Direct-ORB-SLAM: Direct Monocular ORB-SLAM

Linyan Cui\textsuperscript{*}, Chaowei Ma\textsuperscript{1} and Fei Wen\textsuperscript{1}

\textsuperscript{1}Image Processing Center, School of Astronautics, Beihang University, Beijing, 102200, China

\textsuperscript{*}Corresponding author’s e-mail: cuily@buaa.edu.cn

Abstract. Simultaneous Localization and Mapping (SLAM) plays an important role in the computer vision and robotics field. The well-known ORB-SLAM extracts ORB features and uses feature matching to estimate the camera pose and construct a sparse map. However, the features extracted in the common frame (relative to the keyframe) are not used again except in this camera pose estimation stage, which will cause the waste of computation resources. In this work, we propose a new SLAM framework, i.e. Direct-ORB-SLAM. It uses image intensity for data association in common frames to avoid the redundant computations and performs the relocalization and loop closure detection in ORB-SLAM2 for keyframes. The photometric calibration is also employed to benefit for accuracy and robustness. The popular public dataset EuRoC is adopted to approve the efficiency of proposed method. Results indicate that the proposed Direct-ORB-SLAM algorithm runs two times faster than the well-known ORB-SLAM at the expense of a slight reduction in accuracy.

1. Introduction

In the past several decades, considerable attention has been paid to Simultaneous Localization and Mapping (SLAM) in computer vision and robotics communities [1]. SLAM consists of simultaneous estimation of the state of a robot and reconstruction of an unknown environment the robot exploring. SLAM can be widely used in many emerging technologies, from driverless cars and unmanned aerial vehicles to augmented and virtual reality. So far, the feature-based methods occupy the main position of the SLAM field, while direct methods have lately received great attentions.

1.1. Related work

The SLAM systems compose of front-end and back-end. The front-end aims at initially estimating the motion of camera and establishing the environment models through raw sensor measurements. The back-end optimizes the data delivered by the front-end and detects loop closure. We can divide the SLAM systems into two categories: feature-based methods and direct methods, according to the different strategies in the front-end.

Feature-based methods split the task into two steps. Firstly, a set of features in the image are extracted as intermediate representation, such as SIFT [2], SURF [3], and ORB [4]. Secondly, these extracted features are adopted to estimate camera’s state and reconstruct the environment. It is worthwhile mentioning that the feature-based methods are capable to deal with large frame-to-frame motions due to the distinctiveness of features and they are convenient to employ Bundle Adjustment [5] to optimize camera motion and geometry structure because of the features’ efficiency. Up to now, the representative feature-based SLAM systems include MonoSLAM [6], PTAM [7], and ORB-SLAM [8]. The MonoSLAM presented by Davison et al. is the first monocular SLAM system which
can operate in real-time. But the MonoSLAM is based on Extended Kalman Filter (EKF) which is proved less accurate than non-linear optimization at the same computational cost [9]. Klein and Murray proposed PTAM, a representative feature-based SLAM system. They proposed the idea of front-end and back-end which run in parallel threads to tracking and mapping for the first time. They also proposed to use nonlinear optimization techniques rather than filtering in SLAM system to estimate the motion and structure. However, the PTAM adopted the restrictive keyframe insertion strategy which makes the tracking easy to lost. The well-known ORB-SLAM presented by Mur-Artal et al. is the peak of feature-based monocular SLAM system. It runs in three parallel threads: tracking, local mapping and loop closing. Then, they proposed ORB-SLAM2 [10] which is a SLAM system for monocular, stereo and RGB-D cameras. It is one of the most complete and robust SLAM systems. However, all threads are based on ORB features in ORB-SLAM2, so the time-consuming feature extraction step should be carried out in every frame. In this paper, ORB-SLAM2 will always refer to the monocular ORB-SLAM2.

Direct methods skip the feature extraction step and directly use image intensity to estimate structure and motion. They are capable to utilize image region with large intensity gradient, leading to more robustness and higher accuracy in textureless region than feature-based methods. Meanwhile, direct methods can avoid a series of time-consuming processes include feature extraction and descriptor computation, which can reduce computational cost per frame. However, the loop detection cannot be achieved through a fully direct model. DTAM [11] is a direct dense SLAM system proposed by Newcombe et al. It uses every pixel to reconstruct environment and gets a dense map. The computational cost is so high that DTAM can only be real-time using GPU hardware. LSD-SLAM [12-13] is the work of Engel et al., which can build a semi-dense map using direct methods. The "semi-dense" means the system only utilizes the pixels with large intensity gradient. Compared to DTAM, LSD-SLAM that runs in real-time on a CPU is more reasonable and efficient. However, using the direct methods makes the LSD-SLAM very sensitive to large frame-to-frame motion and the image quality. Furthermore, the features are still needed for loop detection. SVO [14-15] is a direct semi-dense visual odometry system presented by Forster et al. SVO neither computes descriptor of features nor tracks dense points, so it can process more than 100 frames per second. The SVO also modifies a depth filter [16] to estimate the depth of features. Nevertheless, the lack of features descriptor makes it hard to perform relocalization when the tracking is failed. DSO [17], proposed by Engel et al., is a fully direct visual odometry system. DSO samples pixels evenly throughout the region of images in the front-end and maintains a sliding window in the back-end. This makes the DSO run in real-time. However, the lack of map reuse in DSO makes the accumulated error hard to be eliminated.

1.2. Motivation

Since all SLAM tasks in ORB-SLAM2, including tracking, mapping, relocalization and loop closing, rely on ORB features, the features extraction is the first step performed with every frame whether the frame is a keyframe or not. As we all know, features extraction, especially descriptor computation, occupies a large amount of computational resources. However, if a frame is not a keyframe, the features extracted in it are only used to estimate the pose of the frame and just a few tenths of features take part in this step. This wastes a lot of time to compute the useless descriptors. Table 1 shows the ratio of used features to all extracted features. We selected three long sequences from three datasets for the experiment. As there is no difference between keyframes and common frames in tracking thread of ORB-SLAM2, we use the mean data of all frames in the sequences in table 1. It can be seen that the highest percentage of used features is only 29.25%.

| Sequence MH01 in EuRoC dataset | Sequence 00 in KITTI dataset | Sequence fr3_long_office in TUM RGB-D dataset |
|-------------------------------|-------------------------------|-----------------------------------------------|
| The number of extracted       | 1007                          | 2006                                          | 1005                                          |
features

| The number of feature matching in tracking thread | 244 | 212 | 294 |
|-------------------------------------------------|-----|-----|-----|
| The ratio of used features to all features      | 24.23% | 10.57% | 29.25% |
| The ratio of common frames to all frames         | 94.89% | 65.31% | 92.80% |

Directly using pixel intensity rather than features to get the initial state estimation of the camera can avoid the extra computation cost. However, it is hard to perform loop detection to reduce the accumulated errors because it focuses on scene local information and lack scene global information. To deal with the problems mentioned above, we will track the common frames through image intensity and perform the feature extracting and feature matching to track keyframes. Through this strategy, the proposed system saves the computation cost in common frames alongside maintains the accurate loop detection and relocalization of ORB-SLAM2. The saved time can be used in other aspects. The details will be illustrated in section 2.

1.3. Contribution and outline

In this paper we propose a new SLAM framework, i.e. Direct-ORB-SLAM. It uses the image intensity to track common frames and adopts the well-known ORB-SLAM2 to track keyframes and detect loop closure. This framework aims at increasing tracking speed while keeping the accuracy, and it achieves two times faster than ORB-SLAM2 in tracking thread. To alleviate the sensitivity to pixel intensities variation introduced by direct methods, we also apply photometric calibration to weaken the influence of exposure time and brightness variations by estimating the photometric parameters we used. We evaluate our system in popular public dataset EuRoC to verify the performance.

The proposed Direct-ORB-SLAM is described in section 2. Specifically, this comprises the notation in Section 2.1, the system overview in Section 2.2, the direct tracking step and pose refinement step of direct methods in Section 2.3, 2.4, respectively, the keyframe selection policy and photometric calibration model in Section 2.5. Section 3 evaluates the accuracy and speed of our system and compares them with ORB-SLAM2 in public dataset EuRoC. Finally, a summary is provided in Section 4.

2. Direct-ORB-SLAM

We introduce the proposed Direct-ORB-SLAM system in this section. The tracking method is the main aspect of the illustration.

2.1. Notation

Throughout the paper, the intensity image captured by camera at timestamp k is denoted with $I_k$. $\Omega_k$ is the image domain. The rigid body transformation $T_{k,k-1} \in SE(3)$ allows us to map a 3D point $p_{k-1}$ from the camera frame of reference k-1 to k: $p_k = T_{k,k-1} \cdot p_{k-1}$, where the subscript k means the point coordinate is described in camera frame of reference k. The camera projection model $\pi$ maps a 3D point $p_k$ to the pixel coordinates $u_k : u_k = \pi(p_k)$ and it relates with the camera intrinsic matrix which is known after camera calibration. For the sake of derivative of pose in optimization, we use twist coordinate $\xi \in se(3)$ which is mapped to SE(3) by the exponential map: $T(\xi) = \exp(\xi \cdot \varepsilon)$. 
2.2. System overview

The overview of the proposed Direct-ORB-SLAM system can be seen in figure 1. It includes three threads: tracking, local mapping and loop closing, which run in parallel.

The tracking thread is the first step of the system that estimates the pose of every frame and makes the decision whether the frame is a keyframe or not. In the proposed system, the tracking procedure is divided into two branches to process keyframes and common frames, respectively. Keyframes are processed with four steps. (1) Extract ORB features. (2) Estimate pose through minimizing reprojection error. (3) Add features extracted in other keyframes according to feature matching. (4) Deliver the keyframe to local mapping thread for further processing. Common frames are processed with two steps. (1) Pose estimate via minimizing photometric error. (2) Project map points in current frame to refine the camera pose and select map points for tracking next frame. After the initial pose estimation of each frame, the motion-only Bundle Adjustment is performed to optimize the state of cameras. The common frames are processed by image intensity which avoid extracting features, and the time can be saved in this way.

The local mapping thread preserves keyframes, map-points and their relationships. The ORB features in the new keyframe find matches in other keyframes to triangulate new map-points. The local mapping is also in charge of map-point culling and keyframe culling to limit the scale of map, which will benefit the system's real-time performance.

The loop closing thread detects loops once a new keyframe is received. This detection is very strict in order to guarantee that the loop is correct. Once a loop is detected, a 7 degree of freedom Sim(3) transformation is calculated between two looped keyframes. Then, the optimization is performed in a coarse-to-fine way for all the keyframes in the loop and it can reduce the accumulated error and scale drift in the loop.

We use the image intensity in tracking thread to process common frames. In this work, we build on the method of SVO. It contains two main steps: direct tracking and pose refinement. Direct tracking uses sparse image block alignment to get estimation of camera pose, and pose refinement reuse the map to further estimate the more accurate pose which has a long-time consistency. The following two sections will illustrate these two steps in detail.

![Diagram of Direct-ORB-SLAM system overview](image-url)
2.3. Direct tracking
The goal of direct tracking is to estimate the transformation of two consecutive frames $T_{k,k-1}$ by means of minimizing the photometric error:

$$T_{k,k-1} = \arg \min_{T_{k,k-1}} \sum_{p_{k-1,k}} \frac{1}{2} \left\| e_{p_{k-1,k}}(T_{k,k-1}) \right\|^2.$$  

(1)

The photometric error is $e_{p_{k-1,k}}(T_{k,k-1})$, defined by the intensity difference of pixels that are projected by the same 3D point $p_{k-1}$ on images $I_k$ and $I_{k-1}$, respectively:

$$e_{p_{k-1,k}}(T_{k,k-1}) = I_k(\pi(T_{k,k-1} \cdot p_{k-1})) - I_{k-1}(\pi(p_{k-1})), \forall p_{k-1} \in P_{k-1}.$$  

(2)

where $P_{k-1}$ is the set of 3D points that can be observed by image $I_k$ and $I_{k-1}$:

$$P_{k-1} = \{ p_{k-1} \mid \pi(T_{k,k-1} \cdot p_{k-1}) \in \Omega_k \ \& \ \pi(p_{k-1}) \in \Omega_{k-1} \}.$$  

(3)

Equation (1) is a nonlinear least square problem that can be solved by an iterative Gauss-Newton procedure. In practice, we use the inverse compositional formulation [18] of the photometric error and perform the optimization on Lie-manifolds:

$$e_{p_{k-1,k}}(\xi) = I_k(\pi(T_{k,k-1} \cdot p_{k-1})) - I_{k-1}(\pi(\exp(\delta \xi) \cdot p_{k-1})).$$  

(4)

Once we get the right-compositional increment $\delta \xi$, the update step is performed:

$$T_{k,k-1} = T_{k,k-1} \cdot \exp(\delta \xi)^{-1}.$$  

(5)

Thanks to the inverse compositional algorithm, the Jacobian and Hessian matrix can be pre-computed, which lead to an obvious acceleration. We use small patches centered at $p_{k-1} \in P_{k-1}$ instead of only the map points to aggregate the photometric error for robustness.

2.4. Pose refinement
The estimation of the state in direct tracking step is far from accurate even for the common frames, as the information we use of last frame is incomplete. To refine the current state of the camera, we need more map points in local keyframes. Keyframes meet one of the following requirements are called local keyframes: (1) observe the map points tracked by the direct tracking step of the current frame. (2) share observations with keyframes in (1) of the same map points (over a threshold). So, in order to find more map point correspondences, we project all map-points observed by local keyframes on current frame, then utilize the intensity constancy assumption to optimize the projection position ($u_k$) on current image:

$$u_k = \arg \min_{u_k} \sum_{p_{k-1,k}} \frac{1}{2} \left\| e_{p_{k-1,k}}(u_k) \right\|^2,$$  

(6)

where the intensity error is:

$$e_{p_{k-1,k}}(u_k) = I_k(u_k) - I_{k-1}(\pi(T_{k,k-1} \cdot \pi^{-1}(u_k))),$$

$$u_k = \pi(T_{k,k-1} \cdot p_{k-1}).$$  

(7)

Once we get more features correspondences, we perform a Bundle Adjustment to optimize the camera pose while all the positions of map points are fixed. This step minimizes the reprojection error which satisfies the epipolar constraints:
\[ T_{k,k-1} = \arg\min_{T_{k,k-1}} \sum_{p \in \mathcal{P}_k} \frac{1}{2} \| u_k - \pi(T_{k-1,k} \cdot \pi^{-1}(u_k)) \|^2. \]  

where the pixel coordinate \( u_k \) is defined in (7). All the map points involved in this step are used in the direct tracking of next frame.

2.5. Keyframe selection and photometric calibration
Before illustrating the keyframe selection strategy, we explore the process of tracking thread in detail. Once the SLAM system is initialized, some map points are created through triangulation. After the initialization, the common frames estimate their pose according to these map points until the next keyframe is inserted. When a keyframe is delivered to local mapping thread, some new map points are created to adapt to the new environment. So, the tracking of common frame is easy to lost if keyframe insertion is not in time.

The keyframe is inserted if one of the following conditions are met: (1) The camera has rotated more than 30 degrees since last keyframe. (2) The features used in direct tracking step is less than 100. These two conditions ensure the robustness of tracking under poor conditions and the accuracy of tracking in common frames.

The direct methods which rely on brightness constancy assumption are more sensitive to image quality than feature-based methods. For the real imaging scene, the image intensity is not proportional to irradiance due to the non-linear response function and vignetting. To remove vignetting artifacts and account for non-linear response functions, we perform the photometric calibration proposed in [19]. The image formation model is given by

\[ I(x) = G(t V(x) B(x)), \]

where \( G \) is the camera response function, \( V \) is pixel-wise attenuation factor, \( B \) is irradiance image and \( t \) is the exposure time. The photometrically corrected image can be computed by:

\[ \Gamma(x) = t B(x) = G^{-1}(I(x)) / V(x). \]

Note that the most of public datasets do not have the information of photometric calibration, we use affine brightness transfer function proposed in DSO to evaluate the system:

\[ \Gamma(x) = e^{-a}(I(x) - b). \]

This function can rectify the influences of exposure time and mean image brightness through estimating the parameters of \( a \) and \( b \). The photometric error in (2) then becomes:

\[ e_p(T_{k,k-1}) = (I_k(\pi(T_{k-1,k} \cdot p_{k-1,k})) - b_k) - e^{a_k} \Gamma e^{b_k}(I_k(\pi(p_{k-1,k})) - b_k), \forall p \in \mathcal{P}_{k-1,k}. \]

Direct tracking estimates not only the state of camera but also the two parameters of photometric calibration: \( a \) and \( b \). We also use this model in the pose refinement step to find more matches in the local keyframes.

3. Evaluation
We have carried out experiment of our system in public datasets: the EuRoC dataset [20], evaluating the performance of our system. The ORB-SLAM2 runs in the same sequences for comparisons. Our Direct-ORB-SLAM system processes the images at the frame rate they were captured. Experiments were conducted on a desktop computer with an Intel Core i5-6500(4 cores @ 3.20GHz) and 8GB RAM. We run all the sequences five times and always record median results, which can limit the uncertainty of the multithreading system. As absolute scale is unknown in monocular slam system, we
align all the trajectories by means of Sim (3) similarity transformation [21] to compare Direct-ORB-SLAM with ground truth.

The EuRoC dataset contains 11 sequences which were collected from a micro aerial vehicle (MAV) at 20Hz and can be classified into two parts. The first part of datasets was recorded in an industrial machine hall and contains millimeter accuracy position ground truth provided by a laser tracker. The second part of datasets was recorded in a vicon room with two different scenarios. All the sequences are divided into three levels: easy, medium, and difficult according to trajectory length, lighting conditions and flight dynamics. The absolute translation RMSE and the mean time spent by tracking thread of Direct-ORB-SLAM are shown in table 2. At the same time, the results for ORB-SLAM2 are also exhibited for comparisons.

Table 2. Comparisons of accuracy and speed in EuRoC dataset for Direct-ORB-SLAM and ORB-SLAM2.

| Sequence    | Length (m) | RMSE(m) | T(ms) |
|-------------|------------|---------|-------|
|             | Direct-ORB-SLAM | ORB-SLAM2 |       |
| MH01_easy   | 80.6       | 0.049   | 0.045 | 12.11  | 32.33  |
| MH02_easy   | 73.5       | 0.036   | 0.035 | 13.75  | 29.97  |
| MH03_medium | 130.9      | 0.057   | 0.039 | 17.47  | 28.56  |
| MH04_difficult | 91.7     | 0.063   | 0.059 | 17.71  | 25.92  |
| MH05_difficult | 97.6     | 0.098   | 0.053 | 17.32  | 26.44  |
| V101_easy   | 58.6       | 0.095   | 0.096 | 15.81  | 31.83  |
| V102_medium | 75.9       | 0.247   | 0.064 | 17.58  | 26.93  |
| V103_difficult | 79.0     | X       | X     | 17.32  | 26.44  |
| V201_easy   | 36.5       | 0.063   | 0.059 | 14.13  | 27.81  |
| V202_medium | 83.2       | 0.093   | 0.055 | 18.28  | 29.65  |
| V203_difficult | 86.1     | X       | X     | X      | X      |

It can be seen that Direct-ORB-SLAM can process all the sequences except V103 and V203, just like ORB-SLAM2. These two sequences are difficult for monocular SLAM system due to poor illumination conditions and sever motion blur caused by rapid rotations. In V102, Direct-ORB-SLAM get a much higher RMSE than ORB-SLAM2 due to the continuous rotation in the sequence. In other sequences, we achieved approximate 2 times speed up and slightly reduced the accuracy with respect to ORB-SLAM2. It is notable that in all sequences that are marked by easy, Direct-ORB-SLAM performs much better than in medium and difficult sequences. This is caused by sensitivity of direct method to large frame-to-frame motion.

Figure 2 displays the tracking time of Direct-ORB-SLAM and ORB-SLAM2 in MH01. Because the ORB-SLAM2 extract features in every frame, the time used in tracking always at a high level. As for Direct-ORB-SLAM, the common frames tracked directly through image intensity that is much faster than ORB-SLAM2 and the time used in keyframes is similar to ORB-SLAM2. The time spent during tracking in common frames and keyframes is approximately 0.008s and 0.0343s, respectively. So, the mean time we used in tracking depends on the ratio of common frames to keyframes. The higher the ratio, the faster the tracking.
4. Conclusion
In this paper we have presented a new SLAM system, i.e. Direct-ORB-SLAM framework, which reducing computational cost in the front end, maintaining robustness in large motion, reusing of the map, and efficient optimization of camera motion and map structure. Our system runs distinctly faster than ORB-SLAM2 while keeping highly competitive accuracy. We separate the frames into keyframes and common frames to process them with different methods in tracking thread. The keyframe is processed through feature matching to create new map points and try to close loop. The common frames utilize image intensity to estimate their pose to save computational cost. The keyframe selection strategy is very important in this case, since more keyframes can boost robustness and accuracy of tracking but reduce the speed. In addition, we use the photometric calibration model to reduce the influence of camera exposure time and brightness variations. Future extensions might include increasing speed through reducing the keyframes’ size, increasing motion blur robustness, and so on.

References
[1] Cadena, C., Carlone, L., Carrillo, H., et al. (2016) Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. IEEE Transactions on Robotics, 32(6): 1309-1332.
[2] Lowe, D.G. (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2): 91–110.
[3] Bay, H., uyntelaars, T., Van Gool, L. (2006) SURF: Speeded Up Robust Features. In: European Conference on Computer Vision (ECCV). Graz. pp. 404–417.
[4] Rublee, E., Rabaud , V., Konolige, K., Bradski, G. (2011) ORB: an efficient alternative to SIFT or SURF. In: IEEE International Conference on Computer Vision (ICCV). Barcelona. pp. 2564–2571.
[5] Agarwal, S., Snavelv, N., Simon, I., Seitz, S. M., Szeliski, R. (2010) Bundle adjustment in the large. In: European Conference on Computer Vision. Berlin. pp. 29–42.
[6] Davison, A., Reid, I., Molton, N., Stasse, O. (2007) Monoslam: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6): 1052-1067.
[7] Klein, G., Murray, D. (2007) Parallel tracking and mapping for small AR workspaces. In: IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Nara. pp. 225–234.
[8] Mur-Artal, R., Montiel, J., Tardos, J. (2015) ORB-SLAM: a versatile and accurate monocular SLAM system. Transactions on Robotics, 31(5):1147–1163.
[9] Strasdat, H., Montiel, J. M., Davison, A. J. (2012) Visual slam: Why filter? Image and Vision Computing, 30(2): 65-77.
[10] Mur-Artal, R., Tardos, J. (2017) ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Transactions on Robotics, 33(5): 1255-1262.
[11] Newcombe, R.A., Lovegrove, S. J., Davison, A. J. (2011) DTAM: Dense tracking and mapping in real-time. In: IEEE International Conference on Computer Vision (ICCV). Barcelona. pp. 2320–2327.
[12] Engel, J., Stueckler, J., Cremers, D. (2015) Large-scale direct SLAM with stereo cameras. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg. pp. 1935-1942.
[13] Engel, J., Schöps, T., Cremers, D. (2014) LSD-SLAM: Large scale direct monocular SLAM. In: European Conference on Computer Vision (ECCV). Zurich. pp. 834-849.
[14] Engel, J., Sturm, J., Cremers, D. (2013) Semi-dense visual odometry for a monocular camera. In: International Conference on Computer Vision (ICCV). Sydney. pp. 1449-1456.
[15] Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, M. (2017) SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems. IEEE Transactions on Robotics, 33 (2) :249-265.
[16] Vogiatzis, G., Hernandez, C. (2011) Video-based, real-time multi view stereo. Image Vision Computing, 29(7): 434–441.
[17] Engel, J., Koltun, V., Cremers, D. (2017) Direct Sparse Odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40: 611-625.
[18] Baker, S., Matthews, I. (2002) Lucas-Kanade 20 Years On; A Unifying Framework: Part 1. International Journal of Computer Vision, 56(3), 221–255.
[19] Engel, J., Usenko, V., Cremers, D. (2016) A photometrically calibrated benchmark for monocular visual odometry. https://arxiv.org/abs/1607.02555.
[20] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W., Siegwart, R. (2016) The EuRoC micro aerial vehicle datasets. The International Journal of Robotics Research, 35(10): 1157–1163.
[21] Horn, B.K.P. (1987) Closed-form solution of absolute orientation using unit quaternions. Journal of The Optical Society of America A-optics Image Science and Vision, 4(4): 629–642.