electronics

MDPI

Article

Development of Mobile App to Enable Local Update on Mapping API: Construction Sites Monitoring through Digital Twin

Alfredo Valenzuela [1], Jongseong Brad Choi [1,2,*], Ricardo Ortiz [1], Byungkon Kang [3], Michael Kim [4] and Taewook Kang [5]

[1] Department of Mechanical Engineering, SUNY Korea, Incheon 21985, Republic of Korea; alfredojavier.valenzuelabustos@stonybrook.edu (A.V.); ricardo.ortiz@stonybrook.edu (R.O.)
[2] Department of Mechanical Engineering, Stony Brook University, Stony Brook, NY 11794, USA
[3] Department of Computer Science, SUNY Korea, Incheon 21985, Republic of Korea; byungkon.kang@sunykorea.ac.kr
[4] School of Architecture and Building Science, Chungang University, Gwangmyeong 14353, Republic of Korea
[5] Korea Institute of Civil Engineering and Building Technology, Goyang-si 10223, Republic of Korea
* Correspondence: jongseong.choi@sunykorea.ac.kr

Abstract: Unmanned ground vehicles (UGVs) have emerged as a promising solution for reconnaissance missions, overcoming labor cost, frequency, and subjectivity issues associated with manual procedures. However, for dynamic environments such as construction sites, the constantly changing conditions hinder a manager from planning the UGV's paths. For an autonomous monitoring mission, the path planning should be dealt with by having a map with the site's most recent scene. In this study, we develop a mobile app capable of local map updates by overlaying an image on a mapping API (e.g., Google Maps) hence working as a digital twin capable of creating a dynamic representation of the updated terrain over the mapping API. UGV operators can draw a path on such an updated construction scene using a tablet PC or smartphone. Discrete GPS information (e.g., latitudinal, and longitudinal) is executed for the UGV's controller. In the overlaying procedure, the homographic relation between the image and map is automatically computed and then projected. We successfully demonstrated the capabilities of the technique with two construction sites and a soccer field using images from the ground and satellite, respectively. The error generation is quantitatively measured and analyzed.

Keywords: reconnaissance mission; google maps API; autonomous driving; unmanned ground vehicles; GPS; digital twin

Citation: Valenzuela, A.; Choi, J.B.; Ortiz, R.; Kang, B.; Kim, M.; Kang, T. Development of Mobile App to Enable Local Update on Mapping API: Construction Sites Monitoring through Digital Twin. *Electronics* **2023**, *12*, 4738. https://doi.org/10.3390/electronics12234738

Academic Editor: Giambattista Gruosso

Received: 30 August 2023
Revised: 26 September 2023
Accepted: 16 October 2023
Published: 22 November 2023

1. Introduction

Onsite visual inspection in construction sites is one of the regular processes performed by human engineers to ensure the safety and efficiency of the operations. They are often seen as time-consuming and tedious jobs because of how physically demanding they are. Challenging conditions, such as extreme temperatures, confined spaces, elevated heights, cost, or risky terrains, are some of the reasons why inspections are done with such a low frequency and have a high percentage of injuries and fatalities despite the establishment and implementation of safety programs [1,2].

Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs), in conjunction with advanced sensors, have emerged as formidable contenders to revolutionize inspection methodologies. Even more so considering that the operation and maintenance of a building costs about 50–70% of the total life-cycle cost of a project [3]. These systems offer the potential to overcome the limitations of human-driven inspections, allowing for increased efficiency, accuracy, and coverage. However, modern UGVs are highly dependent

on multiple sensorial systems including LiDARs, RADARs, GPS, ultrasonic waves, and Vision Cameras [4]. These sensors have line-of-sight limitations and increase the cost of the robot. Not to mention that sometimes even UAVs are coupled to UGVs because of their spatial limitations [5,6]. Furthermore, the abundance of data collected requires the need of processing and filtering depending on the purpose intended for the data.

By limiting the number of sensors implemented with the UGV to only an RTK-GPS, Real-Time Kinematics GPS, a high-accuracy sensor with a tolerance of 2 cm, will not only suffice for proper navigation but also will reduce the complexity of integrating the whole system (control system, communication protocol, and navigation algorithms) and the amount of data collected won't require post-collected filtering or processing [7]. Enabling the possibility of high-precision navigation allows users such as site inspectors to operate this system with ease due to their knowledge and understanding of the inspection site.

The constant changes in construction sites as shown in Figure 1 [8] due to their fast-paced dynamic nature pose a persistent challenge. Because of this, the need for updated maps is essential for on-site decision-making. Therefore, to be able to operate autonomous inspections constantly would require a system capable of displaying the most updated scene on the route map APIs. As a result, UGVs operated based on a user-designated path drawn on the map would enable reinforcing continuous monitoring of local scenes in construction sites as a whole [9]. Moreover, the rise of mobile apps in the construction industry due to the capability of getting accurate information to make critical decisions [10], has enabled its development in many areas of the industry and much more to come [11,12]. In addition, the increase in building structure modeling research as well as sustainability reassures the need to improve the early stages prior to, or during the time of the construction [13–16]. Combining the capabilities of mobile apps with the benefits of robot-based inspections like objectivity, and improved repeatability could enhance autonomous inspections [17]. Being capable of updating the scene of a site on a mobile app would be a great opportunity for the user to take path planning-related decisions in real-time.

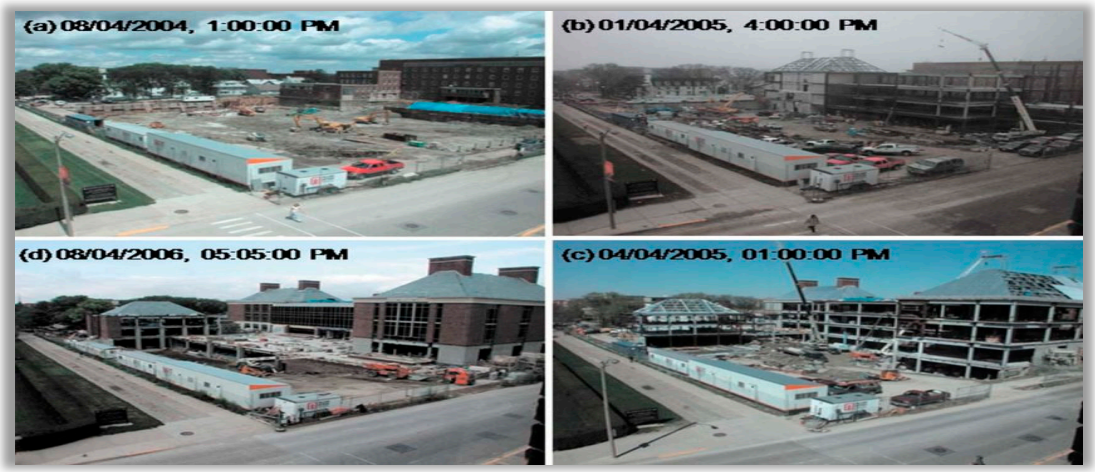


Figure 1. Constant changes in construction sites [18].

In this work, we develop a mobile app capable of local map updates by overlaying an image on a mapping API (e.g., Google Maps) as well as inputting the path plan on the map for UGVs' autonomous monitoring missions. UGV operators can draw a path on such an updated construction scene using a tablet, PC, or smartphone. Discrete GPS information (e.g., latitudinal, and longitudinal) is executed for the UGV's controller. In such an overlaying procedure, for the recent scene on a mapping API, the homographic relation between the image and map is computed and then projected. To do so, our system will be composed of a mobile device with the app, a broker, and the UGV as shown in Figure 2. Due to the conditions that we would be phasing out on a construction site,

a proper communication protocol such as MQTT would be needed that will ensure the sending of all the information needed to the UGV.

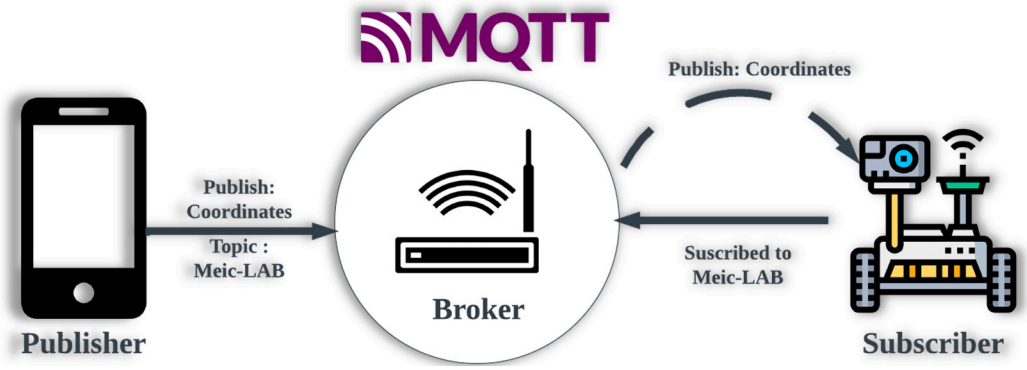


Figure 2. Path coordinates transform to UGV domain.

This development enhances decision-making with real-time visual data and seeks the coverage of specific areas of interest. Furthermore, the mobile app serves as a digital twin platform by allowing the operator to interactively update and visualize real-world terrain and subsequently create a dynamic representation of the terrain. While not a digital twin by definition, the app provides the operator to monitor changes in the site, becoming a tool of high reliability for the operation of autonomous driving in long-term monitoring or surveillance. In addition, it reduces the operational workload of inspections while facilitating the repeatability of the assessment. The dynamic aspect of the application in addition to all the benefits of utilizing a UGV on a construction site makes it a valuable tool to enhance onsite inspection methods.

The remainder of this paper is as follows: Section 2 shows the overview of the system and introduces all the different steps of the algorithm needed to transform the designated path to the UGV domain. Section 3 sets all the criteria considered to demonstrate that the application is properly functioning, and Section 4 reaffirms this by demonstrating other construction site cases. Finally, the remaining sections contain conclusions and future areas of development for this mobile application.

2. System Overview

An overview of the developed technique is presented in Figure 3. It shows the different procedures necessary to obtain the desired path for the autonomous inspection to be sent to the UGV's autonomous monitoring missions. Initially, the mobile app needs a graphic representation of a construction site; Query Scenes I and II are images of the most updated scene and a screenshot of the site taken from our mobile app, respectively. Then, a perspective correction is generated between Query Scene I and II which will be overlayed on mapping API to combine with geolocation information. With the overlayed scene, path planning can be simply created by drawing on a touch screen. Then, discrete GPS information (e.g., latitudinal, and longitudinal) is executed for UGV's controller. Each of these steps is explained in greater depth in the following subsections.

2.1. Query Scene

The objective of this step is to collect the data that we are going to use in the following steps. There are two images needed, Query Scene I and II. Query Scene I refers to the updated scene that the operator has taken to be used as the most recent scene and overlaying on the mapping API. It is important to note that this image could have any type of inclination. A picture from a nearby building or hill, aerial imagery from UAV (e.g., Naver Aerial View), or satellite as long as the entirety of the site can be seen inside a single shot [19]. Query Scene II is a captured image from mapping API that contains the area where the construction site to be monitored is located. It can simply be captured through the screenshot of the location from Google Maps satellite view in a mobile app. Query

Scene II is a planar image of the ground whereas Query Scene I is an image from oblique viewpoints. A rapid estimation of the homographic relationship between those two images for the projection is in the following step.

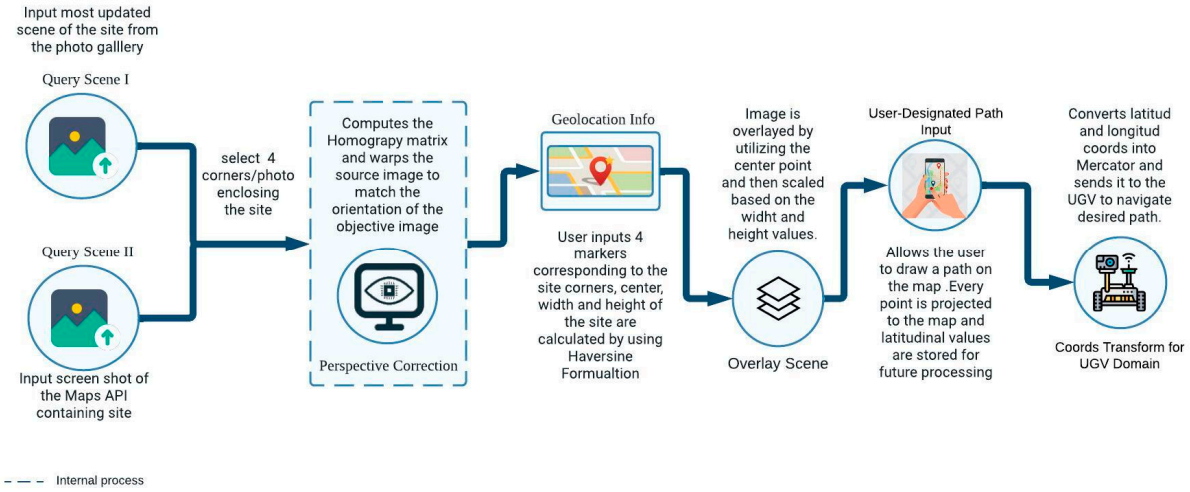


Figure 3. Mobile application workflow.

2.2. Perspective Correction

In this step, the estimation of the homography matrix, a 3 × 3 transformation matrix in computer vision, plays a crucial role in correcting perspective distortion caused by capturing images of planar objects from oblique viewpoints. It describes the perspective transformation between two planar surfaces captured from different camera poses or viewpoints [20,21] as shown in Figure 4.

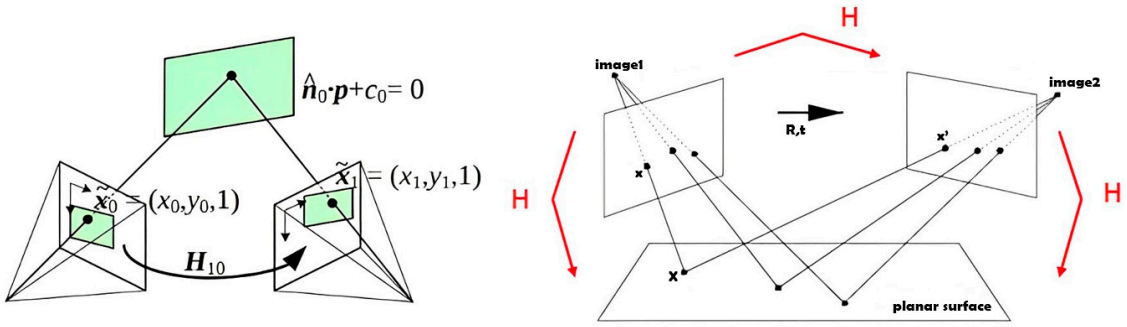


Figure 4. Homography Matrix transformation [22].

By accurately estimating the homography matrix, the transformation between corresponding points in the source and objective images can be determined [23]. Perspective correction involves applying a perspective transformation to the source image, mapping each pixel from the output image back to its corresponding position in the objective image [24]. (Equations (1) and (2)). By utilizing the four points selected on each image by an operator on Query Scene II, the orientation of the recent scene, Query Scene I, is corrected to have the same orientation of the site shown on the mapping API.

$$x' = Hx \tag{1}$$

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2}$$

After the image is warped, a post-process is needed to crop the image so that there are no zero-value pixels. The positions of each corner of the source image, namely (0, 0),

(width, 0), (width, height), and (0, height), along with the center of the quadrilateral, are transformed with the homograph matrix to know their new location as depicted in Figure 5 and Equations (3)–(7).

$$\widehat{P}_1 = H * P_1 = \begin{bmatrix} \widehat{x}_1 \\ \widehat{y}_1 \end{bmatrix} = H * \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{3}$$

$$\widehat{P}_2 = H * P_2 = \begin{bmatrix} \widehat{x}_2 \\ \widehat{y}_2 \end{bmatrix} = H * \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \tag{4}$$

$$\widehat{P}_3 = H * P_3 = \begin{bmatrix} \widehat{x}_3 \\ \widehat{y}_3 \end{bmatrix} = H * \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \tag{5}$$

$$\widehat{P}_4 = H * P_4 = \begin{bmatrix} \widehat{x}_4 \\ \widehat{y}_4 \end{bmatrix} = H * \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} \tag{6}$$

$$\widehat{P}_C = H * P_C = \begin{bmatrix} \widehat{x}_c \\ \widehat{y}_c \end{bmatrix} = H * \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{7}$$

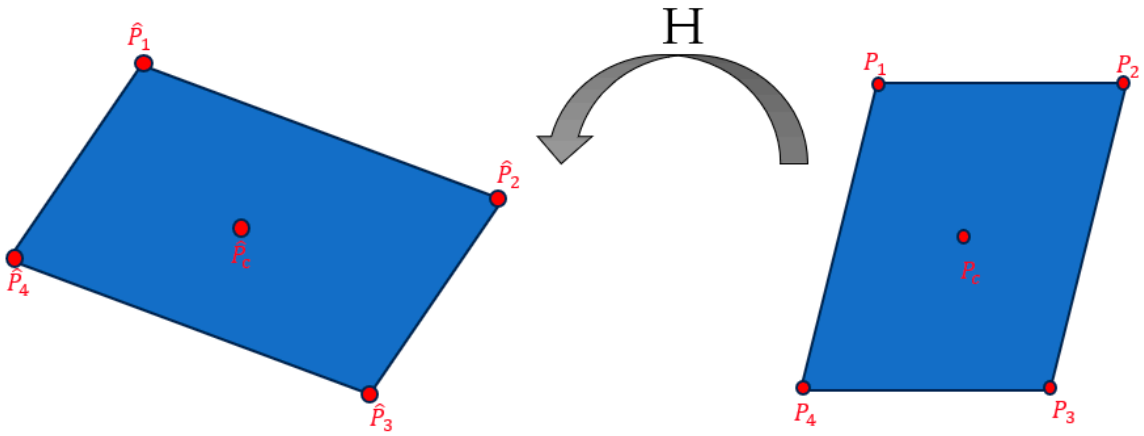


Figure 5. Transformation of planes through the homography matrix.

From these newly computed points, the distance to the center of the quadrilateral is determined for each of the four corners in both the x- and y-axis, and the minimum and maximum value is obtained from the array as shown in Equations (8)–(11).

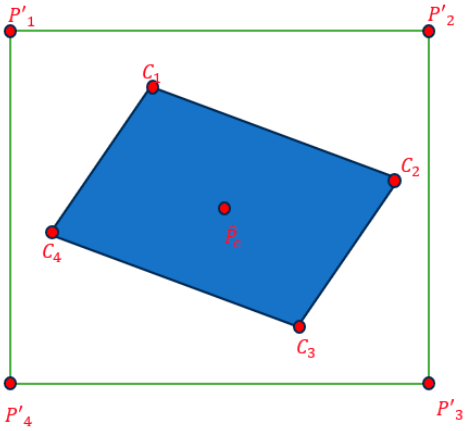
$$d_{x\_min} = min\{|\widehat{x}_1 - \widehat{x}_c|, |\widehat{x}_2 - \widehat{x}_c|, |\widehat{x}_3 - \widehat{x}_c|, |\widehat{x}_4 - \widehat{x}_c|\} \tag{8}$$

$$d_{y\_min} = min\{|\widehat{y}_1 - \widehat{y}_c|, |\widehat{y}_2 - \widehat{y}_c|, |\widehat{y}_3 - \widehat{y}_c|, |\widehat{y}_4 - \widehat{y}_c|\} \tag{9}$$

$$d_{x\_max} = max\{|\widehat{x}_1 - \widehat{x}_c|, |\widehat{x}_2 - \widehat{x}_c|, |\widehat{x}_3 - \widehat{x}_c|, |\widehat{x}_4 - \widehat{x}_c|\} \tag{10}$$

$$d_{y\_max} = max\{|\widehat{y}_1 - \widehat{y}_c|, |\widehat{y}_2 - \widehat{y}_c|, |\widehat{y}_3 - \widehat{y}_c|, |\widehat{y}_4 - \widehat{y}_c|\} \tag{11}$$

Utilizing the shortest distance ensures that the cropped image contains only data, however, there could be a case where utilizing the shortest distance crops part of the area that we are interested in overlaying. Therefore, to have as a result a cropped image that fully displays the area of interest the corners of the site picked by the user (C1, C2, C3, C4) must be inside of the area created by the following points as shown in Figure 6.



$$P'_1 = \begin{bmatrix} \hat{x}_C - d_{x\_min} \\ \hat{y}_c + d_{y\_min} \end{bmatrix}$$

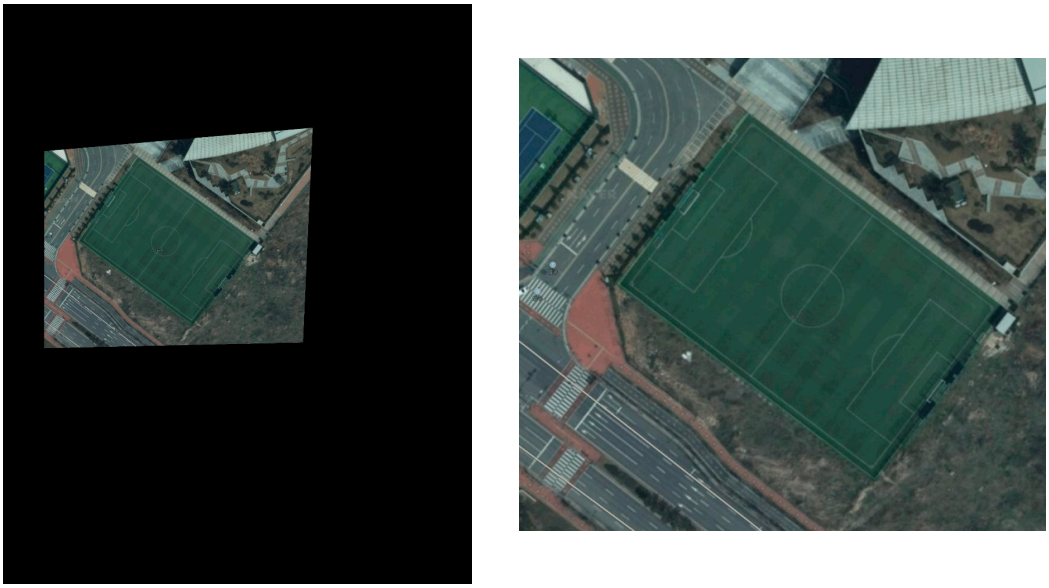
$$P'_2 = \begin{bmatrix} \hat{x}_C + d_{x\_min} \\ \hat{y}_c + d_{y\_min} \end{bmatrix}$$

$$P'_3 = \begin{bmatrix} \hat{x}_C + d_{x\_min} \\ \hat{y}_c - d_{y\_min} \end{bmatrix}$$

$$P'_4 = \begin{bmatrix} \hat{x}_C - d_{x\_min} \\ \hat{y}_c - d_{y\_min} \end{bmatrix}$$

Figure 6. Cropping Area Computation.

If this is the case, Figure 7 shows the before and after of the image. On the other hand, if the condition is not fulfilled, the max distance should be used which reduces the amount of 0 pixels of the image. However, the image must be converted to a PNG type so that the display of the overlayed 0 pixels inside the image is transparent.



(**a**) (**b**)

Figure 7. Image Cropping (**a**) Pre-process Image (**b**) Post-process Image.

2.3. Geolocation Information

Geolocation information will be utilized to overlay the image over the map based on the user input site corners as markers. Since the site will always have a quadrilateral form, the center of the quadrilateral can be calculated by utilizing any of the two opposite side corners. The value obtained will be used as the reference point to overlay the newly obtained image after computing the perspective correction. Furthermore, Haversine formulation (Equations (12)–(14)) is utilized to calculate the distance between two latitudinal points (Figure 8). For this case, the width and height of the site are calculated by taking three out of the four markers input by the user, and consequently, the image is scaled to the displayed size in the mapping API. (φ/λ represents latitude/longitude in radian).

$$a = sin^2(\Delta\varphi/2) + cos\ \varphi 1\ \cdot\ cos\ \varphi 2\ \cdot\ sin^2(\Delta\lambda/2) \quad (12)$$

$$c = 2 \cdot atan2\left(\sqrt{a},\ \sqrt{1-a}\right) \tag{13}$$

$$d = R \cdot c \tag{14}$$

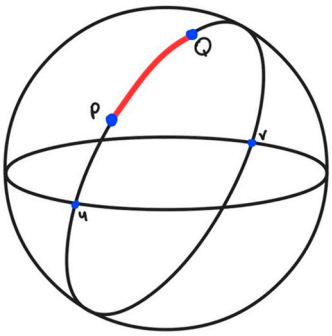


Figure 8. Haversine distance graphical description.

2.4. Overlay Scene

Within the Maps API framework, the Ground Overlay class plays an important role in seamlessly overlaying the designated image (Figure 9) onto the map canvas. To create this object all that is needed is to define both the image and the position. The image itself is defined by two attributes: its dimensions (width and height) which were previously calculated with the Geolocation Information and finally the path to access the image file that is going to be overlayed. Moreover, the positioning of this process is defined by the concept of pivot- the center point around which the image is overlayed. With this, the objective is to align the already perspective-corrected image with the rest of the surrounding elements of the map and have a seamlessly merging image.

Figure 9. GroundOverlay taking SUNY Korea as overlay location.

2.5. User-Designated Path Input

When the user enters path planning mode, the application applies a transparent layer called FrameLayout covering all the screen areas. The user defines the path with free hand drawing and an onTouchEvent displays the path as a polyline on the screen. When the path is drawn over the FrameLayout, every single point of it is projected over the map, and the screen position in pixels of each point is returned to the Map API which converts it to latitude and longitude values (Figure 10).

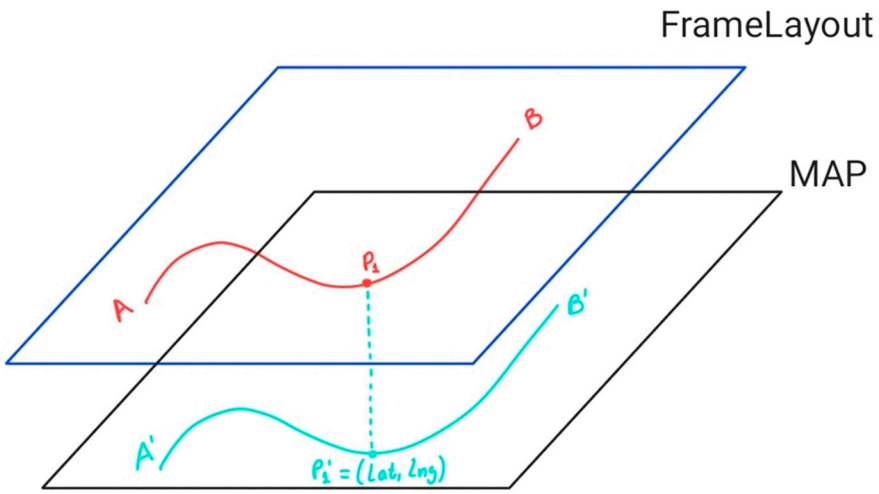


Figure 10. Representation of points transformation.

2.6. Coordinates Transform for UGV Domain

Finally, all the data points collected from the Path input are converted to Mercator (x, y coordinates) for simplicity and accuracy on the UGVs controller side. MQTT is the IoT communication protocol used for the communication between the mobile device and the UGV, it was designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. The QoS (Quality of Service) property of MQTT will ensure the reception of all the messages sent to the user, by applying the highest level of service [25]. This capability is possible because of the bidirectional communication between the publisher and the subscriber through the broker as shown in Figure 11.

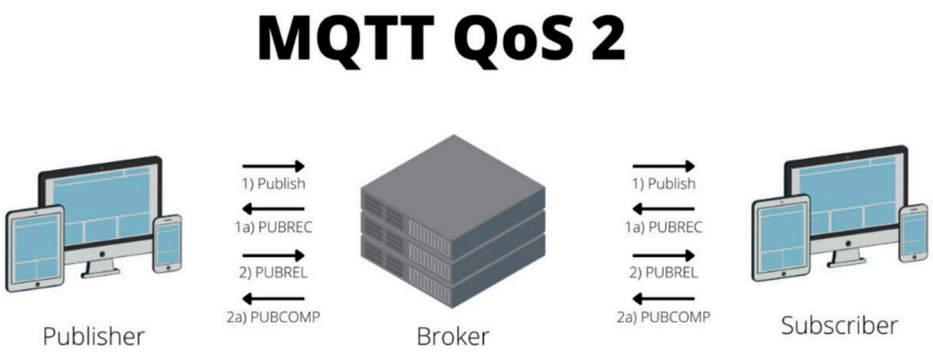


Figure 11. QoS 2 communication between publisher and subscriber [15].

3. Test

3.1. Description of the Test Area

For this section, the algorithm is tested utilizing a soccer field as an example of a construction site. This is due to the fact that the lines of the field can be used as reference points to calculate the area distortion and the offset errors of the overlayed image. The dimensions of the site are 60 × 95 m as shown in Figure 12.

Moreover, it is very important to take into consideration that because the mobile application is very user-dependent, the warped image will depend on the corner selection done by the user, and the overlay of the image is based on both the user markers input and Google Maps GPS error which can get up to 20 m.

Figure 12. Soccer field dimensions.

Please note that the words in Korean on the screenshot image found in both the test section as well as the validation section represent the name of the places and roads near the site. We attempted to remove them however, because of our location the map shows the name of roads and places in that place's language as default.

3.2. AVD Specifications

The simulation of the app was conducted using an Android Virtual Device (AVD) (Figure 13) configured to replicate the specifications of a Pixel 3 smartphone, running on Android API 33. The AVD serves as a software-based emulation of the target hardware, allowing for accurate testing of the application's behavior in a controlled virtual environment. Nonetheless, the application GUI is set to be dynamic and work on any type of mobile device which is why what is more important is that it properly runs on the latest version of Android. The specifications of the device are as follows:

- Android API 33 (Android 13 code name: Tiramisu)
- Device Model: Pixel 3
- Screen size: 1080 × 2028
- Google Maps API v2

3.3. Description of Query Scenes

As explained previously, the query scenes section consists of two images input by the operator through the mobile app. The first image consists of a screenshot of the construction site from the Maps API perspective and the second image is the updated scene. For the initial test, the images used are the once displayed on Figure 14.

For each given image, four corners enclosing the desired site must be selected to internally compute the homography matrix. Figure 15 demonstrates the selection process for Query Scene I and Query Scene II respectively. Note that the dimensions displayed are not the real dimensions of the stored images. The mobile app is designed so that all of the components (image, buttons, corner selection panel, etc.) can be displayed on the screen. Therefore, for an image like a screenshot where its actual size will always be the size of the mobile device screen, it must be manipulated so that all the elements can be displayed.

3.4. Overlay Scene and Error Analysis

Since the path input by the operator is going to be on the overlayed image, it is necessary to know the percentage of the distorted area by calculating the ratio of the area of the site inside Google Maps with the area on the warped image. Since the exact area of the field is already known, it is only necessary to calculate the area in terms of pixels. By utilizing the drawPolygon function in MATLAB, an array with the coordinates of all four corners can be retrieved and then we can use the polyarea function to calculate the area of the quadrilateral by inputting the corners as displayed in Figure 16.

$$error = (A_{pixel\_D} - A_{pixel\_H}) * r_A$$

$$r_A = \frac{A_{real\_site}}{A_{pixel\_D}}$$

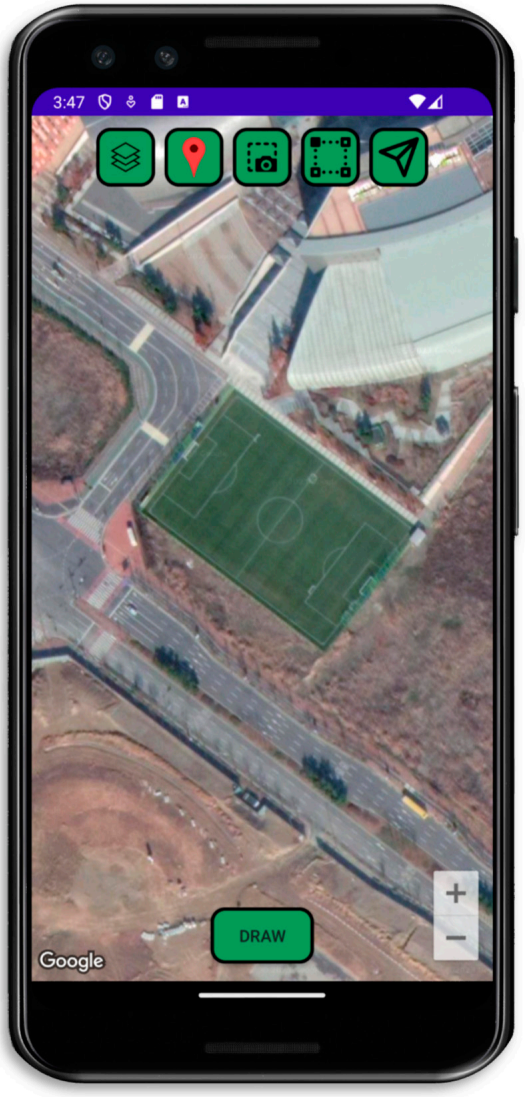


Figure 13. Android Virtual Device (Pixel 3 Model).

Query Scene I **Query Scene II**

Figure 14. Query Scenes Input for Test.

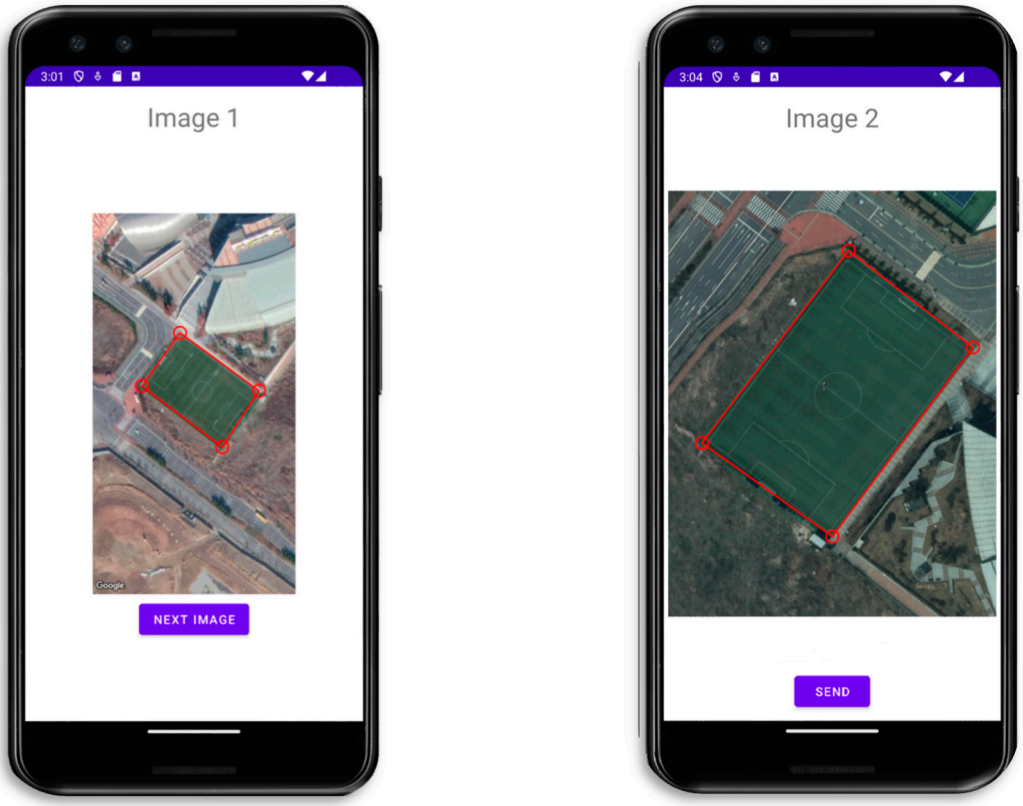
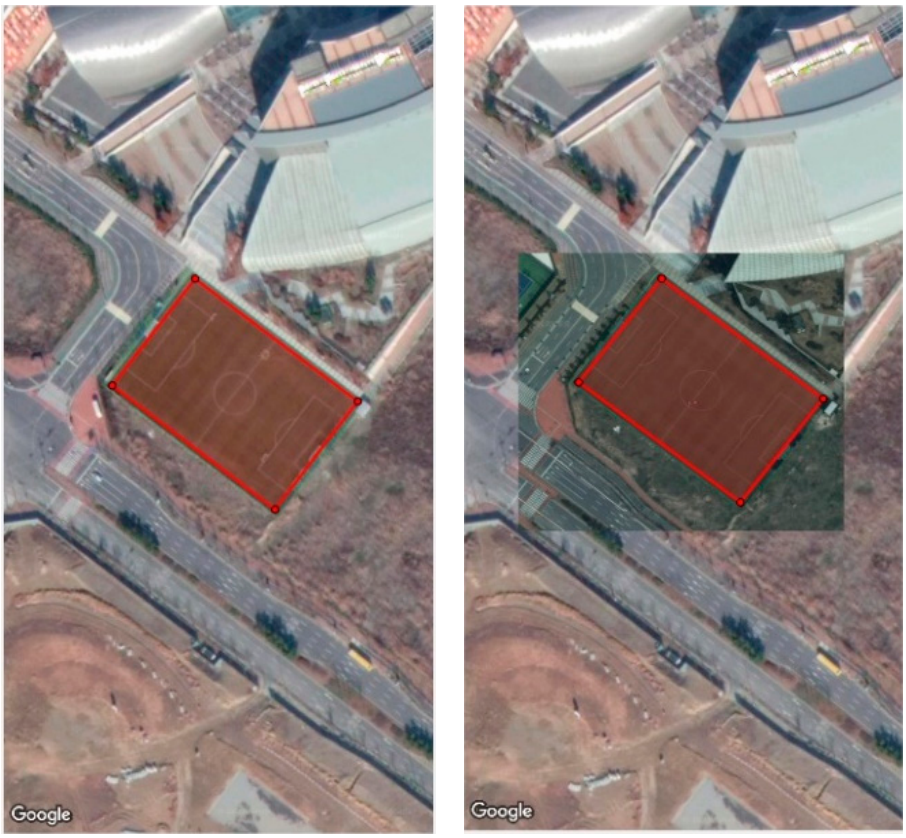


Figure 15. Corner Selection for each Query.



(**a**) (**b**)

Figure 16. Area selection (**a**) Mapping API snapshot (**b**) Overlayed Image.

For the given example, considering the previous equations, the distorted area error is 8.16%. On the other hand, the maximum distance error due to the distortion after warping the image is measured as follows. The warped image is Overlayed on top of the site with a transparency of 50%, this way the offset between the field lines will determine the distance error. By utilizing the same corner selection principle, the maximum and minimum offset distance can be calculated by utilizing the distance between points. This is because both images are of the same pixel size.

$$d(c1_H, c1_D) = \left(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right) * r_d$$

$$r_d = \frac{d_{real}}{d_{pixel}}$$

Figure 17 shows the max and min offset between all 4 corners. The mean error will be the value considered as the error value to subtract when sending the coordinates to the UGV.

3.5. User-Designated Path Input

The path drawn by the operator is projected under the Maps Api and then points are stored for further processing. To ensure that it is accurately stored and that the values transformed from (latitude, longitude) to Mercator (x, y) are correct, data visualization will be used as a method of confirmation.

Data could easily be visualized by creating a marker for each point stored. However, since a particular path could have N numbers of points, a function like linspace in MATLAB or Python is applied for better visualization purposes. The linspace function gets the

first and last values of a given array and then calculates an equidistant rate between the remaining values and displays them as markers as shown in Figure 18.

Figure 17. Distance error.

Figure 18. Points visualization using linspace function.

However, since the objective is to demonstrate that all points scored by the getProjection method are accurate, all the given points should overlay on the path drawn (Figure 19). For the following path example, there is a total of 74 points taken with a mean distance of

30 cm between each point Table 1 shows all the points stored with their respective transformation to X, Y coordinates after being processed so that there are no points repeated.

Figure 19. Points visualization for any given path.

Table 1. Path Drawn Points Coordinates.

					Ground Truth	
#	**Latitude**	**Longitude**	**X [10^6 m]**	**Y [10^6 m]**	**X [10^6 m]**	**Y [10^6 m]**
1	37.3737001430493	126.666829884052	2.9340891	41.3888354	2.93408824	41.38883441
2	37.3737001430493	126.666832230985	2.9340911	41.3888354	2.93409024	41.38883441
3	37.3736980115157	126.666832566261	2.9340914	41.388833	2.93409054	41.38883201
4	37.3736980115157	126.666834577918	2.9340932	41.388833	2.93409234	41.38883201
5	37.3736969457488	126.666838936507	2.934097	41.3888317	2.93409614	41.38883071
6	37.3736969457488	126.666840612888	2.9340985	41.3888316	2.93409764	41.38883061
7	37.3736945477734	126.666843295097	2.9341008	41.3888289	2.93409994	41.38882791
8	37.3736942813316	126.666846312582	2.9341035	41.3888286	2.93410264	41.38882761
9	37.3736916169143	126.666849330067	2.9341061	41.3888255	2.93410524	41.38882451
10	37.3736886860552	126.666852682828	2.9341089	41.3888222	2.93410804	41.38882121
11	37.3736865545213	126.666857041418	2.9341127	41.3888197	2.93411184	41.38881871
12	37.3736838901037	126.666860058903	2.9341153	41.3888167	2.93411444	41.38881571
13	37.3736809592443	126.666861400008	2.9341165	41.3888134	2.93411564	41.38881241
14	37.3736790941518	126.666864417493	2.9341191	41.3888113	2.93411824	41.38881031

Table 1. *Cont.*

					Ground Truth	
#	**Latitude**	**Longitude**	**X [10^6 m]**	**Y [10^6 m]**	**X [10^6 m]**	**Y [10^6 m]**
15	37.3736756304086	126.666866764426	2.9341211	41.3888074	2.93412024	41.38880641
16	37.3736732324325	126.666869781911	2.9341237	41.3888047	2.93412284	41.38880371
17	37.3736697686890	126.666871793568	2.9341253	41.3888008	2.93412444	41.38879981
18	37.3736671042709	126.666872128844	2.9341256	41.3887978	2.93412474	41.38879681
19	37.3736631076435	126.666875146329	2.9341281	41.3887933	2.93412724	41.38879231
20	37.3736604432251	126.666877493262	2.9341301	41.3887903	2.93412924	41.38878931
			.....			
70	37.3734784632248	126.666935160756	2.9341762	41.3885871	2.93417534	41.38858611
71	37.3734763316849	126.666937507688	2.9341782	41.3885847	2.93417734	41.38858371
72	37.3734752659149	126.666940525174	2.9341809	41.3885835	2.93418004	41.38858251
73	37.3734736672599	126.666942536830	2.9341826	41.3885816	2.93418174	41.38858061
74	37.3734726014898	126.666946224868	2.9341859	41.3885804	2.93418504	41.38857941

3.6. Coordinate Transform to UGV Domain and Error Analysis

Once the path is drawn and the points stored, it is important to look into the point transformation from latitude, and longitude to X, Y coordinates. Since the coordinates domain in which the rover is working is Mercator, the transformation should be very accurate to ensure that it will follow the given path.

It's worth mentioning that the offset values between the ground truth (the actual point locations) and the X, Y values are quite close. This occurs because the error for both axes is less than 1 m, and Mercator's origin (0, 0) aligns with the intersection of the Equator and the Prime Meridian. So, for coordinates of this magnitude, the error might seem tiny. However, when we get back to our reference point, a 1-m error, while not huge, should still be considered and adjusted in the obtained coordinates.

Finally, the x and y coordinates are taken to compute the polyval function from MATLAB. The polyval function receives both arrays and computes a N grade polynomial that fits the arrays the best. Figure 20 highly resembles the path drawn in Figure 18 showing that the transformation of the points to Mercator is done correctly.

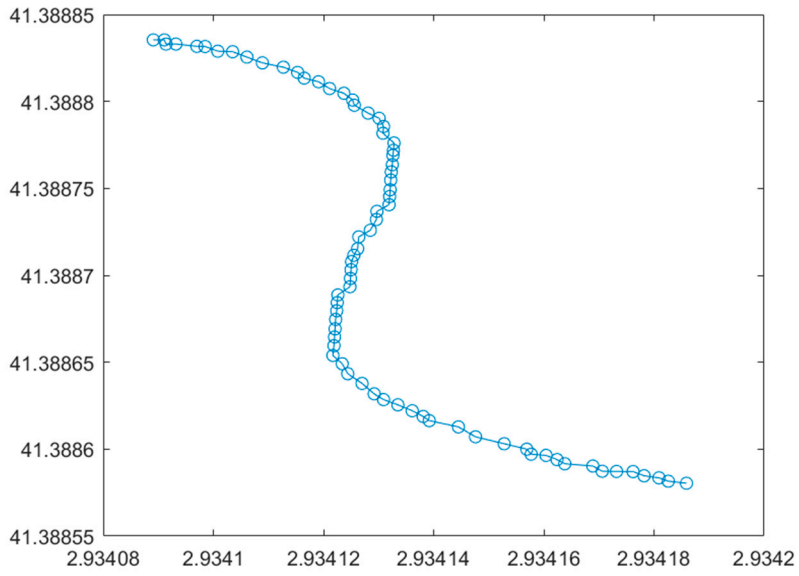


Figure 20. Data Visualization transform to UGV Domain.

Since graphic confirmation from the User-designated Path Input and data visualization from the Coordinate transformation to the UGV domain are working properly, we can ensure that the path input by the user will be stored and processed properly to follow the given instructions if there is no data lost when communicating with the UGV.

4. Validation

4.1. Description of Construction Sites

To validate the proper functioning of the mobile app, two different construction sites were tested. The updated scene image from each site was obtained from Naver Maps Drone view. According to Naver Maps, the gathering of the images was during December of 2018.

The first construction site is situated at 179-4 Songdo-dong, Yeonsu-gu, Incheon. The dimension of the site, approximately 115 × 65 m was estimated utilizing the measurement tool provided as on Figure 21.

Figure 21. Construction Site #1 dimensions.

Figure 22 shows the images that were input to compute the perspective correction as part of Query Scene I and Query Scene II.

The second construction site is located at 75, Songdogyukyo-ro, Yeonsu-gu, Incheon. For this case, the Google Maps measuring tool was used since the satellite view of Naver maps had already been updated and finished its construction. The dimensions of the site are 75 × 75 m (Figure 23).

Similarly, the images on Figure 24 were utilized for Query Scene I and Query Scene II.

From all the previous given cases, the Area distortion error had no linear correlation with the offset error as seen on Figure 25. Generally speaking, the distortion around them will be more noticeable around the corners, which is in fact shown in all the overlayed images. With this in mind, the nearer a corner of the site is to the newly warped image, the more distorted it could be. However, it is very important to note that the offset calculations depend on pixel location which implies that the point locations given by the user will also affect the error calculations.

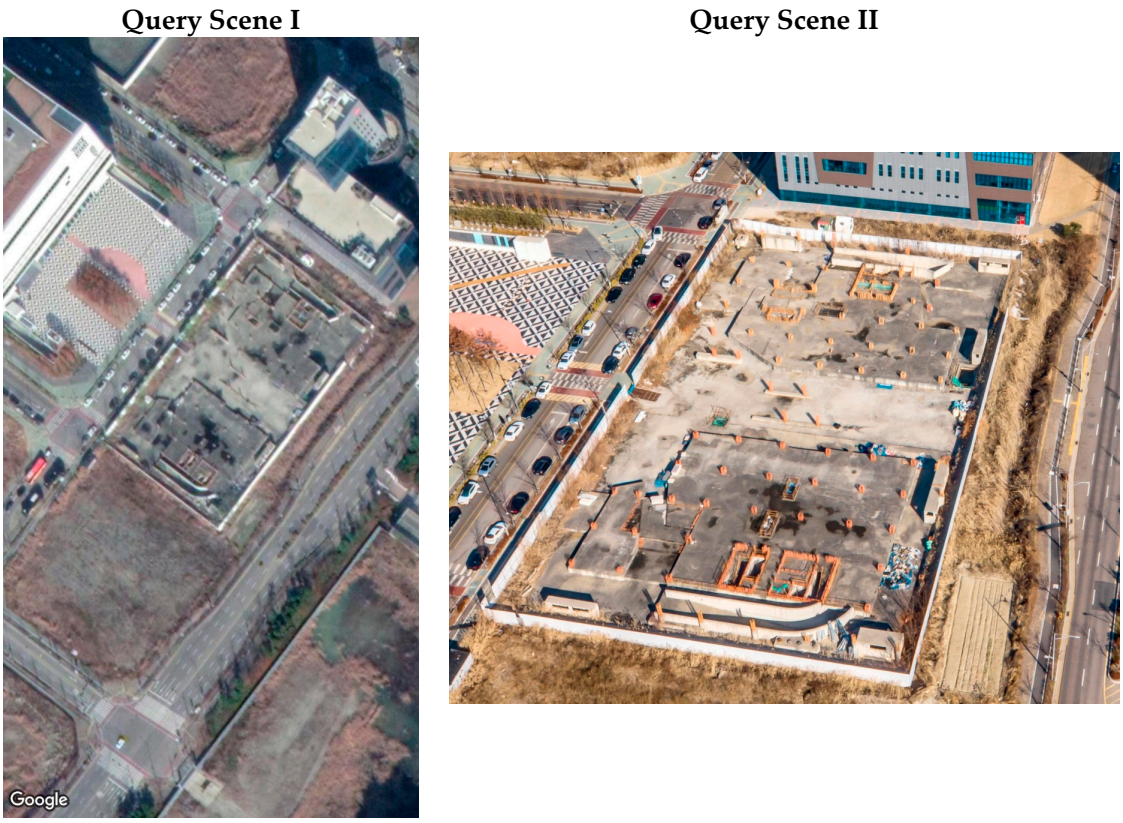


Figure 22. Query Scenes Input for Construction Site#2.

Figure 23. Site 3 Dimensions.

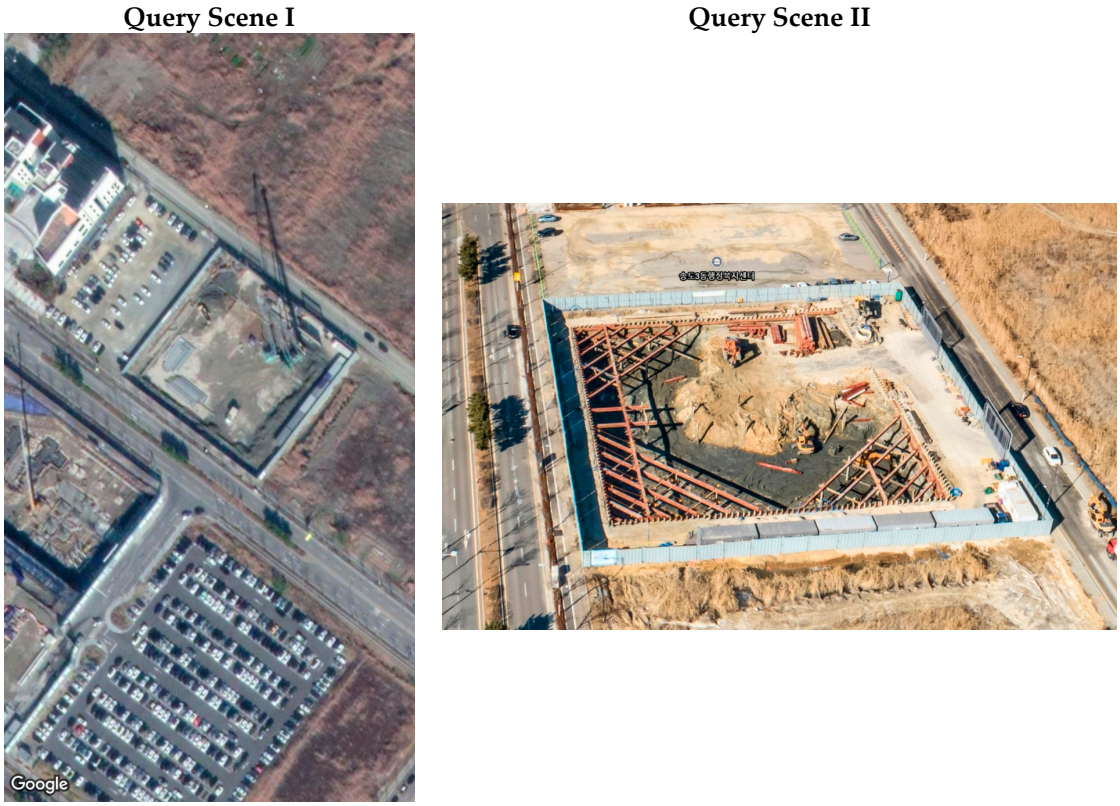


Figure 24. Query Scene Input for Construction Site#2.

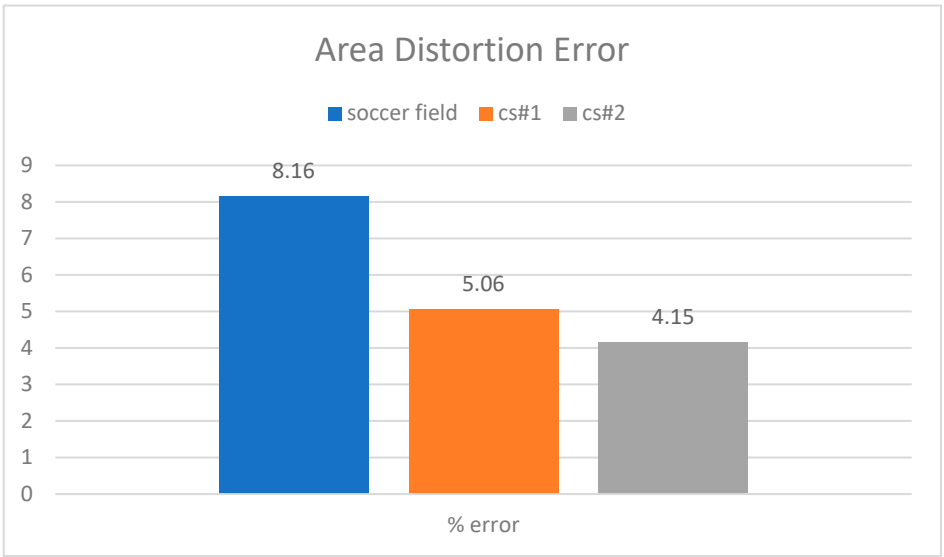


Figure 25. Area Distortion Error.

4.2. Result of Overlayed Scene

4.2.1. Construction Site #1

From the area selection shown in Figure 26, the distortion error is 5.06%. Considering that the site is bigger than the example given, the decrease in the distortion error was unexpected since warped image error depends on the user input corners (Figure 27). However, it is easy to see that it is, in fact, true because the overlayed image offset is less when comparing the overlayed mosaic tiles of the image with the actual Maps Api.

Figure 26. Area Selection for Construction Site#2.

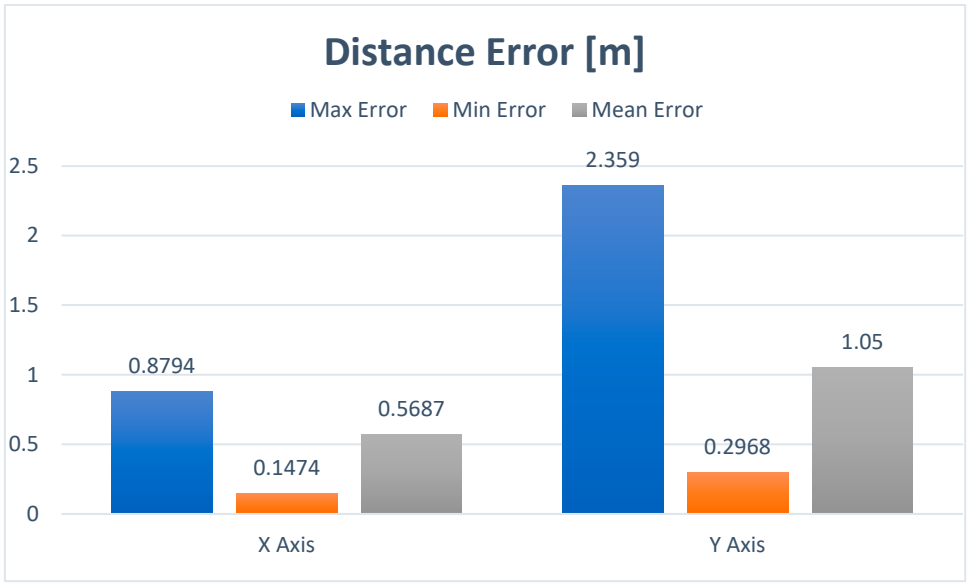


Figure 27. Site 2 Max-Min offset error.

4.2.2. Construction Site #2

From Figure 28 we can observe that both the left top and bottom corners seem to have distorted while warping the image. This occurrence is because of the limitations of the mobile device screen size, the user input, and Google Maps GPS error. Figure 29 shows that despite the site being the smallest one, the offset error for both axes are higher than the two cases previously discussed.

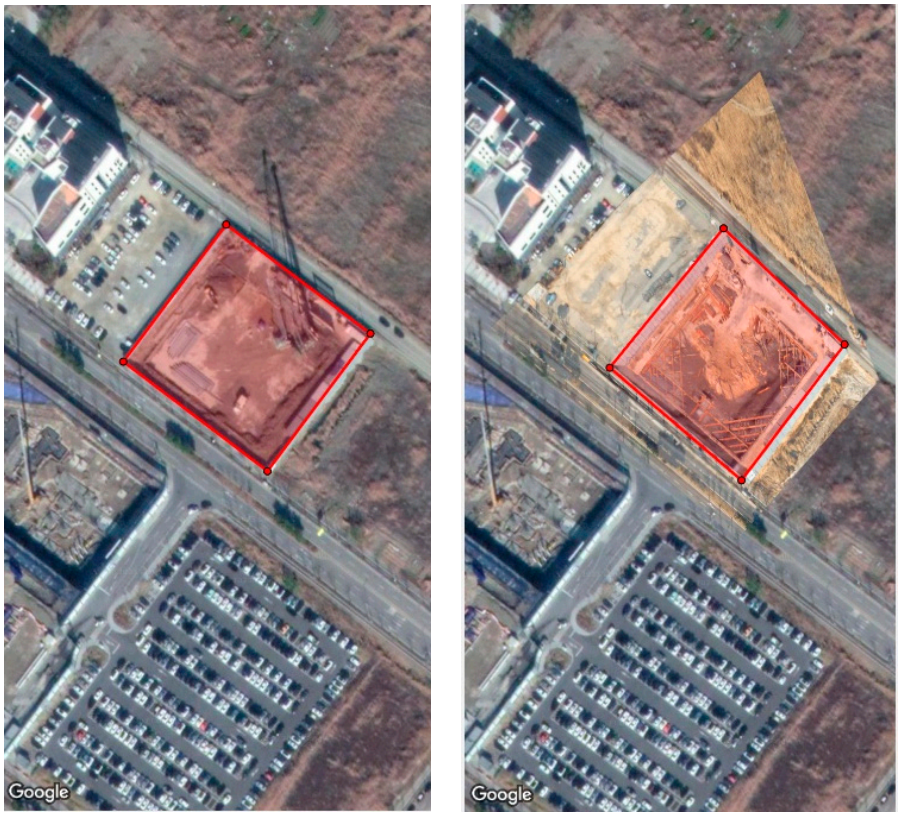


Figure 28. Site 3 Area Selection.

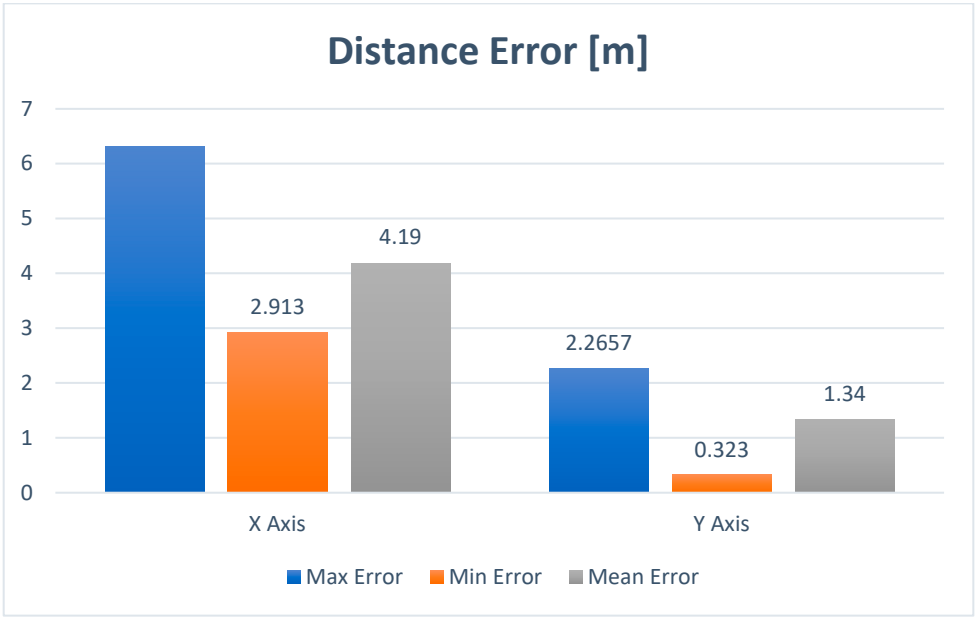


Figure 29. Site 3 Maxi-Min offset errors.

5. Conclusions

The mobile application demonstrates its capability to obtain a warped image, achieved by first computing the precise homography matrix and subsequently adjusting the perspective to the construction site satellite view perspective from Google Maps. An essential part of this transformation involves the normalization of coordinates values obtained on the corner selection section. The obtained values reference is based on the mobile device screen meanwhile the values needed to compute the homography matrix and subsequently warp

the image must be pixel based. Hence a transformation from the screen base domain to the pixel domain was carried out.

The user-designated path undergoes a projection process, resulting in the acquisition of corresponding latitudinal values. These values have an average interval of 30 cm between each other. A filtering process step is implemented to eliminate any instances of redundant points. Subsequently, the transformation of these data points into the Universal Transverse Mercator (UTM) domain is effectively carried out. A graphical confirmation validates the alignment of points, confirming the system's ability to faithfully reconstruct the user-defined pathway, ensuring the procedure's robustness and fidelity.

Applying the proper corner selection directly affects the results of the overlaying of the image, this is because the computation of the homography matrix is based on user selection. Despite this, results show that the area distortion for all cases is lower than 10% and the mean offset values for both axes are under one meter. Furthermore, better-warped images can be obtained by enhancing the corner selection section which will have a positive impact on both the area distortion error and offset errors.

6. Future Work

- With the current version of the app, while functional, the offset errors are propagated during the corners selection of the image to compute the homography matrix. Therefore, one notable area of improvement lies in the integration of a zoom capability. This feature will allow the user to choose more specific pixel points that are not easily noticeable because of the screen's size limitation when projecting the images.
- After warping the source image, an internal cropping algorithm was implemented inside the mobile application. However, during the validation section, we realized that depending on the resolution and size of the images the cropping algorithm does not always return an image without black pixels inside the images. Because of this, it would be necessary to implement a way to return PNG images so that when overlaying it over Google Maps none of the black pixels are displayed.
- Since the mobile application has not been integrated with the UGV, we haven't tested the communication protocol. The selection of the MQTT protocol was made based on the benefits that it offers for the circumstances inside a construction site. The correct integration will ensure that no data is lost so that the UGV can complete the inspection based on the desired path. Furthermore, comparing the odometry of the UGV with the previously drawn path will ensure that the path is properly followed.

Author Contributions: Conceptualization, J.B.C.; Methodology, A.V. and R.O.; Validation, A.V., R.O., B.K., M.K. and T.K.; Investigation, A.V.; Writing—original draft, A.V.; Writing—review & editing, J.B.C., B.K., M.K. and T.K.; Supervision, J.B.C.; Project administration, A.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2022R1F1A106361711 and 2022M1A3C2085237).

Data Availability Statement: The image dataset used to validate the accuracy of the mobile application and its correct implementation were retrieved from NAVER Maps Drone View.The following links will lead to each of the images. Test: https://naver.me/IFI9wWyp Site #1: https://naver.me/GzghQDn7 Site #2: https://naver.me/GEItZtRG.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Awwad, R.; El Souki, O.; Jabbour, M. Construction safety practices and challenges in a Middle Eastern developing country. *Saf. Sci.* **2016**, *83*, 1–11. [CrossRef]
2. Choi, J.; Dyke, S.J. CrowdLIM: Crowdsourcing to enable lifecycle infrastructure management. *Comput. Ind.* **2020**, *115*, 103185. [CrossRef]
3. Lattanzi, D.; Miller, G. Review of robotic infrastructure inspection systems. *J. Infrastruct. Syst.* **2017**, *23*, 04017004. [CrossRef]

4. Kim, J.H.; Kwon, J.W.; Seo, J. Multi-UAV-based stereo vision system without GPS for ground obstacle mapping to assist path planning of UGV. *Electron. Lett.* **2014**, *50*, 1431–1432. [CrossRef]
5. Asadi, K.; Suresh, A.K.; Ender, A.; Gotad, S.; Maniyar, S.; Anand, S.; Noghabaei, M.; Han, K.; Lobaton, E.; Wu, T. An integrated UGV-UAV system for construction site data collection. *Autom. Constr.* **2020**, *112*, 103068. [CrossRef]
6. Choi, J.; Yeum, C.M.; Dyke, S.J.; Jahanshahi, M.R. Computer-aided approach for rapid post-event visual evaluation of a building façade. *Sensors* **2018**, *18*, 3017. [CrossRef] [PubMed]
7. Langley, R.B. Rtk gps. *GPS World* **1998**, *9*, 70–76.
8. Isaac, S.; Edrei, T. A statistical model for dynamic safety risk control on construction sites. *Autom. Constr.* **2016**, *63*, 66–78. [CrossRef]
9. Sud, A.; Andersen, E.; Curtis, S.; Lin, M.C.; Manocha, D. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *IEEE Trans. Vis. Comput. Graph.* **2008**, *14*, 526–538. [CrossRef]
10. Sattineni, A.; Schmidt, T. Implementation of mobile devices on jobsites in the construction industry. *Procedia Eng.* **2015**, *123*, 488–495. [CrossRef]
11. Biel, B.; Grill, T.; Gruhn, V. Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. *J. Syst. Softw.* **2010**, *83*, 2031–2044. [CrossRef]
12. Nah, F.F.H.; Siau, K.; Sheng, H. The value of mobile applications: A utility company study. *Commun. ACM* **2005**, *48*, 85–90. [CrossRef]
13. Choi, K.; Park, S.; Joe, J.; Kim, S.I.; Jo, J.H.; Kim, E.J.; Cho, Y.H. Review of infiltration and airflow models in building energy simulations for providing guidelines to building energy modelers. *Renew. Sustain. Energy Rev.* **2023**, *181*, 113327. [CrossRef]
14. Joe, J.; Im, P.; Cui, B.; Dong, J. Model-based predictive control of multi-zone commercial building with a lumped building modelling approach. *Energy* **2023**, *263*, 125494. [CrossRef]
15. Joe, J.; Im, P.; Dong, J. Empirical modeling of direct expansion (Dx) cooling system for multiple research use cases. *Sustainability* **2020**, *12*, 8738. [CrossRef]
16. Joe, J.; Min, S.; Oh, S.; Jung, B.; Kim, Y.M.; Kim, D.W.; Lee, S.E.; Yi, D.H. Development of Simplified Building Energy Prediction Model to Support Policymaking in South Korea—Case Study for Office Buildings. *Sustainability* **2022**, *14*, 6000. [CrossRef]
17. Nguyen-Huu, P.N.; Titus, J.; Tilbury, D.; Ulsoy, G. *Reliability and Failure in Unmanned Ground Vehicle (UGV)*; GRRC Technical Report 2009-01; Digit. Equip. Corp.: Maynard, MA, USA, 2009.
18. Golparvar-Fard, M.; Peña-Mora, F.; Arboleda, C.A.; Lee, S. Visualization of construction progress monitoring with 4D simulation model overlaid on time-lapsed photographs. *J. Comput. Civ. Eng.* **2009**, *23*, 391–404. [CrossRef]
19. Hyun, M.; Kim, J.-H. Drone View Contents Design Using The GPS-Based Drones and VR. In Proceedings of the Korea Information Processing Society Conference, Seoul, Republic of Korea, 9–10 June 2016; Korea Information Processing Society: Seoul, Republic of Korea, 2016.
20. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Visibridon*; Cambridge University Press: Cambridge, UK, 2003.
21. Szeliski, R. *Computer Vision: Algorithms and Applications*; Springer Nature: Berlin/Heidelberg, Germany, 2022.
22. Bradski, G. The openCV library. *Dr. Dobb's J. Softw. Tools Prof. Program.* **2000**, *25*, 120–123.
23. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
24. Forsyth, D.A.; Ponce, J. A modern approach. *Comput. Vis. Mod. Approach* **2003**, *17*, 21–48.
25. Atmoko, R.; Riantini, R.; Hasin, M.K. IoT real time data acquisition using MQTT protocol. *J. Phys. Conf. Ser.* **2017**, *853*, 012003. [CrossRef]