ICP Localization in 3D Mesh Map Generated by Stitched Aerial Shots and Structure from Motion

Wei-Chin Chien
University of Michigan
Ann Arbor, MI
weichin@umich.edu

Ke-Haur Taur
University of Michigan
Ann Arbor, MI
khtaur@umich.edu

Wei-Fan Tseng
University of Michigan
Ann Arbor, MI
wftseng@umich.edu

Hao-Tsung Lee
University of Michigan
Ann Arbor, MI
haolee@umich.edu

***Abstract*—Our final project for EECS 504 is to attempt localization in reconstructed 3D scenes that are generated via Structure from Motion (SfM) on a series of images captured by aerial drones. Image stitching is also used to preserve localization information in a more compact format.**

***Index Terms*—Image Stitching, Depth Estimation, Structure from Motion, Scene Recovery**

I. Introduction

Localization has always been a difficult problem in the Robotics and Autonomous Driving domain. Various optimization goals such as time to reach the destination, fuel efficiency, and ride-sharing optimization increases the complexity of an already difficult-to-solve problem. We propose a method that could ease the process by incorporating ICP localization with SfM-generated point clouds.

The reason why we decide to present such scheme is because our methods complement an autonomous driving system that also comprises motion planning. Before performing localization on our source object, we would first generate a world map via OpenSfM (an open-source Structure From Motion package) that the object is currently in. Note that our current implementations, more details in the coming section, is built upon only aerial images which is not able to fully reconstruct street views but we will propose our ideas of resolving these issues as future work.

Our localization method is executed in two stages. The first being global positioning, which would provide an initial condition for our ICP algorithm. The subsequent stage, also known as position tracking, takes in the initial pose obtained from the previous stage as input, then localizes our source by accommodating the noise in motion.

II. Implementation

This section provides a glimpse of the working details of our implementation. Detailed results of the following steps would be showcased in section III.

A. Dataset Information

Our dataset (Lafayette Square in St. Louis) is acquired from this site. The drone translates across the city at a fixed altitude then corners around a hairpin bend after cruising off a couple of kilometers while taking aerial shots in set time intervals. Each image also contains the longitude and latitude information of the drone in raw format which comes in handy when setting an initial guess for the ICP algorithm.

B. Image Stitching

It may be easier for other applications that extends from our project to utilize the localized information, since computation on 2D images are considerably cheaper than 3D point clouds – not only do 3D point clouds require special software to view/edit, they also incur substantial computation overhead when performing any sort of optimization or post-processing on them.

The problem, though, is that many extracted features in aerial shots possess similar characteristics, making stitching difficult if only vanilla stitching methods are used. Other grueling implementation nuances such as stitching different sets of scenes or sections of the map require a different set of parameters (e.g., number of features to match) pop up during the process. The extent of error incurred in calculating the homography matrix accumulates after multiple stitches, resulting in an inclined, or in some cases, a totally deformed image though only a small amount of wrong matches are present (visualisation available in later sections). That being said, we still managed to stitch the images together after numerous attempts of trying our own tweaks and applying several off-the-shelf functions.

C. Localization Process

Before we perform localization on our source, we require (1) the map our source ought to match and (2) a representation of the current source location. We can generate a 3D Map in world coordinates by passing our entire bank of static target images to the OpenSfM package. Note that the term ”static” meant that these images should not obtained along with the streamed input/source – it should be slightly different in practice. We used the same set of images for our target map and source due to dataset limitations. A representation of the current source location, on the other hand, is a smaller but also 3D point cloud generated from at least two images that are captured by, in our case, drones or satellites. OpenSfM is not a single-shot depth estimator; hence, we would require at least two images to calculate depth information as depicted in SfM techniques. In our implementations, however, we found

that three images generate a better source representation in terms of overall structural integrity.

Few calibrations concerning scale, rotation and translation have to be made to the target map and source before localization can take place. Once we import our target map, we align the point cloud origin with the center of the Meshlab viewer trackball. Then we would finish calibrating the target map by rotating the map until the normal of the map is perpendicular to the X-Y plane. However, the extent of required calibration is negatively correlated to map sizes – smaller maps expect more manual adjustments. Input sources are then rotated until its surface normal is parallel to that of the target map. We then obtain the scaling magnitude of all incoming input sources by evaluating the difference in scale between an arbitrary input source and the target map reference. In our experimental results, the scale was around 0.85.

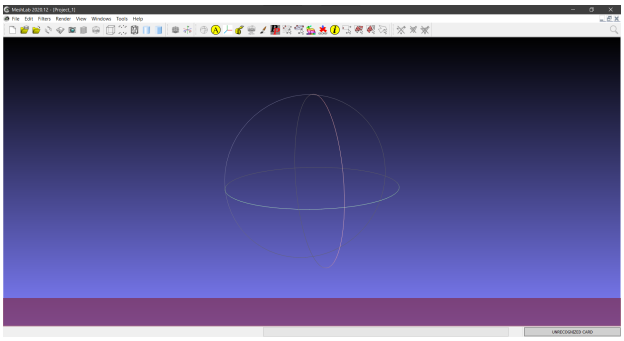


Fig. 1: Meshlab viewer trackball

The ICP algorithm is executed on our ROS setup mentioned previously. Our project pipeline is as follows: ROS would first load the 3D Map point cloud then we will provide an initial guess and the first input to ICP for global positioning. Our dataset contains exact drone longitude and latitude in EXIF format hence we could give a precise initial guess instead of relying on GPS estimates. After a given time step, the second input containing the source's new location arrives and the ICP algorithm would attempt local source tracking based on the previous iteration. Subsequent iterations follow the same reasoning which predicts the source locations on the fly.

D. Workspace Setup

Our project is developed under Ubuntu 18.04 and we also set up a Google Colab notebook for simple point cloud visualizations (see link). Our project dependencies are listed in the following:

- OpenSfM v0.5.1
- Meshlab
- ROS Melodic
- RViz
- Python 3.7
- Cython $\leq$ 0.25.2
- pcl 1.8.1
- OpenCV 4.2.0

III. Experimental Results

This section presents the results of our implementations.

A. Image Stitching

In this subsection, we present our stitching results and the difficulties encountered along the way. Our dataset captures the region highlighted in orange illustrated in figure 3.

Fig. 2: Ground truth of our to-be-stitched region

1) Impact of different set of parameters: In our initial attempt, we extracted ORB features using the FAST key point detector and BRIEF descriptor. However the stitching of different set of images require a different number of feature matches, which can only be done by manual adjustments in our implementation. In addition, though number of required matches may be optimal, some incorrect matches are impossible to remove due to aerial image features being too ambiguous in some cases (See figure 3).

Fig. 3: Feature matching with noticeable errors (prior to using RANSAC)

2) The addition of RANSAC: As an alternative we used the SIFT descriptor to extract features along with RANSAC in order to eliminate outlying features. A good amount of errors are resolved but some distortions are still present. We suspect that slight errors in the homography transform and the inconsistent cruising velocities of the drone (the extent of

overlap between images is not all that persistent) are the main reasons why the results were not as ideal.

Fig. 4: Stitched results with mild distortion

Our stitched result is shown in figure 4. Full stitch not shown due to page constraints.

B. Localization

1) Pre-processing and Target Map Generation: We feed the first hundred images of the dataset to OpenSfM which then generated our target map used in our implementation (See Figure 4). Before continuing on the localization process, we performed rotation and translation calibrations on our target. Scale calibration is then obtained by arbitrarily choosing 3 sequential images as a global source then evaluating its scaling difference between the source and target map. All adjustments can be performed in Meshlab.

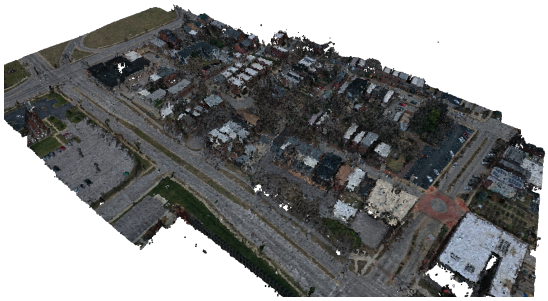


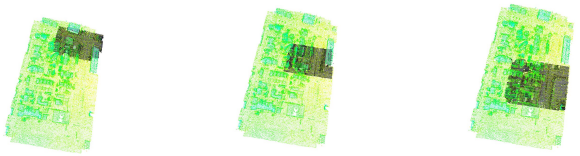
Fig. 5: Generated Map

2) ICP Algorithm and Localization: Upon loading the target point cloud into our ROS environment, we provide the initial guess and local point cloud to the target map. Before we apply the actual ICP algorithm, the map is down-sampled to a manageable size, which is around 70% smaller than that of the original. Down-sampling substantially reduces computational overhead and its effect on performance is merely negligible. The ICP algorithm outputs the result (highlighted portion of Figure 5) then iteratively performs localization for subsequent inputs until a certain number of iterations is reached. A score is evaluated for each iteration and once the score is smaller than a preset value, the algorithm converges.

The green rectangle in Figure 5 represents our target map while the small, black square denotes the localized source. The scores of partial iterations is shown in the Table 1. According to our results and the nature of ICP, the localization process is scalable in terms of correctness.

Iteration	Error/Score (m)	Iteration	Error/Score (m)
1	0.876531	7	0.877143
2	1.318115	8	0.730952
3	0.970332	9	1.34477
4	5.41566	10	0.902253
5	1.16632	11	1.21731
6	1.857	12	0.981078

TABLE I: Table of ICP iteration error/scores, the smaller the better (unit in meters)



(a) Initial Guess (b) Second time step (c) Third time step

Fig. 6: The first three time steps of our localization

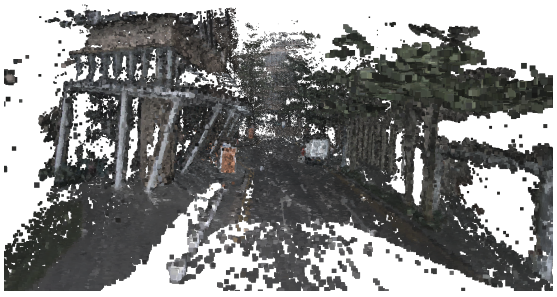
IV. Future Work

We believe that our current project can incorporate dash cameras to reconstruct scenes that were not easy to reproduce without the use of high-end equipment such as LiDAR. It is possible to extend the functionality of our project to reconstruct a complete story of the scene since dash camera images and aerial shots possess complementary information. A practical application or extension of our project would be conducting simulations in unbounded angles/positions in our reconstructed scene. We would then propose encountered difficulties that were currently unresolved.

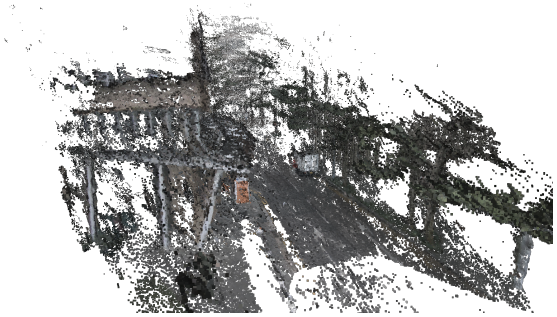
(a) First image (b) Second image (c) Third image

Fig. 7: The ground truth of our street view

We attempted to run OpenSfM on multiple sets of street-view testing images but results were not as promising. Some samples of the original 2D street-view images are shown in figure 7. Note that these street-view images were not taken at the same location as the aerial shots. Failures mostly arose due to non-ideal situations that also present complications to other computer vision problems. The first being non-static objects –mostly vehicles– are present in almost every frame, which breaks down the photogrammetric principles of SfM. Our current ideas of solving this problem is to employ deep learning image in-painting to reconstruct an image without the presence of moving objects. The second issue, as seen in Figures 8 (a) and (b), is that objects such as trees and bushes pose a great challenge to SfM techniques since not only do these different objects overlap in different frames, these objects



(a) Reconstructed Street View Angle 1



(b) Reconstructed Street View Angle 2

Fig. 8: Examples of SfM struggling with trees. (a) and (b) present different viewing angles

have a lot of detailed features that are very similar in color and structure.

We believe that by employing similar calibration methods mentioned in the previous section, we should be able to combine the street-view mesh with the map mesh we conducted our experiments on. The difference of scale between these two meshes is probably the only issue that requires more attention and it should be taken care of once we made the appropriate scale modifications in Meshlab.

ACKNOWLEDGMENT

We would like to thank Prof. Jason Corso for determining the scope of the project and giving technical suggestions. We would also like to show our gratitude to the GSIs, Parker Koch and Oscar De Lima for the assistance in choosing the topic of the project.

APPENDIX: BACKGROUND INFORMATION

This section gives a quick introduction to the relevant techniques we applied in our project.

A. Structure From Motion

Structure from Motion (SfM) is a technique which utilizes a series of 2D images to reconstruct the 3D structure of a scene/object. SfM can produce point cloud based 3D models similar to LiDAR. This technique can be used to create high resolution digital surface models and models of objects with general digital cameras. Together, these advances have now made it feasible for a wide range of users to be able to generate 3-D models without expensive equipment.

OpenSfM, an open-source package we utilized in our project, can automatically identify matching features such as corners or line segments in multiple images. These features are tracked from image to image and are used to produce estimates of the camera positions and orientations and the coordinates of the features. This produces a point cloud of x, y, z coordinates for features. It also consists of basic modules for SfM with a focus on building a robust and scalable reconstruction pipeline. It also integrates external sensor (e.g. GPS, accelerometer) measurements for geographical alignment and robustness.

B. ICP Algorithm

Iterative Closest Point (ICP) is an algorithm employed to minimize the difference between two point clouds. ICP is often used to reconstruct 3D surfaces from different scans or to localize robots and achieve optimal path planning, etc. In the Iterative Closest Point, one point cloud (vertex cloud), the reference, or **target**, is kept fixed, while the other one, the **source**, is transformed to best match the reference. The algorithm iteratively revises the transformation (combination of translation and rotation) needed to minimize an error metric, usually a distance from the source to the reference point cloud, such as the sum of squared differences between the coordinates of the matched pairs. ICP is one of the widely used algorithms in aligning three dimensional models given an initial guess of the rigid transformation required.

REFERENCES

[1] D. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision 60 , no. 2 (2004): 91–110.

[2] Y. Li, Y. Wang, W. Huang, and Z. Zhang, "Automatic image stitching using SIFT," International Conference on Audio, Language and Image Processing, July 2008.

[3] G. Shi, X. Xu, and Y. Dai, "SIFT feature point matching based on improved RANSAC algorithm," International Conference on Intelligent Human-Machine Systems and Cybernetics, August 2013.

[4] J. Gao, S. Kim and M. Brown, "Constructing image panoramas using dual-homography warping," CVPR, August 2011.

[5] T. Partovi, et. al, "Automatic 3-D Building Model Reconstruction from Very High Resolution Stereo Satellite Imagery," Remote Sens. 2019.