Evaluation of SLAM-Based Monocular Camera Calibration for Real-Time Use

Jose A. Medina
Stanford University
Stanford, CA 94305
josem419@stanford.edu

Abstract

Computer vision and robotics researchers have long made use of SLAM as one of the main tools in mobile robotics. SLAM typically expects a well-calibrated robot in terms of the extrinsics and intrinsics of the sensors on the robot. However, these sensors may go out of calibration over time due to wear and tear or other unforeseen circumstances. In the field of autonomous vehicle research, re-calibration of these sensors through lab methods may not be possible. The use of SLAM as a calibration method has be shown to be viable.

Most mobile robotic platforms have a separate sensor which can provide direct state measurements such as an IMU, but this sensor could drift over time without correction. Expedient re-calibration would be necessary to avoid this. This work implements a monocular vision SLAM pipeline to evaluate how long a camera pose will take to converge to the ground truth pose.

1. Introduction

In autonomous platforms, cameras are a commonly leveraged sensor and usually it comes in pairs or sets of cameras on a platform. Calibration of this set of cameras is usually performed under a controlled environment and at the very least an a-priori extrinsic calibration is available from the as-designed model of the sensors relative to the platform. A well calibrated vision system on robotics platforms allows one to make use of many techniques to extract information an reason about the world around it.

This calibration may degrade over time and the severity of this degradation depends a lot on the platform. In aviation, for example, an increase in the use of composites has contributed to more flexible airframes making persistent extrinsic calibrations more challenging. Even factory calibrations on systems as they are installed can not be assumed to be valid for the life time of the vision system. To compensate for this, online calibration methods can allow for these platforms to recover from degraded situations and continue operation either unaffected or well enough until the next chance to perform a controlled calibration.

1.1. Related Work

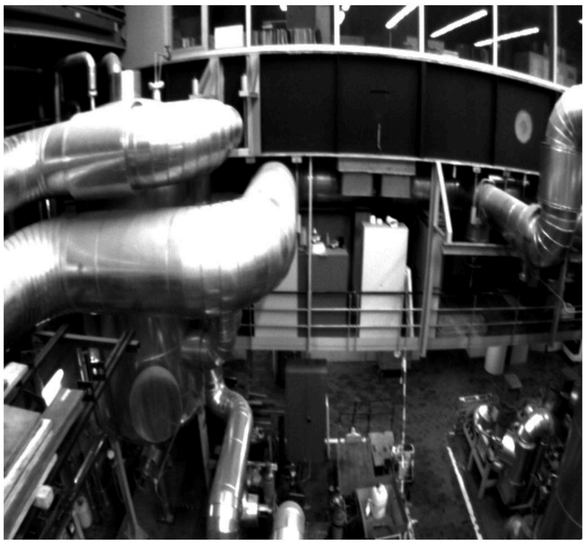
There exists work already related to online calibration, some explicitly leveraging SLAM such as in Carrera *et al.* [3]. This work discusses the use of SLAM and to calibrate an arbitrary set of cameras on a mobile robot. It assumes a-priori intrinsic calibration and it removes the dependence on a known calibration target for the extrinsics. The work proposes a method which performs a a pre-defined set of movements to estimate a relative extrinsic calibration between the set of cameras. It uses MonoSLAM [5], modified to use SURF features, to build a feature map for each camera and estimates the motion individually. The maps are refined independently and then correspondences are made between camera pairs. Then maps are aligned first with RANSAC [7] and then a single joint map is produced. This joint map is the starting point for bundle adjustment to estimate the camera poses.

The predefined movements are practical to reproduce on the platform that the paper used to prove the methods, but such movements would be less practical on aircraft or even some larger vehicles in other settings. The predefined movement such as a 360 may not be guaranteed.

Prior work to this, such as in Carrera *et al.* [4], also used SLAM based method but generalized the approach to also self-calibrate the intrinsic parameters of the cameras, further relaxing assumptions on the camera. The correspondence search in this work uses Bayesian propagation to reduce the risk of mismatches and reduce computation. The work relies on the base feature descriptor defined in MonoSLAM [5] opposed to the SURF feature descriptor used in by Carrera [3].

2. Problem Statement

This work proposes working on the auto-calibration by expanding both the work done by Carrera [3] and Civera [4] where improvements to the original monoSlam method and its feature matching is combined with the sequential


(a) Stereo Pair Example - Cam 0

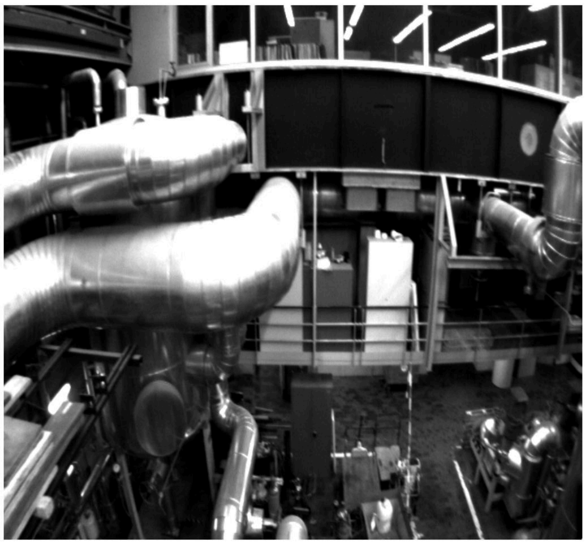

(b) Stereo Pair Example - Cam 1

Figure 1. Stereo Example

bayesian propagation of the state to calibrate extrinsic and intrinsic parameters at once. One important question posed by both of the aforementioned works was evaluation of the methods against degenerate camera motions where feature correspondences may be more sparse.

One desired outcome of this work is an evaluation of how quickly can a rig be re-estimated through the use of natural features. In the case of mobile robotics, there is usually an INS/IMU paired with the platform. The inertials of a the system may only be valid for a short period of time, usually less then 60 seconds, before they drift beyond what is considered useful. This real time re-estimation could be done incrementally until convergence. Similar evaluation is done in Civera *et al*. [4].

2.1. Datasets

To be able to do such an evaluation the dataset used needs to be, of course, a multi-camera platform with some independent means of providing ground truth data. This amount to having an intrinsic calibration for every camera as well as a relative extrinsic calibration for the rig. This calibration could have been estimated through some other more accurate or acceptable means in order to make this evaluation useful even if just as a comparison. This could be either calibration through a target and well-established method such as those described in Hartley and Zisserman [8] or other methods that leverage inertial measurement devices on the platform. Whatever the nature of the calibration so long as there is this ground truth to evaluate against.

One such dataset that meets this criteria are those provided by Burris *et al*. [1] titled ”The EuRoC micro aerial vehicle datasets”. These datasets are available either as rosbags which are easy to manipulate through the robotics framework ROS [12]. The same datasets are also available with individual frames extracted and sorted chronologically. Frames are available for a stereo pair as part of a visual-inertial sensor setup so there is also synchronized IMU measurements available. This dataset also has intrinsic and extrinsic calibration available for each camera which would act as the ground truth for the proposed evaluation.

The datasets are available in different grades of difficulty, where the motion of a the platform ranges from what is considered an ’easy’ motion for SLAM methods to a ”difficult”. There are multiple environments available in the dataset. A sample of only one environment is used, but at least one at each difficulty level is used for the evaluation.

An example stereo pair for this dataset can be found in Figure 1.

For the purpose of this work, all data will be treated as a monocular only dataset for processing. No stereo or visual inertial techniques were used in the implementation of the SLAM pipeline.

3. Technical Approach

The SLAM pipeline implemented borrows a lot of the basic framework from the original MonoSLAM [5] implementation with a few changes on some details. At a high level the implementation follows the flow of image ingestion - feature extraction/mapping - initial pose estimation -

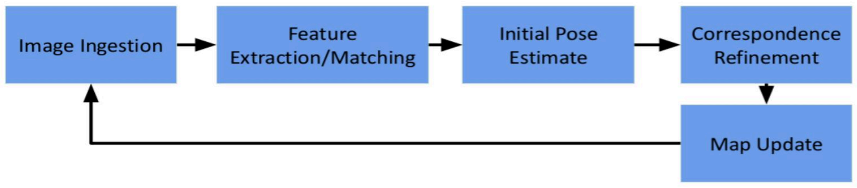


Figure 2. SLAM Implementation

correspondence refinement - map optimization. This diagram is shown in Figure 2

3.1. Image Ingestion

Image ingestion in the pipeline refers to any preprocessing of the input data prior to use. When running in an online pipeline, the ingestion can vary depending on the protocol associated with the cameras used. For experiments done with the aforementioned date, the images are organized through a csv and are read sequentially, providing the SLAM pipeline sufficient time to process each one without it being penalized because of dropped frames.

There was some work done on alignment of the data with IMU data, but was ultimately not used, as part of the experiments as the SLAM pipeline did not include any consideration for the inertials.

The images and associated camera information are then checked at the start of the pipeline. Intrinsic camera matrices and lens distortion coefficients are checked. The image is also transformed to a grayscale encoding in case it isn't already in grayscale. For the implementation, the camera's are assumed to follow a pinhole camera model for the intrinsic calibration parameters. The data chosen supports this assumption.

3.2. Feature Extraction and Mapping

Feature extraction and matching forms the first major step in the SLAM pipeline. In other works, known features have been a primary means of calibration and of establishing scale and anchoring maps in SLAM systems. These known features levy a sometimes unreasonable requirement on the environment which is generally not possible to influence.

Feature matching in this work, leverages "natural" features in which good features in the scene are found with no prior assumption on the scene structure. Two specific feature detectors are ORB [13] and Shi-Tomasi's corner detection method "Good Features to Track" [14].

ORB features were introduced as an alternative to other feature detectors such as SIFT [10], with the characteristic that they are rotation invariant and resistant to noise. The feature is represented as a vector of binary intensity tests performed on points distributed over a patch of the image. The distribution of the points is Gaussian per the original papers claim of it being the best performing overall. The image is also smooth prior to being processed for features. The output of this detector being a pixel location and a feature descriptor unique to that feature.

The Shi-Tomasi method for corner detection [14] was proposed earlier than the ORB feature detection and used princpled methods and measures for textured, dissimilarity and convergence to detect features that would make for good tracking by construction. Because the SLAM problem has a tracking component in it where the ownship and map state is tracked over time, this option appears to be a good choice.

The way in which the features get processed and their expected outputs is identical in this case which makes them interchangeable in the pipeline and allows for a comparison of performance between the two. The expected outputs are the set of all features descriptors and the accompanying pixel location of those features, referred to as keypoints.

To initialize a map for monocular SLAM a single image is not sufficient and the Starkey employed was to simply wait for a few sets of frames and features to be detected and matched prior to map initialization. As part of this initialization the problem of establishing correspondences between frames is necessary.The correspondences across frames are determined for every frame pair in the sequence, in practice it is always the correspondences between the current and previous frame. The matching itself is done by comparing the descriptor for every keypoint. More refined or resource conscience feature matching could be employed, but for this implementation a brute force feature matching was employed where all feature descriptors from the two compared frames are checked against one another. Matches are made through a K-nearest neighbors (KNN) match.

Feature matches are further cleaned up to exclude any incorrect or ambiguous matches. Lowe's ratio test [11] for keypoint matching is implemented. In simple terms, the two best key point matches of the KNN are taken and their distance compared. If the ratio of these distances is less then some threshold, meaning they are not significantly different, then the match is rejected. The matches/correspondences are stored and associated with the frames they came from.

The ratio test expression is reproduced below in (1)

$$\begin{aligned} dist_1 &< t * dist_2 \\ t &= threshold \end{aligned} \tag{1}$$

3.3. Initial Pose Estimate

Once the correspondences between the prior keyframe and current keyframe are settled, an initial camera pose is estimated using these correspondences. The prior camera pose is assumed to be fixed for this fitting, and if the prior pose was the first frames, it is considered to be the origin. This pose estimation is treated as a fitting problem, where the correspondences are used to get a fundamental matrix between the two frames. Once there is a fundamental matrix derived from the correspondences, a set of rotation and translation pairs can be determined and then chosen for the correct one.

The fundamental matrix is found using a classic normalized 8-point algorithm, as defined in Hartley and Zisserman [8].

The normalization of the points is simply to center (2) the centroid of the collection of points in each of the frames around zero and to rescale (3) the values themselves to be between -2 and 2 pixels. This normalization, is done to explicitly correct for ill-conditioned matrices that are commonly encountered in the original 8-point algorithm. The normalization is done for each frame's group of points, individually.

$$\begin{aligned} \boldsymbol{\mu} &= \frac{\sum_i^n \boldsymbol{p_i}}{n} \\ \boldsymbol{p_i}' &= \boldsymbol{p_i} - \boldsymbol{\mu} \end{aligned} \tag{2}$$

$$\begin{aligned} scale = s &= \sqrt{\frac{2n}{\sum_i^n (||p_i'||^2)}} \\ \boldsymbol{T} &= \begin{bmatrix} s & 0 & -\mu_x s \\ 0 & s & -\mu_y s \\ 0 & 0 & -\mu_z \end{bmatrix} \\ \boldsymbol{p_i}'' &= \boldsymbol{T}\boldsymbol{p_i} \end{aligned} \tag{3}$$

After this normalization the 8-point algorithm, is executed as normally, by a singular value decomposition to find the null space, and then a forcing of the rank-2 constraint on the fundamental matrix and an additional singular valu decomposition. The fundamental matrix that is dereived at this point, is de-normalized to the correct scale (4).

$$\boldsymbol{F} = \boldsymbol{T_2^T} \boldsymbol{F} \boldsymbol{T_1} \tag{4}$$

This method of fitting of a fundamental matrix is used in conjunction with RANSAC [7]. Not all correspondences may agree on the best fit of the fundamental matrix and therefore it makes for a more robust solution to employ some amount of model fitting, in this case the use of RANSAC is employed. The algorithm serves as the model to fit against.

The best fit fundamental matrix is used to produce a rotation and translation between the prior and current frame.

3.4. Correspondence Refinement

Of all the correspondences found previously, they are from points which are only a subset of all the detected features in the current image. It is possible many defections made which failed to matches purely through the descriptor or because those features did not have a correspondence in the frame immediately previous to it. For this case, now that there is an estimated transfer between the new pose and the prior pose, and by extension all prior poses, any remaining unmatched point will attempt to be matched with existing points in the map through triangulation.

Any new points that can be triangulated are added as observations to each of their respective frames.

3.5. Map Optimization

The final step in this SLAM pipeline is the map optimization. The map optimization aims to refine the entire pose graph of every camera frame up until the current frame. It does so by adjusting camera poses to minimize some defined loss function.

This leverages work done on graph based optimization to set up the solver. The solver used is a standard non-linear Levenberg-Marquardt . The loss function used is a robust Huber Loss, though a different one could have been used. [9]

4. Experiments

The SLAM pipeline detailed in the previous section was evaluated against the dataset as defined. The first evaluation that was done was to see map quality and a qualitative result of the continuity of the camera trajectory and poses. The initial desire was to evaluate the pipeline's performance using ORB features. A sample taken from the dataset can be seen in Figure3. The picture depicts the ORB features as green circles wherever a feature could be detected.

It can be seen from this picture that the amount of detected features are quite small. As a result this also affects, the number of matches that can be made across frames, results of this will be shown in the following discussion.

The scarcity of features is troublesome because of their effect on map quality. The corresponding map corresponding to figure 3 can be seen in figure 4. The resulting ma has

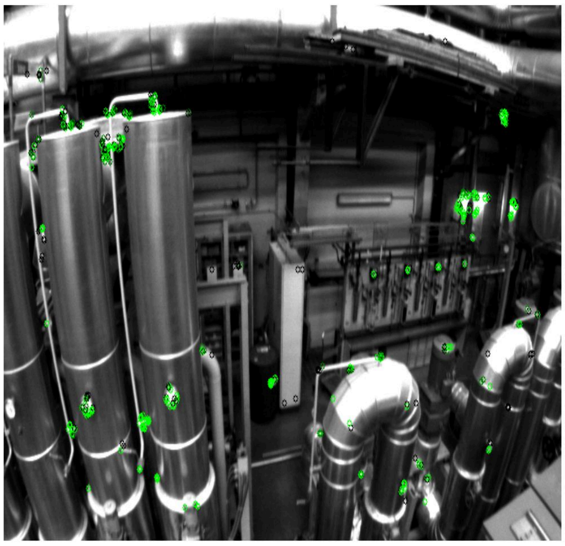


Figure 3. Example of the ORB features detected within the camera frame.

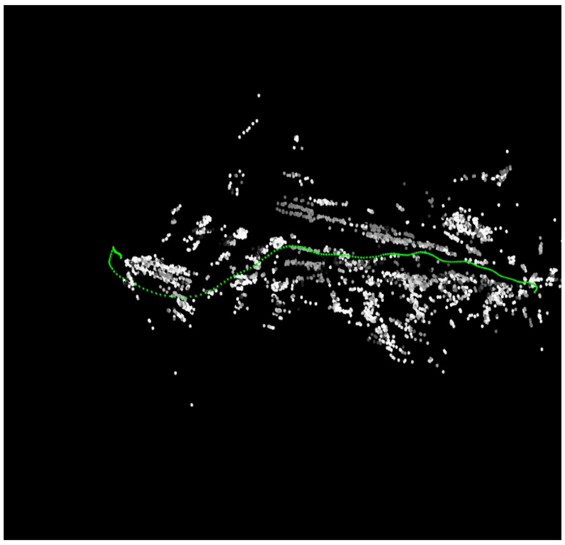


Figure 4. Example of map made through the use of ORB feature correspondences

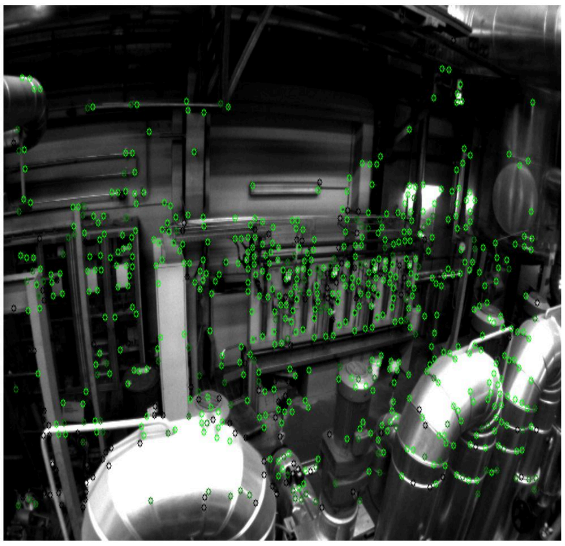


Figure 5. Example of the Shi-Tomasi features detected within the camera frame.

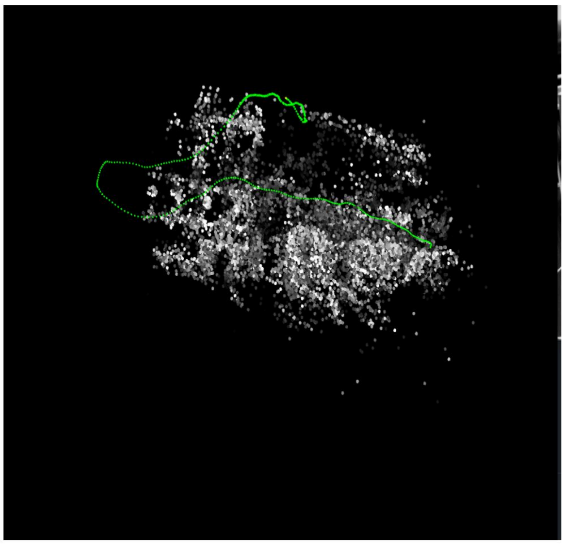


Figure 6. Example of map made through the use of Shi-Tomasi feature correspondences

a fairly continuous trajectory shown by the winding curve of green rectangles. These rectangles are the camera frames and their respective poses.

The difference in density and quality of the map is much more apparent when compared against another feature detector. The same pipeline and segment of the dataset was run using the Shi-Tomasi method. The same two graphics are presented in figures 5 and 6

Its immediately clear that the number of features that could be detected were much greater for the Shi-Tomasi feature detector method. This has cascading effects and was a major cause for performance and map quality in this implementation. The great the number of features, then there was more opportunity to make correspondences between

Method	FPS (Hz)	# Points (at 400 frames)
Our Method (Shi-Tomasi)	0.75	13000
Our Method (ORB)	2	8000
ORB-SLAM3	20	N/A

Table 1. Framerate Comparison

the current and prior frame either through matching feature descriptors or through point triangulation. The lower number of features may be sufficient on there own if they were all of sufficient quality, but after the multiple stages of point filtering to be left with only good inlier points, it was very rare that more than a handful of points were available to add to the solver. This also made the initial pose estimation much poorer and more likely to outright fail as RANSAC could not find good solutions with such few points to search against.

In addition to map density, the consistency of the SLAM pipeline when limited by such few point correspondences as in the case of the ORB detector is very low. The SLAM pipeline results when running in the ORB detector produces inconsistent maps and trajectory on the same dataset from run to run. This is true regardless of where the pipeline starts and stops in the dataset. The run shown in 4 is one of the more stable runs on the same dataset using this feature detector. Most runs would result in maps and trajectories that are visibly broken up and not continuous. Camera poses were also observed to rotate and translate to positions nowhere near the prior. This can be seen in figure 7 where there are 4 distinct trajectories that can be made out as opposed to a single one. The maps points are also incoherent due to bad pose estimates of the keyframes orienting the cameras in strange ways.

For the purpose of real-time calibration, the system as implemented has clear quality limitations which would preclude it from being considered complete for the purposes of calibration. However, the time it took to create the two good maps shown in figures 4 and 6 is compared to a state of the art system such as ORB-SLAM3 [2] in table 1. ORB-SLAM3 can be run on the same dataset under various configurations, but was tested against a monocular SLAM configuration to be most comparable.

From the table it can be seen that, despite the Shi-Tomasi based pipeline, it was the slowest to run through the entire pipeline, slowing down the most at the map optimization step. This is correlated to the number of points which got included in the optimization. However there is nothing special about the points derived from Shi-Tomasi features vs ORB

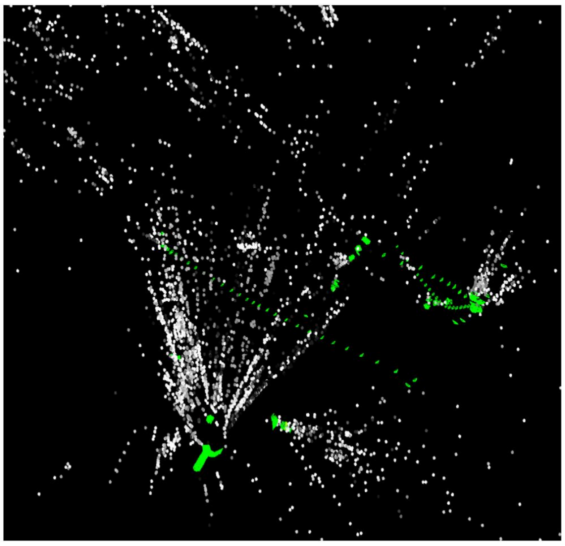


Figure 7. Example of dis-continuous map and trajectory.

features. They are both equivalent in terms of processing the map, the Shi-Tomasi method just happens to produce many more point correspondences which gives more points for the map optimization to consider.

This speaks more to the limitation of the implementation which are discussed further in the conclusions sections. It can be seen that ORB-SLAM3 has no issue with processing the dataset at its full frame rate in this case 20 Hz. The number of points is irrelevant to the comparison with ORB-SLAM3 so it was not included in the table.

5. Conclusions

In this paper, a real-time SLAM implementation was attempted for the purpose of camera calibration. However, many issues with the implementation and quality of the SLAM solution. limited any meaningful contribution to existing work on monocular camera calibration through SLAM.

There are number of proposed follow-on work to this to reach a level where an advancement of the problem could be made. A primary set backs of the implementation is the way the optimizer itself was implemented and the language the pipeline was written in. For true online performance, python as a language is unsuitable but it helps bridge the gap between prototyping and deployment.

The implementation of the optimizer was highly inefficient and could be improved by allowed the bundle adjustment to occur incrementally instead of resolving the entire pose graph at every update.

At many points during the dataset, the pipeline had opportunities to correct itself by detecting loop closure, however there was no loop closure capability implemented which hindered the performance.

For the explicit purpose of camera calibration using SLAM, there is little reason to limit the solution to monocular visual SLAM pipeline alone. On most mobile robotics platforms, it is very reasonable assumption that there would be some inertial measurement system available to be able to leverage in a visual-inertial SLAM pipeline. This would greatly improve the pose estimates, and the stability of the tracks compared to what was seen in the results when correspondences and feature matching performs poorly. One result not explicitly discussed what the map nd track quality during degenerate motions such a purely forward translation or looming motion. This was tested using the aforementioned dataset, and track discontinuities or poor estimates in could be correlated with looming camera motions as there was little parallax from frame to frame.

For visual-inertial SLAM, there are other tools available for pose graph optimization which make use of modern factor graph frameworks such as GTSAM [6]. A visual-inertial SLAM pipeline implementation was started as part of this work but not pursued further because of convergence issues with the optimizer not being resolved in time for the writing of this work.

6. Supplemental Material

Code: https://github.com/Josem419/monoslam_cal

References

[1] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. 2

[2] Carlos Campos, Richard Elvira, Juan J. Gomez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 6

[3] Gerardo Carrera, Adrien Angeli, and Andrew J. Davison. Slam-based automatic extrinsic calibration of a multi-camera rig. In *2011 IEEE International Conference on Robotics and Automation*, pages 2652–2659, 2011. 1

[4] Javier Civera, Diana R. Bueno, Andrew J. Davison, and J. M. M. Montiel. Camera self-calibration for sequential bayesian structure from motion. In *2009 IEEE International Conference on Robotics and Automation*, pages 403–408, 2009. 1, 2

[5] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. 1, 2

[6] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022. 7

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 726–740. Morgan Kaufmann, San Francisco (CA), 1987. 1, 4

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. 2, 4

[9] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. 4

[10] Tony Lindeberg. *Scale Invariant Feature Transform*, volume 7. 05 2012. 3

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 3

[12] "Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng". "ros: an open-source robot operating system". In *"Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics"*, "Kobe, Japan", May 2009. 2

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. 3

[14] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 3