A Depth Camera Based Lightweight Visual SLAM Algorithm

Zhihao Wang, Qingxuan Jia, Ping Ye, Hanxu Sun
Department of Automation
Beijing University of Posts and Telecommunications
Beijing, China

Abstract—**The Visual SLAM algorithm has the goal of estimating the camera trajectory while reconstructing the Environment, which provides great help for autonomous navigation of mobile robot. However, many SLAM systems improve the complexity of the algorithm in order to show high precision, resulting in poor real-time performance. In practical application, considering the problem of robot life-time and the speed of embedded system, SLAM algorithm needs to improve the real-time performance of the algorithm and ensure the accuracy of application requirements at the same time. This paper presents a lightweight SLAM algorithm, which mainly through the management of map points, multi-threaded design and NEON technology to improve the real-tine performance of the algorithm while still maintaining a good localizing accuracy. It has great reference value for the practical application of SLAM algorithm.**

Keywords-component; lightweight; map point; SLAM; NEON

I. Introduction

With the continuous development of robot technology, robots begin to take place of human work in a small range, and have been widely used gradually in all walks of life. For indoor autonomous mobile robots, it is important to be able to navigation accurately in the environment. From the robot's working environment, price, power consumption point of view, the camera is used to be the main sensor of autonomous mobile robots nowadays. The vision-based SLAM algorithm can estimate the camera trajectory and reconstruct the environment at the same time, which provides great help for autonomous navigation of mobile robot [1].

However, many SLAM systems improve the complexity of the algorithm in order to show high precision, resulting in poor real-time performance. To know that we can not put a high-performance PC on the robot, because for such a hardware platform, battery life is weak, the volume can not be reduced, and stability is poor. In this way, the high complexity SLAM system will lost real-time performance in the embedded system, and it is difficult for these SLAM systems to get the practical application.

In this paper, we propose a lightweight visual SLAM algorithm. We use the MoveSense depth camera as the visual sensor, which overcomes the dependence of traditional stereo matching algorithms on complex computing systems such as desktop CPUs and GPUs, enabling all processes from image input to disparity image output on monolithic low cost FPGAs. The process does not depend on any external storage device. MoveSense has the advantages of low cost, low power consumption, small size and light weight, which makes it has great application value and potential in UAV, robot and AR. We used ODROID-XU4 as a hardware platform to verify our real-time performance and localization accuracy. ODROID-XU4 is a new generation of computing device with more powerful, more energy-efficient hardware and a smaller form factor. It is equipped with a clocked at 2GHz Samsung Exynos 5422 eight-core processor with 2GB RAM. As a result of the ARM architecture, ODROID XU4 can run based on the ARM architecture designed operating system.

Our lightweight visual SLAM algorithm greatly reduces the time consuming by reducing the number of participating projection points when estimating the pose of each frame. At the same time, through the management of the map points to ensure that the accuracy of the loss will not even increase. We have also accelerated the code through NEON technology and multi-threaded programming technology.

This paper is organized as follows: The problems faced by SLAM system in practical application are described in Section I. In Section II, our lightweight visual SLAM system is introduced. In Section III, we detail our method about accelerating the visual SLAM algorithm. The conducted experiments and result are shown in Section IV, followed by conclusions in Section V.

II. System overview

Our lightweight SLAM system is divided into front-end and back-end, a total of five threads, as shown in Figure 1.

Taking into account the actual application process, the robot will inevitably be bumpy or do high-speed rotation, relying only on the visual sensor is not guaranteed in the practical application of the robust, so we use the IMU sensor to measure the rotation between inter-frame as an initial estimate of the inter-frame pose transformations [2] [3]. The initial estimate is close to the actual pose, which helps to improve the success rate of descriptor matching.

Now the main SLAM algorithm can be divided into two methods: feature-based and direct method. Our SLAM algorithm is feature-based, using the ORB feature [4] [5], so we need to extract the oFast feature and compute the rBREIF descriptor.

So that our SLAM front-end can be divided into three time-consuming parts: feature extraction, descriptor calculation, and pose estimation. We designed three threads to do feature

This work is supported in part by the National Natural Science Foundation of China under grant No.61305126

extraction, descriptor computation and pose estimation to improve the speed of the program.

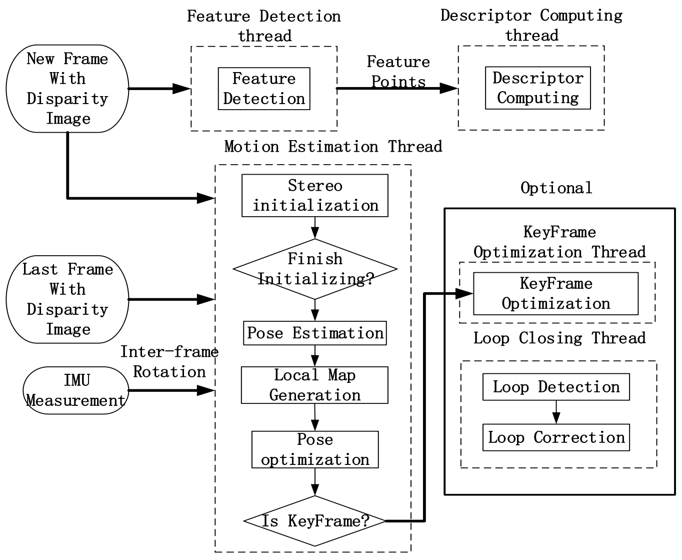

Figure 1. Lightweight SLAM system

In the feature extraction and descriptor computation part, because ODROID has Cortex-A15 processor, support NEON technology, we use the NEON technology to accelerate feature extraction and descriptor computation to improve the efficiency of the program.

In the pose estimation part, the quality of the map points participating in the projection matching is improved by the management of the map points, and the localization accuracy is maintained in the case of reducing the number of map points that participating projection matches and optimizing the current frame pose.

III. LIGHTWEIGHT METHODS

A. *Stereo Initialization*

The depth map we used is obtained by the stereo matching algorithm. According to the principle of stereo matching we can know that further points have higher depth uncertainty [6], and when the image of the scene texture is not rich or dark, stereo matching algorithm will also appear mismatch problems. Experiments have shown that when the camera is placed in a certain position, stationary, the depth of some feature points will change with time, as shown in Figure 2.

We took 150 frames continuously with a camera holding still, and extracted the FAST corner feature in the image. The depth values of each feature were computed by disparity map. We take the point depth value as the mean, and the red circle in the image represent the variance is greater than 1. Thus, it is not the best choice to generate map points by relying only on the depth information at one frame. Through successive frames observation, the points which are continuously matched in the successive frames are selected as the map points.

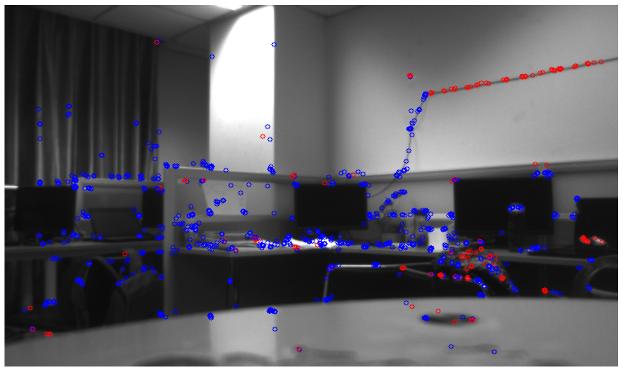

Figure 2. We take the point depth as the mean, and the red circle in the image represent the variance is greater than 1.

B. *Map points Management*

The accuracy of the map points has great influence on the pose estimation of the camera [7] [8]. When we optimize the pose of current frame, if we can choose a small number of high-precision map points to estimate the current frame pose, then this can reduce the time-consuming and also guarantee a certain localization accuracy.

We use FAST algorithm to detect the corner [5]. It is easy to extracted lots of corner features in any scene with rich texture. Actually, many points are not stable. For example, the intersection of some foreground and background objects, with the observation of the angle changed, the point's position in the world coordinate will change [9], and the descriptor will change as well. Or some of the points on moving objects, which position in the world coordinate system is always changing, descriptors are changing as well. If these points are treated as map points to optimize the camera pose, some points cannot find a matching point in the image of the optimized frame because the descriptor changes too much. Some points find a match but play a negative role in estimating the camera pose. In other words, these points will reduce the real-time performance or the accuracy. So these points should not be placed into the map as a map point.

In many SLAM algorithms, it is common to put points that the depth value is valid or smaller than a threshold into map as map points directly when a new keyframe is generated [10] [11]. In fact, we couldn't know the quality of points only through one observation. This will adversely affect the accuracy of the algorithm or performance.

Our method is to treat points that get matched between inter-frames as a candidate map point during pose estimation. These candidate map points will participate in the projection optimization before the next key frame arrives. When a new keyframe is generated, the candidate map points are sorted by the number of successful matches, and the points with higher number of matches are preferentially selected as real map points into the map. Generating new keyframes shows that the current scene has changed significantly compared to the previous keyframe, that is, those points that can be continuously tracked between two keyframes are more likely to be stable.

C. Projection points selection

For the optimization of the current frame pose, it is often done by projecting the map point into the current frame and finding the match, and then obtaining the optimal solution of the camera pose by minimizing the re-projection error [12].

$$\mathrm{T}_c^w = \arg\min_{T_c^w} \frac{1}{2}\sum_i \left\| u_i - \pi\left(T_c^w p_i^w\right)\right\|^2 \tag{1}$$

Where T_c^w is the pose of the camera in the world coordinate system, and π is the mapping of the three-dimensional point p_i^c in the camera coordinate to the two-dimensional point u_i in the image coordinate using the camera projection model.

$$u = \pi\left(p\right) \tag{2}$$

But it is not able to meet the localizing accuracy requirements after this step, we need to further optimize the pose of the current frame. The optimization method is to traverse the inlier map points which find matches in current frame to find out the corresponded keyframes. Then, project the feature points at the center of these keyframes to current frame. If the point can be projected onto the current frame, it can be assumed that the keyframe has large field of view with the current frame, then it can be inferred that there should be a considerable portion of the map point in the keyframe that can be seen in the current frame. Find the keyframes that meet the above criteria and sort them according to the distance from the current frame. The map points in these keyframes are projected to the current frame and the matches is found near the projection point, and the current frame pose is optimized again by using the re-projection error. Experiments show that when we reduce the number of keyframes to 10, it still can guarantee a high localization accuracy.

D. NEON

ARM Cortex-A9 series of processors integrated NEON technology for the first time, can effectively accelerate the multimedia applications. NEON contains 16 128-bit registers, with more than 100 complete instructions, and has a separate register system and a separate hardware execution unit, support 8, 16, 32, 64 and other data types of vector operations, up to 16-channel 8-bit data for parallel computing, can be used for 2D / 3D graphics image acceleration, audio and video codec, digital signal processing applications.

We use the rBRIEF descriptor to describe the feature points. First of all, we select a 31×31 pixel block with the feature point as the center. Then, choosing a set of n_d (x, y)-location pairs uniquely defines a set of binary tests. We consider n_d as 256.

We define test τ on patch p of size 31×31 as (3).

$$\tau\left(I_1, I_2\right) = \begin{cases} 1 & : I_1 > I_2 \\ 0 & : I_1 < I_2 \end{cases} \tag{3}$$

I is the pixel intensity in a smoothed version of p at $x = \left(u, v\right)^T$. θ is the orientation of feature point x.

$$I(x, y, \theta) = I(x\cos\theta - y\sin\theta, x\cos\theta + y\sin\theta) \tag{4}$$

rBRIEF descriptor is a 256-dimensional bit string.

$$f_n(p) = \sum_{1\le i\le n_d} 2^{i-1}\tau\left(I_1, I_2\right) \tag{5}$$

Each feature has its own orientation θ. Therefore, the main task of computing the feature point descriptor is to correctly compute the coordinates of each point pair in the image coordinate system and compare the intensity value of the two points in the same pair. The coordinates of the feature are stored in memory in 16-bit form, and if we use NEON's 128-bit register here, we can handle 8 points at the same time, improve the efficiency of the computation, and reduce the run time of the descriptor computing part of the code.

E. Muti-thread design

Our lightweight SLAM algorithm front-end is divided into three parts: feature extraction, descriptor computation and pose estimation. We put these three parts into three threads in parallel.

After extracting the feature points of a pyramid of the current image, the descriptor is computed and the feature points of the next image pyramid can be extracted at the same time. When the feature points of the nth frame and the descriptor are obtained, the pose estimation of the nth frame can be performed, and the feature extraction and descriptor of the n+1th frame can be computed at the same time. After completing the feature extraction and the descriptor computation, we can solve the pose of the current frame, and at the same time we can compute the feature points and descriptors of the next frame. We have improved the speed of the front-end of the SLAM algorithm in such a parallel way.

IV. EXPERIMENTAL RESULT

We use the EuRoC dataset to verify the accuracy and real-time performance of the algorithm [13]. The first part of EuRoC dataset is taken at a factory using a miniature UAV mounted with a stereo camera. The dataset is divided into simple, medium and difficult sets according to the degree of difficulty. For the most difficult one, the scene changes between inter-frames are large, the intensity of image changes greatly and the image is blurred due to the rapid movement of the UAV, which is very challenging for the SLAM algorithm.

We use ODROID-XU4 as the platform to verify our algorithm.

Figure 3. Lightweight SLAM system

A. LocalMap Generation

It is not able to meet the localizing accuracy requirements by estimating current camera pose by map points in last frame. We need to further optimize the pose of the current frame by more map points. Our method to search map points shows better real-time performance than ORB-SLAM during the pose optimization.

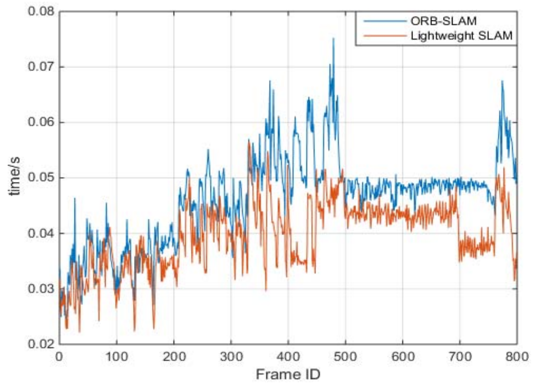

Figure 4. Timing result of pose optimization compared with ORB-SLAM

B. NEON

We have accelerated the process of feature extraction and descriptor computation through NEON technology. As shown in Figure 5, we compare the time-consuming of the feature extraction part with and without NEON by MH02, and the average time consumed by the feature extraction part after using NEON reduced 24ms.

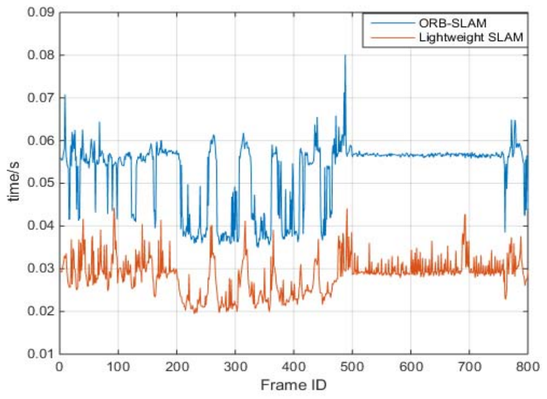

Figure 5. Timing result of using NEON compared with non-NEON in feature extraction part

Figure 6 shows the time-consuming in the descriptor computation part before and after the NEON is applied. The runtime reduced by 29ms per frame after the NEON is applied.

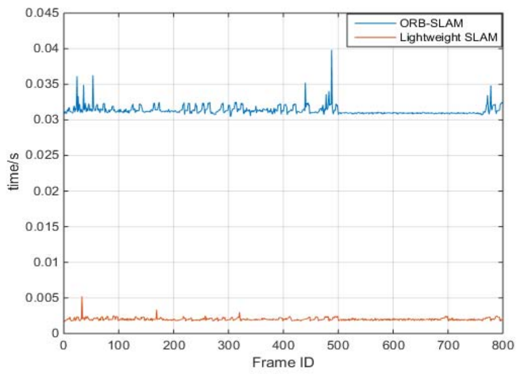

Figure 5. Timing result of using NEON compared with non-NEON in descriptor computation part

C. Algorithm Runtime Evaluation

We compare the runtime of our lightweight SLAM algorithm with the ORB-SLAM algorithm. As shown in Figure 6, our algorithm shows better real-time performance.

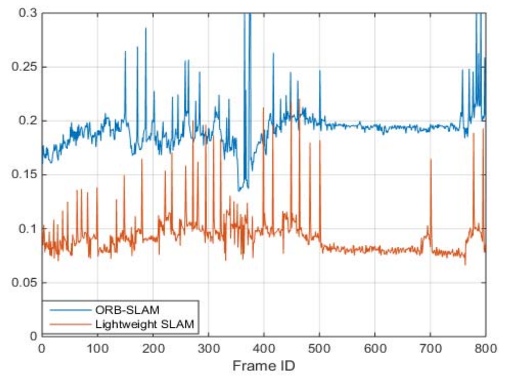

Figure 6. Timing result of pose optimization compared with ORB-SLAM

D. Accuray

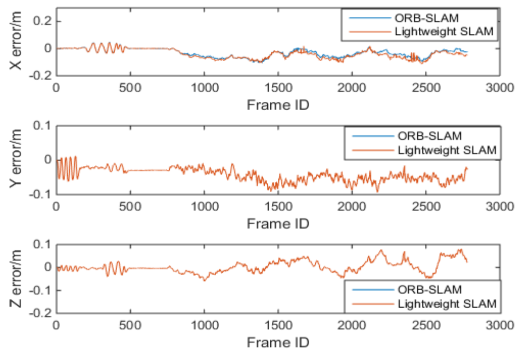

Figure 7. Position drift of lightweight SLAM and comparison against ORB-SLAM.

We evaluate the accuracy of our lightweight SLAM on MH02. The groundtruth for the trajectory originates from a motion capture system. Figures 7 and 8 illustrate the position and attitude error over time. Our algorithm is almost as accurate as ORB-SLAM.

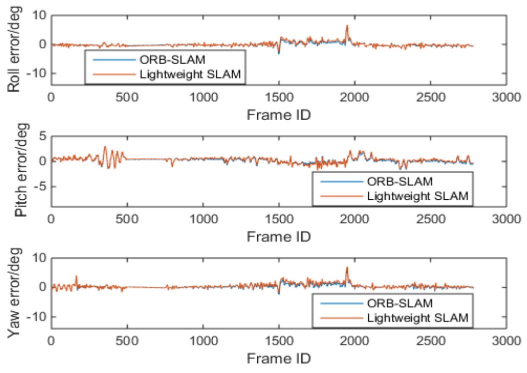

Figure 8. Attitutde drifts of lightweight SLAM and comparison against ORB-SLAM.

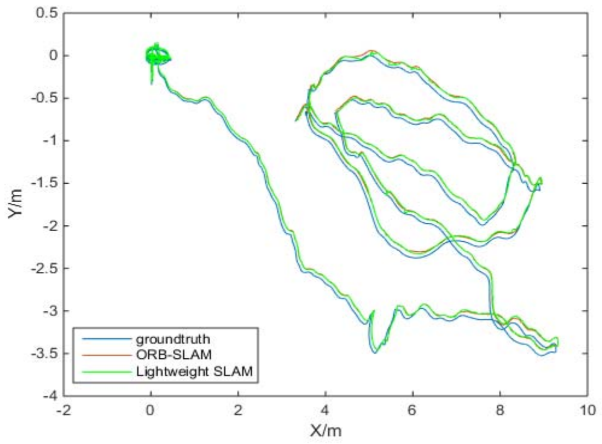

Figure 9. Trajectory of groundtruth, ORB-SLAM and our lightweight SLAM in MH02

TABLE I. POSE AND ROTATION ERROR

	Pos-RMSE [m/s]	Pos-Median [m/s]	Rot-RMSE [deg/s]	Rot-Median [m/s]
Lightweight SLAM	0.030848	0.034014	0.3307	0.3424
ORB SLAM	0.033815	0.04148	0.3517	0.3379

E. Practical application

One of the best ways to verify the performance of an algorithm is to put the algorithm in practice. As shown in Figure 10, we built a complete mobile robot system. The system uses AnyCBot as the robot chassis, MoveSense depth camera as the visual sensor unit, ODROID-XU4 as the hardware platform of the lightweight SLAM algorithm. We have generated the sparse point cloud and occupancy grid map and do autonomous navigation in the indoor environment, showing better real-time performance, and have high localization accuracy.

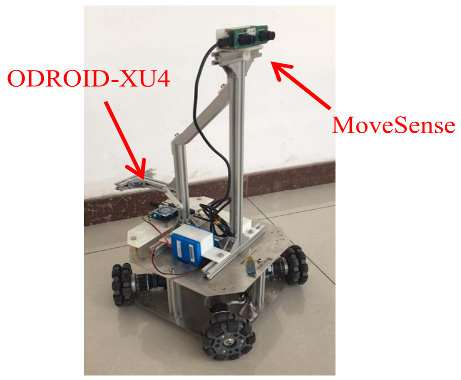

Figure 10. Mobile Robot System

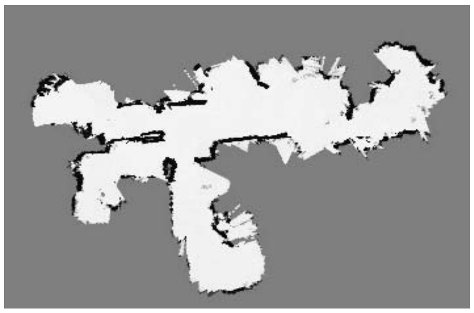

Figure 11. High accuracy occupancy grid map

V. CONCLUSION

From the practical use point of view, we must improve the real-time performance of the vSLAM algorithm and ensure that meet the application requirements of the accuracy if we want to use vSLAM on mobile robot or unmanned aerial vehicles. However, many SLAM systems improve the complexity of the algorithm in order to show high precision, resulting in poor real-time performance. Our lightweight SLAM improves the real-time performance and ensures high localization accuracy. Our algorithm has great reference value for the practical application of SLAM algorithm.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under grant No.61305126.

REFERENCES

[1] Strasdat, H., Montiel, J. M. M., & Davison, A. J. (2010). Real-time monocular SLAM: Why filter?. *IEEE International Conference on Robotics and Automation* (Vol.58, pp.2657-2664). IEEE.

[2] Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., & Siegwart, R. (2014). Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization. *Robotics: Science and Systems* (Vol.34, pp.789–795).

[3] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, & Paul Furgale. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research, 34*(3), 314-334.

[4] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2012). ORB: An efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision* (Vol.58, pp.2564-2571). IEEE.

[5] Rosten, E., & Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. *European Conference on Computer Vision* (Vol.3951, pp.430-443). Springer-Verlag.

[6] Moreno, F. A., Blanco, J. L., & González, J. (2007). An Efficient Closed-Form Solution to Probabilistic 6D Visual Odometry for a Stereo Camera. *International Conference on Advanced Concepts for Intelligent Vision Systems* (Vol.4678, pp.932-942). Springer-Verlag.

[7] Cvišić, I., & Petrović, I. (2015). Stereo odometry based on careful feature selection and tracking. *European Conference on Mobile Robots* (pp.1-6). IEEE.

[8] Shi, J., & Tomasi, C. (2002). Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94. 1994 IEEE Computer Society Conference on* (Vol.600, pp.593-600). IEEE.

[9] Gao, X., & Zhang, T. (2015). Robust rgb-d simultaneous localization and mapping using planar point features. *Robotics & Autonomous Systems,72*, 1-14.

[10] Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. *IEEE International Conference on Robotics and Automation* (pp.15-22). IEEE.

[11] Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics,* vol.31, pp. 1147-1163.

[12] Huai, J., Toth, C. K., & Grejner-Brzezinska, D. A. (2015). Stereo-inertial odometry using nonlinear optimization. *International Technical Meeting of the Satellite Division of the Institute of Navigation*.

[13] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., & Omari, S., et al. (2016). The euroc micro aerial vehicle datasets. *International Journal of Robotics Research,* vol. 35, pp. 1157-1163.