Constrained-covisibility marginalization for efficient on-board stereo SLAM

Matías A. Nitsche[1] and Gastón I. Castro[1] and Taihú Pire[2] and Thomas Fischer[1] and Pablo De Cristóforis[1]

***Abstract*— When targeting embedded applications such as on-board visual localization for small Unmanned Air Vehicles (UAV), available hardware generally becomes a limiting factor. For this reason, the usual strategy is to rely on pure motion integration and/or restricting the size of the map, i.e. performing visual odometry. Moreover, if monocular vision is employed, due to the additional computational cost of stereo vision, this requires dealing with the problem of unknown scale.**

In this work we discuss how the cost of the tracking task can be reduced without limiting the size of the global map. To do so, the notion of covisibility is strongly used which allows choosing a fixed and optimal set of points to be tracked. Moreover, this work delves into the concept of parallel tracking and mapping and presents some finer parallelization opportunities.

Finally, we show how these strategies improve the computational times of a stereo visual SLAM framework called S-PTAM running on-board an embedded computer, close to camera frame-rates and with negligible precision loss.

I. Introduction

With the growing interest in UAVs, there has been an increasing need for localization methods capable of operating on-board and in real-time. Designing systems that provide accurate pose estimation in challenging environment while running on platforms with limited computational resources is thus a key problem in mobile robotics. For this reason, in GPS-denied scenarios such as indoors or outdoors in areas with poor reception, vision-based approaches have been widely used.

However, efficient vision-based localization solutions generally still require considerable computational power. This is particularly difficult when targeting low-payload robots, where only small and low-resource processing hardware can be employed. It is thus worth considering new strategies for reducing computational requirements of vision-based localization methods and allow meeting real-time constraints.

From a methodological point of view, vision-based localization methods can be classified as visual odometry (VO) or Simultaneous Localization and Mapping (SLAM) approaches. VO techniques focus on ego-motion integration to get a camera pose estimate, while SLAM approaches build a global map against which the robot can localize. One of the main drawbacks of VO approaches is that accumulated pose drift is never corrected due to the absence of global map information. In contrast SLAM approaches are able to localize against the map without requiring motion integration.

For the case of SLAM (Simultaneous Localization and Mapping), one widely adopted strategy was one proposed with PTAM [1] (Parallel Tracking and Mapping), where both tracking and mapping tasks are decoupled as separate computing threads. Many recent feature-based visual SLAM approaches embraced this approach and added also a loop-closing thread [2], [3].

Both tracking and mapping tasks have costly operations (such as point-matching and Bundle-Adjustment) which are largely dependent on the number of map-points. Since the map can grow up to thousands of map-points and keyframes, these tasks would not be able to run in real-time if all map is considered at each step. While one simple approach could be to restrict the size of the map and discard old information, this increases pose drift and eliminates the possibility of loop-closure.

Therefore, an efficient method to determine which part of the map is relevant is required. For tracking, the subset of points expected to be observed by the current camera-frame is needed. While for mapping, a subset of related keyframes and their corresponding observations need to be selected to perform local optimization. This selection becomes critical since not only it determines the efficiency of tracking and mapping tasks but also their accuracy and robustness.

The notion of covisibility [4] can be used to determine the relevant keyframes and map points (i.e. mutually observable) to a given camera pose and the currently tracked map. A pair of keyframes are said to be co-visible when they observe at least one visible map point in common. However, it is not yet clear how to best select these keyframes and map points using covisibility.

In this regard, the main contribution of this paper is to present a map marginalization strategy based on covisibility information that can be employed to solve more efficiently some of the most computationally demanding tasks in a SLAM system. We analyze how precision is affected and how this can help running full SLAM on low-resource hardware platforms. Furthermore, this work presents several approaches to minimize contention points and improve parallelism.

II. Related Work

In terms of on-board vision-based localization, a number of recent works employing either visual-odometry or SLAM-based approaches are presented.

[1]Matías Nistche, Gastón Castro, Thomas Fischer and Pablo De Cristóforis are with the ICC, Computer Science Research Institute (CONICET-UBA), Argentina. {mnitsche, gcastro, pdecris}@dc.uba.ar

[2]Taihú Pire is with the CIFASIS, French Argentine International Center for Information and Systems Sciences (CONICET-UNR), Argentina. pire@cifasis-conicet.gov.ar

Sanfourche et al. [5] proposes a stereo visual odometry suitable for UAVs. The method tracks features from successive camera frames while establishing 2D-3D associations with respect to a keyframe-based map. There is no optimization performed over the map, however pose drift grows slower compared to frame-to-frame visual odometry. They achieve 20Hz operation, however, while they claim their approach is suitable for embedded systems, experiments are performed using a relatively powerful Intel Core 2 Duo computer.

Weiss et al. [6] propose an on-board localization method equipped with a single camera that uses a inertial-optical flow approach for speed and IMU bias estimation. As speed integration is prone to position drift, an optimized version of PTAM is used to produce a 6DoF pose estimation. Combining visual and inertial measurements has proven effective for solving localization on UAVs.

Burri et al. [7] build upon visual-inertial odometry (VIO) obtaining dense environment reconstruction suitable for mission planning and exploration. Estimation drift derived from the VIO method is corrected by performing relocalization between local submaps. They obtain 20Hz operation time using a tailor made ARM-FPGA system [8].

Leutenegger et al. [9] proposes a novel visual-inertial SLAM system coined OKVIS. They formulate the problem as one joint optimization where an IMU measurement error term is considered along with the usual reprojection error within the minimization cost function. However, OKVIS is not conceived to work in low-resource hardware platforms and experiments are performed on a powerful laptop computer.

One of the main works on fully on-board stereo vision for UAV navigation was presented by Schauwecker et al. [10], where a visual odometry system based on PTAM is used to estimate the UAV pose. This work was extended in [11], where two stereo cameras were used: one facing forward, used to run a reduced SLAM system, and another facing downward, used for ground plane detection and tracking. Estimations obtained by each sensor are fused using EKF. The authors show that using two stereo cameras significantly increases pose estimation accuracy and robustness.

In terms of reducing the computational cost of SLAM approaches, some works focus on the problem of dealing with a large global map.

In [12] Lynen et al. present a framework for tracking the camera pose relative to a global map. They assert their method can be used for real time localization of mobile platforms with limited resources without the use of an external server. They achieved this by employing a covisibility strategy [13] to efficiently localize against the map.

Strasdat et al. [14] introduces an optimization framework that distinguishes two different keyframe windows. An inner window is composed by those keyframes that will be actively optimized while an outer window of close keyframes will act as fixed constrains. Optimization windows are defined based on the degree of covisibility between keyframes.The authors claim constant time operation when the maximum number of keyframes on each window is restricted.

Mur-Artal and Juan D. Tardós present a SLAM system called ORB-SLAM2 [3] that maintains a covisibility graph and a corresponding minimum spanning tree. These graphs are used to retrieve local windows of keyframes, so that tracking and mapping tasks operate locally while allowing to work on large environments and enabling for pose-graph optimization performed when closing a loop. This covisibility graph used in ORB-SLAM2 is similar to the one introduced by Strasdat et al. in [14].

III. Method

In the context of optimization-based SLAM, the tracking task is in charge of determining frame-to-frame camera pose. The tracking minimizes re-projection errors determined by map point to image-feature matches. Since this optimization requires an initial solution, it is usual to predict camera motion from previous poses and/or using additional proprioceptive sensors (e.g. IMU, wheel encoders, etc.). With this predicted pose, map-to-frame matches need to be established. To do so, map points are projected to the camera frame and then, by nearest-neighbor search in image-descriptor space, matches are obtained. These matches become observations under the optimization framework, representing a set of constraints. Finally, pose-optimization is performed, typically using Gauss-Newton or Levenberg-Mardquardt approaches.

From the previous steps performed by the tracking task, the most computationally demanding are typically: feature detection and descriptor extraction, point-to-feature matching, local map building and pose minimization. In particular, computational time of all these but feature extraction step strongly depend on the number of map points considered.

Moreover, for matching, in principle all map points need to be projected to the camera frame, which scales linearly with the size of the map. Also, this process is wasteful since many points could be actually invisible due to occlusions. Thus, a better approach is to use covisibility information in order to find map points seen by keyframes which share observations to the current camera frame. In other words, it is possible to build a *local map* of points which are highly likely to be currently visible.

When dealing with resource-constrained computing platforms, including all co-visible points to the local map may incur in excessive computational cost for tracking. Additionally, the size of this map can grow unbounded in certain conditions, which represents an undesirable situation in terms of real-time operation. For this reason, in these cases it is desirable to limit this local map.

A difficulty here appears since covisibility information needs to be built empirically during tracking from successful matches: whenever a point is matched to an image-feature in a given camera frame, this point is defined as *visible* from said frame. Covisibility can thus be represented as a graph between keyframes where each edge has a weight corresponding to the co-visibility degree, i.e. the number of shared observations.

Covisibility information built in this manner (via detected matches) cannot be guaranteed to match actual covisibility as would be obtained by exhaustive pair-wise frustum-culling between all keyframes. Thus, using covisibility information to build the local map may not necessarily retrieve the full set of points that could be observed by the current frame. For this reason, some works [3] propose to use not only directly co-visible keyframes but also a second level of keyframes co-visible to the first. This increases the possibility of including points which should be marked as directly co-visible but where this information has not yet been discovered. However, this comes at the expense of a larger local map and thus higher tracking cost.

In the following section, a simpler and more effective strategy for covisibility based local map building is presented, which allows to reach a bounded computational cost of the tracking task. Furthermore, it let balance efficiency vs. tracking precision.

A. Proposed Local Map Building Strategy

The proposed strategy for local map building is outlined in algorithm 1.

This strategy defines how to obtain the set of points M_L and keyframes K_L defining the local map, based on the previous set of successfully tracked points M_T (i.e. matched to image-features). Moreover, a *reference keyframe* K_r is designated during this process. This keyframe is considered to be the closest to the current camera frame in terms of observed points and is later used to determine when a new keyframe should be added.

With this local map, the points contained are then projected to the current image and used for point-to-feature matching. The set of successfully matched points will define the set M_T used for the next iteration. In other words, M_T will always be a subset of M_L.

In general terms, the local map building strategy find points visible by a set of keyframes K_{cov}, co-visible to the reference frame K_r. This reference, in turn, is defined as the keyframe observing the highest number of points in M_T.

In particular, only the first N keyframes with highest covisibility degree with K_r are considered. Moreover, only up to M points observed by these keyframes are added to M_L. Finally, low covisibility keyframes can be ignored using a minimum threshold C_{min}. As a result, the size of the local map is bounded. Moreover, the cost of building this local map scales linearly w.r.t. the number of keyframes co-visible to K_r. This is due to the fact that this term dominates the number of observing keyframes of a given point in M_T. Moreover, both M_T and K_{cov} are bounded by M (in previous iteration) and N, respectively.

It should be noted that M_L is first initialized using M_T, since using the aforementioned limits does not guarantee that all points in M_T will be in the result. This is particularly important when tracking is bad and M_T is small, which would result in a too small M_L.

As a result of applying this local-map building strategy, M_L is bounded by M. Thus, the cost of subsequent matching and minimization operations are also bounded by M.

Algorithm 1: Local-Map building strategy

Input: M_T tracked map
Output: M_L local map, K_L local keyframes, K_r reference keyframe

```
/* initialize with previous tracked map                    */
M_L ← M_T
/* find K_r which observes most points in M_T              */
foreach p in M_T do
    foreach K_i : observingKeyframes(p) do
        count(K_i) ← count(K_i) + 1
K_r ← argmax_{K_i} count(K_i)
/* get N most covisible keyframes to K_r                   */
K_cov ← sort_n(covisible(K_r), N)
/* add up to M pts observed by KFs in K_cov to M_L         */
foreach K_i in K_cov do
    if count(K_i) < C_min then
        continue
    M_L ← M_L ∪ observedPoints(K_i)
    K_L ← K_L ∪ {K_i}
    if #M_L > M then
        break
```

IV. Efficient on-board stereo SLAM

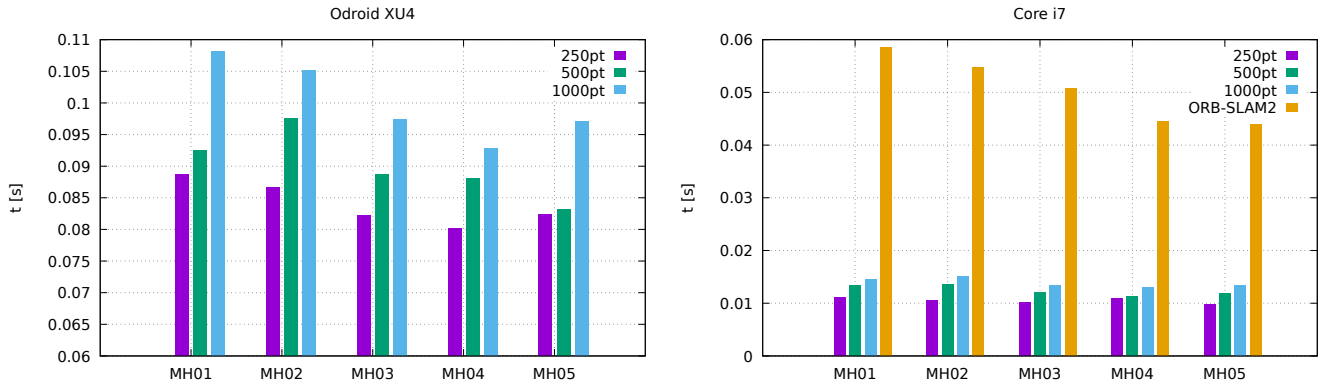
In order to verify the strategy proposed in this work, we build upon the stereo visual SLAM system S-PTAM [2], which has proven to be stable, accurate and suitable for large scale operation. In next sections, we describe some design and implementation considerations to better exploit parallelism in a optimization-based SLAM system. With these improvements S-PTAM is capable of running on-board on low-resource hardware platforms.

A. Efficient Map Access

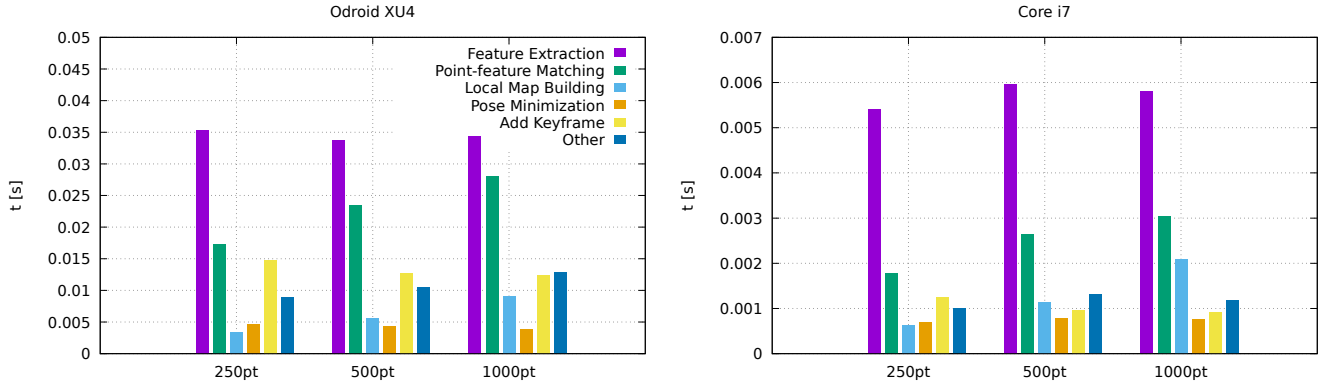
1) Access Requirements: As tracking and mapping tasks communicates through the map, contention points need to be minimized for the purpose of maximizing parallelization. The map is composed of two elements: map points and keyframes, which are related by two different graphs: a visibility bipartite graph between points and keyframes (where edges describe a measurement) and a covisibility graph between keyframes (where edges determine degree of covisibility).

Tracking requires frequent read access to keyframes and map points for local map definition (Section III) and feature-to-point matching. Occasionally, a frame is declared keyframe and its unmatched features are triangulated and added to the map as new points. Only in this case the tracking thread requires write access to maintain consistency of visibility and covisibility graphs.

On the other hand, the mapping task optimizes the map when a new keyframe is created. It requires both read and write access to keyframes and map points. It is also responsible for finding new matches between points and



(a) Execution time for the complete tracking task, for $M = 250, 500, 1000$ points.



(b) Execution time of each step of the tracking task, for $M = 250, 500, 1000$ points, averaged over all MH sequences.

Fig. 1: Mean execution times for different values of the local map size (M), for both processing platforms, over all MH sequences of EUROC dataset.

image features, updating the relations graphs with the new measurements.

Finally, a loop-closing task tries to detect loops upon new keyframe creation and, when detected, performs global pose-graph optimization. This involves write-access to all keyframes and map points. In principle this would involve halting mapping and tracking operations, however this is undesirable.

2) Synchronization strategy: In order to satisfy previous requirements while maximizing parallelism, instead of locking the whole map, map points and keyframes can be individually locked in order to access information such as point position and keyframe poses as well as other relational information such as keyframe covisibility.

In this scenario, when requiring write-access to multiple keyframes and map points simultaneously, such as when closing a loop, one approach could be to lock all of these simultaneously. However, a better choice is to arbitrate access to the map by defining required working regions beforehand. In other words, the mapping task can inform the region where it is currently working on and the loop-closing task can then avoid this region until it is freed.

B. *Feature extraction and matching*

Similarly to ORB-SLAM, features can be extracted on each image using a fixed-size grid. Over each cell, FAST [15] features are detected. If not enough are found in one cell, a lower feature response threshold is used. This process allows to obtain features throughout the whole image. However, this step results in a large number of features and filtering is required. To do so, we recursively divide the image area using a quad-tree, assigning features to each cell accordingly. When a given number of cells is reached, the feature with the strongest response of each cell is returned. In this way, it is possible to limit how many features are used for tracking and have an homogeneous distribution of feature in the image.

Also, to better exploit hardware parallelism, feature extraction (detection and description) and point-to-feature matching can be done concurrently amongst both images of the stereo pair.

V. Experiments

In this section we present the results obtained by the use of the local map building strategy in terms of the resulting performance improvement, particularly when running on low-resource hardware, and of its impact in localization precision. As a reference, we also run the modified S-PTAM

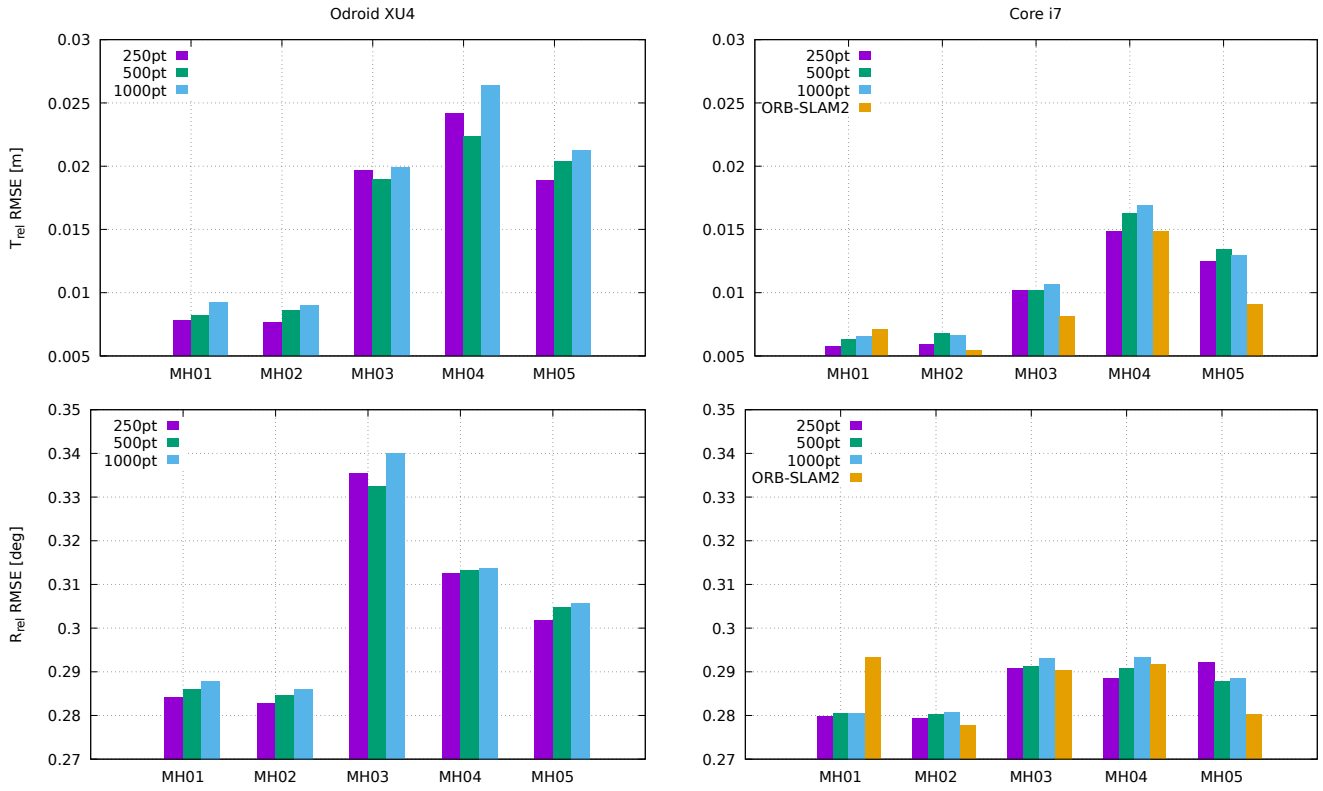


Fig. 2: Impact of different local map sizes on precision: RMSE values for relative translation and rotation errors, for each MH sequence of the EUROC dataset, for both processing platforms. Corresponding error values for ORB-SLAM2 are included.

on a powerful desktop computer and compare the obtained results and that of ORB-SLAM2 [3]. In this case we are not using loop-closing on S-PTAM, for fairness we disable this feature in ORB-SLAM2 as well.

Since the purpose of this work is to ultimately enable on-board and real-time execution of a Visual SLAM system for localization of UAVs, we test the S-PTAM system running on board an Odroid XU4 computer with four Cortex-A15 cores running at 2 GHz and four Cortex-A7 cores running at 1.4 GHz. For establishing a precision baseline without hardware constraints, we also run S-PTAM on an Intel Core i7-7700.

For a realistic and repeatable experimentation, we used the EUROC MAV dataset as input data. Since this is a challenging dataset with rapid camera-motion, we fuse IMU data using the MSF sensor fusion framework [16]. When running S-PTAM on the Odroid XU4, we replay the EUROC rosbag files on a desktop computer and feed data through an Ethernet connection to the Odroid, so as to remove any impact of I/O to the performance of the embedded system. Moreover, we only run MH sequences since V sequences present motion which is too fast for Odroid to follow.

In figure 1, we present the execution times for the tracking task of S-PTAM when using our proposed local map building strategy and that of ORB-SLAM2 (with author's parameters for this dataset), for different values of M (maximum size of local map). We also show the computationally most demanding steps of the tracking task in S-PTAM. We do not present execution of ORB-SLAM2 on-board the Odroid XU4 computer since tracking was quickly lost due to high processing time for each frame.

In figure 2 we present the relative translation and orientation errors of S-PTAM with different values of M, when running on each platform. Moreover, we also measure ORB-SLAM2 tracking precision for reference. It should be noted that since we are interested in using the SLAM system as a real-time localization source for autonomous navigation of UAVs, what is of importance when measuring precision is the error arising from camera pose reported after each tracking iteration, instead of the one obtained after local or even global bundle-adjustment. This is an important difference w.r.t. to other works where error is measured only after the complete dataset is replayed. For this reason, we run ORB-SLAM2 against EUROC dataset while measuring localization error in the same way, using instantaneous camera pose information.

When analyzing the results, a series of considerations can be made. First, it can be seen that the total tracking time (fig. 1a) of S-PTAM is much smaller than ORB-SLAM, around $4x$ to $6x$ faster. Also, the performance of S-PTAM on the Odroid XU4 is around and order of magnitude lower that on the Core i7. In any case, Odroid XU4 manages to track the camera at around 9 to 12 Hz in general, which is close to camera frame-rate. On the other hand, on Core i7, tracking rate is around 66 Hz, which is considerable higher than camera frame-rate. Second, it is possible to observe the

effect of the proposed local map building strategy, where limiting the size of this map reduces computational cost.

In order to better understand the performance improvement obtained by the use of the proposed local map building strategy, we also show the mean execution time of the main steps of the tracking task (fig. 1b). It can be seen that in all cases, the most demanding step corresponds to feature extraction (detection and description). The second most demanding step corresponds to the point to feature matching. Here it can be seen that lowering M has a positive impact on performance. Finally, as expected the cost of the local map building step itself is also lessened the less points are included in the output. On the other hand, lowering M has a slight negative impact on the keyframe creation step. This can be explained since a smaller local map implies that there is a higher chance of adding points which were not successfully matched.

In terms of tracking precision, in figure 2 it can be seen that, in general, reducing the number of points in the local map does not entail a significant impact on translation or rotation relative errors. Moreover, a difference can be observed between execution on Odroid XU4 and the Core i7 computers. This can be explained since on Odroid XU4 there is approximately a 50% frame-loss. Finally, when comparing to ORB-SLAM2 running on the Core i7, it can be seen that the localization performance of the S-PTAM system is quite similar. On the other hand, due to the high computational cost of ORB-SLAM2, measuring the localization precision running on Odroid XU4 was not possible.

VI. Conclusions

This work presents a local map building strategy based on constrained covisibility marginalization of the global map, in the context of an optimization-based SLAM. The purpose of this strategy is to reduce the computational cost of the tracking task in order to restrict the the size of the local map, aiming at on-board and real-time execution of the system on resource-constrained platforms, such as those present on small UAVs.

In order to prove the feasibility of the proposed approach, we implemented this strategy on the state-of-the-art S-PTAM system and performed a series of experiments on the challenging EUROC dataset. We also ran the ORB-SLAM2 system to establish a baseline for performance and precision.

Results show the reduction of computational time of the tracking task, which is of significant importance for on-board execution of the system on UAVs. Moreover, it can be seen how the S-PTAM system manages to track the camera at rates exceeding most standard cameras when running on a more powerful computer. On the other hand, on a resource-constrained platform, while performance is much lower, it is still possible to track the camera with rapid and challenging motions.

VII. Future Work

Analyzing the obtained results, we identify some future work areas. First, it can be seen that feature detection, description and matching are still some of the most demanding steps of a feature-based SLAM system. For this reason, it is interesting to consider finer optimization of these tasks using architecture-specific features, such as ARM's NEON instruction set. Second, we plan to perform closed-loop experiments by performing autonomous navigation of UAVs with the S-PTAM system running on-board and functioning as the main localization source. Finally, since S-PTAM already features loop-closing, we plan to evaluate performance of the system when running it on-board and to introduce optimization to allow for real-time execution.

References

[1] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10.

[2] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-PTAM: Stereo Parallel Tracking and Mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27 – 42, 2017.

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *CoRR*, vol. abs/1610.06475, 2016.

[4] C. Mei, G. Sibley, and P. Newman, "Closing loops without places," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 3738–3744.

[5] M. Sanfourche, V. Vittori, and G. L. Besnerais, "evo: A realtime embedded stereo odometry for mav applications," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2107–2114.

[6] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 957–964.

[7] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 1872–1878.

[8] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 431–437.

[9] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visualinertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[10] K. Schauwecker, N. Ke, S. Scherer, and A. Zell, "Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing," in *Autonomous Mobile Systems 2012*, ser. Informatik aktuell, P. Levi, O. Zweigle, K. Huermann, and B. Eckstein, Eds. Springer Berlin Heidelberg, 2012, pp. 11–20.

[11] K. Schauwecker and A. Zell, "On-board dual-stereo-vision for the navigation of an autonomous mav," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 1–16, 2014.

[12] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization." in *Robotics: Science and Systems*, 2015.

[13] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *European conference on computer vision*. Springer, 2012, pp. 752–765.

[14] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, November 2011, pp. 2352–2359.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer vision–ECCV 2006*, pp. 430–443, 2006.

[16] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2013, pp. 3923–3929.