Incremental 3D Line Segments Extraction from Semi-dense SLAM

Shida He  Xuebin Qin  Zichen Zhang  Martin Jagersand
University of Alberta

Abstract—Despite much interest in Simultaneous Localization and Mapping (SLAM), there is a lack of efficient methods for representing and processing their large scale point clouds. In this paper, we propose to simplify the point clouds generated by the semi-dense SLAM using three-dimensional (3D) line segments. Specifically, we present a novel incremental approach for 3D line segments extraction. This approach reduces a 3D line segment fitting problem into two two-dimensional (2D) line segment fitting problems, which take advantage of both image edge segments and depth maps. We first detect edge segments, which are one-pixel-width pixel chains from keyframes. We then search 3D line segments of each keyframe along their detected edge pixel chains by minimizing the fitting error on both image plane and depth plane. By incrementally clustering the detected line segments, we show that the resulting 3D representation for the scene achieves a good balance between compactness and completeness. Our experimental results show that the 3D line segments generated by our method are highly accurate in terms of the location of their end points. Additionally, we also demonstrate that these line segments greatly improve the quality of 3D surface reconstruction compared to a feature point based baseline.

I. INTRODUCTION

Extracting geometric information from a stream of images is an important yet challenging topic in computer vision. The 3D knowledge of the scene is essential for a variety of applications, such as photogrammetry, robot navigation, augmented reality, etc. With the development of Structure from Motion (SFM) and Multi-View Stereo (MVS), one can easily reconstruct a 3D point cloud from a set of images or a video stream. Alternatively, a 3D point cloud of a scene can also be obtained in real-time using a monocular Simultaneous Localization and Mapping (SLAM) system.

However, there are several limitations in representing a scene using 3D point clouds. First, points in a point cloud are usually stored independently and present no structural relationships. Second, point clouds require large storage space and they are inefficient in terms of representing the geometric structure of a scene. For example, it only requires two 3D points to define a linear structure but there are usually hundreds or thousands of 3D points along the structure in a semi-dense or dense point cloud. These limitations severely reduce the efficiency of post-processing and analysis.

On the contrary, line segments efficiently preserve more structural information of a scene. Several line segment based 3D reconstruction algorithms have been proposed [1], [2]. These methods rely on efficient line segment detection and inter-keyframe matching, which are difficult for certain scenes. However, with the dense or semi-dense point clouds available, one can estimate 3D line segments without explicit matching and triangulation.

In this paper, we develop a novel method to incrementally extract 3D line segments from semi-dense SLAM. Those detected line segments present structural relations among points and prove to be a highly efficient form of 3D representation. We propose a novel edge aided 3D line segment fitting algorithm. We first detect edge segments from keyframes using Edge Drawing [3]. Then 3D line segments are incrementally fitted along edge segments by minimizing the fitting error on both image and depth related planes. Our method produces accurate 3D line segments with few outliers. We apply the 3D line segments extracted by our method to incremental surface reconstruction and improve the quality of reconstructed surface with respect to a feature point based baseline.

The rest of this paper is organized as follows. In Section 2, we review some preliminary and related works about 3D line segment reconstruction. In Section 3, we describe our proposed algorithm in details. We present our experimental results in Section 4 and conclude in Section 5.

II. RELATED WORKS

3D points are the most commonly used features for 3D representation in SFM [4], [5] and SLAM [6], [7], [8], [9], [10]. Compared with 3D points, 3D lines provide more structural information of the environment.

Two dimensional (2D) line segment detection has been studied for a long time. Many algorithms have been proposed, such as Hough transform [11], LSD [9], EDLines [12]. However, these methods are hard to extend to 3D space directly. 3D line segment detection methods can be categorized into three main classes.

1) Direct 3D line fitting from 3D points: Roberts proposed a new representation for a line [13]. Using this representation, Snow developed an algorithm for solving the Total Least-Squares problem of 3D line fitting[14]. This kind of methods are sensitive to noise and outliers. Random Sample Consensus (RANSAC) is relatively robust to small number of outliers [15]. However, in 3D space RANSAC is time consuming and the optimal line fitting is not guaranteed in the presence of a large amount of outliers.

2) 3D line reconstruction from 2D line detection and matching: Bartoli and Sturm proposed a line segment based SFM pipeline [16]. Hofer et al. matched 2D line segments detected from images of different view by pure geometric constraints and reconstructed the 3D line segments by solving a graph-clustering problem [1]. Micusik and Wilder used relaxed endpoint constraints for line matching and developed a SLAM-like line segment based SFM system [17]. Liu et
al. reconstructed 3D wire objects using an ordered set of 3D curve segments [18].

3) 2D cues aided 3D line extraction: Woo et al. [19] detected 2D line segments from 2D aerial images first, and then used their corresponding 3D points on buildings’ Digital Elevation Model (DEM) to fit 3D lines. Given RGB-D sensor, Nakayama et al. [20] transformed 2D points on detected 2D line segments directly to 3D using corresponding depth image. Then, 3D line segments are fitted by RANSAC in 3D space.

Since most of the existing SFM or SLAM systems output point clouds as their mapping results, we prefer to make full use of these results and extract 3D line segments using both image and point cloud information. Hence, our method belongs to the third class. Instead of detecting 2D line segments and finding their corresponding 3D points for fitting 3D line segments afterwards, we fit the line segment in 3D by iteratively fitting its projections in two different planes. In other words, we directly detect 3D line segments by taking both 2D locations and corresponding depth of detected edge pixels into consideration in the line segment detection procedure.

III. METHOD

In order to extract 3D line segments from semi-dense point cloud, our method performs the following steps on each new keyframe (shown in Fig. 1): (1) Compute semi-dense depth map; (2) Edge aided 3D line segment fitting; (3) 3D line segment clustering and filtering.

A. Keyframes and depth maps generation

Our method is based on ORB-SLAM [7] with semi-dense module [21]. ORB-SLAM is a feature based SLAM system which takes the image sequence from a moving camera and computes the camera poses in real time. Mur-Artal and Tards [21] present a semi-dense module which is able to compute a semi-dense depth map for each keyframe. In principle, other keyframe based dense or semi-dense SLAM system could be used to generate the semi-dense depth maps, such as LSD-SLAM [9].

B. Edge aided 3D line segment fitting

Direct 3D line fitting from point clouds are difficult and time consuming. In this paper, we propose to use 2D image information on keyframes to help 3D line segment fitting from semi-dense point clouds.

We first extract edge segments from keyframes using Edge Drawing (ED) [3]. ED is a linear time edge detector which can produce a set of accurate, contiguous and clean edge segments represented by one-pixel-width pixel chains. Then we project the semi-dense depth map to the corresponding keyframe. Now the detected edge segments of the all keyframes can be expressed as $ES = \{ ES_1, ES_2, ..., ES_n \}$ where $ES_k$ denotes the edge segment set of the $k$-th keyframe. $ES_k = \{ es_1, es_2, ..., es_m \}$ where $es_i = \{ p_1, p_2, ..., p_t \}$ is an edge segment formulated as a pixel chain. $p_i$ represents a pixel which is a vector of $(x, y, Z)$ where $x$ and $y$ are the image coordinates and $Z$ is its corresponding depth. The number of keyframes, number of edge segments in a keyframe and number of pixels in an edge segment are denoted by $n$, $m$ and $t$ respectively. It is worth noting that image pixels with high gradient are more likely to be selected for computing depth value in the semi-dense SLAM system. Edge segments are detected based on pixel intensity gradients. Thus, most of the detected edge pixels will have depth values projected from the depth map. The pixels which have no depth values will be considered as outliers in the line fitting process.
Algorithm 1 Edge aided 3D line segment fitting

Input: A set of Edge Segments on the k-th keyframe: \( ES_k = \{es_1, es_2, ..., es_m\} \), \( es_i = \{p_{ij}^1, p_{ij}^2, ..., p_{ij}^n\} \) denotes the j-th pixel of edge segment \( es_i \)

Output: Fitted 3D line segments \( LS_k \)

1: for \( i = 0; i < n; i++ \) do
2:     fit_check = 0, new_line = true, line = null
3:     for \( j = 0; j < m; j++ \) do
4:         if new_line then
5:             for \( t = 0; t < L; t++ \) do
6:                 line.push_back(\( p_{ij}^t \))
7:             end for
8:         end if
9:         if fit_check > L then
10:            if line.size > L then
11:                line.push_back(line)
12:            end if
13:            line = null, new_line = true
14:            fit_check = 0
15:        end if
16:     end for
17: end if
18: end for
19: end for
20: return \( LS_k \)

3D line segments of a keyframe are extracted from those detected image edge segments by Algorithm 1. The main idea of this algorithm is to reduce a 3D line fitting problem to two 2D line fitting problems. For each edge segment, the algorithm initially takes its first \( L \) pixels to fit two 2D lines (\( l_{\text{lm}} \) and \( l_{\text{depth}} \)) in image coordinate frame and the \( 1_p-xz \) coordinate frame using total least square method. The coordinate frames are defined in Fig. 2. The line \( l_{\text{lm}} \) is fitted based on the pixels’ \( (x, y) \) values while \( l_{\text{depth}} \) is fitted based on \( (D, Z) \). \( Z \) is the pixel’s depth and \( D \) is the distance from \( p_1 \) to the pixel’s projection on the x-axis. Total least square 2D line fitting is performed by solving Singular Value Decomposition (SVD) [22]. Given the next pixel in the pixel chain, we compute its distances to \( l_{\text{lm}} \) and \( l_{\text{depth}} \) in their corresponding coordinate frames. It is worth noting that \( D \) and \( Z \) have different units. To have the same unit as \( Z \), we denote as \( ls = \{ls_1, ls_2, ..., ls_w\} \). Here \( w \) denotes the total number of 3D line segments on all keyframes. Directly registering all of 3D line segments of each keyframe will produce redundant and slightly misaligned 3D line segments. We address this problem by proposing a simple incremental merging method.

The main idea of our merging method is clustering closely located 3D line segments and fitting those cluster sets with new 3D line segments. As illustrated in Fig. 3, the angle and distance measures are used for clustering. The angle measure \( \alpha \) is defined as:

\[
\alpha = \cos(\theta) = \frac{p_{ij}^1 \cdot p_{ij}^2 \cdot p_{ij}^3}{\|p_{ij}^1\|\|p_{ij}^2\|\|p_{ij}^3\|}
\]

The distance measure \( d \) is computed as:

\[
d = \min(d_1, d_2)
\]

\[
d_1 = \|p_{ij}^1\| + \|p_{ij}^2\| - \|p_{ij}^3\|
\]

\[
d_2 = \|p_{ij}^1\|^2 + \|p_{ij}^2\|^2 - \|p_{ij}^3\|^2
\]

Specifically, we take the first 3D line segment \( ls_1 \) as the initial cluster \( C_1 \). Then, we compute the angle and distance measure between the initial cluster (single line segment) and the next 3D line segment \( ls_2 \). If the angle \( \alpha \) and distance \( d \) are smaller than certain thresholds (\( \lambda_\alpha \) and \( \lambda_d \) respectively), we add \( ls_2 \) to the cluster \( C_1 \). Otherwise, we create a new cluster \( C_2 \). For each cluster, if it contains more than one
3D line segments, we will fit a new 3D line segment to represent the cluster. The direction of the new line segment is determined by performing SVD on the matrix consisting of points in $P_{ep}$, where $P_{ep}$ denotes the set containing all the end points of line segments in this cluster. A new 3D infinite line is then determined by the direction together with the centroid of $P_{ep}$. Our objective is to obtain a 3D line segment from this infinite line. We project end points $P_{ep}$ onto the newly generated infinite 3D line and compute the furthest projections with respect to the centroid in both directions. The 3D line segment between these two furthest projection points is taken as the fitted line segment of the cluster. This process is repeated until all the line segments in $I$ are clustered. Clusters with small size (num($C_i$) < $\lambda_C$) are filtered out in the end. In this way, we can merge a large number of line segments into fewer clusters and generate new 3D line segments with higher quality.

IV. EXPERIMENTS

In this section, we present the results of our 3D line segment extraction method on image sequences from the TUM RGB-D dataset [23] and the EuRoC MAV dataset [24].

A. Implementation

The experiments in this section are performed on a desktop computer with a quad-core Intel i7-6700k CPU. We use the open source ORB-SLAM2 [25] as our base system. We implement the semi-dense module in C++ as described in [21]. The parameters in ORB-SLAM are kept as default, and the parameters for semi-dense module are set as presented in [21]. Parameters in Algorithm 1 and incremental line segment clustering are fixed as follows in all our experiments: $L = 10$, $e_1 = 1.0$, $e_2 = 1.5$, $\lambda_d = 10$, $\lambda_d = 0.02$, $\lambda_C = 3$.

B. Qualitative Comparison

The results of our 3D line segment extraction method on the test sequences are illustrated in the last two rows of Fig. 4. Our results accurately fit the semi-dense point clouds shown in the second row of Fig. 4. They still capture the major structures of the scene while reducing the number of 3D elements greatly.

We first compare our results with those from Line3D++ [1]. The results of Line3D++ on the test sequences is shown in the third row of Fig. 4. In our experiments, Line3D++ uses line segments detected by EDLines together with the keyframe images and camera poses output by ORB-SLAM to construct 3D line segments. However, it is fragile in some cases, such as complex indoor environment or scenes without long, straight line segments. Therefore, Line3D++ tends to produce a large number of outliers due to the ambiguity of geometric line matching in such cases. On the contrary, our method utilizes the accurate semi-dense depth map. Since the depth maps are checked multiple times and filtered to produce confident points, the results of our method have fewer outliers.

In contrast to Line3D++, semi-dense points can cover regions with large image gradient, such as boundaries and contours, where straight lines may be absent. Since our method takes both intensity and depth information into consideration, it is robust to outliers caused by intensity noise so that it can extract shorter line segments than EDLines. Thus, our results fit curves better and captures finer details than Line3D++.

To further demonstrate the capability of our method, we compare it to a decoupled 3D line segment fitting method using 2D line segment given by EDLines [12]. Given detected line segments and the depth information on some of the pixels along the line segments, we can easily estimate the 3D line segment position by performing a single 2D line fitting. In this case, there are a fixed number of pixels on the line segment since we do not need to iteratively search along pixel chains and extend line segments. Therefore we can efficiently perform RANSAC in 2D to remove outliers before the line fitting process. With the fitted line, we compute the 3D location of the endpoints and reconstruct the 3D line segment. Note the result of this method is equivalent to directly performing a RANSAC in 3D to fit all 3D points on the line segment. However, fitting a line in 2D is faster because fewer parameters are required to represent the line and the search space is much smaller.

The results of decoupled line segment fitting are presented in the forth row in Fig. 4. Compared to the edge aided 3D line fitting which tries to utilize pixel position and depth simultaneously, the decoupled fitting essentially fits lines in the image plane and depth plane in two steps. The error from line fitting in the image plane will be propagated to the error of 3D line segment position, which result in an inaccurate reconstruction compared to our method. It is worth mentioning that the decoupled fitting tends to generate longer segments since only the pixel position is considered in the image plane line fitting process. Longer segments will make the error propagation even worse because the total error of line segments in image space might be larger. Another source of error is that EDLines may detects a long line segment which is not a continuous line in 3D space. Trying to fit a single 3D line segment onto the 2D segment in this case will result in a large error. On the other hand, in our method, if either of the two errors of line fitting grows higher than the threshold, we stop the line fitting and start a new line fitting process. In this way, the errors accumulated from image plane and depth are bounded, and therefore prevent the line segments from being far away from the 3D points.

| Method                     | Vicon Room 101 | Vicon Room 201 |
|----------------------------|----------------|----------------|
| Line3D++                   | 84.10 mm       | 78.63 mm       |
| Decoupled 3D fitting       | 21.48 mm       | 23.45 mm       |
| Edge aided w/o clustering  | 13.91 mm       | 17.41 mm       |
| Edge aided w/ clustering   | 13.93 mm       | 17.63 mm       |
Fig. 4. Experimental results. Top to bottom: sample image from the sequence, semi-dense point cloud, results of Line3d++, results of decoupled 3D fitting using EDLines, results of our edge-aided 3D fitting without clustering, results of our edge-aided 3D fitting with clustering. Left to right (four sequences): EuRoC MAV Vicon Room 101, EuRoC MAV Machine Hall 01, TUM RGBD fr3-large-cabinet, TUM RGBD fr1-room.
C. Quantitative Comparison

1) Distance to surface: To demonstrate the accuracy of our method, we compute the average distance of line segment endpoints to the ground truth surface in two EuRoC MAV sequences, as shown in Table I. We take the provided precise 3D scanning of environment as ground truth. Since the output of ORB-SLAM have coordinates different from the ground truth surface data, we estimate the global Euclidean transform and scale change by performing ICP to align the semi-dense point cloud to the ground truth point cloud. The same Euclidean transform and scale change are applied to all the output line segment data before calculating distance, so that all the distances calculated are in the coordinates of the ground truth data. It can be seen in Table I that the result of our method fit to the surface better than other methods.

2) Compactness: For easier handling and manipulation, it is desired to have fewer 3D elements while they can still represent most of the environment. In the surface reconstruction pipeline, a smaller number of vertices will also greatly reduce the running time. As shown in Table II, the point clouds are greatly simplified with our edge aided 3D line fitting algorithm. The results are simplified further to present a clean structure of the scene using our 3D line segments clustering process. Note that although Line3D++ produces the fewest number of vertices in the reconstruction, the completeness of reconstruction is generally worse than our method as shown in Fig. 4.

3) Running Time: Table III presents the average running time of 3D line segments fitting on the sequences shown in Fig. 4. Our line segment fitting method is run-time efficient while utilizing large amount of depth information. Compared to the running time of edge aided 3D fitting, decoupled 3D fitting requires additional computation time for performing RANSAC. Because the segments are usually much longer in decoupled 3D line segments fitting, RANSAC is necessary in order to obtain a good fit for the larger pixel set on the line segments. Although our 3D line segment fitting algorithm is fast enough to be real-time, our clustering process is relatively slower. The complexity of clustering a single line segment is $O(N)$, where $N$ is the number of existing clusters. Generally, in a sequence with 200 keyframes, clustering can take about 0.5s per keyframe.

D. Surface Reconstruction

The resulting line segments of our method can be used to improve the quality of surface reconstruction. We integrate our method to the incremental space carving surface reconstruction algorithm presented in [26]. The algorithm incrementally reconstructs the surface by marking discretized 3D space as free or occupied using the visibility information of interest points. We compare the reconstructed surface using the line segment end points from our proposed method versus using the map points of ORB-SLAM. The result running on EuRoC Vicon Room 101 sequence is shown in Fig. 5. Since more points are available, our method yields much smoother surfaces. Also thanks to the structural information and fewer outliers provided by our method, major structures in the room are much more obvious.

V. CONCLUSIONS

In this paper, we present an incremental 3D line segment based method that uses underlying structural information to simplify the semi-dense point cloud output by keyframe-base SLAM system. The main contribution lies in the novel edge aided 3D line segment extraction algorithm which solely relies on the image and the semi-dense depth map of individual keyframes. Our method is fully incremental. It tries to minimize the line fitting error on both image plane and depth plane simultaneously as the line segment grows. By incrementally clustering the line segments detected on each keyframe, we can obtain a compact and complete 3D line segment reconstruction for the scene. Compared to using the line segments produced by 2D image detectors and minimizing the line fitting error on the depth plane afterwards, our method achieves better accuracy in terms of the location of the reconstructed vertices. We show that the result of our method can be used in incremental surface reconstruction to improve the quality of 3D surfaces.

The performance of our method relies on that of the SLAM algorithm. The current implementation is not real-time due to the clustering process and the delay of ORB-SLAM in its semi-dense depth map output. We plan to improve the speed in the future.
Fig. 5. Reconstructed surface of sequence EuRoC MAV Vicon Room 101 in different views.

REFERENCES

[1] M. Hofer, M. Maurer, and H. Bischof, “Efficient 3d scene abstraction using line segments,” Computer Vision and Image Understanding, vol. 157, pp. 167–178, 2017.

[2] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfelici, and F. Moreno-Noguer, “PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines,” in International Conference in Robotics and Automation, 2017.

[3] C. Topal and C. Akinlar, “Edge drawing: A combined real-time edge and segment detector,” J. Visual Communication and Image Representation, vol. 23, no. 6, pp. 862–872, 2012.

[4] C. Wu et al., “Visualsfm: A visual structure from motion system,” URL http://www.cs.washington.edu/homes/ccwu/vsfm, vol. 14, 2011.

[5] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, “Openmvg: Open multiple view geometry,” in International Workshop on Reproducible Research in Pattern Recognition. Springer, 2016, pp. 60–74.

[6] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007, pp. 225–234.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: a versatile and accurate monocular slam system,” IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147–1163, 2015.

[8] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “Dtm: Dense tracking and mapping in real-time,” in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011, pp. 2320–2327.

[9] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in European Conference on Computer Vision. Springer, 2014, pp. 834–849.

[10] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[11] R. O. Duda and P. E. Hart, “Use of the hough transformation to detect lines and curves in pictures,” Commun. ACM, vol. 15, no. 1, pp. 11–15, 1972.

[12] C. Akinlar and C. Topal, “Edlines: A real-time line segment detector with a false detection control,” Pattern Recognition Letters, vol. 32, no. 13, pp. 1633–1642, 2011.

[13] K. S. Roberts, “A new representation for a line,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 1988, 5-9 June, 1988, Ann Arbor, Michigan, USA., 1988, pp. 635–640.

[14] K. Snow and B. Schaffrin, “Line fitting in euclidean 3d space,” Studia Geophysica et Geodaetica, vol. 60, no. 2, pp. 210–227, 2016.

[15] M. A. Fischer and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” Commun. ACM, vol. 24, no. 6, pp. 381–395, 1981.

[16] A. Bartoli and P. F. Sturm, “Structure-from-motion using lines: Representation, triangulation, and bundle adjustment,” Computer Vision and Image Understanding, vol. 100, no. 3, pp. 416–441, 2005.

[17] B. Micusik and H. Wildenauer, “Structure from motion with line segments under relaxed endpoint constraints,” International Journal of Computer Vision, vol. 124, no. 1, pp. 65–79, 2017.

[18] L. Liu, D. Ceylan, C. Lin, W. Wang, and N. J. Mitra, “Image-based reconstruction of wire art,” ACM Trans. Graph., vol. 36, no. 4, pp. 63:1–63:11, 2017.

[19] D. Woo, S. S. Han, Y. Jung, and K. Lee, “Generation of 3d building model using 3d line detection scheme based on line fitting of elevation data,” in Advances in Multimedia Information Processing - PCM 2005, 6th Pacific-Rim Conference on Multimedia, Jeju Island, Korea, November 13-16, 2005. Proceedings, Part I, 2005, pp. 559–569.

[20] Y. Nakayama, H. Saito, M. Shimizu, and N. Yamauchi, “3d line segment based model generation by RGB-D camera for camera pose estimation,” in Computer Vision - ACCV 2014 Workshops - Singapore, Singapore, November 1-2, 2014, Revised Selected Papers, Part III, 2014, pp. 459–472.

[21] R. Mur-Artal and J. D. Tardós, “Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM,” in Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015, 2015.

[22] G. H. Golub and C. F. Van Loan, “An analysis of the total least squares problem,” SIAM Journal on Numerical Analysis, vol. 17, no. 6, pp. 883–893, 1980.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in Proc. of the
International Conference on Intelligent Robot Systems (IROS), Oct. 2012.

[24] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The euroc micro aerial vehicle datasets,” The International Journal of Robotics Research, 2016.

[25] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” IEEE Transactions on Robotics, 2017.

[26] D. I. Lovi, “Incremental free-space carving for real-time 3d reconstruction,” in Fifth International Symposium on 3D Data Processing Visualization and Transmission(3DPVT), 2010.