Automatic Interpretation of Unordered Point Cloud Data for UAV Navigation in Construction

M.D. Phung[a], C.H. Quach[a] , D.T. Chu[a], N.Q. Nguyen[b], T.H. Dinh[c] and Q.P. Ha[c]

[a]University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
[b]National Institute of Information and Communications Strategy, Ministry of Information and Communications, Vietnam
[c]School of Electrical, Mechanical and Mechatronic Systems, University of Technology Sydney, Australia
E-mail: duongpm@vnu.edu.vn, Quang.Ha@uts.edu.au

Abstract –

The objective of this work is to develop a data processing system that can automatically generate waypoints for navigation of an unmanned aerial vehicle (UAV) to inspect surfaces of structures like buildings and bridges. The input includes data recorded by two 2D laser scanners, orthogonally mounted on the UAV, and an inertial measurement unit (IMU). To achieve the goal, algorithms are developed to process the data collected. They are separated into three major groups: (i) the data registration and filtering to generate a 3D model of the structure and control the density of point clouds for data completeness enhancement; (ii) the surface and obstacle detection to assist the UAV in monitoring tasks; and (iii) the waypoint generation to set the flight path. Experiments on different data sets show that the developed system is able to reconstruct a 3D point cloud of the structure, extract its surfaces and objects, and generate waypoints for the UAV to accomplish inspection tasks.

Keywords –

Point cloud data processing; UAV navigation; Infrastructure monitoring; Bridge inspection

1 Introduction

Unmanned aerial vehicles (UAVs) are expected to yield automatic solutions for inspecting and monitoring large and hardly accessible structures like bridges, towers, dams, culverts, wind turbines and heritage monuments due to their flexibility in operating space and ability to carry specialized sensory equipment. In [1], a micro air vehicle system was employed to scan buildings using a high resolution camera. Pictures taken were successfully stitched with sufficient quality for crack and damage detection. In [2], an advanced UAV system were developed to evaluate the state of historical monuments including a chimney built in early 1980th and a natural stone masonry tower inclined 4.7^0. Images captured after processing revealed damages in both monuments. A control system for navigation of the UAV from an initial to a final position in an unknown 3D environment was used to monitor and maintain bridges [3]. UAVs were also used to inspect and monitor oil-gas pipelines, roads, power generation grids and other essential infrastructure [4].

In order to complete those inspection missions, the UAV system typically carries out three steps including creating waypoints from coarse data of the environment, controlling the UAV to follow waypoints and collect data of the structure, and processing the data to detect defects or damages. Among these steps, the generation of waypoints is mainly manual with little support of automation. In [1], the UAV was directly controlled by a pilot. In [2], the flight plan was set by the user while work [3] focused on a control strategy without concerning obstacles. As the result, the deployment process is time consuming and the data collected are often redundant and ineffective. For example, among more than 12,000 images taken in [2] only several hundred images were finally valid for stitching and inspection.

In this study, we introduce an automatic solution to generate waypoints for the UAV to inspect and monitor built infrastructure. Input data includes range information acquired by two laser scanners and orientation measured by an IMU. The system then can align the range data into point clouds and register them to represent a 3D model of the structure. Based on the model, the system detects surfaces, extracts borders and clusters obstacle objects. For each surface to be inspected, the system will generate waypoints which are optimized for path planning and obstacle avoidance. The total processing time, from loading data to generating waypoint, is several minutes for the surface with area of hundreds of squared meters.

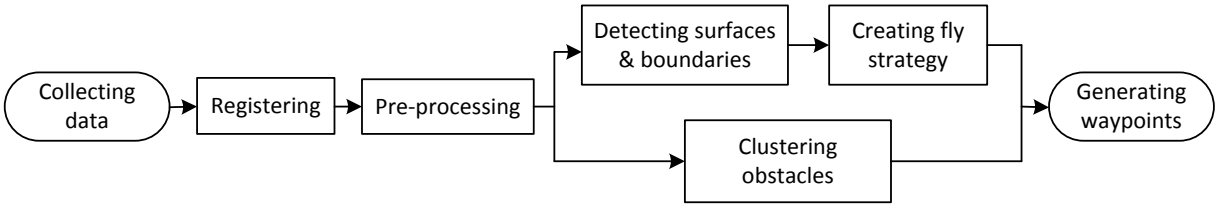


Figure 1. An overview of the system

2 Methodology

Figure 1 presents an overview of the system, consisting of seven stages. Details of each stage are explained as follows.

2.1 Data collection

Data used for the system includes range information measured by laser scanners. Due to the limitation in payload and energy capability, the 3D laser scanner is not suitable for the UAV. Instead, we use two 2D laser scanners and an IMU to collect information of the structure. The arrangement is shown in Figure 2. The two laser scanners are placed orthogonally to each other and symmetrically with respect to the center of the UAV. One scans horizontally and the other scans vertically. The IMU is fixed at the center of the UAV. The data collection is carried out by yawing the UAV 360^0. During yawing, points acquired by vertical scanning will be used to generate a 3D point cloud of the structure.

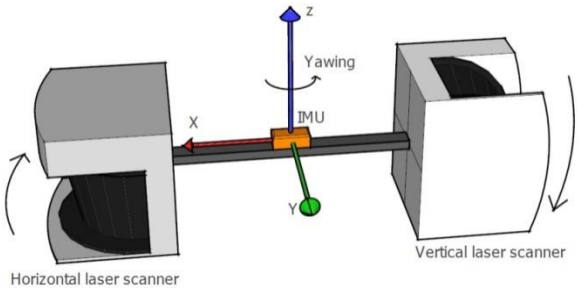


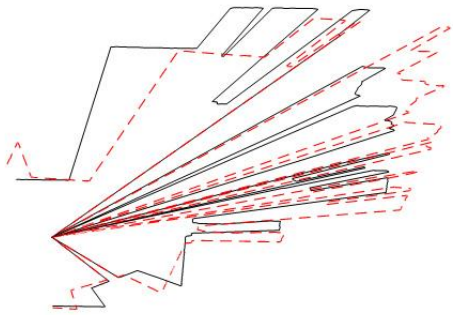
Figure 2. Hardware configuration for data collection

Let (ρ_i, α_i) be the distance and angle of point P_i measured by the vertical scanner. Its Cartesian coordinates in the local frame attached to the center of the scanner are given by:

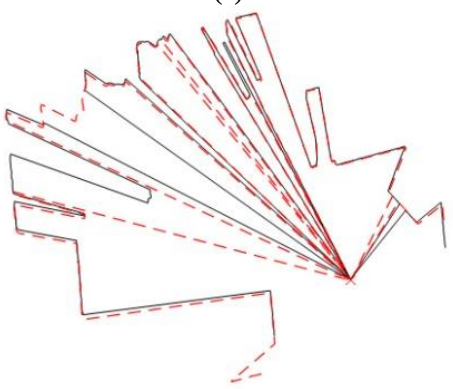
$$\begin{aligned} x_i^* &= -\rho_i \cos\alpha_i \\ y_i^* &= 0 \\ z_i^* &= -\rho_i \sin\alpha_i. \end{aligned} \tag{1}$$

As the scanner moves during acquiring data, it is required to map local coordinates to a fixed global frame. Let the origin of the global frame be the center of UAV at the initial scan, associated with $x-$, $y-$, and $z-$ coordinates as shown in Figure 2. The laser scanner motion is then the combination of two components: rotation caused by yawing and translation caused by the external forces like wind acting on the UAV. Let R and T be the rotation and translation matrices of the motion. The coordinate vector of point P_i in the global frame, $\mathbf{x}_i = [x_i \; y_i \; z_i]^T$, is given by:

$$\mathbf{x}_i = R_i \mathbf{x}_i^* + T_i. \tag{2}$$



(a)



(b)

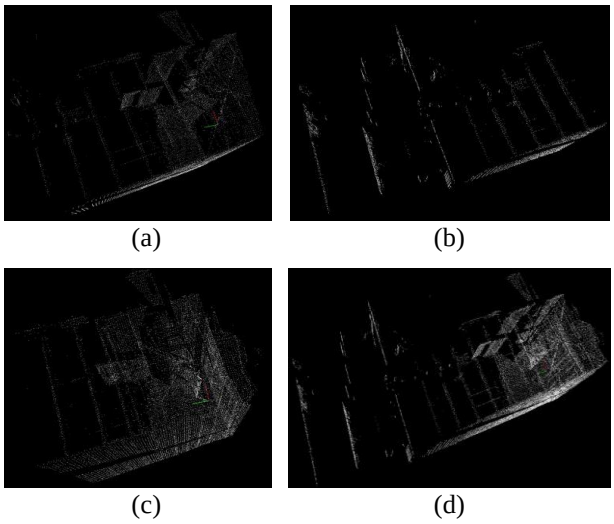
Figure 3. Horizontal scans: (a) before ICP, and (b) after ICP

In the system, matrix R is directly measured by the IMU while T is determined by the horizontal scanner as follows: During yawing, the horizontal scanner will scan the same surface of the structure with small changes in position and orientation. The scans therefore

have much in common (Figure 3(a)) so that we can use the iterative closest point (ICP) algorithm [5] to match them and extract the translation and rotation (Figure 3(b)). As the rotation is already known, we incorporate it by rotating the raw scans before applying the ICP. This customization can improve the accuracy of the ICP and reduce the computation load.

2.2 Point cloud registration

After collecting data, point clouds recorded at different positions need to be combined to a complete 3D model of the structure. This process is called registration. The key idea is to identify corresponding points between the point clouds and to find a transformation that minimizes the distance between them. Figures 4(a)-(c) show point clouds collected at three different positions in an office. They include both overlapping and distinct areas. We first predict the overlaps by merging point clouds from their recorded positions. The result is then refined by applying the ICP to the predicted overlaps (Figure 4(d)).



(a) (b) (c) (d)

Figure 4. Point cloud registration: (a) – (c) point clouds recorded at different positions, and (d) merged point cloud

2.3 Pre-processing: point cloud filtering and voxelization

Due to characteristics of the laser scanner, the point cloud generated in the registration process typically varies in densities and contains sparse outliers. This complicates the estimation of point cloud features such as surface normals or curvature changes, leading to erroneous values, which, in turn, might cause failure in further processing steps. The objective of the pre-processing process is hence to filter sparse outliers and equalize point densities.

The sparse outlier removal is carried out based on computation of the Gaussian distribution of distances from each point to its neighbors. By comparing the mean μ and standard deviation σ with pre-defined global thresholds, a point could be considered as an outlier and trimmed from the dataset if it falls outside the $\mu \pm d_t \sigma$, where d_t is a threshold coefficient [6]. For density equalization, the voxelization technique is employed. A uniformly spaced 3D voxel grid is created over the input point cloud. All points within a voxel are then presented by the centroid of the voxel. This process can largely reduce the amount of points, but may introduce a small amount of geometric error due to quantization.

Figure 5 demonstrates an example of filtering and applying voxelization on a data set. Here, the number of scan points in the original cloud was cut down from 90,396 points (left) to 80,114 points (right) while isolated points were considered as outliers and removed.

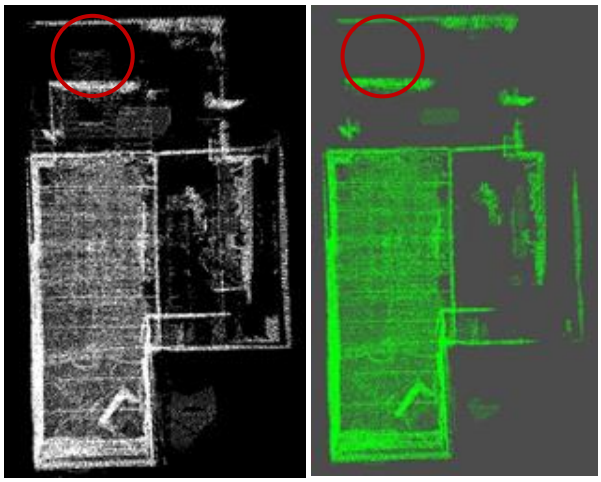


Figure 5. A point cloud before and after filtering and voxelizing

2.4 Surface and boundary detection

Surfaces are the main target to be inspected so they need to be extracted from the point cloud. We use the RANSAC (random sample consensus) method and planar model for this task due their simplicity and robustness [7]. Since a plane model ($ax+by+cz+d=0$) is given, $M=[a,b,c,d]^T$ is the parameter vector to be identified. The estimate procedure is a repeat of the following steps:

- First, a random subset of the original data is selected.
- A planar model is then fitted to the selected subset.

- Remaining points are then tested against the model. A point is considered as an inlier if its distance to the model is smaller than a pre-defined threshold. The fitting model is reasonable good if it has sufficient inliers.
- The fitting model is refined to better fit all found inliers.

In our system, a threshold value of 20 cm and a repeat of 200 times give compromised performance.

After detecting surfaces, the boundaries need to be determined. For each surface, corresponding inliers are first projected on the found planar model. The 2D convex hull algorithm [8] is then employed to fit a bounding polygon to the projected inliers.

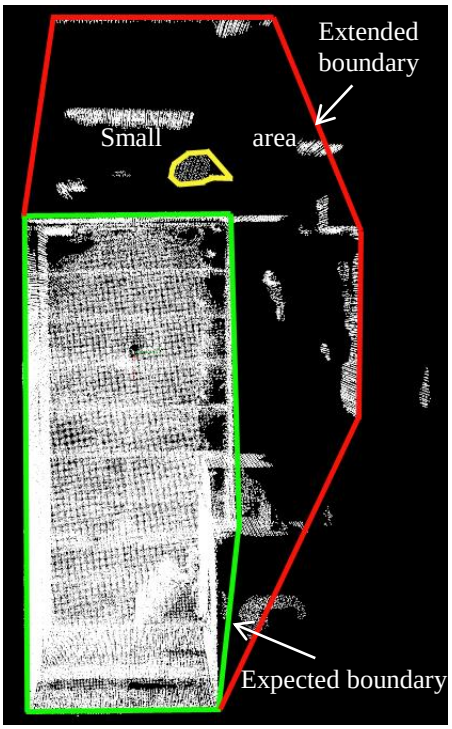


Figure 6. Problems related to surface and boundary detection

In practice, two problems may raise during using the RANSAC and convex hull algorithms as illustrated in Figure 6. First, the detected surface may contain isolated clusters causing the boundary to be extended. Second, when detecting small surfaces, the RANSAC may cause confusion between surfaces belonging to the structure and objects. In order to deal with the first problem, we require a high point density of a surface. We therefore trim out the isolated groups by clustering the surface and only keep the largest one. The clustering algorithm will be presented Section 2.5. For the second problem, the inlier quantity and density are used to compute the area of each detected surface, which is limited to a threshold set by the user. Figure 7 shows a flowchart of our surface and boundary detection.

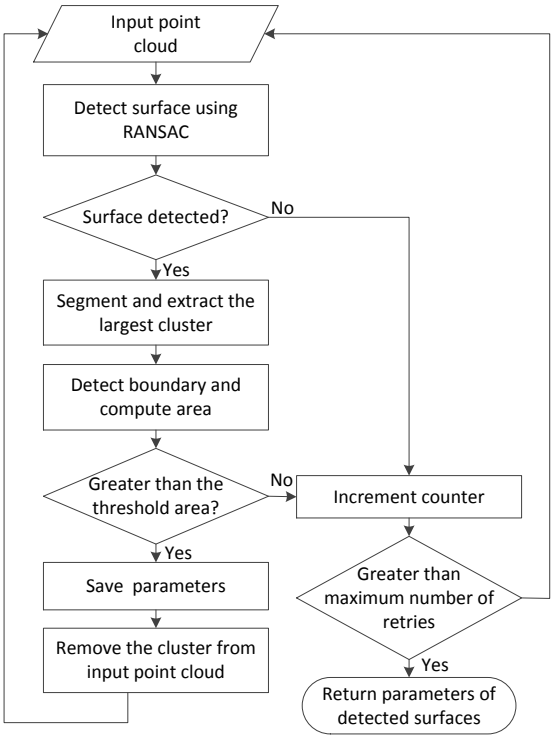


Figure 7. Flowchart of the surface and boundary detection algorithm

2.5 Obstacle clusterization

The scanned points that do not belong to any surfaces maintain their importance in UAV navigation. It is suggested here to cluster them into small groups as obstacles in order to support the path finding algorithm and the UAV's operator to have a better vision of the environment. The clusterization is carried out by finding the nearest neighbour that is essentially similar to a flood fill algorithm [9]. We use a 3D Kd-tree structure to implement the algorithm as follows:

- Create a 3D Kd-tree structure from the input point cloud dataset, P.
- Set up an empty list of clusters C, and a queue of the points that need to be checked Q.
- For every point $p_i \in P$, perform the following steps:
  - Add p_i to the current queue Q;
  - For every point $p_i \in Q$, do:
    - Search for the set P_i^k of point neighbors of p_i in a sphere with radius $r < d_{th}$;
    - For every neighbor $p_i^k \in P_i^k$, check if the point has already been processed, and if not, add it to Q;

- When the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to an empty list;

- The algorithm terminates when all points $p_i \in P$ have been processed and are now part of the point clusters list C.

We then use the octree structure to represent the clusters for visualization, as shown in Figure 8.

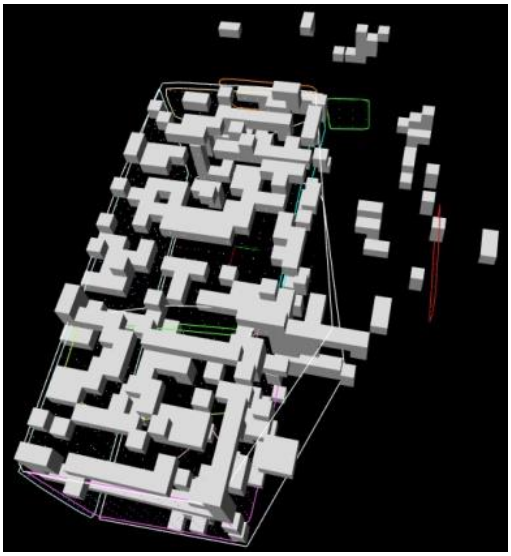


Figure 8. Visualization of obstacle objects

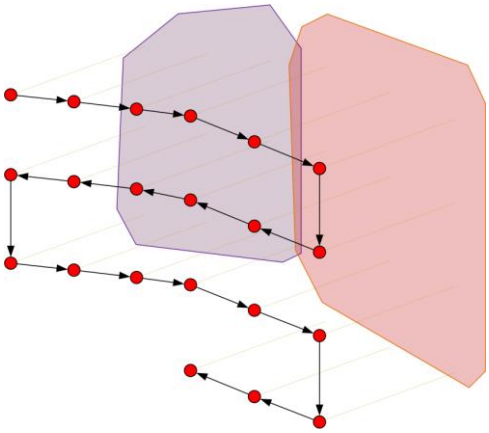


Figure 9. A flight strategy to take photos of surfaces for inspection

2.6 Flight strategy creation

A flight strategy is a set of positions at which the UAV has to pause to collect data. In order to create a flight strategy, it is required to know the assigned tasks, requirements on data collected, and capability of the sensory devices. For instance, a common situation would be taking photos of a structure and sticking them together for defect or damage detection. In this case, requirements may include the minimum resolution of photos, at which defects are still detectable and the overlap percentage between consecutive photos feasible for sticking and mosaicking. For this, the capability of a sensory device includes the resolution of the camera, its open angle and the number of degrees of freedom of the mechanism on which the camera is mounted. Given those requirements and parameters, we can compute the area that each photo captures, positions to take photo, their order, number of positions needed to cover the surface and thus the flight strategy, as illustrated in Figure 9.

2.7 Waypoint generation

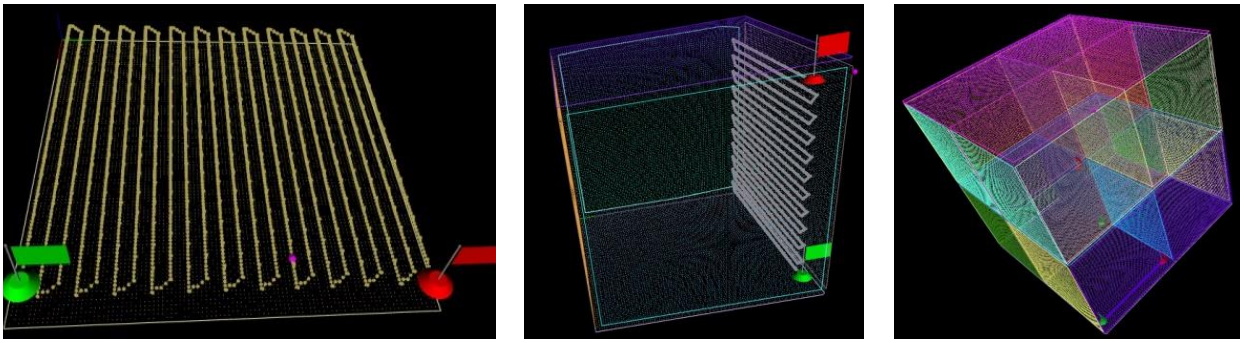
The final stage in our system is the generation of waypoints. They are intermediate points the UAV follows to complete a flight strategy. Given the flight strategy, we can determine waypoints by using a path finding algorithm. We implement it by dividing the work space into a grid of voxels. Each voxel has the free or occupied status corresponding to the existence or not of object in that voxel. In order to consider the UAV as a particle moving without collision between voxels, we mark all free voxels in a sphere with the radius equal to the largest dimension of the UAV as occupied. We then use the A-star algorithm [10] to find the shortest path between stop points. In each step, the cost to move from one voxel to another surrounding neighbor is computed as:

$$C(i,j,k) = \gamma_1 i^2 + \gamma_2 j^2 + \gamma_3 k^2, \qquad (3)$$

where indices $i,j,k \in \{-1,0,1\}$ indicate the position of neighbor, and the coefficients γ assign a particular weight to each direction.

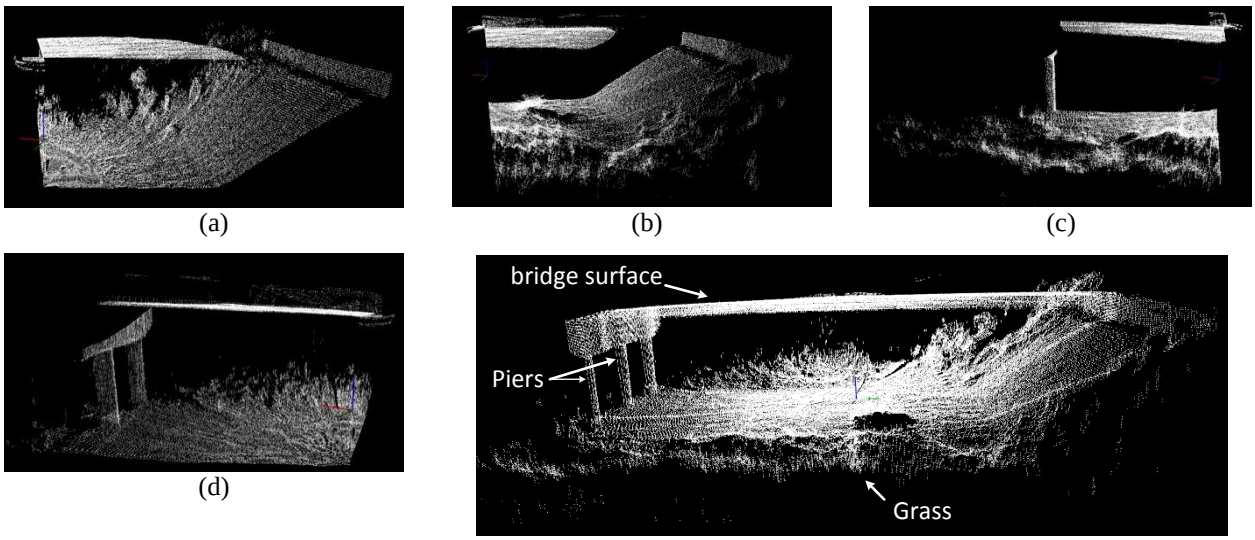
3 Experiments

The system was evaluated on various sets of data recording different structures like an office, a building or a bridge. The UAV used is a quadcopter with the size of 60 cm × 60 cm × 40 cm equipped with a camera mounted on a two degree-of-freedom gimbal, two laser scanners, an IMU and other electronic boards.



(a) (b) (c)

Figure 10. Experiments on hypothesis data sets: (a) a single surface; (b) a single cube; (c) a cube with crossed planes



(a) (b) (c) (d) (e)

Figure 11. Point cloud registration: (a) – (d) registered point clouds from raw data, and (e) merged point cloud

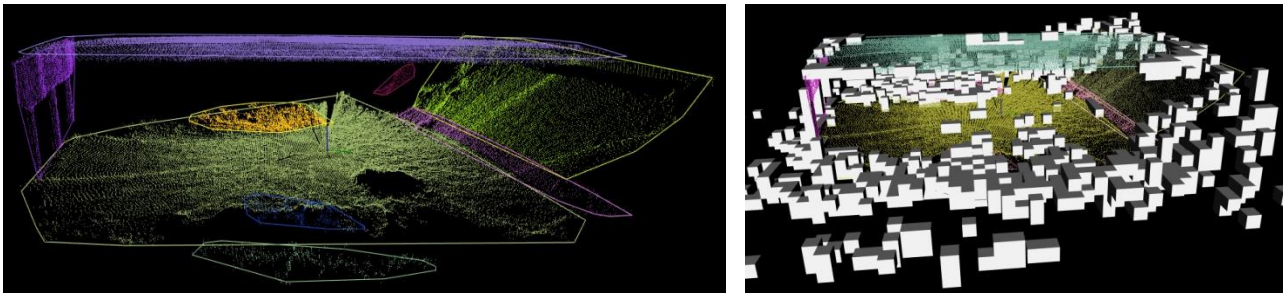
The laser scanner used is Hokuyo UTM-30LX with the scanning range of 30 m, detection angle of 270^0, angular resolution of 0.25^0 and scan period of 25 ms [11]. The IMU is Xsense Mti-10 outputting the rotation matrix with the frequency of 2 KHz, accuracy of 0.50, and no accumulated error [12]. The software is written in C++ and run on a PC with the processor Intel Core2 Dual 2.53 GHz and 4 GB RAM.

3.1 Experiments on hypothesis data sets

Experiments are first carried out on hypothesis data sets to evaluate the behavior of system in regular and irregular cases. They include the point clouds containing a single point, a single line, a single surface, a single cube and a cube with crossed planes. For the two first cases, no surface is detected. For the two later cases, all surfaces and their boundaries are detected. The flight strategy and waypoints for each surface are also created as shown in Figures 10(a)-(b). For the last case, all surfaces and boundaries are detected, but waypoints are not generated because the surfaces are blocked by cross planes (Figure 10(c)).

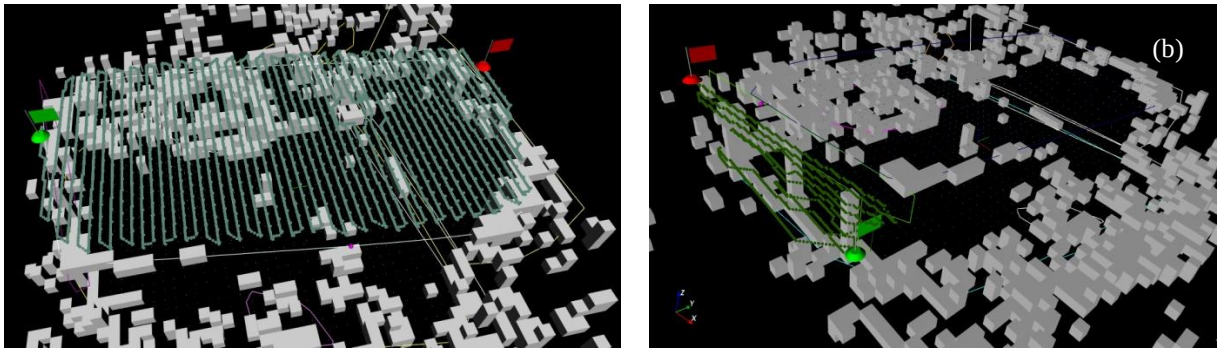
3.2 Experiments on real data sets

After evaluating the behavior of system on hypothesis data, experiments on real data sets are carried out. Figure 11 shows results of aligning raw scans into four point clouds (Figures 11(a)-(d)) and registering them to become a complete part of the bridge with the size of 22 m × 10 m × 4.5 m (Figure 11(e)). The alignment error, measured by yawing 360^0 and comparing the first and last scans, is 10 cm. Figure 12 represents the planes and objects detected with the area threshold of 2 m^2.



(a) (b)

Figure 12. Surface and obstacle detection: (a) planar surfaces; (b) obstacle objects



(a) (b)

Figure 13. Waypoints generated to navigate the UAV to take photos: (a) bridge surface, and (b) pier

Besides planes created by grasses and trees, all main planes including the bridge surface, pier, sides and land surface are detected. The boundaries are reliable for the convex surfaces. However, for the concave surfaces like pier, extra areas should be accounted for. We deal with this problem by adding a function to allow manually modification of the boundary. In practice, this function is useful for not only the concave but also convex surfaces as it allows the operator to ensure the accuracy of detected boundaries before generating waypoints for inspection.

Figure 13(a) shows the flight strategy and waypoints generated for the task of taking photos of the bridge surface with the coverage of 60 cm × 40 cm for each photo and the overlap of 20% between consecutive photos. A total of 1,146 stop points and 6,699 waypoints were generated to cover the area of 220 m^2. The path generated does not collide with any obstacles appearing in scans. The result is similar for the pier surface with 75 stopover points and 891 waypoint generated (Figure 13(b)). The processing time for a surface, from loading raw data to generating waypoints, averages 3 minutes in which 83% is consumed on aligning and registering point clouds. This result is very encouraging because manually defining waypoints for a surface often takes a lot of time or even becomes impractical for large structures like kilometer-long bridges. The waypoint generation will be further improved in a future work involving the deployment of a group of UAVs [13] in coordinated formation [14] to enhance the scanning coverage.

4 Conclusion

In this study, we have introduced a paradigm to collect and process data for automatic navigation of UAVs in built infrastructure inspection. A hardware configuration was proposed to acquire sufficient information to reconstruct the 3D model of the structure while feasible to equip on a micro UAV. The stages necessary to process point cloud data and extract semantic information from them are introduced. At each stage, we have presented details of algorithms used and customization to enhance the performance. Through experiments, it is shown that the proposed system provides a quick and reliable solution for the automatic operation of the UAV used in built infrastructure inspection.

Acknowledgement

The first author would like to acknowledge an Endeavours Research Fellowship by the Australian Government for the support (ERF_PDR_142403_2015), in part, for this work.

References

[1] Eschmann C., Kuo C.M., Kuo C.H., Boller C. Unmanned Aircraft Systems for Remote Building Inspection and Monitoring. In *Proceedings of* the *6th European Workshop on Structural Health Monitoring*, pages 1–8, Dresden, Germany, 2012.

[2] Hallermann N. and Morgenthal G. Un-manned aerial vehicles (UAV) for the assessment of existing structures. In *Proceedings of the 36th IABSE Symposium*, volume 101, number 14, pages 266–267, Kolkata, India, 2013.

[3] Metni N., Hamel T. A UAV for bridge inspection: Visual servoing control law with orientation limits. *Automation in Construction*, 17 (1):3–10, 2007.

[4] Rathinam S., Kim Z.W., Sengupta R. Vision-Based Monitoring of Locally Linear Structures Using an Unmanned Aerial Vehicle. *Journal of Infrastructure Systems*, 14(1):52–63, 2008.

[5] Besl P.J. and McKay N.D. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–254, 1992.

[6] Rusu R.B., Marton Z.C., Blodow N, Dolha M., Beetz M. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11): 927–941, 2008.

[7] Fischler M. and Bolles R. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6): 381–395, 1981.

[8] Barber C.B., Dobkin D.P., Huhdanpaa H. The quick hull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[9] Rusu R.B. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *Doctoral Dissertation*, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[10] Hart P.E., Nilsson N.J., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[11] Hokuyo Automatic Co., Scanning range finder UTM-30LX. Online: https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html, Accessed: 23/03/2016.

[12] Xsens, MTi 10-series. Online: https://www.xsens.com/products/mti-10-series/, Accessed: 20/03/2016.

[13] Woods A.C., La H.M. and Ha Q.P. A novel extended potential field controller for use on aerial robots, In *Proceedings of the 12th IEEE International Conference on Automation Science and Engineering*, Fort Worth Texas, USA, August 21–24, 2016.

[14] Nguyen A.D., Ha Q.P., Huang S., Trinh S. Observer-based decentralized approach to robotic formation control, In *Proceedings of the 2004 Australasian Conference on Robotics and Automation*, pages 1–8, Canberra, Australia, 2004.