Evaluation of Self-localization Estimation in Indoor Environments Using Cameras for Small Drones

Genki Higashiuchi[1,*], Tomoyasu Shimada[1,†], Hiroki Nishikawa[2,†], Xiangbo Kong[3,†] and Hiroyuki Tomiyama[1,†]

[1]*Graduate School of Science and Engineering, Ritsumeikan University*
[2]*Graduate School of Information Science and Technology, Osaka University*
[3]*Department of Intelligent Robotics, Faculty of Information Engineering, Toyama Prefectural University*

Abstract
Recently, the demand of drones has increased dramatically. Drones are employed in variety fields, e.g., border security, search and rescue, surveying and recreational activities. Drones have become indispensable tools in each of these fields. Self-localization is an essential function that enables UAVs to fly autonomously. While Global Positioning System (GPS) is effective outdoors, GPS is dysfunctional in the indoor environment because of signal blocking or irregular reflection. Additionally, small drones valued for their agility in narrow spaces. However, they are often limited by weight and cost constraints, restricting their sensor capabilities. In this paper, we investigates self-localization for small drones using monocular camera images. Specifically, we will employ Visual Odometry [1] to estimate the drone's position. By analyzing the impact of varying the number of image frames used per estimation, we aim to evaluate potential improvements in self-localization accuracy quantitatively. For the three pre-defined paths, the average error when using 25 images per estimation was found to be 89.17 cm.

Keywords
Drone, Tello, Self-localization

1. Introduction

There has been a surge in recent research activity on drones. Due to their increased affordability and convenience, drones are expected to find applications in a lot of fields, e.g., border security, search and rescue, surveying, and recreational activities [2]. Self-localization is an essential method for enabling autonomous drones flight. GPS is a standard and high-precision self-localization method in outdoor environments. However, GPS often lacks function in indoor environments because of signal blocking and irregular reflection. Therefore, GPS is insufficient for small drones, which is sufficient in indoor environments. However, there are limitations to installing sensors on small drones. In order to tackle this issue, we employ a monocular

The 6th International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT 2024), August 19-22, 2024, Kusatsu, Shiga, Japan
*Corresponding author.
†These authors contributed equally.
ri0117vi@ed.ritsumei.ac.jp (G. Higashiuchi); ri0080xs@ed.ritsumei.ac.jp (T. Shimada); nishikawa.hiroki@ist.osaka-u.ac.jp (H. Nishikawa); kong@pu-toyama.ac.jp (X. Kong); ht@fc.ritsumei.ac.jp (H. Tomiyama)

Figure 1: Tello

camera to estimate self-lo localization for small drones. We utilize visual odometry to estimate the relative position of moving drones based on images. In order to improve a self-localization accuracy, we will experiment with varying the number of image frames used per estimation and analyze the resulting performance changes. In the experiments, we use Tello, showed in Figure 1. Tello is an affordable and programmable drone suitable for educational purposes. This small drone weighs 80 grams, measures 98×92.5×41 mm, and mounts a monocular camera with a field of view of 82.6 degrees. The camera can capture 720p resolution video at 30 fps. Tello can operate at approximately 13 minutes of continuous flight time.

The contributions of this paper are as follows:

1. By utilizing monocular camera images, we enable drone position estimation even in GPS-denied indoor environments. This extends the operational range of drones and allows them to function in areas where GPS is unavailable.
2. Utilizing only a monocular camera eliminates the need for expensive GPS receivers and Inertial Measurement Units (IMUs). This approach results in a cost-effective self-localization system, making the technology more accessible for various applications.

The structure of this paper is as follows. Chapter 2 discusses related studies to this research. In Chapter 3, we describe self-localization using Visual Odometry. Chapter 4 shows the experimental results. Chapter 5 concludes this paper and discusses future challenges.

2. Related Study

This chapter presents a review of related work on self-localization. Monocular SLAM was initially tackled using filtering techniques [3, 4, 5, 6]. This approach involves processing each frame through a filter to jointly estimate the positions of map features and the camera's pose (orientation and location). However, this method suffers from drawbacks such as computational inefficiency when dealing with consecutive frames containing limited new information, and the accumulation of errors due to linearization. In contrast, keyframe-based approaches estimate the map using only a subset of selected frames (keyframes). This allows for the application of more computationally expensive but highly accurate Bundle Adjustment (BA) optimizations. Unlike filtering methods, keyframe-based approaches decouple map updates from frame rate, enabling more flexibility. Study by Strasdat et al. [7] demonstrate that keyframe-based techniques achieve superior accuracy compared to filtering approaches at similar computational costs. PTAM, developed by Klein and Murray [8], is arguably one of the most well-known keyframe-based SLAM systems. This system introduced the concept of parallel threads for camera tracking and mapping, enabling real-time performance in small-scale augmented reality applications. More recently, Mur-Artal et al. [9] proposed ORB-SLAM2, a real-time, feature-based monocular visual odometry system. This method combines FAST keypoint detection with BRIEF rotation-invariant feature descriptors to achieve simultaneous estimation of the camera's ego-motion (self-motion) and the surrounding environment map. Engel et al. [10] introduced Direct Sparse Odometry (DSO), which directly utilizes the brightness gradients within images, rather than relying on features, to estimate the camera's ego-motion. By leveraging depth maps for direct pose estimation, this approach achieves high accuracy and excels in fast-moving or large-scale environments due to its ability to exploit visual depth information for ego-motion estimation. Building upon both feature-based and direct methods, Forster et al. [11] proposed Semi-Direct Visual Odometry (SVO). This technique employs simultaneous feature tracking and direct methods for optical flow estimation, achieving real-time visual odometry. SVO is characterized by improved computational efficiency and exceptional real-time performance, demonstrating its suitability for various indoor and outdoor environments.

3. Visual Odometry for Self-Localization

This chapter delves into the methodology and experimental setup for estimating the self-localization of the Tello using Visual Odometry.

3.1. Visual Odometry

In this section, we will explain Visual Odometry. Visual Odometry is a technique used to estimate the self-position and movement trajectory of moving robots or drones using image data from vision sensors (typically cameras). This technology is particularly useful in environments where GPS is not available, such as indoors or between buildings. Self-position estimation is performed through six steps illustrated in Figure 2:

1. Reading of the camera's intrinsic parameters: Load the intrinsic parameters of the camera that captured the images.
2. Loading the image to be processed: Load the images captured during the previous flight.
3. Detection and tracking of keypoints: Detect feature points in specific pairs of image frames using the ORB (Oriented FAST and Rotated BRIEF) algorithm. ORB is a popular feature detector and descriptor in computer vision, commonly used for tasks like SLAM (Simultaneous Localization and Mapping) and Visual Odometry, where camera pose estimation is crucial. Additionally, the BRIEF (Binary Robust Independent Elementary

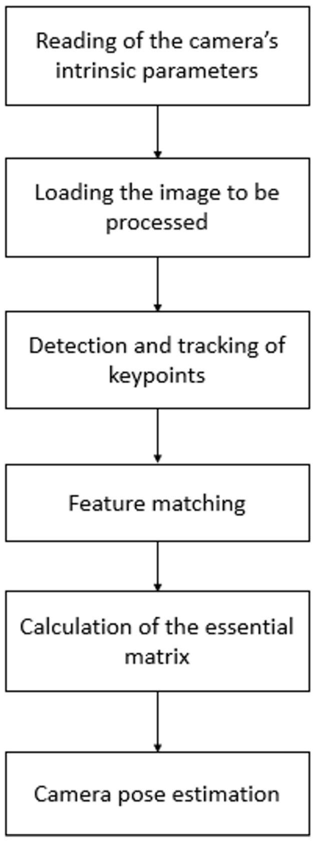

Reading of the camera's intrinsic parameters
Loading the image to be processed
Detection and tracking of keypoints
Feature matching
Calculation of the essential matrix
Camera pose estimation

Figure 2: The procedure for self-localization

Features) algorithm computes feature descriptors for each detected point, representing the surrounding features numerically.

4. Feature matching: Utilize the FLANN (Fast Library for Approximate Nearest Neighbors) library to compare and match detected feature points. FLANN is a widely used library for efficient nearest neighbor searches in computer vision and machine learning applications. To enhance matching accuracy, the distances between the two nearest neighbor points are compared, and the closer point is chosen as the match.
5. Calculation of the essential matrix: Calculate the essential matrix from the matched feature points. This matrix represents the geometric relationship between the two images, encoding information about the relative positions and rotations of the features.
6. Camera pose estimation: Decompose the essential matrix into a translation vector and a rotation matrix. These elements can be combined to form a transformation matrix that represents the camera's position and orientation in space. This information ultimately allows for estimating the self-position and movement trajectory of the robot or drone.

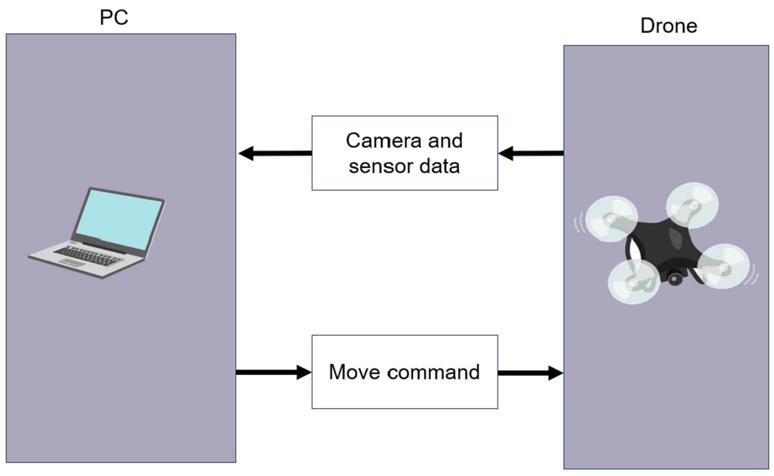


Figure 3: The overall flow if controlling Tello

3.2. Self-Localization Using Visual Odometry with the Tello

Figure 3 illustrates the overall flow of controlling the Tello. The Tello establishes a Wi-Fi connection with a PC, enabling remote control through a publicly available API. The PC receives camera images and sensor data from the Tello. This data is processed for image and sensor fusion, followed by the calculation of the Tello's movements. Finally, commands are sent to the Tello to execute the planned movements.

The Tello's self-localization process is illustrated in Figure 4. It involves a cyclical sequence. The Tello launches, receives flight commands specifying direction and distance, and then flies. Images are continuously captured while in motion. After movement, the Tello hovers, and self-position estimation is based on the previously captured images. If processing continues, the Tello flies again capturing images. If processing terminates, the Tello lands. This cyclical process repeats to continuously update the Tello's self-localization. The frame acquisition can be stopped by terminating command execution on the PC's command prompt.

When performing self-localization using the Tello, the loading of the camera's intrinsic parameters depicted in Figure 4 involves loading the pre-acquired intrinsic parameters of the Tello. The procedure for obtaining the intrinsic parameters is illustrated in Figure 5. The camera's intrinsic parameters are obtained by performing calibration using camera images in Matlab. The images used for this purpose are captured from different directions and positions, as shown in Figure 6, using a checkerboard pattern. The detectCheckboardPoints function included in Matlab's Computer Vision Toolbox is used to detect the corners of the checkerboard in each image and obtain the image coordinates of these corners. Image coordinates represent the pixel positions on the digital image. The real-world coordinates are obtained from the length of one square of the checkerboard, representing the position of objects in physical 3D space. The correspondence between the image coordinates of each corner and their corresponding real-world coordinates is established to obtain the camera's intrinsic parameters.

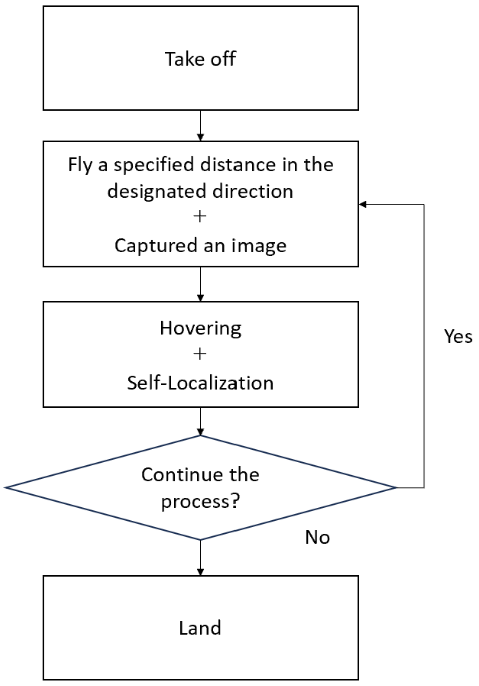


Figure 4: The overall process of performing self-localization

4. Evaluation Experiment

This section aims to quantitatively assess the performance of the self-localization algorithm by varying the number of images used in a single estimation. The objective is to verify the impact of varying image count on the accuracy of the task.

4.1. Evaluation Methods

The evaluation method for self-localization involves assessing the error between the estimated position and the true (actual) position. This error is the distance between the two points shown in Figure 7, and it is calculated using equation (1).

$$Error = \sqrt{(a-c)^2 + (b-d)^2} \tag{1}$$

The research plan includes flying the pre-planned Paths 1 to 3, as depicted in Figures 8 to 10, and evaluating their performance. Along these paths, the Tello will maintain a constant orientation, without executing any turns. Error calculation will be performed at the red checkpoints designated along Paths 1 to 3. The average error values from these checkpoints will be calculated and compared.

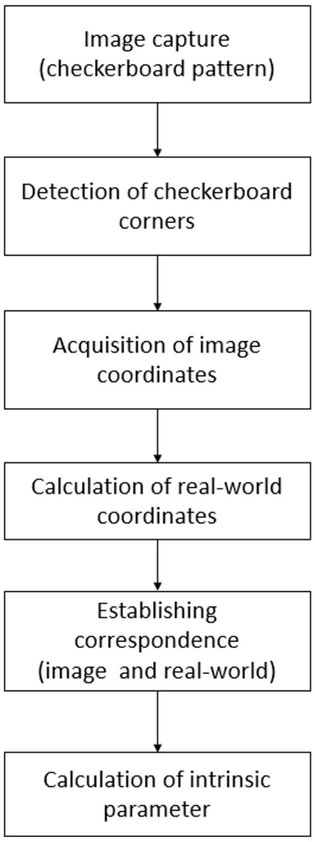


Image capture
(checkerboard pattern)
Detection of checkerboard corners
Acquisition of image coordinates
Calculation of real-world coordinates
Establishing correspondence
(image and real-world)
Calculation of intrinsic parameter

Figure 5: Procedure for obtaining intrinsic parameter

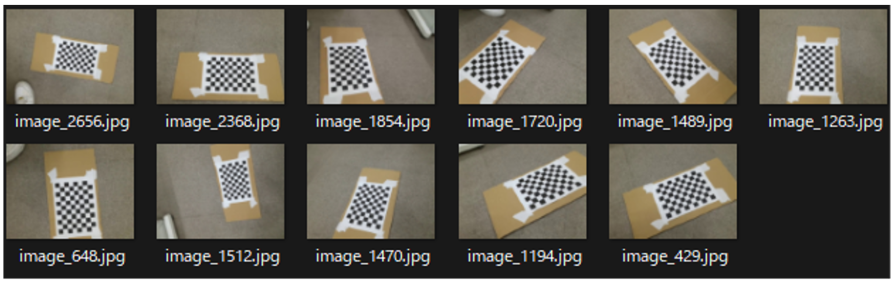


Figure 6: Images of a checkerboard captured from different directions and positions

4.2. Results and Discussion

Figures 11 to 13 present the results for Paths 1 to 3 when using 25 images in a single estimation. Each figure comprises two panels: (a) Flight path: This panel depicts the trajectory of the Tello during the flight. (b) Estimation results: This panel shows the estimated positions of the drone (represented by blue plotted points) and the actual paths measured with a ruler (represented by green arrow lines).

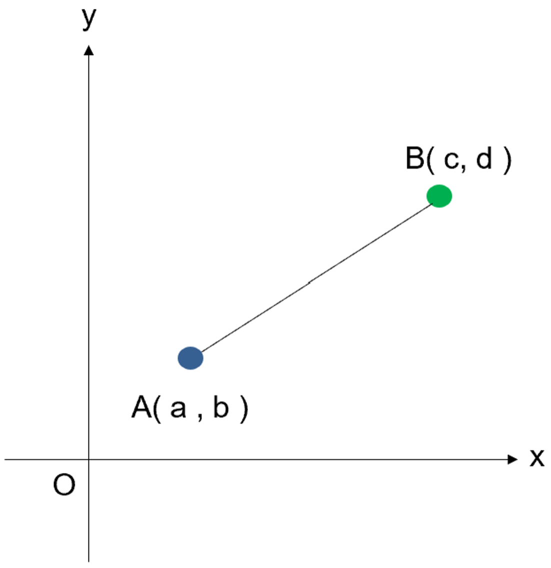


Figure 7: Distance between the two points

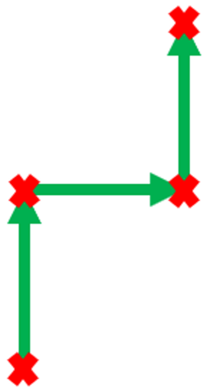


Figure 8: Path 1

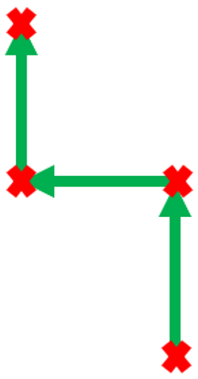


Figure 9: Path 2

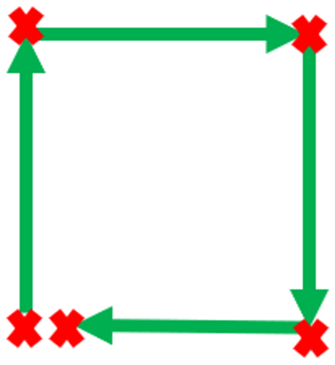
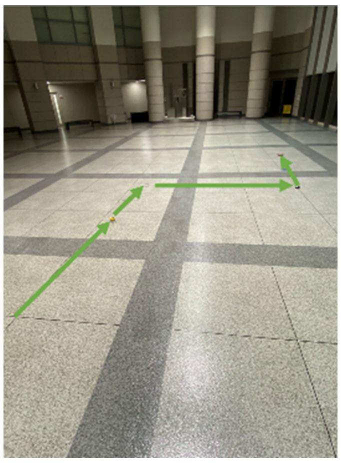
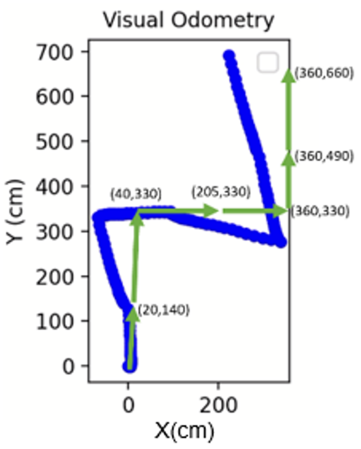


Figure 10: Path 3

Table 1 demonstrates that using 5 images per estimation minimizes the average error for path 1, while using 25 images per estimation minimizes the average error for paths 2 and 3. Moreover, as evident from Figure 13, the accuracy of backward estimations significantly decreases. Consequently, path 3 exhibited a larger estimation error compared to the other paths. Table 2 further illustrates the processing time for each estimation with different image counts (5, 25, and 50). As expected, the processing time increases with the number of images used in each estimation.

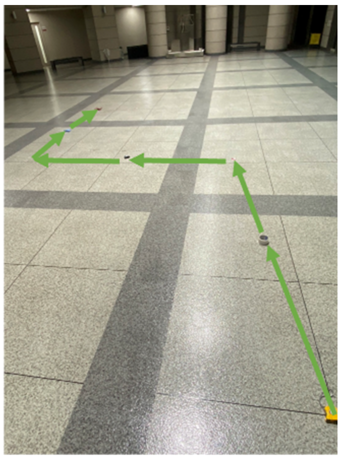


(a) Flight path

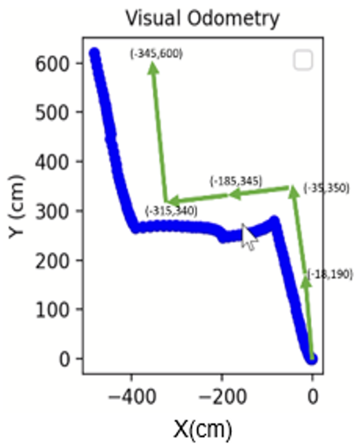


(b) Estimation path

Figure 11: Flight path and estimation results for path 1



(a) Flight path

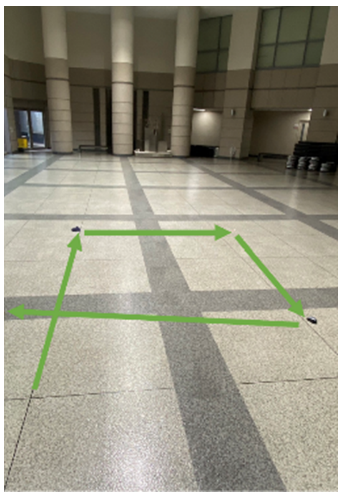


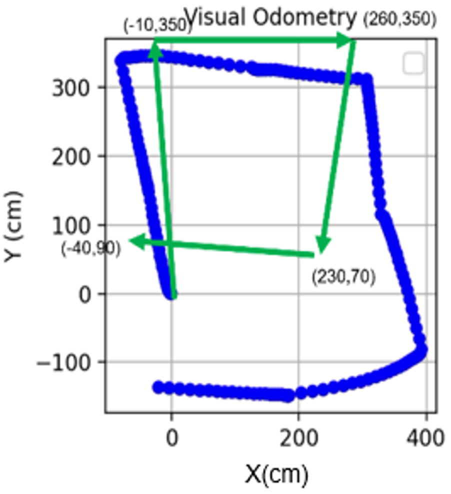
(b) Estimation path

Figure 12: Flight path and estimation results for path 2

5. Conclusion

In recent years, small drones have been widely used in various fields such as logistics, agriculture, and disaster relief. Among them, The Tello have gained popularity among individual users due to their affordable price and ease of operation. However, self-localization is essential for the Tello to fly autonomously. In this study, we aimed to realize self-localization using the Tello. The Tello are equipped with a monocular camera, and we performed self-localization using a technique called visual odometry by utilizing the information of the image frames captured by this camera. Visual odometry is a technique for estimating the camera's movement by identifying corresponding features between consecutive images. In the experiment, we flew


(a) Flight path



(b) Estimation path

Figure 13: Flight path and estimation results for path 3

Table 1
Average absolute error

Number of Images	5	25	50
Average Error for Path 1 (cm)	48.25	69.73	149.07
Average Error for Path 3 (cm)	95.87	82.13	180.71
Average Error for Path 3 (cm)	175.86	115.64	151.92

Table 2
Estimated time per iteration

Number of Images	5	25	50
Estimated Time per Iteration (Seconds)	1.8 ~2.0	9 ~11	23 ~24

the Tello along a pre-set path and verified the accuracy of self-localization. The results show that using 5 images per estimation minimizes the average error for path 1. Additionally, it was found that using 25 images per estimation minimizes the average error for paths 2 and 3. These results clarify the relationship between the number of images and accuracy/processing time and contribute to the optimization of the self-localization model for the Tello. However, further improvement in the accuracy of self-localization remains a challenge to be solved. In response to these challenges, we plan to improve the accuracy by complementing the information from sensors other than the monocular camera, such as a gyroscope and a barometer.

Acknowledgments

This work is partly supported by a research grant provided by the First Bank of Toyama, Ltd. and partly commissioned by NEDO (Project Number JPNP22006).

References

[1] D. Scaramuzza, F. Fraundorfer, Visual odometry [tutorial], IEEE robotics & automation magazine 18 (2011) 80–92.

[2] A. Couturier, M. A. Akhloufi, A review on absolute visual localization for uav, Robotics and Autonomous Systems 135 (2021) 103666.

[3] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, Monoslam: Real-time single camera slam, IEEE transactions on pattern analysis and machine intelligence 29 (2007) 1052–1067.

[4] J. Civera, A. J. Davison, J. M. Montiel, Inverse depth parametrization for monocular slam, IEEE transactions on robotics 24 (2008) 932–945.

[5] A. Chiuso, P. Favaro, H. Jin, S. Soatto, Structure from motion causally integrated over time, IEEE transactions on pattern analysis and machine intelligence 24 (2002) 523–535.

[6] E. Eade, T. Drummond, Scalable monocular slam, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 1, IEEE, 2006, pp. 469–476.

[7] H. Strasdat, J. M. Montiel, A. J. Davison, Visual slam: why filter?, Image and Vision Computing 30 (2012) 65–77.

[8] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, in: 2007 6th IEEE and ACM international symposium on mixed and augmented reality, IEEE, 2007, pp. 225–234.

[9] R. Mur-Artal, J. M. M. Montiel, J. D. Tardos, Orb-slam: a versatile and accurate monocular slam system, IEEE transactions on robotics 31 (2015) 1147–1163.

[10] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE transactions on pattern analysis and machine intelligence 40 (2017) 611–625.

[11] C. Forster, M. Pizzoli, D. Scaramuzza, Svo: Fast semi-direct monocular visual odometry, in: 2014 IEEE international conference on robotics and automation (ICRA), IEEE, 2014, pp. 15–22.