EFFICIENT UAV FLIGHT PLANNING FOR LOD2 CITY MODEL IMPROVEMENT

Yu-Lun Wu*, Bashar Alsadik, Sander Oude Elberink, George Vosselman

Faculty of Geoinformation Science and Earth Observation (ITC), University of Twente, Enschede, The Netherlands
y.wu-2, b.s.a.alsadik, s.j.oudeelberink, george.vosselman @utwente.nl

Commission II, WG

KEY WORDS: UAV flight planning, imagery data acquisition, point cloud, 3D reconstruction, LoD2 city model.

ABSTRACT:

Geometric errors in LoD2 building models can be caused by the modeling algorithm but are often related to the quality of input data. One approach to tackling the modeling errors caused by the quality of input data is to collect additional data with a UAV and remodel the buildings. However, no flight planning approach exists specifically designed for efficient data recollection for model improvement. In this paper, we propose an innovative flight planning approach for this purpose. Contrary to the conventional method that recollects the data covering the entire building roof, our approach only collects the data over the erroneous region and uses it to improve the erroneous model part later. Our algorithm utilizes the existing LiDAR survey data to automatically detect model errors and design the camera networks by considering the roof geometry. We optimize the trajectory that connects the viewpoints with a genetic algorithm and develops an obstacle avoidance function with ray-casting to ensure a collision-free path. The proposed flight plan is implemented in a real-world scene. Our result shows an improved point cloud created through dense image matching with the collected UAV image data. The generated point cloud is successfully used for creating partial building models for improving the original models.

1. INTRODUCTION

1.1 Motivation

The foundation of the city model is the individual building model. According to the City GML standard (Kolbe, 2009), building models can be classified into categories from LoD0 to LoD4 based on the level of detail (LoD). Among these categories, the LoD2 model is required to contain roof geometries and semantic classes such as roofs and walls, while the LoD3 model further requires to include the facade structures (e.g. doors, windows). An example of the mentioned LoD models is illustrated in Figure 1.

Nationwide 3D models are usually constructed based on point clouds obtained by airborne LiDAR data or dense matching of aerial photographs. Compared to LoD3 models, the automatic reconstruction of LoD2 models using point clouds is relatively mature. However, the automatically generated LoD2 building models may still contain some geometric errors. For instance, in the Dutch nationwide city model, 3D BAG (3D BAG, 2021), around 10% of the reconstructed LoD2 models contain invalid geometry in the release of March 2021 (Dukai et al., 2021). The factors for incorrect modeling can be two-fold: 1) the reconstruction algorithm, or 2) the quality of the input point cloud data. Since the model errors to be repaired are scattered and UAVs are suitable for locally collecting additional data, one approach to tackling the modeling errors caused by the quality of input data (e.g., low point density, lack of data) is to recollect the data with UAV and then re-model the buildings. However, to the best of our knowledge, there are no existing UAV flight planning methods specifically designed for data collection for LoD2 model improvement.

For LoD2 model correction, using commercial flight planning software such as Pix4D (Pix4D, 2022), which employs conventional planning techniques for data acquisition has several disadvantages. First, it requires manually identifying the erroneous models and delineating the regions of interest (ROIs). Second, the viewpoints are generated in a grid shape following a predefined overlap rate without considering the object geometry, therefore causing self-occlusion during data acquisition. Third, the data are manually collected one ROI after another discontinuously, which is time-consuming. Additionally, while only a small part of the model contains errors, the traditional method requires recollecting all the data for remodeling the entire building, which causes redundancy.

In this paper, we propose an innovative approach to efficient UAV flight planning to tackle the above-mentioned problems. We utilize the existing airborne LiDAR survey data, which is used to generate the original LoD2 models, to automatically detect model errors, build up the camera networks, and avoid obstacles in the pre-planned path. Contrary to conventional methods that collect the full data and remodel the entire building, our method collects data that merely covers the erroneous model part and its surroundings. The newly collected data is then used to create a partial model to replace the erroneous part of the original model. Our UAV flight planning is able to generate reliable point clouds to represent the scenes through dense image matching. While the focus of this paper is flight planning, we also demonstrate the result of an improved LoD2 model at the end of the paper. We envision an automated repeated onboard cycle of model reconstruction, error detection, and additional data acquisition for LoD2 city model improvement.

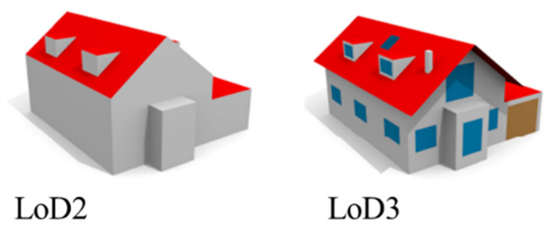

Figure 1. Illustration of the LoD2 and LoD3 models. (Biljecki et al., 2016)

1.2 Related Work

To achieve accurate 3D reconstruction from UAV images through dense image matching, a crucial component is the camera network, which forms the core of flight planning. The factors affecting the quality of 3D reconstruction results may include ground sampling distance (GSD), forward and sideward image overlap rates, observation angles, and the avoidance of occlusion. Conventional flight planning methods that follow simple polygon or grid-pattern nadir viewpoints are limited to reconstructing the roof parts. Due to the lack of considering the object's surface geometry, they usually lead to 3D reconstruction problems such as noisy boundaries or lack-of-data areas.

There are several studies focusing on optimizing camera positions and orientations based on the initial coarse 3D geometry of a targeted object. Nagasawa et al. (2021) proposed a multi-UAV coverage path planning for the 3D reconstruction of post-disaster damaged buildings. Koch et al. (2019) formulate path planning as a graph-based optimization problem that maximizes the information gain along the planned trajectory. Another category of optimizing the camera poses is next-best-view planning (NBV), which iteratively decides the next camera pose based on the currently available geometric information (Bircher et al., 2016; Vasquez-Gomez et al., 2014).

One of the path-planning algorithms used to navigate the UAV through all the waypoints is the genetic algorithm (Nagib & Gharieb, 2004). The genetic algorithm is an optimization technique that contains an iterative process that mimics natural selection, genetics, and evolution. The iterative process involves ranking the current population (i.e. possible solutions) given a fitness function, and then reproducing new solutions from highly-ranked parent solutions by crossover and mutation. The genetic algorithm can be used to solve the traveling salesman problem where every waypoint is only visited once.

For obstacle avoidance in a known static environment, Greiff & Robertsson (2017) conducted obstacle avoidance in 2D space based on finding the boundary points of the inflated obstacle by projection. However, this method would fail if there are multiple objects in the scene between two waypoints. Han (2019) proposed an efficient obstacle avoidance planning method by finding a subset of grid points that can fully surround the obstacles in 3D. However, the altered waypoints are generated on grid points instead of a free space, which may not achieve an optimized path. Liu et al. (2020) proposed an elliptic tangent graph method to generate two possible paths when confronting an obstacle and select one path based on the heuristic rule. The process iterates until reaching the target. Nevertheless, the research is limited to 2D. In our research, we adopt the genetic algorithm to connect the generated viewpoints and develop a method to find a collision-free path that connects two waypoints in 3D space.

2. METHOD

Our UAV flight planning is for collecting roof imagery for 3D reconstruction in a nadir-view setup. It tackles the problem that conventional flight planning methods are inefficient in collecting data over multiple scattered ROIs. By utilizing existing LiDAR data, we are able to automatically detect modeling errors, generate viewpoints that consider roof geometries, and conduct pre-planned obstacle avoidance.

Following the error detection to identify where the existing model parts require improvement and new data collection, the ROI polygons are first clustered together based on proximity and roof height differences. Initial gridded viewpoints are then generated locally for each cluster. The final viewpoints are decided by considering the roof inclination and the visibility test result. We realize an optimized path that connects all the viewpoints by adopting the genetic algorithm. We also develop an obstacle avoidance function with ray-casting to ensure a collision-free pre-planned trajectory. Figure 2 shows the flowchart of the proposed flight planning approach. The components and procedures of the flight planning approach are further explained in the following sections.

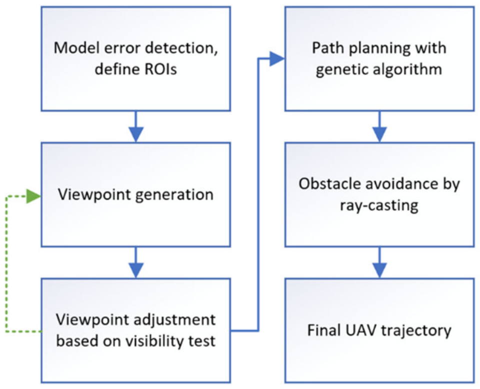
Model error detection, define ROIs
Viewpoint generation
Viewpoint adjustment based on visibility test
Path planning with genetic algorithm
Obstacle avoidance by ray-casting
Final UAV trajectory

Figure 2. Overview of the proposed flight planning approach.

2.1 Error Detection of the Existing Models

To automatically identify modeling errors, we compare the building models and the point cloud used for their reconstruction. Two types of mismatches are being detected. First, we identify where the modeled areas lack data support. This is done by rasterizing the modeled area and finding where the cell that is modeled contains no point cloud data. Second, we calculate the orthogonal distance between the data point to the corresponding model face as done by Oude Elberink & Vosselman (2011) to find the outlier points. When outliers form a cluster that exceeds a certain size, the area is enclosed with a polygon to represent an erroneous region. A sample result of error detection is shown in Figure 3.

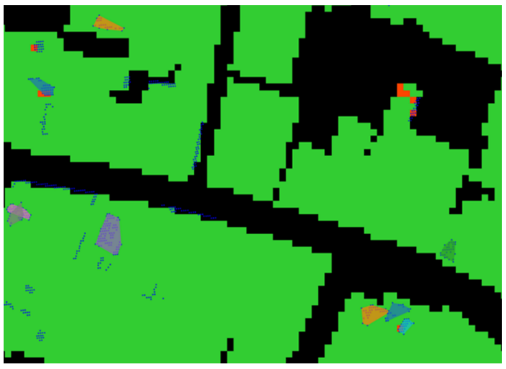

Figure 3. Automatic error detection. The green pixels are the modeled area; the orange pixels are the no-data pixels. The dots are the outlier points, and the translucent polygons indicate larger erroneous areas.

In order to allow an efficient observation of nearby polygons without redundant viewpoints, the local viewpoints are generated together for nearby polygons. The polygons are first clustered based on proximity and roof height differences. The neighbouring polygons are grouped together if the projected distance between them is below a threshold. However, to ensure optimal image scales, a polygon would be split again if it contains large differences in roof height.

2.2 Viewpoint Generation

The generation of viewpoints contains two steps. First, gridded viewpoints are generated locally for a polygon cluster. Second, the algorithm removes redundant viewpoints that fall outside a certain range of the original polygon boundaries. However, it retains boundary viewpoints that support the observation of highly inclined roof faces.

- Generation of the initial gridded viewpoints

The gridded viewpoints are generated locally for each polygon cluster. These viewpoints are generated within a rectangular bounding box that encloses a cluster and is extended by one viewpoint baseline. The baselines are calculated from the pre-defined viewpoint height and image overlap rate, with both end-lap and side-lap being set to the same rate. Initially, the rate is set to 70%, which is the typical overlap rate set for 3D reconstruction while having no prior geometric knowledge of the scene. However, the overlap rate can be gradually increased following the visibility test that is explained later in Section 2.3.

The point cloud density is directly proportional to the image resolution and the overlap between images. Therefore, if the observation height is too high, the image resolution decreases, resulting in a lower point cloud density. This can lead to incomplete 3D models and inaccuracies in measurements. On the other hand, if the observation height is too low, the overlap between images decreases, resulting in lower point cloud density. Therefore, determining the maximum observation height relative to the building roof is crucial in achieving the desired point cloud density and safeguarding the required image GSD.

$$H_{max} = \frac{GSD \cdot f}{pixel\ size} \quad (1)$$

Where H_{max} is the maximum allowed observation height relative to the roof, and f is the camera focal length, which is a camera constant. The viewpoint heights are defined within this threshold by also considering the size of the polygon cluster. For smaller polygons, we further reduce the relative observation height for optimized viewpoint sampling to match the ROI boundaries, with the benefit of achieving an even higher image GSD.
Accordingly, the flying height of the UAV should be chosen based on a balance between the GSD, camera parameters, and practical limitations, with the understanding that other factors such as height accuracy resulting from dense matching will also play a role in the final data quality.

- Redundant viewpoint removal by considering roof inclination.

While removing the redundant viewpoints that fall outside a certain range of the polygon boundaries, the algorithm preserves viewpoints that support observing highly inclined roof faces (i.e., over 60 degrees), which is commonly seen in European alike residential buildings. The reason for keeping these viewpoints is that a large observation angle (i.e., the angle between the normal of a roof plane and the line connecting the viewpoint to the roof point) results in a small projected area on the image, which is not ideal for dense image matching. Finding the viewpoints that have a smaller observation angle could result in a better 3D reconstruction even in a nadir-view setup. Figure 4 illustrates the idea. Viewpoint 2, which facilitates the observation of the steep roof part is preserved during the removal even if it falls outside a certain range of the polygon boundary.

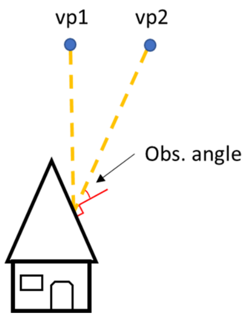

Figure 4. Viewpoint position affects the observation angle and results in different projected area sizes.

To calculate the orientation of the roof faces, the algorithm first rasterizes the ROI into 1 × 1 m^2 cells. This ensures a minimum area to define a cell representing a highly-inclined roof part. The orientation of the roof face within each cell is derived from the existing LiDAR point cloud. Firstly, the surface normal of each roof point is calculated based on a plane fit to neighboring points. Then, a matrix that represents the hemispheric sections is created for each cell to find out the major direction of the normal vectors, which is then defined as the cell orientation. Figure 5 shows how the hemisphere is divided into sections that contain 6 intervals in elevation (15°/per interval) and 8 intervals in azimuth (45°/per interval).

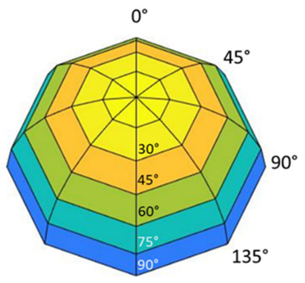

Figure 5. The division of hemispheric sections for orientation statistics.

To examine whether the potentially redundant viewpoint is beneficial for observing highly inclined roof faces, the algorithm creates a search range to identify if within the range exists any cell that has a high inclination and is facing the viewpoint. The viewpoint is preserved if it fulfills these two criteria. The search range is set to a space between 15 to 30 degrees from the plumb line of the viewpoint, which is the space between two cones (Figure 6). For deciding whether the cell azimuth is pointing to the viewpoint, we examine whether the projected vector from the cell to the viewpoint is within the same azimuth interval as the cell azimuth.

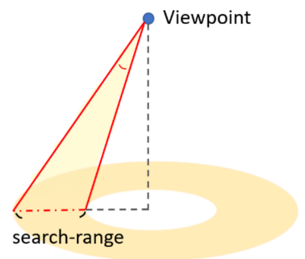

Figure 6. Illustration of the search range (yellow-shadowed) for deciding whether a viewpoint supports observing highly inclined roofs.

The thresholds for defining highly inclined roof faces and the search range were set by considering multi-view stereo (MVS) principles recommended by Koch et al. (2019) and Wenzel et al. (2013). They point out that an observation angle larger than 75 degrees will greatly deteriorate the 3D reconstruction result through dense image matching. Therefore, an additional viewpoint is only helpful when the observation angles are kept within this threshold.

The roof inclination threshold was set to 60 degrees instead of 75 degrees because a local viewpoint is not necessarily located right above the inclined roof face, which can possibly increase the observation angles. Therefore, a 15-degree interval is added as a buffer. For determining the threshold of the search range, since the upper bound of the search range angle is restricted by the horizontal and vertical field-of-view (FOV) of the camera, we set the maximum search angle to 30 degrees. The lower bound of the search range angle is set to 15 degrees to meet the recommended observation angle of 75 degrees while approaching a maximum inclination of 90 degrees.

Figure 7 illustrates the geometry following the defined thresholds. Given a roof inclination of 60 degrees, a 15 to 30-degree search range will result in observation angles between 30 to 45 degrees to the roof plane. While the roof inclination gradually increases to a maximum of 90 degrees, with the same search-range setup, the observation angles increase to between 60 and 75 degrees. This is still within the recommended threshold of 75 degrees.

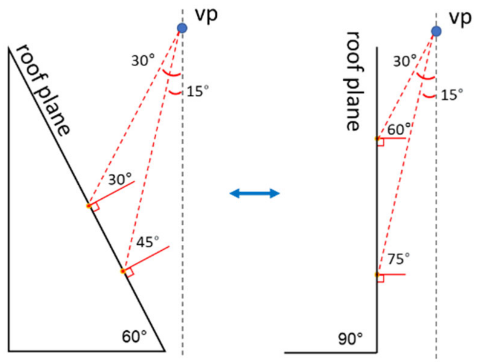

Figure 7. The geometry of the observation angles for a highly inclined roof plane. From a 60-degree inclination to a maximum of 90-degree inclination.

2.3 Viewpoint Adjustment by Visibility Test

Dense 3D reconstruction requires at least 3 views from different perspectives for robust triangulation (Furukawa & Hernández, 2015). Due to self-occlusions from complex roof geometries, some areas of the roof surface may not be seen from at least three of the generated viewpoints. This can increase triangulation uncertainty or even cause data loss over such areas. A clear object boundary in the point cloud is crucial for correctly modeling the roof geometry. However, the boundary areas of the roof structures are more often suffering from bad data quality due to self-occlusions. We try to minimize this problem by reasonably densifying the generated viewpoints given the visibility test result.

The visibility test is used to find the surface points that are visible from a camera without occlusion. We adopt the hidden point removal method from Katz et al. (2007) to find surface points that are visible from at least three viewpoints. The hidden point removal method directly works on point clouds to find the occluded points. Because the existing nadir-view LiDAR data primarily covers the top sides of the roofs with the vertical sides often unrepresented, directly applying this method to the data can cause the problem that actually occluded points are considered visible. Therefore, we first convert the LiDAR point cloud to a mesh, where the vertical sides are filled. The mesh vertices are then resampled to a regular point density to homogeneously represent the object's surface.

Since the process of LoD2 building modeling only requires the point cloud that represents the non-vertical faces of the building, and the nadir-view camera is not designed for facade information acquisition, the surface points on vertical sides are excluded in the calculation of the visibility rate. Additionally, because some of the walls under the eaves are still represented by LiDAR points, it can generate mesh facets facing downward at those local regions during the Poisson reconstruction. Since these mesh facets are invisible from the nadir-view cameras all along, we also exclude them from the calculation by excluding surface points that have normal vectors pointing downward. This results in the definition of the visibility rate as:

$$visibility\ rate = \frac{N_{vp}}{N_{sp}} \tag{2}$$

Where N_{vp} stands for the number of non-vertical surface points visible from at least three viewpoints, and N_{sp} is the number of surface points that are not on the vertical side and their normal vectors not pointing downward.

The local viewpoints are generated as described in Section 2.2 with the image overlap rate initially set to 70%. After conducting the visibility test, we used the results of a visibility test to determine whether to increase the overlap rate further. Our algorithm iteratively regenerated denser viewpoints and conducted visibility tests until the visibility rate did not increase significantly after a new iteration or the image overlap rate reached 90%. This approach allowed us to optimize the viewpoints to ensure maximum visibility.

2.4 Path Planning with Genetic Algorithm

After deriving the final viewpoints for all scattered ROIs, we adopt the genetic algorithm from a Python package scikit-opt (scikit-opt, 2023) to connect these viewpoints by solving the traveling salesman problem in three dimensions. The genetic algorithm is introduced briefly in Section 1.2. To apply this algorithm, we set the initial population size to 50 and set the fitness function to evaluate the total travel distance. This allows us to generate an optimized trajectory for the UAV based on the overall travel length.

2.5 Obstacle Avoidance by Ray-casting

During the local viewpoint generation process, we ensure the generated viewpoints are positioned above any objects within each ROI. This guarantees a collision-free path locally. However, as the UAV moves between different ROIs, there may be some high-rising obstacles on its path that could cause a collision. To ensure a safe trajectory, the algorithm further examines if the linear path between two successive waypoints intersects any obstacle in between. We draw inspiration from (Greiff & Robertsson, 2017) in altering the path by inserting the found boundary points of the inflated obstacle, which ensures a space buffer to the original obstacle. Nevertheless, our modified method mimics the real-world laser sensor by iteratively conducting ray-casting to find the inflated obstacle boundaries in 3D, it is not restricted by the presence of multiple objects in the scene as the work done by Greiff & Robertsson (2017). The procedure is explained below and is illustrated in Figure 9:

1. **Mesh dilation**
   The existing airborne LiDAR point cloud is first converted to mesh with the Poisson surface reconstruction method (Kazhdan et al., 2006). The mesh vertices are then shifted along the normal vectors of the mesh vertices by a certain distance. These new points are used to conduct the Poisson reconstruction again to derive the dilated mesh to represent the scene.

2. **Ray-casting to find the boundary points**
   The algorithm conducts ray-casting to find the candidate waypoints that detour around the obstacles. When a line connecting two successive waypoints, point A and point B, intersects a 3D obstacle, the algorithm generates three ray sets along different directions for scanning to find the three boundary points. The three scanning directions are 1) from A to B and gradually increase the ray elevation, 2) from A to B and gradually point to clockwise/counterclockwise directions. An illustration is shown in Figure 8. In each scanline, the point where the last ray hits the closest intersected inflated mesh is considered the boundary point. Technically, the found boundary point has to be shifted away from the mesh along the surface normal by a certain small distance in order to continue ray-casting. We continue the iteration and cast new ray sets from the shifted boundary point toward point B to find the next candidate waypoints until forming a collision-free path to B. This generates multiple candidate paths that connect point A and point B without collision in the form of a tree structure. The shortest candidate path is then selected and used to alter the original path that intersects the obstacles.

This method is able to find a collision-free path even in the presence of multiple obstacles. However, the success of the mesh reconstruction plays an important role in this function, the failure of correct mesh reconstruction can lead to incorrect path alteration. An example of the path alteration result using the proposed method is shown in Figure 10.

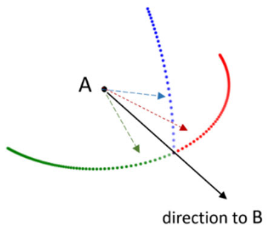

Figure 8. Illustration of the three ray-casting extending directions.

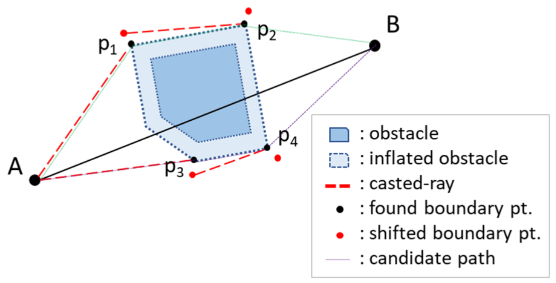

Figure 9. The procedure of finding the candidate path from A to B. The found candidate paths are: A, p1, p2, B and A, p3, p4, B in this example.

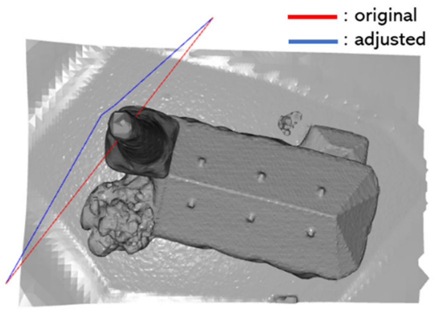

Figure 10. A sample result of the obstacle avoidance function.

3. EXPERIMENT

We implemented the proposed flight planning method for data acquisition in a real-world scene. In the experiment, the Dutch nationwide 3D BAG serves as the LoD2 base model. The Dutch nationwide airborne LiDAR data, AHN (AHN, 2022), which is also used for the reconstruction of 3D BAG, is used for error detection of models and serves as prior knowledge of the scene.

The observation location is selected on the main campus of the University of Twente. In order to comply with the privacy restrictions and avoid a great number of occlusions caused by trees next to buildings, we manually select the ROIs with the aid of automatic error detection mentioned in Section 2.1. The minimum size of the detected erroneous parts was set to 2 m^2. However, due to the limited number of available candidates caused by the mentioned restrictions, we included some visually erroneous model parts that were not automatically detected as our ROIs. We finally defined 6 polygons that contain erroneous model parts in 3D BAG for data acquisition. Because a building section larger than the erroneous model part must be re-modeled

to ensure the integrity of the generated partial model, the polygons were delineated by deciding which building section required being re-modeled and were extended by 1 to 2 meters as the buffer zone.
This buffer zone is included to ensure the derived point cloud covers both the partial building and the ground areas of its surroundings; since the building modeling algorithm requires both the roof and ground points to decide the height of the building. This resulted in polygon sizes ranging from 39m^2 to 227m^2. In our experiment, the defined polygons are located far from each other. Therefore, no polygons were grouped during the polygon clustering process. The defined ROIs can be seen in Figure 13.

The camera parameters for viewpoint generation are set according to the camera onboard DJI Phantom 4 Pro V2.0, where the focal length is 8.8 mm, and the sensor resolution is 3648 × 5472 pixels with a pixel size of 2.41 µm. Our targeted image GSD is equal to or better than 2cm. Under an ideal condition, it can generate a point cloud with a point spacing equal to or smaller than 2cm. Nevertheless, the final point cloud density is affected by factors including the texture and the processing software setups. While the maximum allowed observation distance for achieving 2cm GSD is 73m, we set the local viewpoints 65m above the average roof height for two larger ROIs, and 40m above the rest four smaller ROIs. The different settings of viewpoint heights are because the projected area covered by a camera is proportional to the observation height. Our viewpoint baseline length is generated based on the image overlap rate, thus is also proportional to the observation height. For a smaller ROI, a large observation height would generate a viewpoint baseline much longer than the ROI itself. This may expand the camera network and result in longer path length while generating more redundant viewpoints. Although we are able to gradually reduce the baseline length after each visibility test, reducing the baseline length would also reduce the parallax for a point viewed from two adjacent viewpoints, and thus affect 3D reconstruction accuracy. In order to maximize the 3D reconstruction accuracy and minimize the local camera network, we set a lower viewpoint height for small ROIs. This also results in an even smaller GSD that can generate point clouds with higher point density.

For each ROI, the initial overlap rate was set to 70% for generating the local viewpoints. Following the visibility test, the algorithm generates a denser camera network with an overlap rate increased by 5% in every new iteration. If the visibility rate in this new iteration is not increasing by 3% compared to the previous, the algorithm will consider the previous local viewpoints as optimal and use them as the final local viewpoints. However, technically, in order to avoid a local minimum, we add one iteration to check if the visibility rate is not increased further. Figure 11 shows the resulting visibility rates for each ROI given a different overlap rate.

In addition to the visibility rate, in the experiment, it was found that in order to achieve a slight increase in visibility rate by about 3% in the new iteration, the number of viewpoints can significantly increase by more than 50% compared to the previous iteration. To strike a balance between point cloud accuracy and the number of required viewpoints, a penalty was set for a drastic increase in viewpoint numbers. If the number of viewpoints increases by 50% or more in the new iteration compared to the previous one, even if the visibility rate increases by 3%, the previous viewpoint set will be used in the final viewpoints. This approach was taken to avoid an excessive increase in viewpoint numbers, which may not be practical in terms of time and resources. The specific thresholds for the visibility rate and the penalty for the increase of viewpoint number were chosen empirically based on the experimental results. Following these rules, the algorithm finally outputted 166 viewpoints across the scenes in total. All the viewpoints were automatically examined to ensure they keep a certain distance from the targeted object surfaces. Some statistics for the final viewpoints over each ROI are shown in Table 1.

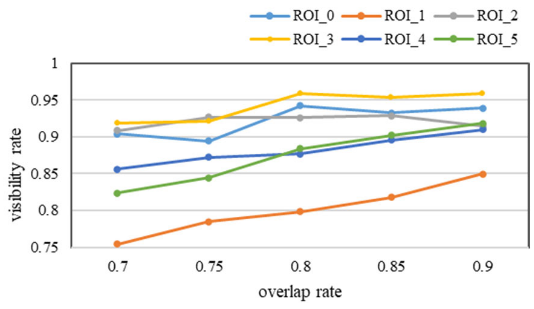

Figure 11. Visibility rates are based on different overlap rates for each ROI.

ROI	0	1	2	3	4	5
Rel. obs. height (m)	40	65	40	40	40	65
Overlap rate	0.8	0.85	0.75	0.8	0.8	0.8
Number of viewpoints	16	59	16	18	23	34
Visible rate (%)	94.2	81.8	92.7	95.9	87.7	88.4

Table 1. Statistic information of the final viewpoints for each ROI.

The viewpoints were then connected with the genetic algorithm to form an optimized UAV trajectory. We set the maximum number of iterations to 20,000 in order to derive a visually smoothed trajectory. Figure 12 illustrates the decrease in path length following the iterations. We can observe that after the rapid decrease, the total path length reduced much slower after about 3000 iterations. Nevertheless, the path pattern kept becoming more structural and simpler in our observation. The derived path was examined by the obstacle avoidance function to ensure a collision-free path. No obstacles were detected on the computed path. The final UAV trajectory for the six ROIs is shown in Figure 13.

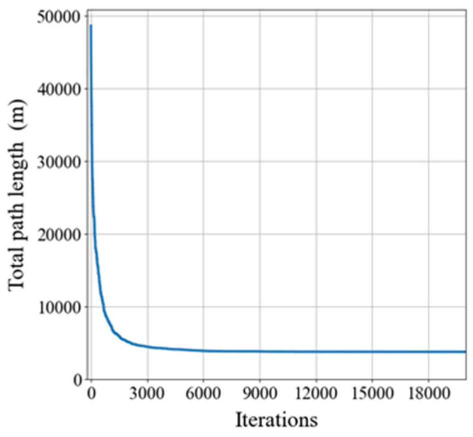

Figure 12. The change of total path length following the iterations using the genetic algorithm.

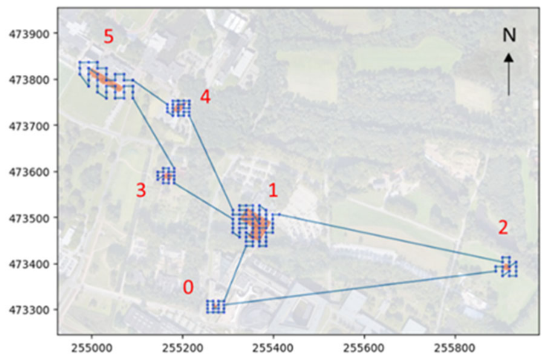

Figure 13. The final UAV path. The orange polygons are the defined ROIs. The Blue dots are the viewpoints.

Due to the EU regulation which does not allow flying the UAV beyond the visual line of sight (VLOS), the real data was collected individually for each ROI by inputting the viewpoint coordinates to the UAV control system. The images were processed with the commercial software Pix4D (Pix4D, 2022) following the default setting, which uses half image scale to generate the dense point clouds. We demonstrate the generated point cloud over one ROI in comparison to AHN3 in Figure 14. Compared to AHN3 data which has a point density of 10-14 pts/m^2, we achieved a point density of at least 300 pts/m^2 over the targeted regions generated from the UAV images without counting in the wall points. This high point density is beneficial for delineating the boundaries for more detailed structures. Our dense point clouds show clear boundaries of roof structures over almost all the scenes.

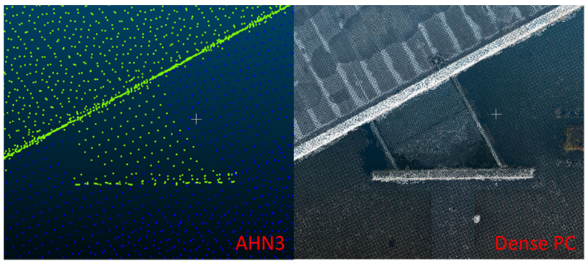

Figure 14. The original AHN3 LiDAR data and the dense point cloud derived from the collected UAV images with the proposed flight planning.

The dense point cloud was used to generate the partial 3D model. We correct the LoD2 model by replacing the erroneous part with the new partial model. Figure 15 demonstrates a sample result of the model improvement.

4. DISCUSSION

The proposed flight planning approach can generate an efficient UAV trajectory for image acquisition of building roof parts. In the experiment, the resulting point clouds were accurately reconstructed, with less self-occlusion occurring at the object's lower boundaries. While the results are promising, there is still room for improvement in certain areas.

First, the photogrammetric point cloud generated from UAV images and AHN3 LiDAR point clouds may both contain orientation and positional drift relative to real-world coordinates. Accordingly, one approach to ensure accurate 3D modeling is to perform local registration of the photogrammetric point cloud generated from UAV images to the AHN3 point cloud. This local registration can improve the alignment between the new model parts and the already accepted parts of the older model, ensuring accurate and consistent modeling throughout the entire area of interest. The use of automatic co-registration methods, such as the interactive closest point (ICP) algorithm, can help to achieve local registration by matching the geometric features of the point clouds. In the experiment, due to the privacy restriction and tree occlusions limiting the selection of ROIs, we manually defined the ROIs with the aid of automatic error detection. We purposely increase the buffer zone and drew larger polygons as the ROIs. This resulted in wider coverage of the scene, which can possibly include more geometric features that are beneficial for point cloud co-registration. However, it remains a question that how much buffer is required to include sufficient geometric features for automatic co-registration, since geometric features distribute differently among different targeted objects and also have different scales.

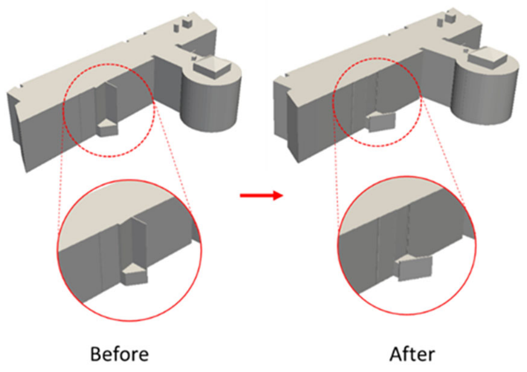

Figure 15. The improvement of the LoD2 model.

Second, the viewpoint heights in the experiment were manually defined by considering factors including the maximum allowed observation height given a targeted GSD, the ROI size, and camera parameters. This process should be further automated and the method for deciding the viewpoint heights given the ROI size needs to be further refined.

Finally, we achieved an obstacle avoidance function with ray-casting. Because we restricted the scanlines to three directions, the process of finding the alternative path with our ray-casting method is computationally efficient. However, the process of reconstructing the scene as a mesh with the existing LiDAR point cloud is slightly memory and computationally intensive. The success of the mesh reconstruction also decides whether the function can find a correct path to avoid obstacles.

5. CONCLUSION

In this paper, we proposed an innovative flight planning pipeline for LoD2 model error correction. Our algorithm detects modeling errors automatically by comparing the models to the original point clouds used for their reconstruction. We generate nadir-view camera networks by considering the geometry of the building roofs and the visibility test result. We optimize the UAV path with the genetic algorithm to connect all the viewpoints by solving the traveling salesman problem. An obstacle avoidance function was developed to ensure a collision-free path. We implemented our flight planning in a real-world scene. The

generated photogrammetric point cloud shows an accurate and clear boundary with minimized occlusions. We also demonstrated an improved LoD2 model corrected with the UAV data we collected.
Our proposed flight planning pipeline addresses the challenge of LoD2 model error correction, which is a critical step toward achieving accurate 3D modeling for various applications, such as urban planning, building inspections, and disaster management. Our pipeline offers an efficient solution for data recollection for model improvement, as it only collects the data over the erroneous region and uses it to improve the erroneous model part later. This approach reduces the time, cost, and labor involved in data acquisition and processing, making it a viable option for a wide range of applications.
One of the key features of our flight planning pipeline is the nadir-view camera network generation. This feature is crucial for efficient and accurate data collection. Our algorithm considers the geometry of the building roofs and the visibility test result when generating the camera networks, resulting in better coverage and less self-occlusion occurring at the object's lower boundaries. This feature is particularly useful for urban environments, where buildings are often close together, and the visibility of the targeted objects can be limited.
While our paper focuses on flight planning, we plan to explain and analyze the process of model improvement in future work. Model improvement is an essential step toward achieving accurate and efficient 3D modeling. Our approach can be extended to include facade image acquisition and LoD3 modeling, providing a comprehensive solution for 3D modeling in urban environments.

REFERENCES

3DBAG. (2021). 3D BAG, 3D Geoinformation, Delft University of Technology. https://www.3dbag.Nl.

AHN. (2022). Actueel Hoogtebestand Nederland. https://www.Ahn.Nl/.

Biljecki, F., Ledoux, H., Stoter, J., 2016. An improved LOD specification for 3D building models. Computers, Environment and Urban Systems 59, 25-37.

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R., 2016. Receding Horizon "Next-Best-View" Planner for 3D Exploration, 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1462-1468.

Dukai, B., Peters, R., Vitalis, S., van Liempt, J., Stoter, J., 2021. Quality Assessment of A Nationwide Data Set Containing Automatically Reconstructed 3D Building Models. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XLVI-4/W4-2021, 17-24.

Furukawa, Y., Hernández, C., 2015. Multi-View Stereo: A Tutorial. Foundations and Trends in Computer Graphics and Vision 9, 1-148.

Greiff, M., Robertsson, A., 2017. Optimisation-based motion planning with obstacles and priorities IFAC-PapersOnLine 50, 11670-11676.

Han, J., 2019. An efficient approach to 3D path planning. Information Sciences 478, 318-330.

Katz, S., Tal, A., Basri, R., 2007. Direct visibility of point sets, ACM SIGGRAPH 2007 papers. Association for Computing Machinery, San Diego, California, pp. 24–es.

Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). Poisson Surface Reconstruction. In A. Sheffer & K. Polthier (Eds.), Symposium on Geometry Processing. The Eurographics Association.

Koch, T., Körner, M., Fraundorfer, F., 2019. Automatic and Semantically-Aware 3D UAV Flight Planning for Image-Based 3D Reconstruction. Remote Sensing 11, 1550.

Kolbe, T.H., 2009. Representing and Exchanging 3D City Models with CityGML, in: Lee, J., Zlatanova, S. (Eds.), 3D Geo-Information Sciences. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 15-31.

Liu, H., Li, X., Fan, M., Wu, G., Pedrycz, W., Suganthan, P.N., 2022. An Autonomous Path Planning Method for Unmanned Aerial Vehicle Based on a Tangent Intersection and Target Guidance Strategy. IEEE Transactions on Intelligent Transportation Systems 23, 3061-3073.

Nagasawa, R., Mas, E., Moya, L., Koshimura, S., 2021. Model-based analysis of multi-UAV path planning for surveying postdisaster building damage. Scientific Reports 11, 18588.

Nagib, G., Gharieb, W., 2004. Path planning for a mobile robot using genetic algorithms, International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC '04., pp. 185-189.

Oude Elberink, S., Vosselman, G., 2011. Quality analysis on 3D building models reconstructed from airborne laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing 66, 157-165.

Pix4D. (2022). Pix4D. https://www.pix4d.com/

Vasquez-Gomez, J.I., Sucar, L.E., Murrieta-Cid, R., Lopez-Damian, E., 2014. Volumetric Next-best-view Planning for 3D Object Reconstruction with Positioning Error. International Journal of Advanced Robotic Systems 11, 159.

Wenzel, K., Rothermel, M., Fritsch, D., Haala, N., 2013. Image Acquisition and Model Selection For Multi-View Stereo. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XL-5/W1, 251-258.