Multi-Agent Monocular SLAM

Pieter Beerten[1], Charles Hamesse[1,2] and Rob Haelterman[1]
[1]*Department of Mathematics, Royal Military Academy, Belgium*
[2]*imec - IPI - URC, Ghent University, Belgium*

Keywords: SLAM, Bundle Adjustment, Multi-Agent, Monocular Vision, Optimization.

Abstract: This article describes the development of an optimization method for multi-agent monocular SLAM systems. These systems allow autonomous robots to create a map of an unknown environment and to simultaneously localize themselves within it. The proposed multi-agent system combines measurements made by independent agents to increase the accuracy of the estimated poses of the agents and the created map. Our method is based on the single-agent monocular ORB-SLAM2 framework, and we develop a complete multi-agent optimization post-processing algorithm, effectively refining all camera trajectories and map points. Our experiments on the EuRoC machine hall dataset show that we can successfully combine the information of multiple SLAM agents to increase the accuracy of the estimated trajectories.

1 INTRODUCTION

Autonomous robots are more popular today than ever in our modern society. To name a few, the fields of automotive, civil engineering, security, and leisure all reveal a dynamic landscape of innovation where autonomous robots play an important role. The progress in 3D perception hardware allows robots to map, understand, and navigate within various types of environments. They have the ability to perform tasks that were typically tedious, dangerous, or complex for humans, such as surveying construction sites, monitoring the health of agriculture fields, or moving boxes and packages. In all of these applications, the robot needs a way to know, at any given time, where it is with respect to its surroundings, that is, to localize itself. For these robots to be able to do so and safely navigate through unknown territories, different algorithms have been developed over the course of the last decades. One very popular method for this is Simultaneous Localization And Mapping, or SLAM in short. This technique allows a robotic agent to determine its location within an unknown environment based on the input it captures via one or multiple onboard sensors. Simultaneously, the locations of distinct points in the environment are estimated, thus creating a map of the scene.

The goal of this work is to develop an optimization pipeline that refines the accuracy of the pose estimates and the created map, based on the application of bundle adjustment to a combination of measurements of multiple agents. To this end, ORB-SLAM2 (Mur-Artal et al., 2017) is used as a baseline to obtain the initial SLAM measurements. The popular *EuRoC* dataset (Burri et al., 2016) is used to test and validate the proposed algorithm, rather than relying on our own measurements.

The article is structured as follows: first, in Section 2, a theoretical introduction to the concept of SLAM and the considered optimization method will be provided. Next, in Section 3 the single-agent bundle adjustment pipeline will be discussed. This will serve as a building block to finally arrive at the multi-agent bundle adjustment pipeline. In Section 4, the results for both bundle adjustment algorithms will be discussed. This article concludes with a general conclusion of the results and some suggestions for further research.

2 RELATED WORK

We start with a broad overview of SLAM, then dive deeper into bundle adjustment, the optimization technique on which our multi-agent optimization pipeline relies.

Beerten, P., Hamesse, C. and Haelterman, R.
Multi-Agent Monocular SLAM.
DOI: 10.5220/0012320000003636

In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 1*, pages 213-220
ISBN: 978-989-758-680-4; ISSN: 2184-433X

2.1 SLAM

SLAM algorithms are used to estimate the pose (position + orientation) of an agent in an (unknown) environment, while simultaneously building a map of this environment. This allows autonomous agents to navigate safely through the environment and to create a map that can be used for various applications. To this end, SLAM algorithms use data provided by onboard sensors, such as visual sensors, GPS, lidar, or radar. In this work, a single monochrome camera (in the visual part of the spectrum) is used to provide the necessary data.

One of the main issues at hand for SLAM algorithms is the dual nature of the problem they are addressing. Indeed, to create a map based on data captured by an agent, the complete pose of the agent should be known. However, to know the exact pose of the agent, an accurate map is needed. Consequently, errors in the pose will inevitably lead to errors in the map, and vice versa. This can easily lead to accumulating or drifting errors. An illustration of this accumulating effect is given in Figure 1. To address this issue, map refinement techniques, such as pose graph optimization, can be used to significantly reduce the drifting error. This is indicated by the white arrow. However, in this work, bundle adjustment is chosen as the optimization technique to refine the estimates and reduce the error.

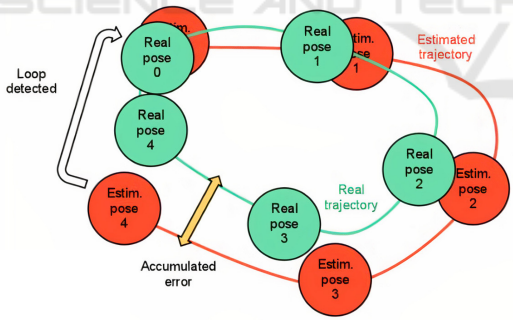


Figure 1: Illustration of the accumulating error. (Wong, 2014).

As said, our work relies on monocular visual SLAM algorithms. In general, these algorithms can be broken down into five major parts, as represented schematically in Figure 2. The work carried out for this paper can be situated in the "Optimization" part. The visual variant of SLAM has been a subject of extensive research and development over the years. The inception of monocular SLAM can be traced back to 2007 when Davison introduced MonoSLAM (Davison et al., 2007). This algorithm utilized an extended Kalman filter (EKF) to achieve real-time localization by leveraging large image patch features and sparse prior scene knowledge. Although Kalman filter-based odometry systems are relatively simple and elegant, optimization-based methods have been introduced to increase accuracy. Odometry systems based on bundle adjustment optimization have been adopted with great success, albeit at the cost of increased computational complexity (an increase that can be controlled with a smart use of keyframe-based and sliding window optimization). VINS-Mono (Qin et al., 2018) is a robust visual-inertial odometry estimation system that maintains recent states of map points and cameras in a sliding window, optimized with local bundle adjustment. Visual features are tracked using KLT and optical flow. ORB-SLAM (Campos et al., 2021) uses ORB features and a descriptor-based approach for matching, which is more expensive computationally but can offer better long-term data association performance.

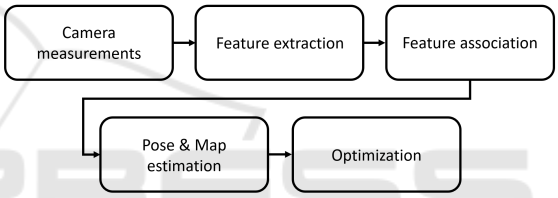


Figure 2: Key components of a visual SLAM algorithm.

2.2 Bundle Adjustment

Bundle adjustment, often abbreviated as BA, can be considered a general optimization technique for the estimated trajectory and map. In robotics terms, it is a state estimation technique used to jointly refine the estimates of the 3D location of the observed points and the camera poses. This refinement is performed by minimizing the reprojection error, which expresses the difference between the projection of the estimated 3D map points in the camera image and the actual location in the image where the point is observed, assuming that the exact pose of the cameras and the coordinates of the points in the environment are known. By combining the data captured in several consecutive frames, the estimates of the camera poses and the World Coordinates of the observed features can be simultaneously optimized. Mathematically, the expression of the initial situation for a bundle adjustment problem is given in (1):

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \mathbf{K} \cdot \mathbf{R} \cdot \begin{pmatrix} \mathbf{I}_{3\times 3} \mid \mathbf{t} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{X}_W \\ 1 \end{pmatrix} \quad (1)$$

The 3×1 column vector $(u \ v \ w)^T$ at the left-hand side of this equation represents the three-dimensional pixel coordinates of an observed point. Note that the third component is added to express the vector in homogeneous coordinates. On the right-hand side, the 3×3 matrix $\mathbf{K}$ is the intrinsic matrix of the camera, the 3×3 matrix $\mathbf{R}$ is the rotational matrix, and the 3×1 column vector $\mathbf{t}$ represents the translation of the camera center. $\mathbf{I}_{3\times3}$ represents the identity matrix of dimension 3. The final 4×1 column vector $(\mathbf{X}_W \ 1)^T$ represents the observed 3D point in homogeneous coordinates in the World Coordinate System.

Moreover, the residual $\mathbf{r}$ can be expressed as:

$$\mathbf{r} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \frac{u}{w} \\ \\ \frac{v}{w} \end{pmatrix} \tag{2}$$

where $(x \ y)^T$ represents the observed pixel coordinates of a feature. These coordinates will be provided as input to the BA algorithm. The second vector $\left(\frac{u}{w} \ \frac{v}{w}\right)^T$ is calculated from the estimated homogeneous coordinates of the point in the three-dimensional WCS and represents the reprojection of an observed point in the image plane. By changing the values of $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$, these reprojected coordinates will vary accordingly.
In essence, a bundle adjustment algorithm will perform a nonlinear least squares (NLLS) optimization in order to minimize the sum of the squared residuals $\mathbf{r}$ over all considered points and camera poses:

$$\min_{\mathbf{C},\mathbf{X}} \sum || \ \mathbf{u}_{ij} - \pi(C_j, X_i) \ ||^2 \tag{3}$$

In (3), $\mathbf{u}_{ij}$ represents the observations in pixel coordinates. This is inherently equal to $(x \ y)^T$ defined in (2). $\pi(C_j, X_i)$ represents the projection of a point X_i into the camera plane of camera C_j. Hence, this term corresponds to $\left(\frac{u}{w} \ \frac{v}{w}\right)^T$ defined in (2). Note that this is a nonlinear operation.

3 METHOD

As a first building block, a pipeline to perform bundle adjustment based on the measurements made by a single agent is proposed. This is referred to as Single-Agent Bundle Adjustment (SABA). We start with a description of this algorithm, before moving on to the Multi-Agent Bundle Adjustment (MABA) system.

3.1 Single-Agent Bundle Adjustment

As illustrated by (3), the first step is to reproject the observed features in the camera plane. This can be easily achieved by multiplying the estimated World Coordinates of the points with the camera pose matrix $\mathbf{T} \in SE(3)$, then with the camera intrinsic matrix to obtain the pixel coordinates that are ultimately used to calculate the residual.

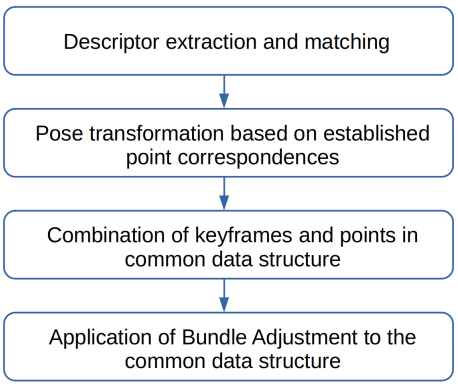


Figure 3: MABA algorithm pipeline.

Once all features are transformed to the Pixel Coordinate System (PCS), the actual bundle adjustment is carried out using the *Ceres* library. This library allows for the solution of NLLS problems using various solution methods, such as the Gauss-Newton method or the Levenberg-Marquardt algorithm. Optionally, the resulting camera trajectory can be smoothed to reduce the effect of outliers, caused by an incorrect pose estimation. This will result in a smaller Absolute Pose Error (APE) and Relative Pose Error (RPE) between the estimated trajectory and the ground truth. When calculating the optimal camera poses and point locations in the map, the algorithm can be run in two different modes. The most intuitive approach is to run the algorithm in the so-called 'unbounded' mode, which imposes no additional constraints on the refined parameters. While being the least constrained problem, it might result in some camera positions being heavily displaced in order to minimize the global cost function. To avoid this effect, which has a negative impact on the APE and RPE after alignment with the ground truth, the algorithm can be run in the 'bounded' mode. Now, the displacement of the camera poses and the point coordinates is limited to avoid huge outliers. In any of the three principal directions, the maximal displacement of the camera positions is clamped to $[-\delta, \delta]$, where $\delta \in \mathbb{R}^+$ is a user-input parameter. Camera rotations and 3D point coordinates are clamped in a similar manner.

While imposing additional constraints to the optimization problem, this method tends to avoid having great outliers, hence improving the alignment of the estimated trajectory with the ground truth.

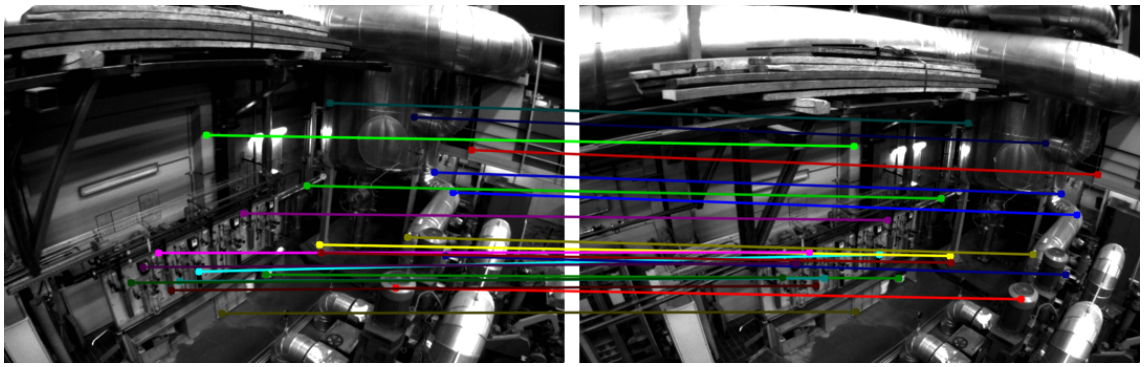


Figure 4: Illustration of point correspondences between images captured by different sequences. The colored lines connect the matching features in the images.

3.2 Multi-Agent Bundle Adjustment

The goal of this algorithm is to combine the data from multiple sequences such that the estimates of the camera poses and the coordinates of the points are refined jointly for these sequences. By relying on measurements obtained with multiple agents, the uncertainty of the estimates can be reduced, which should lead to more accurate results.

A key component of our pipeline is the Scaled Point Cloud Registration (SPCR) system that we use to make the outputs of each agent's SLAM system coherent. In fact, monocular SLAM suffers from scale ambiguity. Moreover, the different sequences do not share the exact same initial pose. Therefore, we need to apply a Similarity transform ($Sim(3)$: an $SE(3)$ transformation with an extra degree of freedom for scale) to the poses and map points to make the monocular SLAM outputs of different agents coherent.

In Figure 3, the pipeline of the MABA algorithm is presented. The first step will be to match the descriptors of the points of the different sequences based on a multi-dimensional distance metric. Once the point correspondences are established correctly, the SPCR is performed between the matching points in order to estimate the rigid transformation parameters that map the points observed by one sequence to the reference frame of the other.

3.2.1 Descriptor Extraction and Matching

Every point considered by ORB-SLAM2 is given a 32-element descriptor vector. These vectors can be used to establish point correspondences by calculating the minimum distance between the points seen in different sequences. As ORB uses binary descriptors, the Hamming distance is considered as a metric in this article.

To increase the robustness of the point matching, Lowe's ratio test (Lowe, 2004) is implemented, which discards ambiguous matches. In a practical application, the distance between the closest and the second-closest neighbor for a certain query descriptor is often very small. Consequently, incorrect matches often occur whenever there is noise in the considered images. The idea behind this ratio test is that a match is only accepted if the ratio of the distance to the closest and second closest neighbor is smaller than a certain threshold:

$$\text{Accept if } r = \frac{d_{NN}}{d_{SNN}} < \text{threshold} \tag{4}$$

where d_{NN} represents the minimal (Hamming) distance (*Nearest Neighbor*) between a query descriptor and all descriptors against which it is matched. d_{SNN} represents the second lowest distance (*Second Nearest Neighbor*). If this ratio is smaller than the acceptance threshold, one can assume that the matching has overcome the effects of noise or other sources of inconsistencies in the calculation of the descriptors.

An example of point correspondences between two images captured by different sequences is provided in Figure 4.

3.2.2 Pose Transformation Based on Established Point Correspondences

We describe the Scaled Point Cloud Registration (SPCR) system that we use to make the outputs of each agent's SLAM system coherent. Since the camera poses and the estimated point positions for each agent are expressed relative to their first camera pose, considered as the origin, the different sequences are often expressed in totally different coordinate frames, making it difficult to compare the poses or the points in an absolute way. To overcome this issue, the rigid pose transformation parameters are estimated from a set of matching points. By definition, these matching points should have the exact same coordinates when expressed in a common coordinate frame. This

knowledge can be used to estimate a $Sim(3)$ transform, i.e., 3×3 rotation matrix $\mathbf{R}$, a 3×1 translation vector $\mathbf{t}$ and a 3×1 scale vector $\mathbf{s}$ that map the points from one coordinate system to the other:

$$\begin{pmatrix} \mathbf{X}' \\ 1 \end{pmatrix} = \mathrm{diag}([\mathbf{s}, 1]) \cdot \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \quad (5)$$

where $\mathbf{X}$ and $\mathbf{X}'$ represent the point coordinates in the original reference frame and the transformed coordinates, respectively.

The results of the SPCR algorithm for the point correspondences visualized in Figure 4 are presented in Figure 5. There is indeed an apparent initial scale difference between the estimated point coordinates for the two sequences. However, after the transformation, both sets of points coincide almost perfectly. In this figure, the source points represent the points that should be transformed to a different coordinate system. The target points are the points from the other sequence with which the source points match.

Finally, the estimated transformation can be applied to the camera poses as well in order to transform them to a common reference frame.

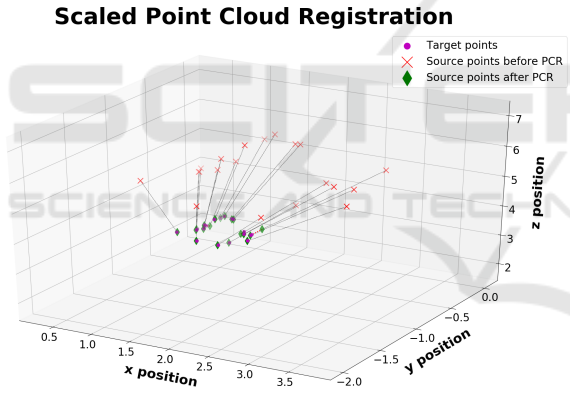


Figure 5: Illustration of the SPCR algorithm.

3.2.3 Combination of Keyframes and Points in a Common Data Structure

Once all data is transformed to the same reference system, a common data structure can be created for the different sequences. This way, the multi-agent problem is translated to one larger optimization problem, that involves the data of multiple sequences. The observations of matched points are combined, while unique points are individually added to the optimization problem. An illustration of the combined trajectory of sequences MH01 and MH02 is given in Figure 6. Notice the similarity in scale between the two sequences, enforced by the SPCR algorithm.

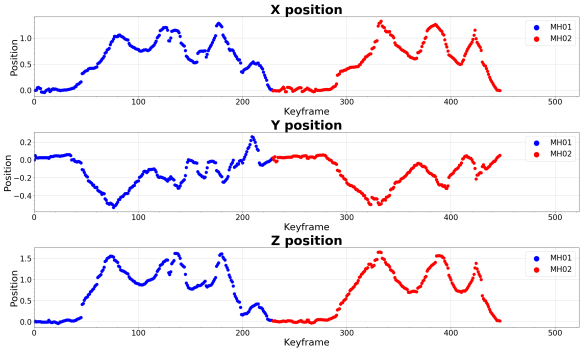


Figure 6: Resulting combined trajectory after SPCR.

4 EXPERIMENTS AND DISCUSSION

As explained in the introduction, ORB-SLAM2 is used to obtain the initial estimates of the camera poses and the locations of the points in the created map for each agent. Our system is evaluated on the EuRoC dataset (Burri et al., 2016). Since ORB-SLAM2 results do not typically exhibit significant drifting errors (longer data sequences would be needed to better observe accumulating errors), our proposed system will be validated through the deliberate addition of noise.

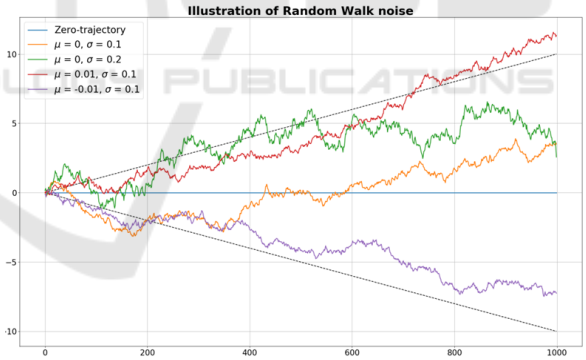


Figure 7: Illustration of the random walk noise.

Two different noise models are used to distort the original estimates. First, a zero-mean Gaussian noise is applied to the data, for which the effects of the standard deviation of the applied noise on the obtained results are analyzed. In a second set of experiments, a random walk noise is applied to the camera positions. This simulates the effect of an accumulating error, which results in a more realistic noise. This effect is illustrated by the red and purple curves in Figure 7.

In Table 1, an overview of the different scenarios is given.

Table 1: Overview of the considered scenarios.

Scenario	Noise type	BA type	Discussed noise parameter	Affected quantity
Scenario 1	Gaussian	Motion-only	σ	Camera positions
Scenario 2	Gaussian	Motion-only	σ	Point coordinates
Scenario 3	Gaussian	Full	σ	Camera positions Point coordinates
Scenario 4	Random Walk	Motion-only	μ σ	Camera positions
Scenario 5	Random Walk	Motion-only	σ	Point coordinates
Scenario 6	Random Walk	Full	σ	Point coordinates

4.1 Results for the SABA Algorithm

In Figure 8, the results of a sensitivity analysis of the unbounded SABA algorithm are given for scenario 1. Note that for this scenario, the algorithm is run in the motion-only mode, indicating that only the camera poses are refined. Especially for sequences MH01 and MH02, it is clear that the algorithm does not perform accurately for values of $\sigma > 0.15$. In these cases, the APE after optimization is larger than before, due to the presence of extreme outliers in the optimized camera trajectory. By applying the smoothing filter, which reduces the effect of these outliers, the APE is indeed considerably smaller.

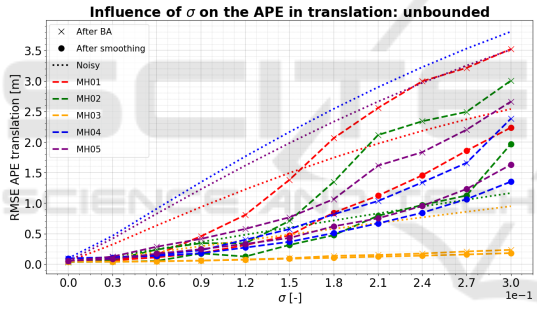


Figure 8: Results of the unbounded SABA algorithm for scenario 1.

The estimates can be further improved by applying the bounded SABA algorithm, as shown in Figure 9. In all of our experiments, the bound parameter is set to $\delta = 0.3$. Now, even for the largest value of σ, the APE after optimization is significantly smaller than the APE of the noisy trajectory, which suggests the refined trajectory is closer to the ground truth after alignment.

4.2 Results for the MABA Algorithm

Finally, the bundle adjustment algorithm can be applied to the common data structures as explained in Section 3.2.3. To guarantee a maximal overlap between the sequences, a combination of sequences MH01 and MH02, and sequences MH04 and MH05 has been chosen in this article. For each of these, the

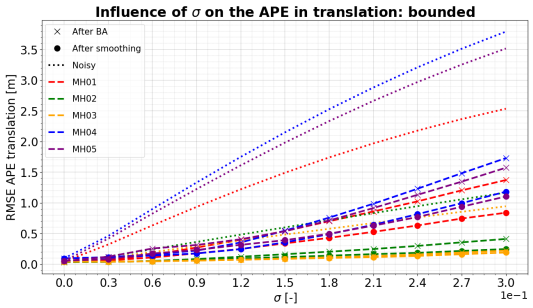


Figure 9: Results of the bounded SABA algorithm for scenario 1.

coordinate system and scale of the first sequence, respectively MH01 and MH04, have been chosen as a reference.

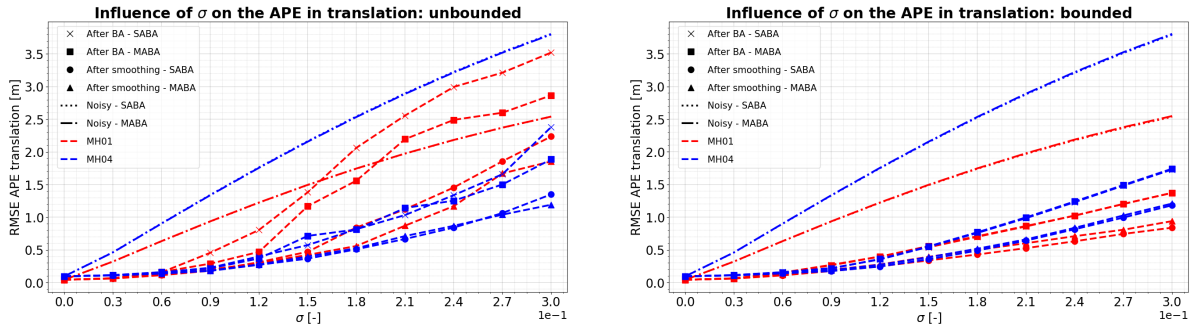
A comparison of the performance of the SABA and MABA algorithm for scenario 1 is visualized in Figure 10. There, it is clear that the resulting APE is lower for the unbounded MABA algorithm than for the single-agent variant for the same value of σ, indicating that the extension to multiple agents indeed has a positive effect on the optimized trajectory. However, for the bounded algorithm, the difference between SABA and MABA is negligible. This is due to the higher level of limitation of the mathematical problem, which reduces the potential to increase the accuracy of the estimates by applying an optimization algorithm.

For scenario 3, which considers the 'full' bundle adjustment algorithm, results tend to show a trend similar to scenario 2. This is visualized in Figure 11. Now, the improvements are especially apparent when the unbounded algorithm is applied to sequence MH01, at values of $\sigma > 0.09$. For the bounded algorithm, the difference is again negligible.

Notice that while the results obtained with the MABA algorithm are generally better than for the SABA algorithm, the computation time is significantly ($\sim 10\times$) larger for the MABA algorithm, computing the optimized poses and points for all agents simultaneously.

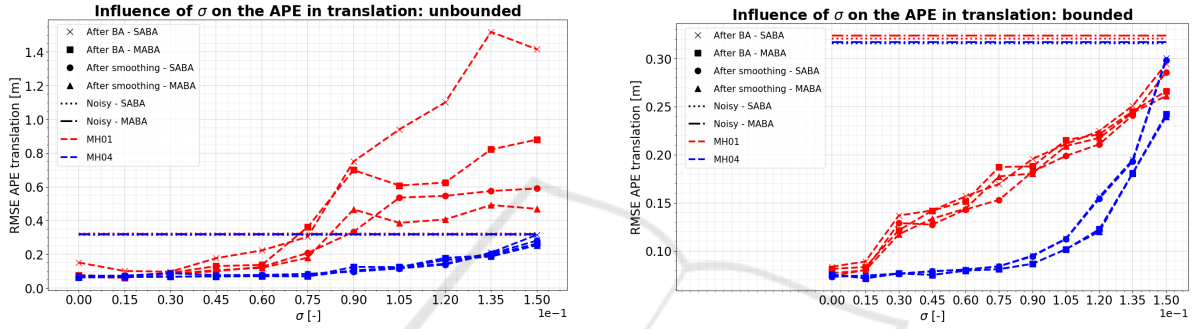
5 CONCLUSION

This article discusses a bundle adjustment pipeline to improve the estimated camera poses and locations of points in the map, either based on measurements provided by a single agent or by combining the measurements of multiple agents. For a setup involving monocular visual SLAM, it is shown that the accuracy of the estimates, after the deliberate addition of noise,



(a) (b)

Figure 10: Scenario 1: Comparison of the results of the SABA and MABA algorithm for the APE in translation.



(a) (b)

Figure 11: Scenario 3: Comparison of the results of the SABA and MABA algorithm for the APE in translation.

improves when the data of multiple agents are considered. However, a trade-off between the improved accuracy and an increase in computation time should always be considered.

6 FUTURE WORK

In this article, a multi-agent bundle adjustment pipeline has been proposed for monocular visual SLAM systems. In further research, this could be extended to integrate the data of other sensors, such as GPS, IMU, or lidar, to improve the accuracy of the estimates. Moreover, an analysis of the scalability of the algorithm could be performed. As explained, the computation time increases with a factor 10 when the algorithm is expanded to consider two independent sequences. Although the extension to additional agents should be straightforward, computation time might become an issue when too many agents are considered. Therefore, future research could focus on speed improvements for descriptor matching and the solution of the nonlinear least-squares optimization problem. A final suggestion would be to perform an analysis of the parameters of the *Ceres* solver. Fine-tuning these parameters, such as the type of linear solver, the step size of the Levenberg-Marquardt solver, or the type of loss function, could significantly impact the results. Conducting a comprehensive sensitivity analysis of these parameters across diverse datasets would therefore be valuable to determine the best settings for different scenarios.

REFERENCES

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*.

Campos, C., Elvira, R., Gomez, J. J., Montiel, J. M. M., and Tardos, J. D. (2021). ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890.

Davison, A. J., Reid, I. D., Molton, N., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1052–1067.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Mur-Artal, J., Montiel, R., and Tardós, J. (2017). Orb-slam2: an open-source slam system for monocular,

stereo and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

Wong, W. (2014). Autonomous 3d mapping and exploration using a micro aerial vehicle. pages 138–142.