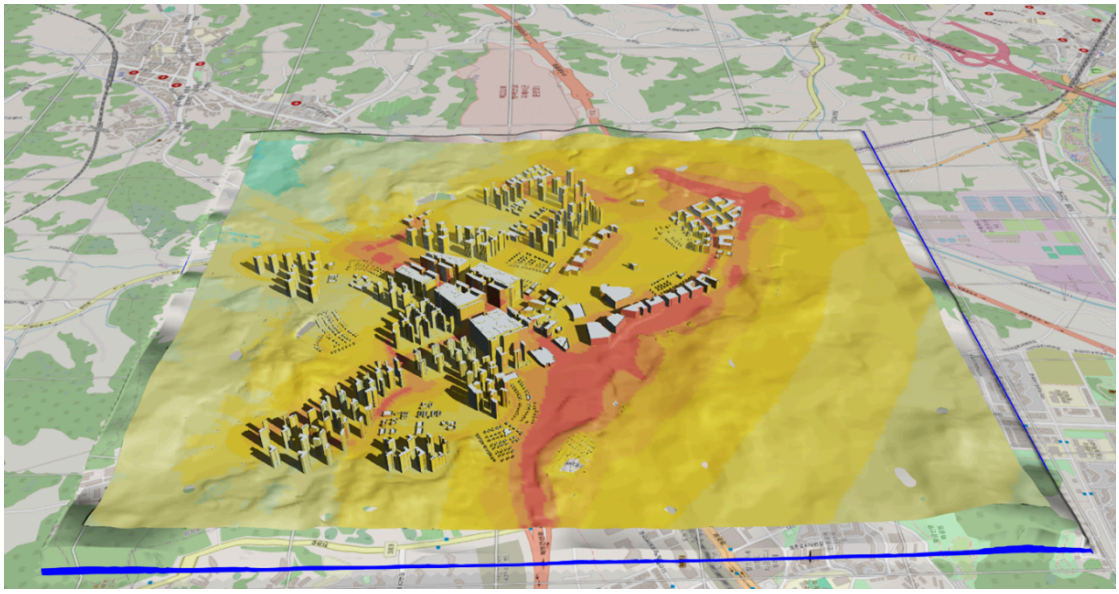

Figure A.52 — Noise analysis results visualized on top of OSM with terrain/buildings in Cesium

A.5.1.6. Legend

The legend for the noise analysis was configured based on the criteria outlined below.

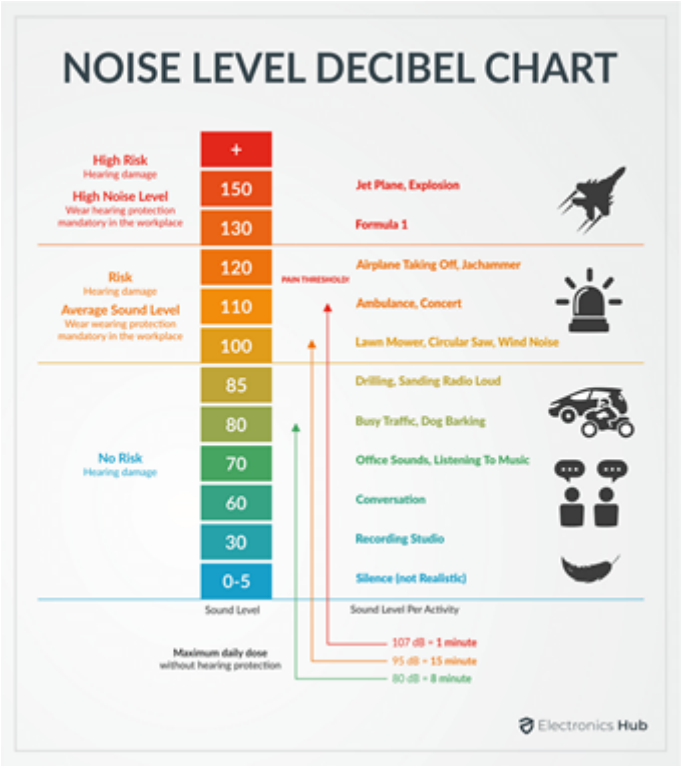

Figure A.53 — Legend in Noise Analysis

A.5.1.7. Converting CityGML into 3D Tiles

We converted the CityGML sample files into 3D Tiles by adding the ability to handle CityGML 3.0 files to the 3D Tiles creation tool developed by Gaia3D. (The 3D Tiles creation tool is open source and available at https://github.com/Gaia3D/mago-3d-tiler)

The schematic process flow for converting CityGML to 3D Tiles in this tool is as follows

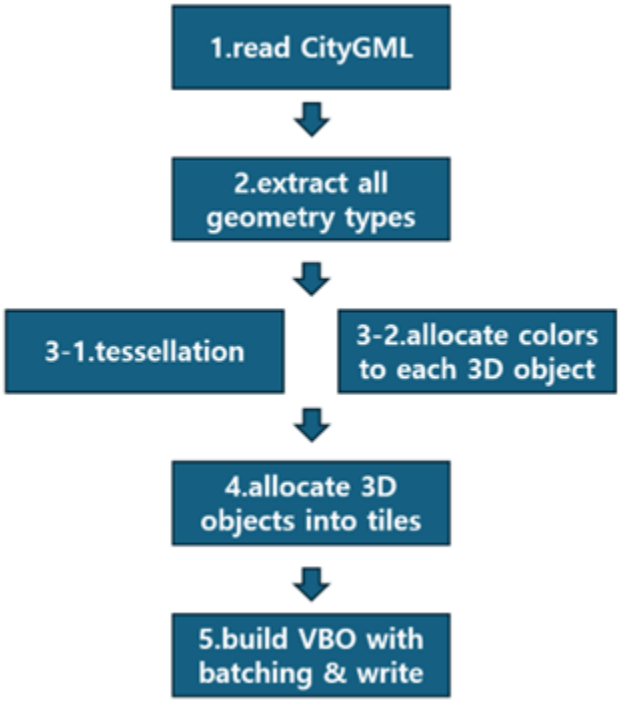

Figure A.54

1) read CityGML

2) extract all geometry types Extract all visible three-dimensional objects from the digital twin.

3-1) tessellation

CityGML uses polygons to describe surfaces, and the task of breaking up the polygons and converting them into a list of triangles for GPU rendering is done.

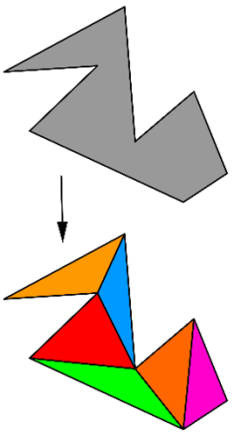

Figure A.55 — Example of a tessellation.

*Image source: https://en.wikipedia.org/wiki/Polygon_triangulation

3-2) allocate colors to each 3D object

To colorize three-dimensional objects according to their logical division, it extracts information about each object type and applies its own color assignment rules.

4) allocate 3D objects into Tiles
Categorize three-dimensional objects by size and assign them to tiles of the appropriate depth. When creating CityGML as 3D Tiles, we create a tileset that is 'additive', meaning that we do not allow duplicates and ensure that any three-dimensional object has only one tile that it belongs to.

5) build VBO with batching & write
In order to finally write to the 3D Tiles, they are processed into Vertex Buffer Objects (VBOs). At this point, a "stitching" operation is performed to combine the VBOs of all the 3D objects into a single mesh.

A.5.1.8. Converting DEM to Quantized Mesh specification

We developed a tool to convert DEM (GeoTIFF) to Quantized Mesh specification, a service format for 3D terrain visualization in CesiumJS published by Cesium.

*The tool is called mago 3DTerrainer, and as of the time of writing this ER, we are working on cleaning up the source code for the tool and plan to release it as open source before the end of this project.

*About Quantized Mesh: https://github.com/CesiumGS/quantizedmesh Quantized Mesh is a specification that has the same logical hierarchy as a tile pyramid made from 2D gridded data, except that each tile is assigned a 3D mesh data set of vertices/triangles instead of a 2D raster of pixels.

In our tool, we generated a terrain mesh in the form of RTIN (Right-Triangulated Irregular Network) based on GeoTIFF, assigned it to each tile of the tile pyramid, and saved the tiles as a Quantized Mesh specification. The bounding area and hierarchy of the tile pyramid applied in this tool are borrowed from OGC's GlobalCRS84Scale. (https://docs.ogc.org/is/17-083r4/17-083r4.html#toc43)

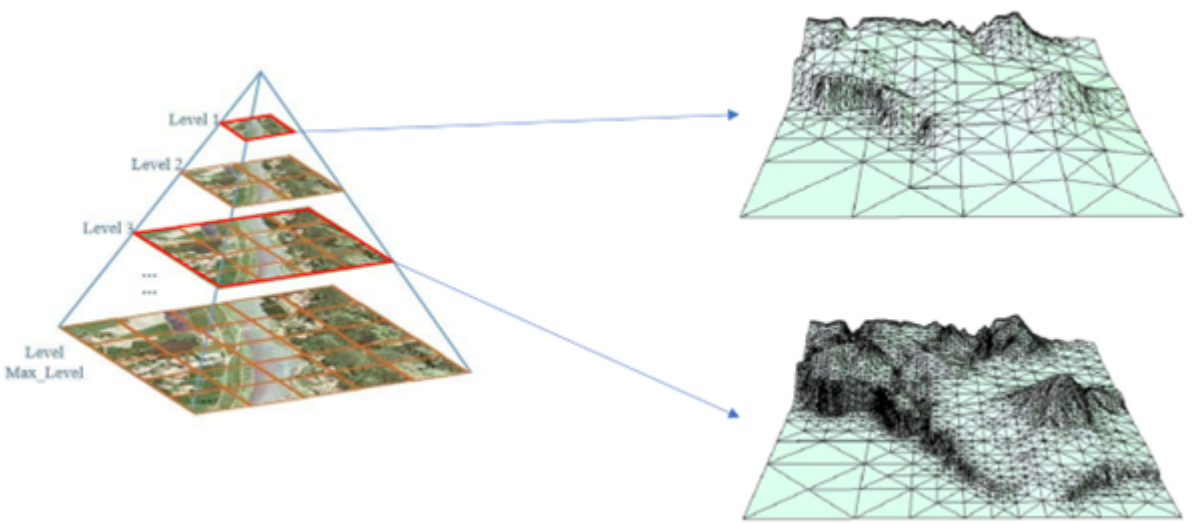

*Image source: https://www.sciencedirect.com/science/article/pii/S0895717708001040

Figure A.56

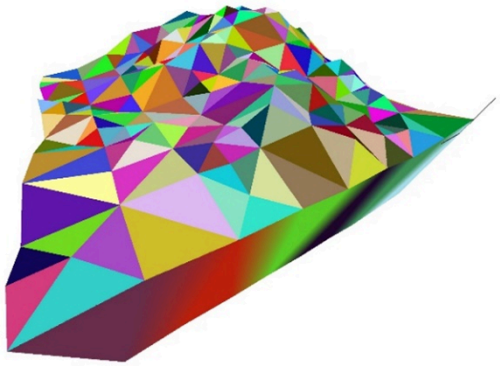

Figure A.57

A realistic view of a single Quantized Mesh tile generated with mago 3DTerrainer. Triangles are given a random color to differentiate between them

A.5.1.9. Noise Modeling

1. Noise modeler

   a) Noise modeler

      To obtain noise prediction analysis results, we delivered two GeoJSON files and a GeoTiFF of the primary workpiece mentioned in Annex A.1 to Createch, Inc. (Haeundae-gu, Busan, Republic of Korea, http://www.createch.co.kr/en/), which is collaborating with us to combine digital twins and noise prediction results, as input materials, and Createch analyzed the results using its specialized noise prediction analysis tool "ENPro, indoor & outdoor environment noise simulation S/W" and provided us with the analysis results along with specification information. The noise prediction model information supported by the specialized noise prediction analysis tool is as follows. ISO-9613-1:1993 Acoustics — Attenuation of sound during propagation outdoors Part 1: Calculation of the absorption of sound by the atmosphere (https://www.iso.org/standard/17426.html) ISO-9613-2:1996 — Attenuation of sound during propagation outdoors Part 2: General method of calculation (https://www.iso.org/standard/20649.html) * Part 2 is now ISO-9613-2:2024 (https://www.iso.org/standard/74047.html), Edition 2 has been published and 2:1996 is deprecated.

   b) Noise modeler's output structure and conversion to 3D Tiles Noise modeler uses DEM (digital elevation model), building data, and noise source data as input data, places noise prediction points on the ground and building surface, and predicts noise at each noise prediction point. The noise prediction results are provided in the form of a list organized by each receiving point.

      It also provides information about the rectangular network, which is a set of four noise estimation points that form a square. Each square is divided into two triangles, resulting in a triangular network with a noise prediction value assigned to each vertex. The final converted triangulation can be converted to a 3D mesh in the form of a heatmap by applying color assignment rules to it. The process is illustrated below

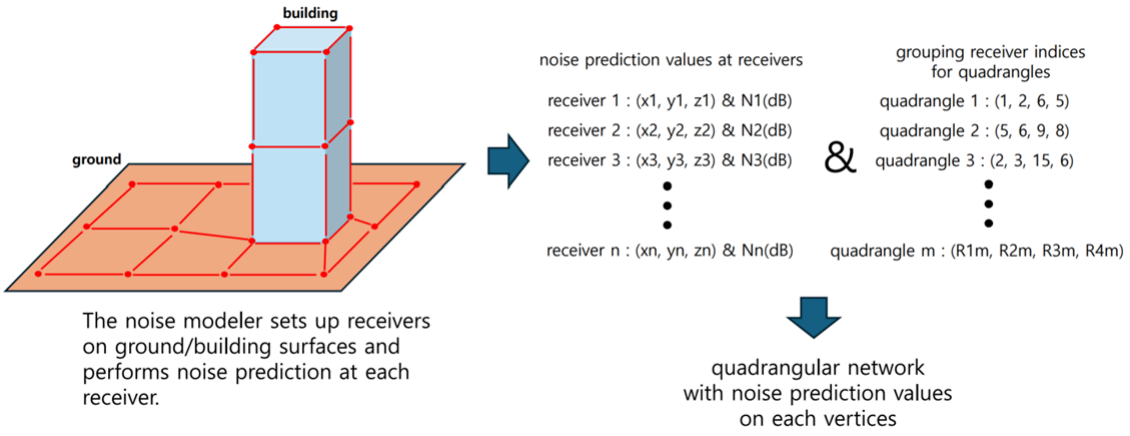

Figure A.58 — Output structure of the noise modeler

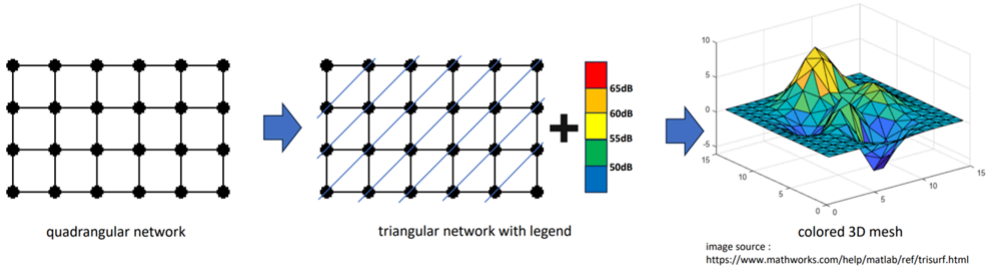

Figure A.59 — Converting the quadrangular network to 3D mesh

Figure A.60 — Legend in Noise Analysis

*Image source: https://www.electronicshub.org/noise-level-decibels-chart/ The final converted 3D mesh was converted to glTF and then converted to 3D Tiles using a homegrown 3D Tiles creation tool described in [Annex A.2].

A.5.2. D104 — WiTech

WiTech's 3D visualization client (https://project-udtip-2024.vercel.app/) effectively leverages the Cesium JS framework to integrate urban noise data from OGC API services, particularly derived from Deliverable D103. This client seamlessly combines various geospatial data formats, enhancing its capability to provide detailed visualizations.

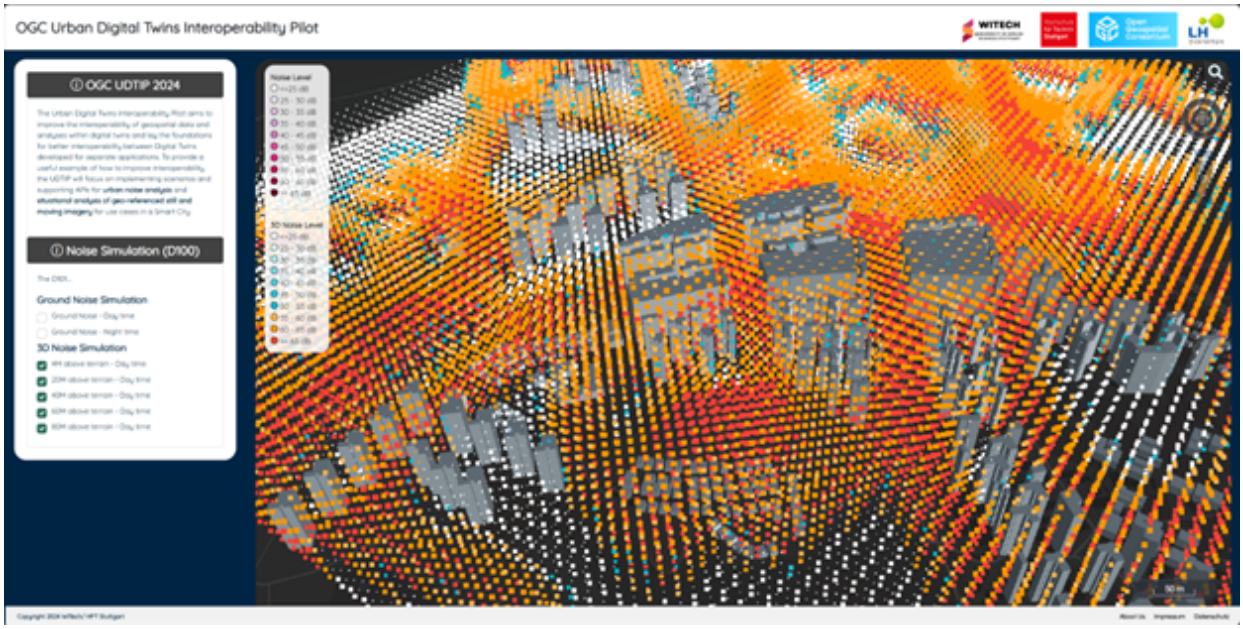

Figure A.61 — WiTech UDTIP Client Visualizing 3D City Model with 3D Multi-Level Noise Data

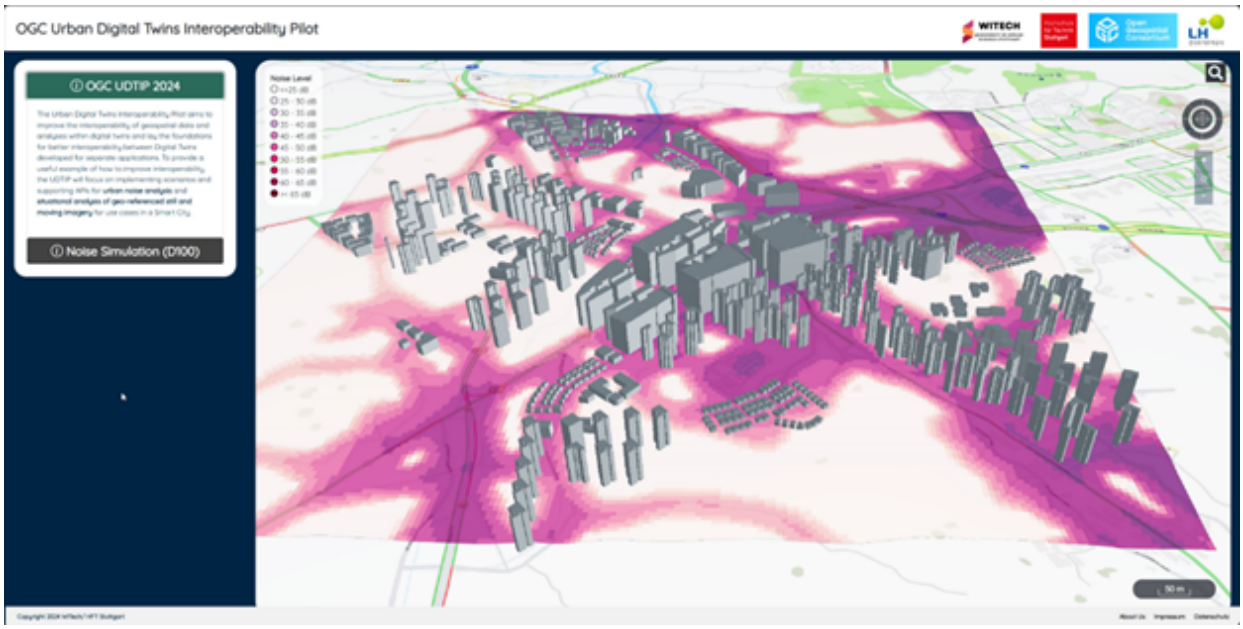

Figure A.62 — WiTech UDTIP Client Visualizing 3D City Model with Ground-Level Noise Data

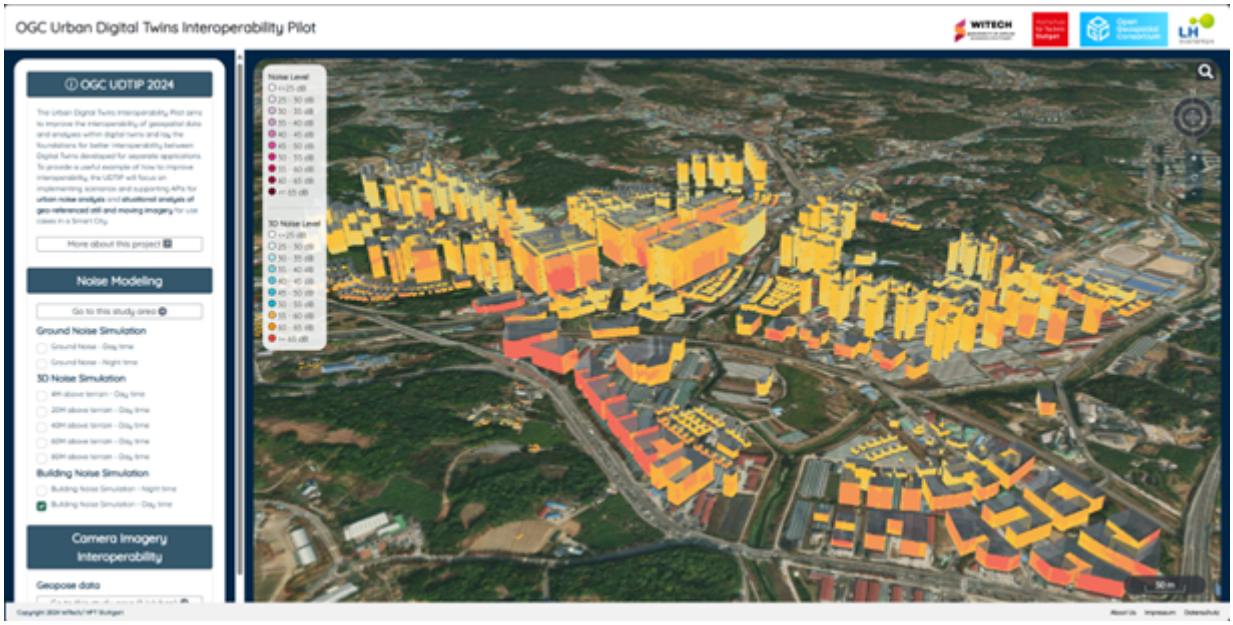

Figure A.63 — WiTech UDTIP Client Visualizaing 3D City Model with Noise Simulated on 3D Model

B

ANNEX B (NORMATIVE) HEALTH SERVICE RESEARCH

ANNEX B (NORMATIVE) HEALTH SERVICE RESEARCH

B.1. Health Use Cases for Urban Digital Twins

B.1.1. Introduction

Digital twins represent a groundbreaking innovation in the realm of technology, enabling the creation of precise, dynamic virtual models that mirror physical entities. From individual components of a building to entire cities, digital twins integrate real-time data, advanced analytics, and visualization tools to simulate and optimize the performance of physical systems. Originally rooted in manufacturing and engineering, the application of digital twins has expanded into various sectors, including urban planning, healthcare, and environmental management. As cities become increasingly complex and interconnected, and as cities house more and more of the global population, the role of digital twins in enhancing operational efficiency, decision-making, and predictive analysis is becoming indispensable.

In the rapidly evolving landscape of urban development, digital twins have emerged as a revolutionary tool to model, simulate, and optimize the intricate systems that constitute modern cities.

This pilot, funded by the Land and Housing Corporation (LH) in Korea and the United Nations (UN) and managed by the Open Geospatial Consortium (OGC), explores both multiple use cases for digital twins within urban environments, as well as the integration between use cases. Also explored is the application of digital twins to healthcare, and the extraction from non-health use cases of data supporting healthcare decision makers in improving public health and healthcare delivery.

B.1.2. Defining a Digital Twin

A digital twin consists of several key components that work together to create a virtual representation of a physical object, system, or process flow (called here 'entity' for simplicity) that can be small, such as a single floor of a building or component of an HVAC system, and as large and complex as an entire city.

The components of a Digital Twin include:

1. Data Collection: Sensors and Internet-of-Things (IoT) devices to collect real-time data from the entity, including operational metrics, environmental conditions, and other relevant information.

2. Data Integration: The process of aggregating, processing, and integrating data from various sources (e.g., multiple sensors and devices). This may involve data cleaning, transformation, and storage, and must ensure the data is consistent and usable.

3. Communication Interface: The connection between the entity and its digital twin. This interface ensures continuous data flow between the two, allowing the digital twin to stay updated with real-time information.

4. ML/AI: Advanced machine learning and artificial intelligence algorithms, analytics, simulation tools, and models used to analyze the collected data, predict outcomes, and optimize performance. These tools help in understanding the behavior of the entity under different conditions.

5. Visualization: Tools and platforms that provide a visual representation of the digital twin. This may include dashboards, 3D models, and augmented or virtual reality interfaces to help users "see" and interact with the digital twin.

6. User Interface: The actual mechanism by which users interact with the digital twin. This can be integrated with the systems providing visualization of the digital twin.

7. Feedback Loop: The mechanism that allows the digital twin to influence the entity. Based on the analysis and insights generated, actions can be taken to adjust or optimize the real-world entity. While not necessary, this component helps ensure any changes are reflected in the digital twin and should be considered for later phases or generations of the digital twin.

8. Security and Privacy: Measures to ensure that data and control/commands exchanged between the entity and its digital twin is secure and that privacy is maintained. This includes encryption, authentication, and access control mechanisms.

These components collectively enable the digital twin to provide representation, monitoring, operational analysis, scenario or what-if analysis of outputs based on different inputs into the entity, and the optimization of the underlying real-world entity in real-time.

B.1.3. Digital Twin Use Cases for Cities

A city-wide Digital Twin can serve as an integrated platform for up-to-date location-aware data, process-based modeling, and visualization with detailed and interconnected representations of every aspect of city life – including its transportation systems, lighting, waste management, public services and resources, healthcare infrastructure and facilities, and more. Specific areas where digital twins can provide value to urban environments include and are not limited to:

- Urban Mobility: including use cases of Traffic Management, Public Transit, Ride Share, Micro-Mobility, as well as Drones and Autonomous Vehicles (land, air, and water).
- Urban Climate: including use cases of Air Quality/Pollution monitoring, Urban Heat, Tree Canopy, Environmental Factors, Sustainability, Energy Efficiency, and supporting and tracking Decarbonization efforts.
- Urban Infrastructure: including use cases related to the management and operations of Building, Transportation Systems, Utilities, Energy, Energy Grid Resilience, Telecommunications, Data Networks, 5G, Sensors, IoT, IoMT, and Underground Networks.
- Urban Operations: including use cases related to Urban Planning, Waste Management, Water Management, Lighting Systems, Noise, Public Works, Public Safety, Disaster Management and Emergency Response.
- Governance: including use cases related to Budgeting and Funding, Commerce and Economic Development, Land Use, Real Estate Property Management, Construction Approval, Heritage Preservation Public Policy, and Citizen Engagement.
- Digital: including use cases related to the management and operations of Data Networks, Big Data, Cloud Infrastructure, Edge Infrastructure, AI/ML, 5G, Sensors, IoT, IoMT, and Cybersecurity.

As can be seen, these categories are not mutually exclusive and do overlap. As an example, an Urban Mobility-focused use case described below provides value to many areas:

B.1.3.1. Urban Mobility

Urban digital twin (UDT) technology visually simplifies complex city processes and ensure the systemic impact of planning decisions can be predicted and therefore enables fine-tuning of those decisions before implementation. Further, the sensor infrastructure of UDTs fuel real-time intelligence gathering and support decision making, while Artificial Intelligence (AI), with its ability to reason over vast quantities of Big Data, can uncover deep insights and augment human expertise helping inform better policy and the improved delivery of key public services.

Universal Design and Accessibility

Leveraging universal design and accessibility for workflow optimization is an example. UDTs leverage AI to enhance workflows for surveying, documenting, and modeling public infrastructure in a manner that identifies and addresses accessibility challenges.

- Geospatial and Reality Capture: Technologies like aerial photogrammetry and ground photography provide detailed, accurate representations of public spaces. These tools are instrumental in assessing and improving accessibility features such as ramps, crosswalks, and parking spaces.
- Accessibility Planning: Detailed digital models of pedestrian access routes and public spaces help in designing and positioning features like curb ramps, crosswalks, and transit stops. For example:

- Sidewalks and Shared-Use Paths: Digital twins aid in planning and optimizing these pathways to ensure they meet accessibility standards and are navigable for all users, including those with mobility impairments.
- Crosswalks: Implementation of detectable warning surfaces on curb ramps enhances safety for visually impaired pedestrians and improves overall crosswalk design.
- Transit Stops: Digital models help ensure that transit stops are designed with accessibility in mind, including proper sizing and positioning of both sidewalk-level and elevated boarding platforms.
- Parking Areas: Virtual measurements of parking spaces ensure that they are appropriately sized and strategically located to facilitate easy access from vehicles to sidewalks.

Integrating these technologies meets the needs of a broad coalition of users within a city ecosystem including City Planners, Engineers, and Architects, Public Service Departments, as well as Policymakers. And further, serve city residents, labor force, and tourists by creating more inclusive and functional urban environments, by ensuring that infrastructure is accessible to everyone and that planning decisions are data-driven and well-informed. There is also research suggesting that well-planned communities are both safer and lead to higher home values [10, 11].

B.1.4. Digital Twins in Healthcare

A digital twin in healthcare is a dynamic, real-time digital representation of a patient, medical device, medical facility (e.g., an individual ward or entire hospital), or healthcare process that mirrors its physical counterpart. By integrating real-time data from various sources potentially including electronic health records (EHRs), wearable devices, medical imaging, building information management (BIM) systems, and IoT and IoMT sensors, the digital twin provides a comprehensive view of the entity allowing for continuous monitoring, predictive analytics, resource allocation, and personalized healthcare with the goal of improving patient outcomes, operational efficiency, cost containment, and decision-making.

Key characteristics of a healthcare digital twin are listed below, and it should be noted that the use case plays a role in what characteristics must be included. For instance, a patient care use case involves additional characteristics.

1. Data Integration: To be most effective, the digital twin must present decision makers with the most current data available.
2. Predictive Analytics: Advanced ML/AI algorithms can help predict disease progression, potential complications, the impact of interventions, outcomes, as well as enable what-if analysis.
3. Simulation and Modeling: Allows healthcare professionals to simulate treatment plans or administrative decisions, optimizing care plans, resource allocation, and operational efficiency.

4. Operational Efficiency: Enhances the management of healthcare facilities by monitoring the utilization of resources such as hospital beds, medical equipment, and staff, improving workflow and reducing inefficiencies.

5. Decision Support: Provides healthcare providers with actionable insights and recommendations, supporting clinical decision-making across a targeted or comprehensive decision space.

B.1.4.1. Patient Care

Patient care use cases involve additional characteristics.

1. Patient-Centric: To support decisions involving patient care, the digital twin must provide a holistic view of an individual patient's health status, including medical history, current conditions, treatments, and responses to interventions.

2. Remote Monitoring: Particularly essential for those with chronic conditions or post-operative needs, remote monitoring supports continuous monitoring of patients medication adherence, vital signs, and other health metrics by ingesting input from medical devices and wearables.

3. Personalized Medicine: Tailor treatments and interventions to the unique characteristics and needs of each patient, improving care efficacy.

With these characteristics defined, the full range of health use cases can be supported either as a stand-along digital twin or as components of a large urban digital twin.

B.1.4.2. Health Use Case for an Urban Digital Twin

As recognized in the Call for Proposals to the Urban Digital Twin Infrastructure Pilot 2024, "Innovations in the planning, design and management of urban environments are essential to enable their residents to live healthier, safer, happier lives."

A Digital Twin in the healthcare space can provide shared visibility into the status, location, and utilization of healthcare resources including but not limited to:

- Hospital beds by bed type (Emergency Room, Observation, Admission, Critical Care, etc.)
- Medical facility (hospital, clinic, surgical center, operating rooms, etc.) and occupancy
- Medical devices (e.g., MRIs, ventilators)
- Medical supplies (e.g., PPE, intravenous saline)
- Staffing (across all patient-facing clinical positions)
- Ambulances

- Climate and environmental factors known to impact patient health such as air pollution and heat

A UDT with health use cases can also support the remote monitoring of a wide variety of patients who may no longer require hospitalization but will still benefit from a heightened level of care within their homes.

The data from IoT and IoMT sensors that underlie the UDT can be leveraged to build health applications allowing for faster recognition of and response to emerging health risks, whether social, clinical, climate-induced, or environmental, at both the population level and at the individual patient level.

These applications can both improve health outcomes and reduce overall healthcare costs. For example:

- **Air Quality Impacts on At-risk Patients:** Patients with asthma, COPD, or other respiratory conditions can be advised on specific and granular areas to avoid during days and times of poor air quality. Additionally, patient health education materials and efforts can be directed towards at-risk populations using the same data.
- **Operational Efficiency:** By tracking bed occupancy, or more general medical facility utilization, patient transfer through the healthcare system can be streamlined. Visibility into hospital/health facility status will also aid disaster response scenarios. Perhaps more importantly, the sources of inefficiency in the healthcare system that hampers patient flow can be identified — enabling solutions to improve the pathways of care and patient throughput.
- **Medical Inventory and Supply Chain Management:** Leveraging IoMT sensors to track medical supply levels, current staffing, staff training, skill set, and experience, as well as current medical device utilization/availability – ambulances can be routed toward the facilities where patients will experience the shortest wait times and receive the best care.
- **Remote Patient Monitoring:** Monitoring of patients discharged from hospital or medical facility care on a continuous or periodic basis.
- **Chronic Disease Management:** Continuous monitoring and management of chronic conditions such as diabetes, hypertension, and heart disease, along with potential environmental stressors.
- **Surgical Planning:** Preoperative simulation and planning to optimize surgical process and outcomes.
- **Patient Monitoring:** Remote monitoring of patients outside of the hospital (e.g., in their homes or workplace), reducing hospital admissions and utilization and freeing beds for those who need acute care.
- **Facility Management:** Optimization of hospital operations, process flow, resource utilization, and emergency response.

The most exciting aspect of a health digital twin is its ability to enable What-If analysis on clinical and social mitigation strategies to assess in advance their effect on health outcomes.

Anticipating the degree of success (or failure) of mitigation strategies and tweaking them through a digital twin helps both ensure the efficiency and effectiveness of such efforts in improving patient outcomes and overall population health, as well as reduce overall healthcare spending.

This is important as between 1970 and 2022, health expenditure increased an astronomical 208,900% [1]. Healthcare represents the highest proportion of Korea's general government expenditure (GGE) at 13.6% in 2022 [2].

B.1.4.3. Remote Patient Monitoring

A digital twin can serve as a powerful tool in healthcare to integrate broad sets of data in real-time or near real-time, perform advanced analytics, enable planning and what-if analysis with the goal of enhancing patient care, support personalized medicine, and streamlining operations ultimately leading to increased quality and better health outcomes while lowering costs.

An essential use case is remote patient monitoring – as in-patient care at a hospital facility is expensive and often provides a higher level of service than is required. In other words, as patients recover from illness, trauma, or surgery, they may no longer require hospitalization but would benefit from a lesser level of medical attention. In addition, seniors, the chronically ill, at-risk individuals, and those who require sub-acute care may well need medical attention, but not hospitalization. Remote patient monitoring involves a medical team tracking any or all personal electronic devices, wearables, at-home environmental & building sensors, camera feeds, as well as community and environmental sensors, to provide remote care. Such care can include but is not limited to:

- Turning on/off lights, water, gas
- Raising/lowering room temperature
- Tracking medication compliance
- Updating medication dosages
- Dispatching emergency response as needed

This provides those individuals with the needed care without separation from their families and confinement to medical homes or assisted living facilities. It also frees up hospital beds for more critical patients.

B.1.4.3.1. Technical and Operational Requirements

The technical and operational requirements for remote patient monitoring include [19, 20, 21, 22, 23]:

1. Devices and Connectivity: RPM involves various medical devices like blood pressure monitors, pulse oximeters, as well as consumer electronic devices such as weight scales, step monitors, and smart watch – all transmitting data to healthcare providers. Devices should be able to support real-time or near-real-

time data transmission, often via wireless or Bluetooth technology. While not all metrics are required in real-time, but the capacity to know the current patient status is important. It's essential that these devices are easy for patients to use and have reliable connectivity to function effectively.

2. Data Security and Privacy: RPM systems must comply with HIPAA and other data protection regulations. This includes encrypted data transmission, secure cloud storage, and stringent access controls to protect patient information.

3. Integration with Health Systems: The data collected from RPM devices needs to integrate smoothly with existing Electronic Health Records (EHR) and other clinical data systems. This ensures that healthcare providers can easily access, monitor, and analyze patient data, enabling timely intervention.

4. Training and Support: Providers, patients, as well as their caregivers will need training on how to use RPM devices and interpret the data. A remote patient monitoring scenario should be selected only once all parties are confident in their ability to correctly operate the monitoring equipment at all times, and that technical support is available if needed.

5. Billing and Reimbursement: Understanding billing codes (e.g., CPT 99453-99458) is critical to ensure the proper documentation and compliance with regulations to receive reimbursement and maintain the financial viability of RPM programs.

B.1.4.3.2. Workflow

A possible workflow for a remote patient monitoring use case can be the following:

1. Patient Enrollment: Patients are identified and enrolled based on eligibility, and appropriate monitoring devices are selected.

2. Device Setup: Devices are set up, and providers, patients, and caregivers are trained to use them, ensuring proper data transmission.

3. Data Collection and Transmission: Patients regularly use the devices, and health data is automatically sent to healthcare providers.

4. Data Review and Alerts: Data is analyzed; alerts are generated for abnormal readings, and providers review trends.

5. Patient Engagement and Follow-Up: Providers communicate with patients to discuss data and adjust care plans as needed.

6. Billing and Reimbursement: Healthcare providers document data reviews for billing purposes.

7. Ongoing Monitoring: Continuous data collection and monitoring, with adjustments made as needed.

While a remote patient monitoring system can be independent of a UDT, integration would provide access to a broader set of devices and sensors making the program more effective overall, as well as simplifying the inclusion and support for a larger number of patients.

B.1.4.3.3. Similarities with Traditional UDT Use Cases

There are, for example, opportunities to leverage the camera systems and sensors used to enhance mobility within an urban environment to also inform remotely monitor patient solutions. Cameras as well as traffic light and other roadway sensors can be used to guide emergency medical technicians (EMTs) to reach and transport at-risk patients requiring medical intervention.

Sensors in and around patient habitats used to monitor for potential exposure to harmful air quality and excessing heat can also inform broader city-wide efforts keep abreast of these considerations.

B.1.4.4. Existing Health Digital Twins

There are several implementations and ongoing projects of health digital twins. These initiatives highlight the potential of digital twins to transform healthcare by improving patient outcomes, enhancing operational efficiency, and supporting personalized medicine. Some notable examples include:

1. **Philips' Digital Twin for Heart Disease [3]:**
    - **Implementation:** Philips has developed a digital twin model for the heart, which simulates cardiac function and predicts the impact of various treatments on heart disease patients.
    - **Purpose:** To help cardiologists tailor treatments and interventions to individual patients, improving the management of heart conditions and optimizing treatment plans.
2. **Siemens Healthineers' Digital Twin Technology [4]:**
    - **Implementation:** Siemens Healthineers is developing digital twin technology to model facilities and processes to pilot solutions in a virtual environment before applying the real world.
    - **Purpose:** To enhance and improve hospital and medical facility operational practices.
3. **Dassault Systèmes' Living Heart Project [5]:**
    - **Implementation:** Dassault Systèmes has created a highly detailed digital twin of the human heart through its Living Heart Project.

- **Purpose:** To advance cardiovascular science, improve the design of medical devices, and support the development of personalized treatments for heart disease.

4. **Boston Children's Hospital's Digital Twin for Pediatric Care [12]:**

   - **Implementation:** Boston Children's Hospital is developing digital twins to model and simulate the health conditions of pediatric patients.
   - **Purpose:** To optimize treatment strategies, improve surgical planning, and enhance patient monitoring, particularly for children with complex medical conditions.

5. **GE Healthcare's Digital Twin for Hospital Operations [13]:**

   - **Implementation:** GE Healthcare has implemented digital twin technology to model hospital operations and workflows.
   - **Purpose:** To optimize resource utilization, streamline patient flow, and improve overall hospital efficiency, particularly in managing patient admissions, discharges, and bed occupancy.

6. **Mayo Clinic's Digital Twin for Predictive Health [14, 15]:**

   - **Implementation:** Mayo Clinic is leveraging digital twins to predict patient outcomes and personalize treatment plans.
   - **Purpose:** To improve the management of chronic diseases, enhance surgical outcomes, and support the development of new therapies through advanced simulation and modeling.

7. **Microsoft's Project Premonition [16]:**

   - **Implementation:** Microsoft's Project Premonition uses digital twin technology to model and predict the spread of infectious diseases.
   - **Purpose:** To improve disease surveillance, enhance public health response, and support the development of strategies to mitigate the impact of outbreaks.

These implementations demonstrate the diverse applications of digital twin technology in healthcare ranging from diagnosis and personalized patient care planning to operational efficiency and disease prevention. As the technology continues to evolve, more healthcare organizations are expected to adopt digital twins in one or more of these areas to enhance their capabilities and improve outcomes.

B.1.4.5. Defining a Health Digital Twin

For a health digital twin to be effective, it must provide a comprehensive understanding of the physical entity being represented, the data sources, the intended applications, and the technological infrastructure. The key details needed to define a health digital twin include:

1. **Objective and Scope:**
    - **Objective:** Clearly define the purpose of the digital twin, such as improving patient outcomes, enhancing diagnostic accuracy, or optimizing hospital operations.
    - **Scope:** Determine the specific aspects of healthcare to be modeled (e.g., patient flow within a medical facility or across the care continuum, medical device utilization, healthcare facility operation).
2. **Physical Entity Description:**
    - **Entity:** Specify what the digital twin will represent (e.g., a patient, a medical device, an operating room, an entire hospital) along with its attributes, characteristics and, behaviors (e.g., patient demographics, device specifications, facility layout).
3. **Data Sources:**
    - **Real-Time Data:** Identify sensors, IoT/IoMT devices, and other data sources (e.g., wearable health monitors, medical imaging devices).
    - **Historical Data:** Include electronic health records (EHRs), medical histories, and previous treatment data.
    - **Environmental Data:** Incorporate data on environmental factors impacting health (e.g., air quality, weather conditions).
4. **Data Integration and Management:**
    - **Data Aggregation:** Plan for collecting and aggregating data from multiple sources.
    - **Data Storage:** Determine how data will be stored and managed (e.g., cloud storage, databases).
    - **Data Quality:** Ensure data accuracy, completeness, and timeliness.
5. Modeling and Simulation:
    - **Digital Model:** Develop a virtual model that accurately represents the physical entity.

- **Simulation:** Use advanced algorithms and machine learning to simulate behaviors and predict outcomes.

6. **Analytics and Insights:**

   - **Analytics Tools:** Identify tools and platforms for analyzing data (e.g., machine learning algorithms, predictive analytics).
   - Insights: Determine the types of insights needed (e.g., predictive health outcomes, operational efficiencies, what-if analysis).

7. **Visualization and Interaction:**

   - **Visualization Tools:** Choose tools for visualizing the digital twin (e.g., 3D models, dashboards).
   - **User Interface:** Design an interface for users, including healthcare professionals and patients, to interact with the digital twin.

8. **Communication Interface:**

   - **Connectivity:** Ensure reliable data communication between the physical entity and the digital twin.
   - **Protocols:** Define communication protocols and standards (e.g., HL7, FHIR).

9. **Feedback Loop:**

   - **Actionable Insights:** Develop mechanisms for the digital twin to provide actionable insights and recommendations.
   - **Implementation:** Plan for implementing strategic or operational tactics and practices developed in the digital twin into the physical entity (e.g., adjusting treatment plans, optimizing resource allocation).

10. Security and Privacy:

    - **Data Security:** Implement measures to protect data integrity and confidentiality (e.g., encryption, access controls).
    - **Compliance:** Ensure compliance with healthcare regulations and standards (e.g., HIPAA, GDPR).

11. **Stakeholder Involvement:**

    - **Stakeholders:** Identify key stakeholders (e.g., healthcare providers, patients, administrators).

- **Collaboration:** Plan for stakeholder collaboration and input throughout the development process.

12. **Evaluation and Refinement:**

    - **Performance Metrics:** Define metrics to evaluate the digital twin's performance and effectiveness.
    - **Continuous Improvement:** Develop a plan for continuous monitoring, evaluation, and refinement of the digital twin.

The above details provide for a well-defined health digital twin that accurately represents the physical entity and provides valuable insights to improve healthcare outcomes and quality as well as lower costs.

Extracting data relevant to health applications expands the value of an UDT use case – providing further justification for the resource (e.g., funding, infrastructure, etc.) requirements for designing and deploying the use case. In addition, it expands the user community for the UDT overall.

B.1.5. Integration with Urban Digital Twin Use Cases

Health use cases also integrate with traditional UDT use cases – reflecting the overlap between health considerations and most other issues and challenges impacting human society. For instance, there is overlap between health concerns and both the urban noise analysis and situational analysis of geo-referenced still and moving imagery use cases included in the current UDTIP pilot. Incorporating health needs expands the applicability of traditional use cases to the community and to daily life.

While the domains are different, there are similarities in the process and information required to build a health-focused and traditional use case.

B.1.5.1. Urban Noise Analysis

This use case aims to support urban planning and management aimed at mitigating traffic noise. One of the reasons for mitigating traffic noise is the impact excess or constant noise can have on quality of life and residential property values. Noise can also have deleterious effects on health outcomes.

B.1.5.2. Specific Health Applications

Exposure to loud and prolonged noises can have health implications. Data on noise levels within a city and in and around residential areas can help identify correlations with health issues such as sleep disturbances, cardiovascular problems, stress levels, mental health, and post-trauma or post-operative recovery, informing public health interventions, treatment plans, and patient

care. This can lead to the creation of quiet zones or noise-reduction programs near and around residential areas, schools, and medical facilities promoting better health and well-being. In addition, hospitals, rehabilitation care facilities, and post-operative recovery centers can be located within quiet zones so healthcare outcomes are not impacted by noise levels.

As such, an UDT can use noise analysis from across the city to both study the impact of noise pollution on public health, as well as guide urban planning and policy decisions to mitigate its adverse effects. For example, identifying areas with high noise pollution and poor health outcomes can guide efforts to either remove or mitigate the noise (e.g., through the creation of sound barriers), or by moving the adversely impacted facilities.

Such solutions may help lower nationwide healthcare spending. In 2022, health spending in South Korea amounted to approximately 209 Trillion KRW, representing an increase from the previous year and continuing a trend of annual growth [3].

The Urban Noise analysis is expected to result in a 3D heat map for a city — with more noisy regions of the city appearing darker and taller, while less noisy/more quiet places resolving as lighter and shorter or flatter. This data can be resolved to 2D input that can be ingested into our health risk indices. Specifically, data on the average and maximum noise level by region of the city is of most interest.

B.1.5.3. Situational Analysis of Geo-Referenced Still and Moving Imagery

This use case involves the detection and identification of unwanted objects throughout a city environment. There are numerous potential health applications of such information.

B.1.5.3.1. Public Health Management:

- Disease Spread Tracking: Geo-referenced imagery can track population movement and density, helping to model and predict the spread of infectious diseases in urban areas.
- Health Infrastructure Utilization: Imagery analysis along with geolocation analysis can monitor the usage patterns of healthcare facilities, helping to optimize resource allocation, placement, as well as identify areas needing additional support.

B.1.5.3.2. Environmental Health:

- Monitoring Environmental Changes: Leveraging imagery, digital twins can allow for what-if analysis on changes in the urban environment, such as deforestation, water levels, and air quality, to assess and inform policies to mitigate any potential negative impacts on health outcomes.For instance, a digital twin can allow for designing city landscapes where trees can be used to create shaded areas shielding residents from UV radiation and consequent health impacts, as has been modeled in Seoul, Korea [8].

B.1.5.3.3. Emergency Response and Disaster Management:

- Real-Time Situational Awareness: Digital twins can enable integration of real-time data for situational analysis during emergencies, helping responders to assess damage, identify affected areas, and allocate resources more effectively.
- Evacuation and Rescue Operations: Digital Twins can aid planning for recovery operations as well help build evacuation plans and guide rescue operations by providing up-to-date information on road and infrastructure conditions, crowd locations, and safe routes.

B.1.5.4. Specific Health Applications

In disaster response scenarios, real-time information on accessibility of roadways and transportation networks is critical for evacuation efforts and medical supply routing. The analysis of still and moving imagery performed in this pilot effort along with maps of existing roadways and transportation networks can be leveraged to supply this information.

Data on transportation networks can be sourced from public sources such as departments of transportation as well as international organizations such as Open Street Maps. Citizen science approaches to real-time, on-the-ground, data collection through mobile phones can also be effective.

The images in such scenarios can be located through GeoPose and potentially through embeddings in images. For instance, imagery of fallen trees or other debris blocking a roadway can be located on a map so first responders and evacuating residents avoid the impacted route until road crews are able to clear the obstruction.

When fed into a routing engine, the updating of evacuation routes can be updated in real-time. This can speed and streamline the overall evacuation process as it is known that damage to transportation networks affects accessibility and travel time during the emergency response [9].

B.1.5.5. OGC Standards Supporting Interoperability

Interoperability refers to an Urban Digital Twin's ability to seamlessly integrate and communicate with other systems, software, devices, and data sources. It involves establishing standardized protocols, interfaces, and data formats to enable the exchange of information and actions between different components of the overall UDT ecosystem. Interoperability is crucial because it allows for the integration of diverse data sources, sensors, IoT and IoMT devices, and software applications that contribute to the UDT's functionality. Further, it enables the Digital Twin to receive data inputs from various sources, such as building automation systems, sensors, weather stations, or maintenance databases, and to provide data outputs to other systems for further analysis or control.

The following standards play a role in enabling interoperability among UDTs across use cases and the systems they support.

- CityGML — City Geography Markup Language
- CityJSON — City Java Script Object Notation
- GeoPose
- SensorThings API
- MovingFeatures
- MUDDI — Model for Underground Data Definition and Integration / May not be applicable.
- BIM/IFC — Building Information Modeling/Industry Foundation Classes. While not an OGC standard, this enables the integration and conversion of building data to GIS data.
- CIM — City Information Modeling

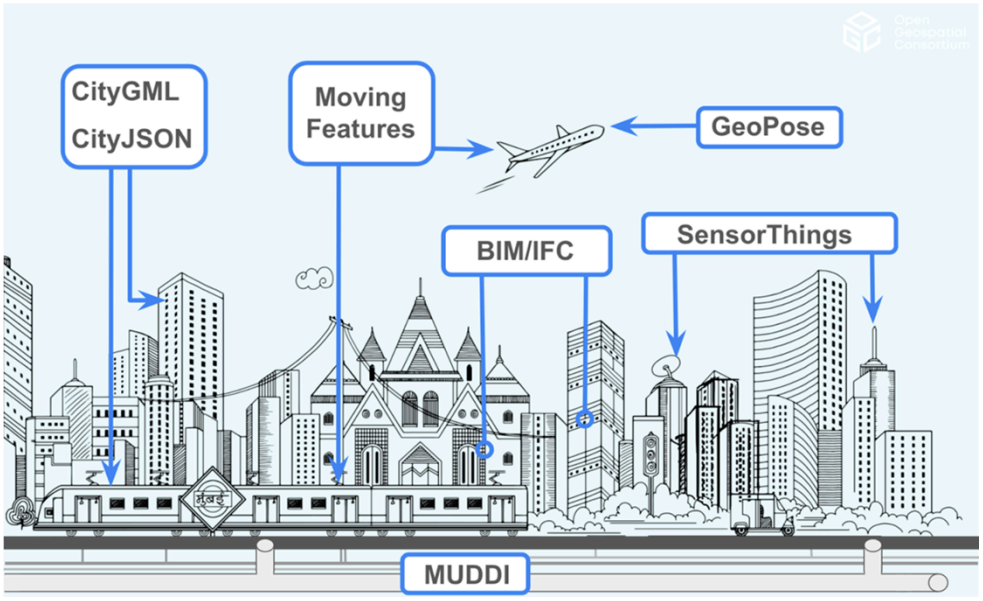

Figure B.1 — Standards playing a role in UDT Interoperability.

B.1.5.6. Challenges with Interoperability

A specific barrier to interoperability is data conversion. Integrating BIM and GIS data has the potential to combine a large-scale and a micro-scale built environment [24]. However, successful integration requires two common conversion paths, IFC-to-CityGML and IFC-to-shapefile, or to other geo-based formats. The conversion and integration of data is complicated by the vast differences between these data models across five components: encoding, semantic coherence, geometry validation, coordinate reference system, and topological accuracy.

The second issue is software compatibility. Integrating geospatial and building data information, such as importing 3D GIS data into BIM models, is a potential but challenging and continuous application. BIM software does not always provide support for geospatial data compatibility. Although some commercial software has been developed to support the integration and export

of models, all functionality is not realized. For example, Esri's CityEngine can export 3D city models to various data formats, but it does not directly support semantic compatibility with the CityGML standard [25, 28].

B.1.6. Benefits to Health Digital Twins

The benefits of a Health Digital Twin are far reaching and accrue to many parties, only partially including:

- **Patients** — Through the synthesis of distributed sensor data that can be a part of a Digital Twin and health data, patients benefit from a more insightful, holistic approach to health care delivery leading to improved health outcomes.
- **Caregivers** — These solutions enable friends, family, and neighbors to provide care to their loved ones, even if remote, partially reducing the caregiving burden often known to lead to burnout and to cost caregivers their own health.
- **Hospitals/health systems** — Will be able to use digital twin models to track procedural or utilization tendencies and to better understand patient needs and reduce costs.
- **Insurers** — Both public and private payers can utilize digital twins to better understand their populations and develop approaches to help individuals make healthy choices.
- **Governments** — Will be well positioned to understand the sources of costs and inefficiencies in the medical infrastructure overall to develop approaches and policies to reduce costs long-term.

Ultimately, a Digital Twin focused on healthcare needs will help communities and individuals lead their healthiest lives. And the return on investment on healthier, happier communities will reap rewards for generations.

B.1.7. Conclusion

Digital twins can have a transformative impact on health outcomes, optimize resource allocation, and reduce costs in urban environments. By leveraging real-time data and advanced analytics, health digital twins can provide actionable insights into patient care, facility management, and public health interventions. As urban environments become more complex, the ability to model and simulate health scenarios within a digital twin framework will be essential for anticipating and addressing the challenges of modern healthcare.

As the technology continues to evolve, its adoption in the healthcare sector will likely expand, driving innovation and improving the quality of life for urban populations.

Key use cases and benefits of health digital twins have been presented along with how non-health use cases can support decision making within healthcare – expanding the value of a city-wide digital twin infrastructure.

The challenges of interoperability notwithstanding, we strongly encourage the Land & Housing Corporation to consider the inclusion of health use cases in future pilots as well as continue

to explore extracting health insights from non-health use cases. By incorporating urban noise analysis and situational analysis of geo-referenced imagery into digital twins, the Land and Housing Corporation can enhance their ability to monitor, analyze, and respond to various challenges, ultimately improving public health, safety, and overall quality of life. As additional use cases are developed, they will also be able to provide input supporting health risk analytics.

Further, we recommend surveying the Korean healthcare sector to identify potential use cases of interest to the industry.

B.1.8. References

1. Macrotrends.net, https://www.macrotrends.net/global-metrics/countries/KOR/south-korea/healthcare-spending.
2. Healthsystemfacts.org, https://healthsystemsfacts.org/national-health-systems/national-health-insurance/south-korea/south-korea-health-system-financing-expenditures/
3. Van Houten, Henk, Chief Technology Officer (Ret.), Royal Philips, How a virtual heart could save your real one, November 12, 2018.
4. Siemens Healthineers, The value of digital twin technology.
5. Dassault Systemes.
6. Statista Research Department, Sept 27. 2023, https://www.statista.com/statistics/978137/south-korea-current-health-spending/
7. Sun, Tianze, et al, The Digital Twin in Medicine: A Key to the Future of Healthcare?, Frontiers in Medicine, 13, July 2022
8. Na, Hang Ryeol, et al, Modeling of urban trees' effects on reducing human exposure to UV radiation in Seoul, Korea, Urban Forestry & Urban Greening 13, 2014, 785-792.
9. Chen, S, and Wu, Y, Traffic Resilience Modeling and Planning of Emergency Medical Response, Mountain-Plains Consortium, U.S., Department of Transportation, June 2022.
10. Krieg, Dillan, et al, Why Master-Planned Communities Outperform: The Lifestyle Commands a Price Premium, John Burns Research & Consulting, January 27, 2023.
11. Hopkins, Erin A., The Impact of Community Associations on Residential Property Values: A Review of the Literature, November 2015, Virginia Tech.
12. Digital World, Using a heart's digital twin to reduce cardiac surgeries, February 13, 2024.
13. GE HealthCare Command Center, Digital Twins.

14. Halamka, John, MD, and Cerrato, Paul, Can Digital Twins Improve Patient Care?, Mayo Clinic Platform, July 12, 2022.

15. Sahilbhutada, Case Studies and Real-World Examples of Digital Twin Implementation, Medium, May 28, 2023.

16. Microsoft Premonition.

17. Biot, Claire, Digital twins to predict disease, Dassault Systems, November 10, 2022.

18. Loehr, Steven, Mixed-Use, Mixed Impact: Re-Examining the Relationship between Non-Residential Land Use & Residential Property Values, Columbia University, May 2013.

19. U.S. Department of Health and Human Services, Telehealth and Remote Patient Monitoring.

20. The American Medical Association, AMA Remote Patient Monitoring Playbook.

21. Great Plains Telehealth Resource & Assistance Center, Telehealth Resource Center RPM Toolkit.

22. California Telehealth Resource Center, Remote Patient Monitoring Toolkit.

23. Prevounce, RPM Billing Guide.

24. ScienceDirect, Building Information Modeling Technology.

25. Lei, Binyu, et al., Challenges of urban digital twins: A systematic review and a Delphi expert survey, Automation in Construction, Volume 147, March 2023.

26. Geospatial World, Unlocking the Potential of Digital Twins for the AEC Industry, 2023.

27. International Electrotechnical Commission, City information modelling and urban digital twins, December 2021.

28. Esri, ArcGIS CityEngine.

C

ANNEX C (NORMATIVE) ROAD CLASSIFICATION

ANNEX C (NORMATIVE) ROAD CLASSIFICATION

Suggested by Alessandro Palmas, consultant c/o United Nation Global Service Centre

Before instructing an AI/Deep Learning system, a classification method should be agreed.

As discussed before, without reinventing the wheel, we adopt some of the OpenStreetMap's Map Feature classification[2] to describe roads' conditions.

The two main classes used to classify road conditions are:

- surface: https://wiki.openstreetmap.org/wiki/Key:surface
- smoothness: https://wiki.openstreetmap.org/wiki/Key:smoothness

and, applied to a road, they can be used alone or togheter as they describe different features.

In practical terms:

Surface could be useful to understand how good/fast a vehicle could go on that road, but especially how its condition could change depending on the weather. An asphalted road in UN is considered an all weather road, regardless heavy rains fallen some days before, while an unpaved is considered a fair weather road, as it could be not passable after rain or snow, or even during the whole rain season in some Countries.

The most used — for our scope — properties for surface tag (class) are:

Table C.1 — Detail of Tag Descriptions

MORE GENERIC TAGS	DESCRIPTIVE TAGS	DESCRIPTION
asphalt		Short for asphalt concrete.
	paved	Other solid surfaces (cement, etc.)
unpaved		

[2] https://wiki.openstreetmap.org/wiki/Map_features

MORE GENERIC TAGS	DESCRIPTIVE TAGS	DESCRIPTION
	compacted	A mixture of larger (e.g., gravel) and smaller (e.g., sand) parts, compacted (e.g., with a roller), so the surface is more stable than loose gravel.
	gravel	This tag has very large meaning range. Used for cases ranging from huge gravel pieces like track ballast used as surface, through small pieces of gravel to compacted surface.
	rock	Big pieces of rock used to improve path quality or exposed bare rock.
	dirt	We can include grass Used for where surface is exposed soil, also commonly referred to as earth or dirt, but it is not sand, gravel, or rock.
	sand	Small to very small fractions (less than 2 mm) of rock.

Text extracted by the OpenStreetMap wiki page

Smoothness usually help to understand which kind of vehicle is able to travel on that road. This, for our scope is a bit much; we group the several classes by 3.

Table C.2 — Comparison of Classification Categories

OUR CLASSIFICATION	OSM TAG	KIND OF VEHICLE	DESCRIPTION
good	excellent	(thin_rollers)	As-new asphalt or concrete, smooth paving stones with seamless connections, etc.
	good	(thin_wheels)	Asphalt or concrete showing the first signs of wear, such as narrow (<1.5 cm) cracks.
intermediate	intermediate	(wheels) city biker, Scooter	Asphalt and equivalent that shows signs of maintenance such as patches of repaired pavement, wider cracks (>2 cm). The pavement may contain small potholes.
	bad	(robust_wheels) trekking bike, normal cars	Heavily damaged paved roads that badly need maintenance: many potholes, some of them quite deep. The average speed of cars is less than 50% of what it would be on a smooth road.
	very_bad	(high_clearance) Car with high clearance, light-duty off road vehicles	Unpaved roads with potholes and ruts, but still passable with an average SUV with a ground clearance of at least 18 cm.*
bad	horrible	(off_road_wheels) heavy-duty off road vehicles and all below	Unpaved tracks with ruts, rocks etc that need a ground clearance of at least 21 cm. Skid plate protection is advisable.

OUR CLASSIFICATION	OSM TAG	KIND OF VEHICLE	DESCRIPTION
	very_horrible	(specialized_off_road_wheels) tractor, tanks and all off-highway vehicles	Tracks with deep ruts and other obstacles that need a ground clearance of at least 24 cm.

Text extracted by the OpenStreetMap wiki page

C.1. Example Photos

Figure C.1 — asphalt; smoothness=good

Figure C.2 — asphalt; smoothness=excellent

Figure C.3 — asphalt; smoothness=good

Figure C.4 — surface=compacted; smoothness=good

Figure C.5 — surface=compacted; smoothness=good

Figure C.6 — surface=gravel; smoothness=good

Figure C.7 — surface=gravel; smoothness=good

Figure C.8 — surface=dirt; smoothness=fair

Figure C.9 — surface=dirt; smoothness=fair

Figure C.10 — surface=dirt; smoothness=fair

Figure C.11 — surface=dirt; smoothness=good

Figure C.12 — surface=dirt; smoothness=good

Figure C.13 — surface=rock; smoothness=fair

Figure C.14 — surface=dirt; smoothness=good

Figure C.15 — surface=rock; smoothness=bad

Figure C.16 — surface=dirt (or mud); smoothness=bad

Figure C.17 — surface=mud; smoothness=bad

D

ANNEX D (NORMATIVE) OPEN-SOURCE SOFTWARE USED OR DEVELOPED FOR UDTIP

ANNEX D (NORMATIVE) OPEN-SOURCE SOFTWARE USED OR DEVELOPED FOR UDTIP

UDTIP has been implemented based on open-source software and the outcomes are also available as open source. The list of open-source software used in the project or developed by the project is as the table below:

Table D.1 — Open-source Software Used or Developed for UDTIP

NAME	DESCRIPTION	MODULE USING THIS OPEN SOURCE	LICENSE	SITE URL
QGIS ope Noise	QGIS python plugin for noise level analysis	D100: Noise Analysis	GPL-3.0	https://github.com/Arpapiemonte/openoise-map
CVAT	Image labeling tool for training data	D101: Geo-AI	MIT	https://www.cvat.ai/
Label Studio	Image labeling tool for training data	D101: Geo-AI	Apache 2.0	https://labelstud.io/
Sensor Logger	cross-platform data logger that logs readings from common motion-related sensors on smartphones.	D101: Geo-AI	MIT	https://github.com/tszheichoi/awesome-sensor-logger
OpenCamera Sensors	Android application for synchronized recording of video and IMU data. It records sensor data (accelerometer, gyroscope, magnetometer) and video with frame timestamps synced to the same clock.	D101: Geo-AI	GPL 3.0	https://github.com/MobileRoboticsSkoltech/OpenCamera-Sensors
FFMPEG	FFmpeg is a collection of libraries and tools to process multimedia content such as audio, video, subtitles and related metadata.	D101: Geo-AI Mostly LGPL 2.1	Some modules are MIT, X11, BSD	https://github.com/FFmpeg/FFmpeg

NAME	DESCRIPTION	MODULE USING THIS OPEN SOURCE	LICENSE	SITE URL
GPS Logger	A GPS logger for Android mobile devices.	D101: Geo-AI	GPL-3.0	https://github.com/BasicAirData/GPSLogger
PyTorch	PyTorch is an open source deep learning framework built to be flexible and modular for research, with the stability and support needed for production deployment.	D102: Geo-AI	BSD-3	https://github.com/pytorch
Road Classification API	Geo-AI tool for classifying road surfaces developed by Helyx	D102: Geo-AI	Apache 2.0	https://github.com/Natte2110/road-classification-api-D102
CesiumJS	3D visualization engine	D104	Apache 2.0	https://cesium.com/platform/cesiumjs/
Mago3D	3D visualization engine developed by Gaia3D	D104	Apache 2.0	https://github.com/Gaia3D/mago3d-suite
pygeoapi	pygeoapi is a Python server implementation of the OGC API suite of standards	D103	MIT	https://pygeoapi.io/
OGC 3D GeoVolumes	Open-source Implementation of OGC API — 3D GeoVolumes / Installation by Node.js or Docker	D103	Apache 2.0	https://github.com/JoeThunyathep/OGC-API-3D-GeoVolumes
OGC API Features	A modern, RESTful standard for accessing and querying geospatial feature data.	D103	OGC Copyright License	link:++https://github.com/opengeospatial/ogcapi-features