Robotics and Autonomous Systems 116 (2019) 192–205

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Efficient on-board Stereo SLAM through constrained-covisibility strategies

Gastón Castro [a,*], Matías A. Nitsche [a], Taihú Pire [b], Thomas Fischer [a], Pablo De Cristóforis [a]

[a] University of Buenos Aires (UBA-CONICET-ICC), Argentina
[b] French Argentine International Center for Information and Systems Sciences (CONICET-UNR), Argentina

HIGHLIGHTS

- A fully functioning Stereo SLAM system with loop closure capabilities is presented.
- Efficient covisibility-based map culling strategies are introduced.
- Map optimization and loop closure policies employing shared covisibility information.
- Suitable for low-resource processing units, such as those found on-board of MAVs.
- Detailed description of every concurrent module with proper evaluation of each task.

ARTICLE INFO

Article history:
Available online 2 April 2019

Keywords:
Stereo SLAM
Constrained covisibility
Loop closure
Real-time embedded SLAM
MAVs

ABSTRACT

Visual SLAM is a computationally expensive task, with a complexity that grows unbounded as the size of the explored area increases. This becomes an issue when targeting embedded applications such as on-board localization on Micro Aerial Vehicles (MAVs), where real-time execution is mandatory and computational resources are a limiting factor.

The herein proposed method introduces a covisibility-graph based map representation which allows a visual SLAM system to execute with a complexity that does not depend on the size of the map. The proposed structure allows to efficiently select locally relevant portions of the map to be optimized in such a way that the results resemble performing a full optimization on the whole trajectory. We build on S-PTAM (Stereo Parallel Tracking and Mapping), yielding an accurate and robust stereo SLAM system capable to work in real-time under limited hardware constraints such as those present in MAVs.

The developed SLAM system in assessed using the EuRoC dataset. Results show that covisibility-graph based map culling allows the SLAM system to run in real-time even on a low-resource embedded computer. The impact of each SLAM task on the overall system performance is analyzed in detail and the SLAM system is compared with state-of-the-art methods to validate the presented approach.

1. Introduction

The recent growing interest in MAV platforms has triggered an increasing need for localization methods capable of operating on-board and in real-time. Designing systems that provide accurate pose estimation in challenging environments while running on platforms with limited computational resources has therefore become a key problem in modern mobile robotics.

Vision-based approaches seem to be the preferred solution due to their low power consumption, low weight and good performance in a broad spectrum of environments. However, accurate and consistent vision-based localization algorithms still require considerable computational power. This becomes an issue when targeting low-payload robots, where only small and low-resource processing hardware can be employed. Therefore it becomes interesting to consider new strategies for reducing computational requirements of vision-based localization methods to allow meeting real-time constraints.

From a methodological point of view, vision-based localization methods can be classified as *Visual Odometry* (VO) or visual *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

* Corresponding author.
E-mail addresses: gcastro@dc.uba.ar (G. Castro), mnitsche@dc.uba.ar (M.A. Nitsche), pire@cifasis-conicet.gov.ar (T. Pire), tfischer@dc.uba.ar (T. Fischer), pdecris@dc.uba.ar (P. De Cristóforis).

https://doi.org/10.1016/j.robot.2019.03.015
0921-8890/

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 the need of motion integration and are able to correct the pose drift over long trajectories by performing loop detections and closures.

A widely adopted real-time strategy for Visual SLAM is the one proposed by PTAM [1] (Parallel Tracking and Mapping) where the localization and map optimization tasks are decoupled as separate computing threads. The former, usually referred to as *tracking*, is expected to provide real time localization whereas the latter, further referred to as *mapping*, aims to keep the map as consistent and precise as possible, by performing a non-linear optimization technique called Bundle Adjustment (BA). Most recent feature-based Visual SLAM approaches adopt this strategy and usually extend it by adding a third thread [2,3] to perform loop detection and closure.

Nevertheless, tracking, mapping and loop closure tasks rely on operations which are largely dependent on the number of landmarks and processed camera frames. Since these numbers grow potentially unbounded with the size of the explored area, these tasks would not be able to run in real-time if the whole map is considered at each step. While a simple approach could be to restrict the size of the map by discarding old or faraway information, this restricts the possibility of performing large loop-closures, thus being tantamount to Visual Odometry.

Therefore, efficient methods to isolate portions of the map relevant to the current task at hand become necessary. The notion of covisibility [4] can be used to efficiently select a portion of the map that is strongly related by mutual observation to one or more query camera poses. A pair of camera frames are said to be covisible to the nth degree when they observe at least n landmarks in common (Fig. 1 illustrates this).

The present work is a follow up research based on proposed contributions of Nitsche et al. [5]. In this regard, the covisibility-based map culling strategy, originally employed only during the tracking process, is extended to all other tasks involved in the stereo SLAM system. This improvements are incorporated to the S-PTAM SLAM system [2,6]. The main contributions of this paper can be summarized as:

- A more extended and detailed description of every concurrent module in the system including a series of parallelization insights with proper evaluation of each task involved.
- An efficient local optimization policy around most recent map areas, allowed by shared covisibility information computed during the tracking process.
- A fully functioning stereo SLAM system is presented, with enabled loop closure capabilities and able to run in real-time on an embedded low-resource processing unit, such as those found on-board of MAVs.

2. Related work

In terms of on-board vision-based localization, several approaches either SLAM-based or employing visual-odometry have been proposed in recent years. Sanfourche et al. [7] propose a stereo visual odometry suitable for MAVs. 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. While this method achieves 20 Hz operation and authors thus claim their approach is suitable for embedded systems, experiments are performed using a relatively powerful Intel Core 2 Duo computer.

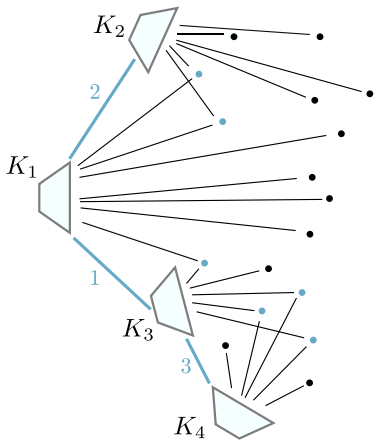


Fig. 1. The figure depicts a small map consisting of a set of polygons representing camera frames and a set of landmarks represented by dots. Landmark measurements are represented by black edges, whereas blue edges show the covisibility degree between frames. Blue dots depict landmarks that are observed by more than one camera pose, thus contributing to covisibility relations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Weiss et al. [8] propose an on-board localization method based on a single camera which employs an 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 6 DoF pose estimation. In general, combining visual and inertial measurements has proven effective for solving localization on MAVs [9–12]. Burri et al. [13] build upon visual-inertial odometry (VIO) obtaining dense environment reconstruction suitable for mission planning and exploration. Estimation drift resulting from the VIO method is corrected by performing relocalization between local submaps. They obtain 20 Hz operation time using a tailor made ARM-FPGA system [14].

Bloesch et al. [15] present a robust direct monocular visual-inertial odometry (ROVIO) framework based on a visual-inertial EKF-SLAM formulation [16,17]. In this work, multilevel patch features are directly tracked within the EKF. The new locations of the multilevel patch features are estimated by considering a IMU-driven motion model in the prediction step. The subsequent update step computes intensity errors as visual measurements by evaluating the discrepancy between the projection of the multilevel patch into the image frame and the image itself. The direct use of the error terms within an EKF would make it computationally intractable. In order to tackle this issue, the authors apply a QR-decomposition on the linear equation system resulting from stacking all error terms. The proposed approach uses a purely robocentric representation of the full filter state producing estimates of the current robot pose without maintaining an extensive map of the environment neither a history of the trajectory poses. The camera extrinsic parameters as well as the additive IMU biases are co-estimated. The developed framework can be run on-board a UAV equipped with an Intel Core i7-2760QM using a small number of features (equal to 50) at a frame rate of 20 Hz.

Leutenegger et al. [18] propose 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 visual 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 MAV navigation was presented by Schauwecker et al. [19], where a visual odometry system

based on PTAM is used to estimate the MAV pose. This work was extended in [20], 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. Obtained estimations from both camera rigs are then fused using an EKF. The authors show that using two stereo cameras significantly increases pose estimation accuracy and robustness.

On the matter of reducing the computational cost of SLAM, various works focus on efficient ways to deal with large global maps. Lynen et al. [21] present a framework for tracking the camera pose of a mobile system relative to a global 3D map. First, in an offline stage, a map of the environment is reconstructed from a set of database images by extracting image features that correspond to 3D landmarks and applying standard Structure-from-Motion (SfM) techniques. This 3D point cloud is then compressed by removing less important landmarks and quantizing the 3D point descriptors before being stored on the mobile device. The system runs a keyframe-based visual inertial EKF-SLAM method to smoothly track the movement of the camera. For each keyframe, image features are extracted and their descriptors are matched against the descriptors of the map. The resulting 2D-3D matches are then used to robustly estimate the camera pose using RANSAC.

Lynen et al. [21] uses a covisibility strategy based on [22] to efficiently localize against the global map by filtering out landmarks from the model. The idea is to define an undirected, bipartite visibility graph, where the two sets of nodes correspond to the database images and the 3D landmarks in the map. A landmark node and a image node are connected if and only if the landmark is visible in the corresponding database image. The landmarks from a given set of 2D-3D matches and their corresponding database images then form a set of connected components in this visibility graph. Then, the covisibility filter removes all matches whose 3D point does not belong to the largest connected component. These correspondences are then used to estimate the camera pose.

Strasdat et al. [23] introduces an optimization framework that distinguishes two different windows of constraints. An inner window of pose-to-point constraints supported by an outer window of pose-to-pose constraints. Optimization windows are defined based on covisibility between keyframes, prioritizing those with higher degree of covisibility. The authors claim constant time operation when the maximum number of keyframes considered for 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 locally connected keyframes, so that tracking and mapping tasks operate locally while allowing to work on large environments and enabling pose-graph optimization of the map when closing a loop. They achieve average processing time below camera frame-rate, on an Intel Core i7-4790 desktop computer with 16 Gb RAM.

Unlike the aforementioned works, in this paper we present a fully operational stereo SLAM system targeting an on-board embedded computer without restricting the size of the map, which allows performing loop detection and closure. As a result of the proposed covisibility strategy, our system is able to obtain high localization accuracy while maintaining its ability to operate in real time without the need of any prior information or model of the environment.

3. Efficient on-board stereo SLAM

In order to achieve a complete stereo SLAM solution capable of running in real-time on low-resource hardware platforms, it is crucial to identify the most computationally demanding operations. Fig. 2 outlines modules involved, along with their most common procedures, of a typical feature-based visual SLAM system which localizes the camera and maintains a global sparse reconstruction of the environment.

The Tracking module is in charge of determining map to camera-frame poses minimizing 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, these constraints are optimized refining the predicted camera pose, typically using Gauss–Newton or Levenberg–Marquardt algorithms. Once the current pose is estimated, a frame could be selected to be a keyframe if the Tracking module determines that it is about to lose track of the map. This decision involves a policy based on how many map points are successfully tracked between frames: keyframes should be created if the camera does a rapid change of orientation towards an unexplored area. Unmatched image features from the stereo frame are then triangulated and the map is expanded with new 3D points and a keyframe pose.

In a separated thread, the Local Mapping module continuously refines the map by means of minimizing the re-projection error between map points and observing keyframes. In contrast to the Tracking module, where only the most recent camera pose is optimized, in this case various keyframes are jointly optimized. However, considering the whole map during this process is not feasible for real-time applications. To face this problem, most successful approaches defines an area that will be actively adjusted, supported by an outer area that contributes fixed constraints [1,3,6]. Another approach is to optimize relative transformations between keyframes while marginalizing map points [24]. Others authors propose to use both point-to-pose and pose-to-pose constraints [23].

After a local area has been selected and optimized, a search for new map points observations is performed. Map points are matched against image features of recently adjusted keyframes, and in this way, map connectivity is improved. The process could also reveal outlier map points that show large re-projection errors on several keyframes. These points are flagged as badly estimated and removed from the map.

The third module found in most SLAM systems is the Loop Closing module. As the explored environment grows, visual SLAM systems tend to accumulate error on localization, keyframe poses and map point estimates. Even when the Tracking module is able to find enough map-to-frame matches and estimate an optimized pose of the current camera frame, the map could become globally inaccurate and lose fidelity with the real environment. Given the absence of global external information (GPS-denied scenarios) a successful approach is to recognize places that have been visited more than once (trajectory loops). This is a very valuable information that allows to measure the accumulated drift up to that moment and apply corrections to produce a globally consistent map model.

Most successful methods of place recognition on visual SLAM seek to classify appearance of keyframe images and evaluates similarities among them to establish possible loop candidates [25,26]. An appearance keyframe database is incrementally built allowing to quickly evaluate if two given keyframe images belong to the same place. After a loop candidate between two

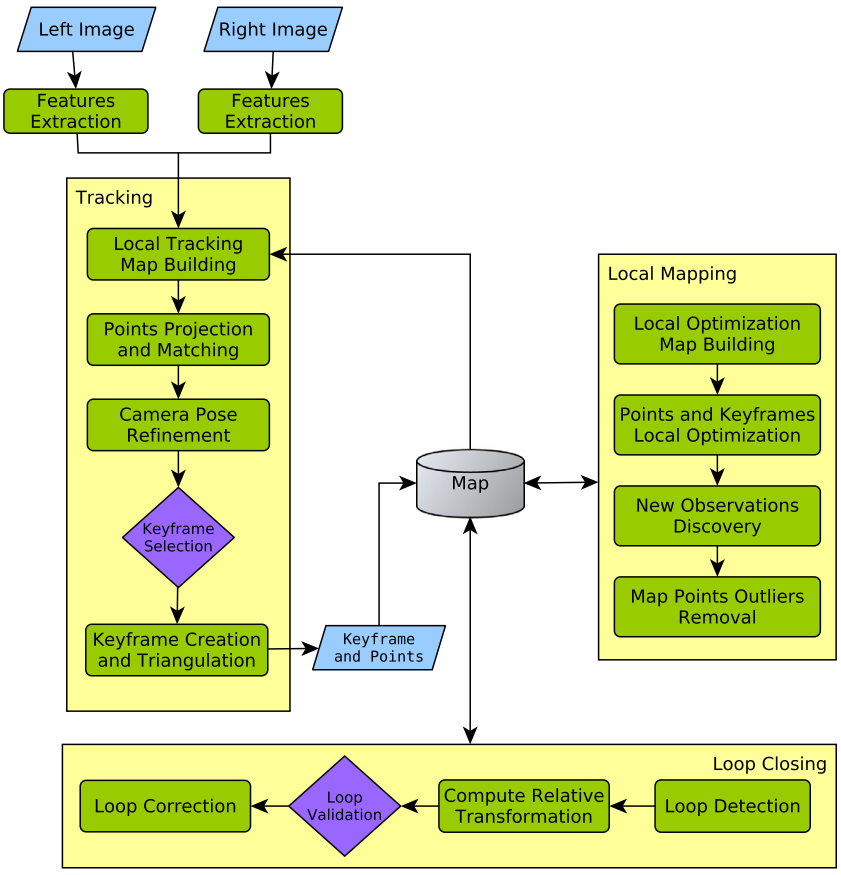


Fig. 2. Stereo visual SLAM overview.

keyframes has been established, an exhaustive geometric verification is employed to compute a better estimate of the relative transformation between those keyframes. Map points related with both keyframes are cross-projected and matched against the corresponding image features and the relative transformation is calculated using Perspective-n-Points [27] methods, usually under a RANSAC [28] model checking scheme.

In following sections we delve into each module proposing several covisibility-based techniques that allow to leverage a fully operational SLAM system including loop detection and closure on resource-constrained hardware, without limiting the global map size. A series of synchronization insights and considerations between modules are also presented.

3.1. Map structure and concurrency requirements

The map is composed of two main elements, map points and keyframes. These are related by two different graphs, a visibility bipartite graph between points and keyframes and a covisibility graph only between keyframes. The visibility graph models observation dependencies, where edges describe measurements between keyframe camera poses and tracked map points of the environment. The covisibility graph relates keyframes which share at least one observed map point, and edges store the degree of covisibility (total amount of shared map points). Using the covisibility information every module will continuously declare which is its active working area. This map segmentation allows to maximize concurrency on the map as modules will adjust to work efficiently over non-conflicting areas of the map. Fig. 3 exhibits a possible situation where three common operations are happening at the same time on the map. In this example, the Tracking module is performing current camera-pose estimation while the Local Mapping module is performing local optimizations. At the same time, the Loop Closing module is correcting older keyframes that are not being actively used by any other module. These operations will be further explained on following Sections 3.2.1, 3.3.1 and 3.4.1.

Maintaining which areas are being actively used at all times allows for specific tailoring of algorithms involved on each module, but access to individual map elements could still be required by several execution threads at the same time. Tracking requires frequent read access to keyframes and map points for local map definition and feature-to-point matching. A frame is declared keyframe when the number of tracked points is less than 90% of the points tracked in the last added keyframe. When a frame is selected as a keyframe, 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 thread optimizes the map constantly. It requires both read and write access to keyframes and map points. It is also responsible for finding new matches between points and image features, updating the relations graphs with the new measurements. Finally, a loop closing thread 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 order to satisfy the aforementioned requirements while maximizing parallelism, every map element stores its visibility relations and, instead of locking the whole map, points and keyframes can be individually locked allowing efficient information access.

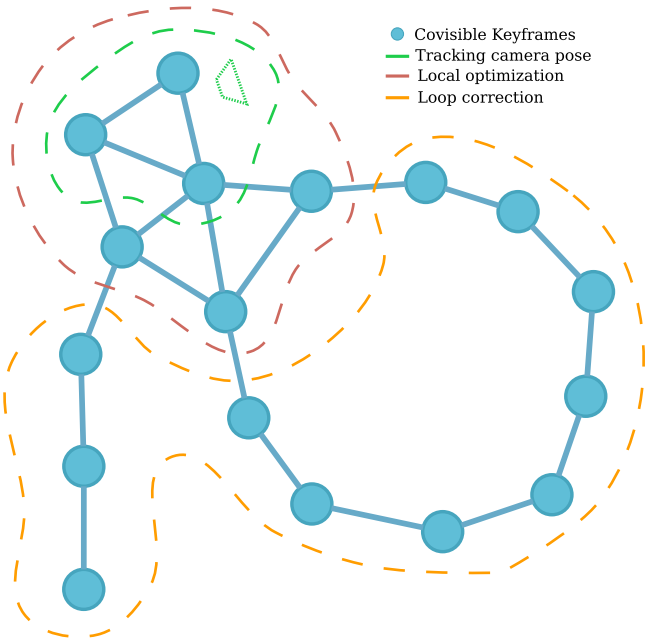


Fig. 3. Map segmentation example on a situation where different areas are being actively used for different purposes.

3.2. Tracking module

The most computationally demanding operations carried by the Tracking module are: feature detection and descriptor extraction, point-to-feature matching, local map building and pose optimization.

Regarding feature extraction, a robust and fast solution is to used a fixed-size grid as performed by ORB-SLAM2 [3]. Then, features are detected over each cell. If not enough features 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 maximum number of 200 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. To better exploit hardware parallelism, feature extraction (detection and description) and point-to-feature matching can be done concurrently among both images of the stereo pair.

While 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. When dealing with resource-constrained computing platforms, including all map points to the local map needed for tracking may incur in excessive computational cost. Additionally, the size of this map can grow unbounded as the explored area increases, which represents an undesirable situation in terms of real-time applications. For this reason, in these cases it is desirable to limit this local map. The approach followed in this work 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.

Covisibility information is 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. This information can thus be represented as a graph between keyframes where each edge has a weight corresponding to the covisibility degree, i.e. the number of shared observations.

However, a difficulty arises here since information built in this manner (via detected matches) cannot be guaranteed to match actual covisibility as would be obtained if the complete map was known beforehand. 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 covisible keyframes but also a second level of keyframes covisible to the first. While this increases the possibility of including points that should have been marked as directly covisible, it comes at the expense of a larger local map and thus higher tracking cost.

Besides building the covisibility relational graph, the Tracking module is also in charge of updating a pose-chain graph. This graph is actually a sub-graph of the covisibility graph that connects every keyframe with its most covisible one. To iteratively construct this pose-chain graph, the tracking maintains a reference to the most covisible keyframe with respect to the current camera frame. When a new keyframe is created, the Tracking module connects the reference keyframe with it and the recently created one is declared as the new reference.

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 allows to better balance efficiency and the tracking accuracy.

3.2.1. Local tracking map building

The proposed strategy for local map building is outlined in algorithm 1. This strategy presents a way to obtain the set of points M_L defining the local map, based on the previous set of successfully tracked points M_T (i.e. matched to image-features). First, a reference keyframe K_r is defined as the keyframe that observes most points in M_T. Second, the set of N most covisible keyframes to K_r is found and sorted, which is called K_{cov}. Third, up to M points observed from keyframes of K_{cov} are incorporated into M_L. Low covisibility keyframes of K_{cov} can be ignored using a minimum threshold C_{min}. Fig. 4 shows the selection process of map points to be considered in the local map.

Once the local map M_L is built, contained points are 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 this way, M_T will always be a subset of M_L.

It is important to note that only the first N keyframes with highest covisibility degree with K_r are considered. And only up to M points observed by these keyframes are added to 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.

The cost of building this local map scales linearly w.r.t. the number of keyframes covisible 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. Both M_T and K_{cov} are bounded by M (in previous iteration) and N, respectively.

It should also 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 relevant when tracking is bad and M_T is small, which would result in a too small M_L.

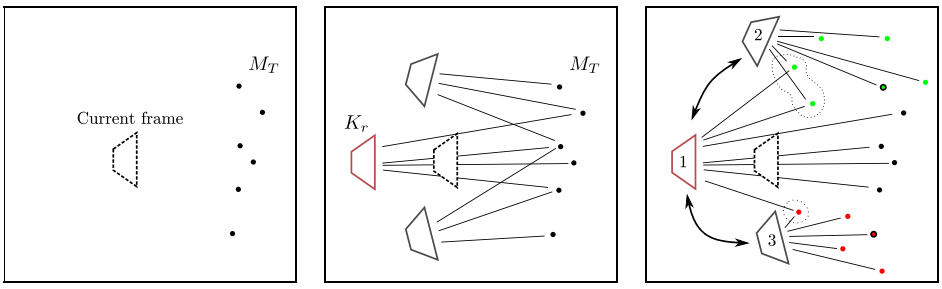


Fig. 4. (a) Map points tracked by the previous frame (M_T). (b) Selection of the reference keyframe K_r, this is, the keyframe that observe the major amount of map points in the M_T set. Extension of the M_T with the map points observed by the most covisible keyframes of K_r. The number depicted in the keyframes, reflects the order in which the map points are included to the local map.

Algorithm 1: Local Map building strategy

Input: M_T tracked map
Output: M_L local map, K_r reference keyframe,
K_{cov} most covisible keyframes to K_r

```
/* 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)
    if #M_L > M then
        break
```

3.3. Local mapping module

Adjusting keyframe poses and map point position through minimization of local re-projection errors is a very demanding task that highly depends on the amount of keyframes and map points selected. A precise keyframe selection policy must be decided for resource-constrained hardware. Before determining that policy, it is worth defining what is the main objective of this local optimization.

The Local Mapping module will be in charge of enhancing local consistency of areas being actively used by the tracking. Instead of working as a sliding window through the entire map, it will prioritize areas in close vicinity with the current camera pose. In this way, every task performed will have immediate impact on localization.

As the tracking relies on pose estimation methods that minimize re-projection errors, we work with the optimization policy described in [2] which defines an area that will be actively adjusted supported by an outer area that adds fixed constraints. Only pose-to-points constraints will be included in the optimization as adding pose-to-pose constraints will increase the computational cost. On the other hand, using only pose-to-pose constraints could result in a less demanding approach, but would not ensure better re-projection errors on the tracking task.

In the following section a covisibility-based strategy is presented to select the keyframes to be included in the optimization.

3.3.1. Local optimization map building

For the selection of keyframes to be used in the optimization, the mapping thread exploits information already computed during tracking. The current reference keyframe K_r is retrieved, along with the ordered set of the most covisible keyframes K_{cov}. K_r and a predefined number of keyframes from K_{cov} are selected to construct the set of keyframes that will get actively adjusted. The selection is carried in such way that those with highest degree of covisibility are chosen first to be actively adjusted. Remaining keyframes are then used to define the set of fixed keyframes. The amount of fixed keyframes is also predefined and if there is not enough keyframes in K_{cov} to satisfy it, a search is carried among covisible keyframes to those already selected for adjusting. As before, the selection is done by degree of covisibility.

As a result of this strategy, keyframes and map points in close vicinity of the current camera pose are prioritized for optimization. This allows improvements to be quickly available, enhancing the local consistency around the map area being tracked. When moving around already mapped areas where new keyframes are not required to be created at high rates there could be some spare time for optimization of older keyframes. In this case, the mapping task keeps track of not yet optimized keyframes and adjusts them in FIFO order.

3.4. Loop closing module

Once a loop is found and validated, a newly estimated transformation between keyframes is included into the pose-graph. This could generate a spatial inconsistency as the relative transformation may not conform with the global keyframe poses in the map. To deal with this, a correction is propagated through the entire map, relaxing all transformations between keyframes using a pose-graph optimization. It is paramount that the correction takes into account all map points and maintain consistency with already established keyframe-to-point observations. As a result, the map improves its fidelity with respect to the real environment.

In terms of computational cost, while very efficient appearance classification algorithms and loop detection methods exist, loop correction and map update are very time consuming processes that are not easily tackled as they require to work over the entire map.

Following an efficient covisibility-based strategy is presented for loop correction (see Fig. 2), which impacts on both the pose-graph optimization and map update performance. We build upon the loop closure method described in [2] adding a policy that divides map keyframes in several windows allowing to maximize concurrency of the system and to reduce the time during which the Tracking and Local Mapping modules must be paused.

3.4.1. Covisibility-based map segmentation

Introducing strong changes to the map may harm the performance of other system modules. As the goal of the system is to maintain proper localization, the Tracking module must remain operational in real-time at all times, being able to access map points as needed and to create new keyframes if required. It is also desirable that the Local Mapping module is able to perform local optimizations over keyframes and map points to enhance local consistency of an area nearby to the current camera pose. To ensure all this while correcting the map, the Loop Closure module divides the map in several windows. The main objective is to allow operation of both Tracking and Local Mapping in a defined map area while loop-closure optimizations and corrections over map areas that are not being used happen concurrently.

After a loop has been detected and validated, a safe tracking and mapping window is defined with the current tracking reference keyframe K_r and all its covisibles keyframes. No loop correction to keyframes nor map points will be introduced within this area. The Local Mapping module is also notified, so it can ensure that it will not perform optimization outside the safe window.

Loop correction and optimization are performed in this work as described in [2] over an internal copy of the entire pose-chain of keyframes. An initial correction is computed by propagating a relative transformation through keyframes applying a linear pose interpolation. Thereafter, a pose-chain optimization is carried to get a more accurate solution. After the optimization, each map point is corrected by applying the same transformation that was applied to the keyframe that originally triangulated it.

Fig. 5 characterizes a possible situation where a recently created keyframe K_t gets related with a loop keyframe K_ℓ by a relative transformation T_{tl} which denotes the pose of K_ℓ w.r.t. K_t coordinate system. Map is then divided into three windows: the safe tracking and mapping window, the safe loop closing window and the out of loop closure window. Keyframes on the safe tracking and mapping window are not necessarily time related as they are selected by covisibility. Loop closing window is then defined as the remaining keyframes that were captured by the internal pose-chain copy. Finally, the out of loop closure window includes keyframes that were created after the internal copy has been made.

The Tracking module is allowed to create new keyframes and map points during the loop closure and optimization process, and thus, there could be some keyframes that will not be considered on the pose optimization. In this regard, these keyframes are added to the out of loop closure window and a special update policy is applied.

After an optimized correction has been computed for all the considered keyframes, the map update process follows in three stages:

1. Update corrected keyframes and map points of the safe loop closing window.
2. Update corrected keyframes and map points inside the safe tracking and mapping window, applying any modification that may have been introduced by the local mapping since the start of the loop closing process.
3. Update corrected keyframes and map points of the out of loop closure window.

The first update stage is performed right after the pose-chain optimization without any synchronization between modules. Only during the second and third stage, Tracking and Local Mapping modules must be paused, where only a small fraction of the map is updated.

Keyframe states inside the safe mapping window are stored at the process beginning, as they are necessary for bookkeeping changes introduced by the Local Mapping module while the loop is being corrected. These changes are then re-applied before updating the safe window. Keyframes in the out of loop closure window are corrected using the same rigid transformation applied to the nearest keyframe on the pose-chain that has been included on the loop optimization.

3.5. Inter-modules synchronization and information sharing

After the Tracking module builds his local map, the reference keyframe K_r and K_{cov} set are stored and kept available for sharing with the Local Mapping as they represent valuable information that will be used to define an efficient optimization policy (discussed in Section 3.3.1). Apart from the reference keyframe, the Tracking module does not make use of pose information of other keyframes as these are only used to retrieve a set of surrounding map points to be used as map-to-frame matching candidates and pose estimation. Furthermore, map points are maintained in global coordinates. As a result, changes on keyframes from Local Mapping and Loop Closing optimization tasks do not invalidate map points information, and hence, further synchronization is not required regarding map points during tracking tasks. However, during map-to-frame matching a reasonable prediction of the current camera pose is necessary. The reference keyframe pose serves as an origin from which to construct the prediction integrating last camera pose with estimated instantaneous velocities (e.g. from wheel odometry or a decaying velocity model).

A loop correction may change the reference keyframe pose drastically, therefore, the Loop Closing module must pause all tracking and mapping operation while updating the map area that contains it (see Section 3.4.1) and ensure that the most updated state is always used. In cases where the camera is moving quickly during the loop closure process, or changes its orientation towards unexplored areas, the safe tracking window could not be sufficient to enclose the required area to be tracked. This situation could produce that the Tracking module creates new keyframes over already mapped areas leading to duplicated map points. This is acceptable and can be dealt with a guided search and fusion for duplicated points after loop correction.

4. Evaluation

In order to verify the proposed constrained-covisibility strategies in this work, we build upon the stereo visual SLAM system called S-PTAM [2], which has proven to be stable, accurate and suitable for large scale operation.

We analyze the resulting system in terms of performance improvement, particularly when running on low-resource hardware, and of its impact in localization precision. Since the purpose of this work is to ultimately enable on-board and real-time execution of a stereo SLAM system for localization of MAVs, we test the modified 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 evaluate performance on an Intel Core i7-7700.

The presented system is agnostic regarding to the image feature extraction method used. For reasons of efficiency and robustness, we use FAST [29] for feature detection and BRIEF [30] for keypoint description.

For a realistic and repeatable experimentation we used the EuRoC MAV dataset [31], as it delivers accurate ground-truth position estimates using a Leica MS50 laser that tracks a prism mounted on top of the MAV. Since EuRoC presents a challenging scenario with rapid camera motion and S-PTAM requires a camera pose prediction to allow for features-to-map matching,

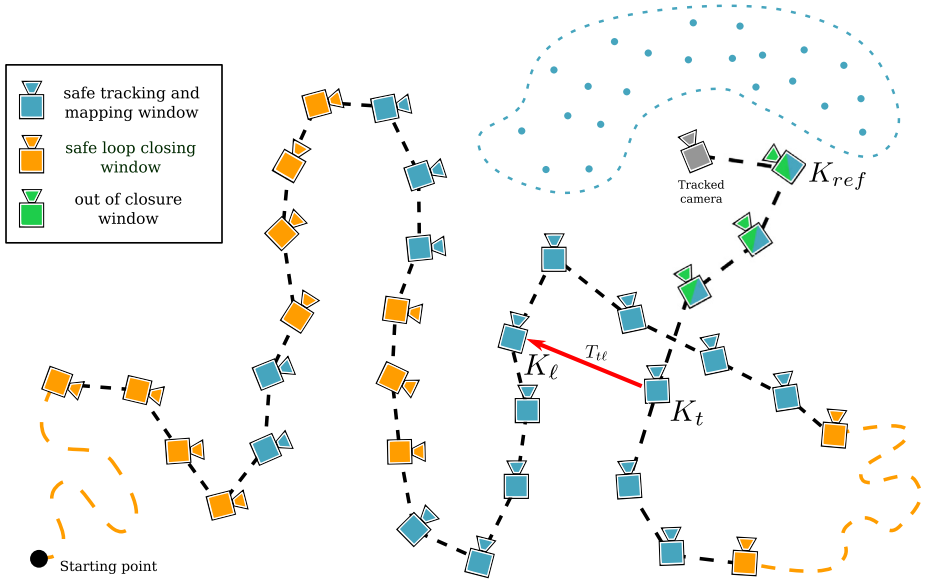


Fig. 5. A map segmentation scenario while performing loop closure. Orange dashed lines represents big map areas that are not actively used by Tracking nor Local Mapping modules. Blue dots represents map points retrieved by the tracking local map building. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

instead of employing a simple decay velocity model we choose to employ instantaneous velocity estimated from the IMU data. For this purpose, we use the MSF sensor fusion framework [32] that takes as input the IMU measurements and the current camera pose estimated by S-PTAM; and produces the camera velocity. The resulting velocity is integrated to obtain an initial guess for the camera pose in the following frame. We notice that, after the features-to-map matching, the initial guess is not used as a prior in the pose optimization. Furthermore, to measure camera pose estimation precision, we use the pose estimated by S-PTAM and not the fused pose produced by MSF. In other words, MSF is only used to aid in features-to-map tracking.

When running S-PTAM on the Odroid XU4 we replay the EuRoC dataset from a desktop computer and feed data through an Ethernet connection to the Odroid, so as to remove any impact of I/O in the performance of the embedded system. We only run MH sequences since V sequences present motion which is too fast for Odroid to follow.

4.1. Evaluation of constrained-covisibility strategies

In this section we present the results obtained from the proposed covisibility-based strategies employed for the Tracking and Local Mapping modules. As a reference, we also run the modified S-PTAM on a powerful desktop computer and compare the obtained results and that of ORB-SLAM2 [3]. Since for these tests we are not performing loop closure on S-PTAM, for fairness we disable this feature in ORB-SLAM2 as well. For all experiments, 200 FAST+BRIEF features are extracted and local optimization selects 6 keyframes for the active window and 20 keyframes for the fixed window.

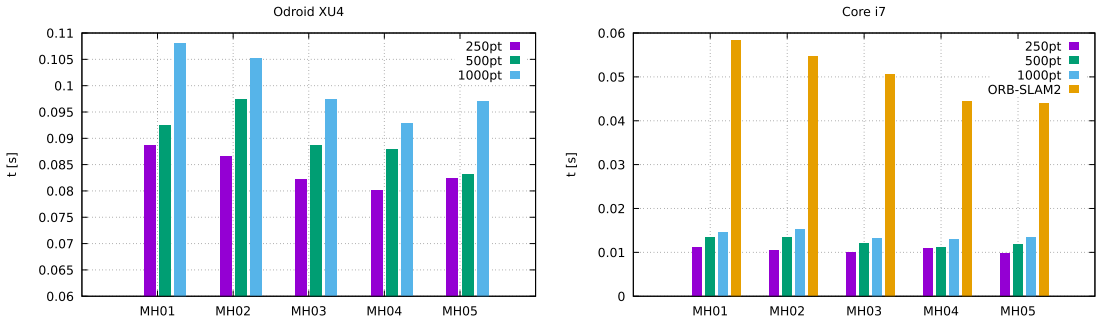
Fig. 6 shows execution times for the tracking task of S-PTAM when using our proposed local map building strategies and that of ORB-SLAM2 (with author's parameters for this dataset), for different values of M (maximum size of local tracking map). We also show the computationally most demanding steps of the tracking task in S-PTAM. Note that we do not present execution of ORB-SLAM2 on-board the Odroid XU4 computer since tracking was quickly lost due to high processing time demanded for each frame.

In Fig. 7 we present the relative translation and rotation errors of S-PTAM with different values of M, when running on each platform. Moreover, we also measure the ORB-SLAM2 tracking accuracy 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 MAVs, 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 with respect to other works where the 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.

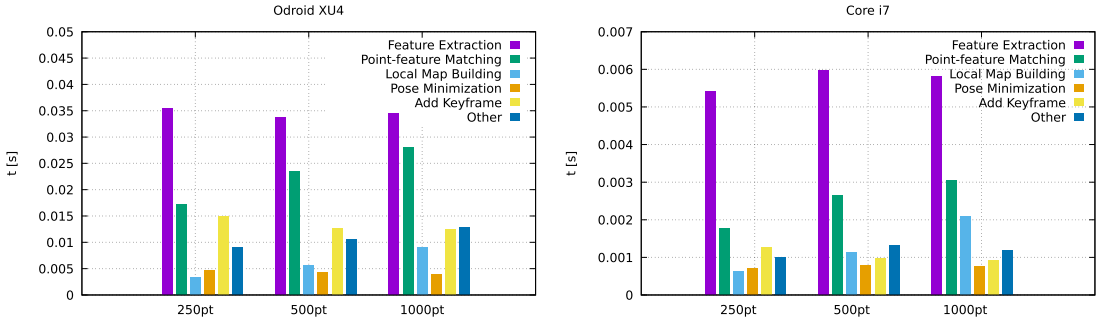
When analyzing the results, a series of considerations can be made. First, it can be seen that the total tracking time (Fig. 6a) of S-PTAM is much smaller than ORB-SLAM2, around 4× to 6× faster. Also, the performance of S-PTAM on the Odroid XU4 is around and order of magnitude lower that on the Core 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. 6b). 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 when 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 Fig. 7 it can be seen that, in general, reducing the number of points in the local map does



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

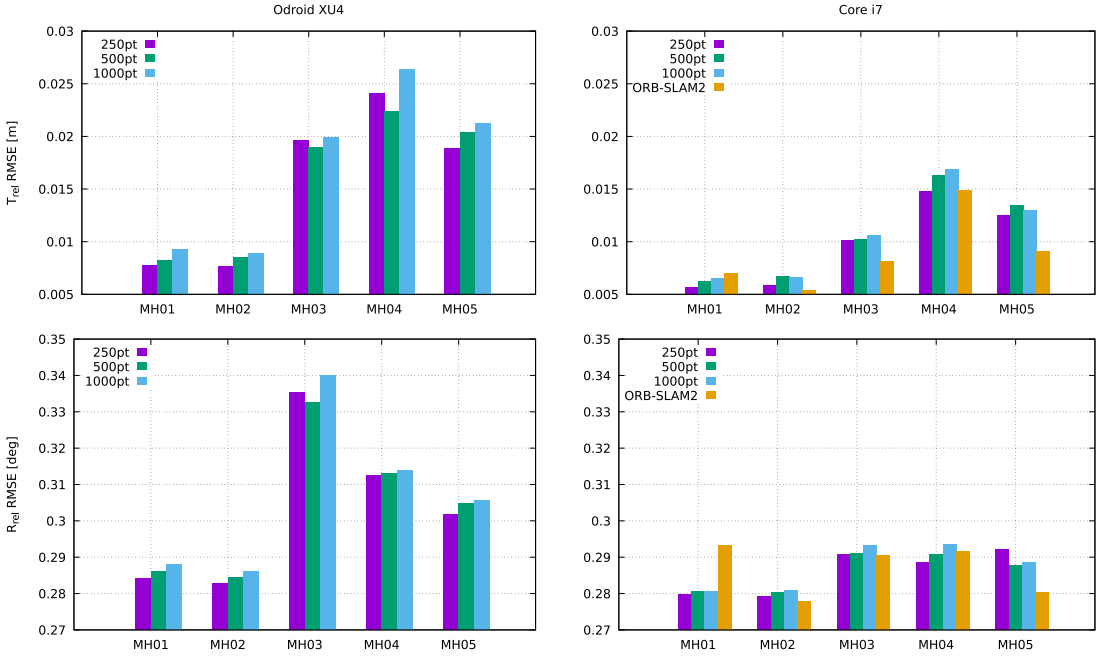


Fig. 7. Impact of different local map sizes on precision: RMSE values for relative translation and rotation errors, for each MH sequence of the EuRoC dataset, for both processing platforms. Corresponding error values for ORB-SLAM2 are included.

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.

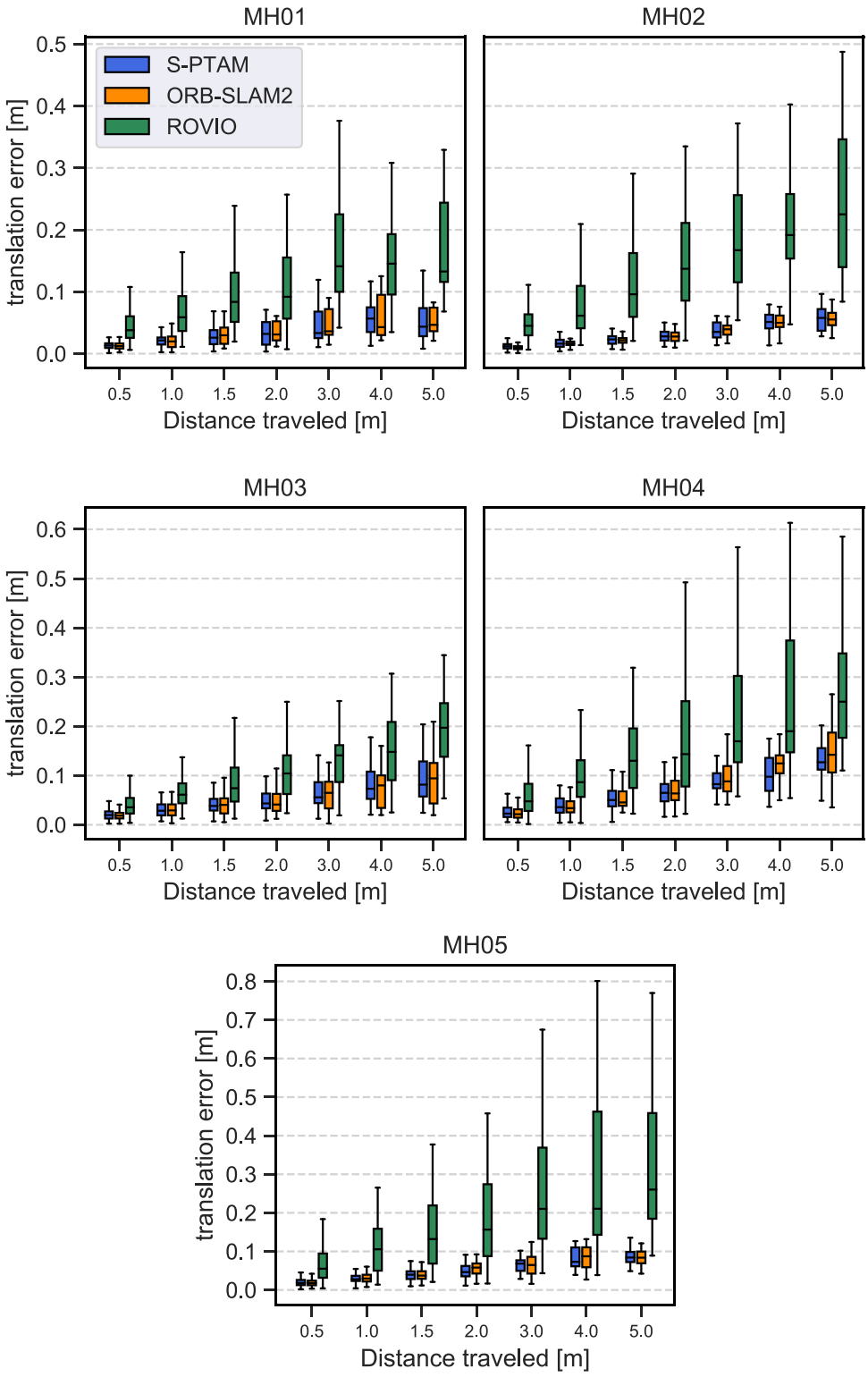


Fig. 8. Translation errors for several segment lengths using odometry metric proposed in [33].

4.2. Evaluation of odometry accuracy on intel core i7

Following we evaluate the accuracy of S-PTAM against ORB-SLAM2 and the state-of-the-art visual-inertial odometry system ROVIO [15] on the Intel Core i7 platform. As ROVIO is a monocular system without loop closure capabilities, we disable Loop Closing modules in S-PTAM and ORB-SLAM2 for all evaluations. Moreover, the EuRoC MH sequences are processed when MAV start exploring. S-PTAM parameter M (maximum size of local tracking map) has been set to 250 points, based on the results obtained in the previous Section 4.1.

Fig. 8 exhibits translation error on each sequence using the metric proposed in [33]. We compute the errors using the *evo* evaluation framework.[1] The accuracy of an algorithm is evaluated

[1] https://github.com/MichaelGrupp/evo.

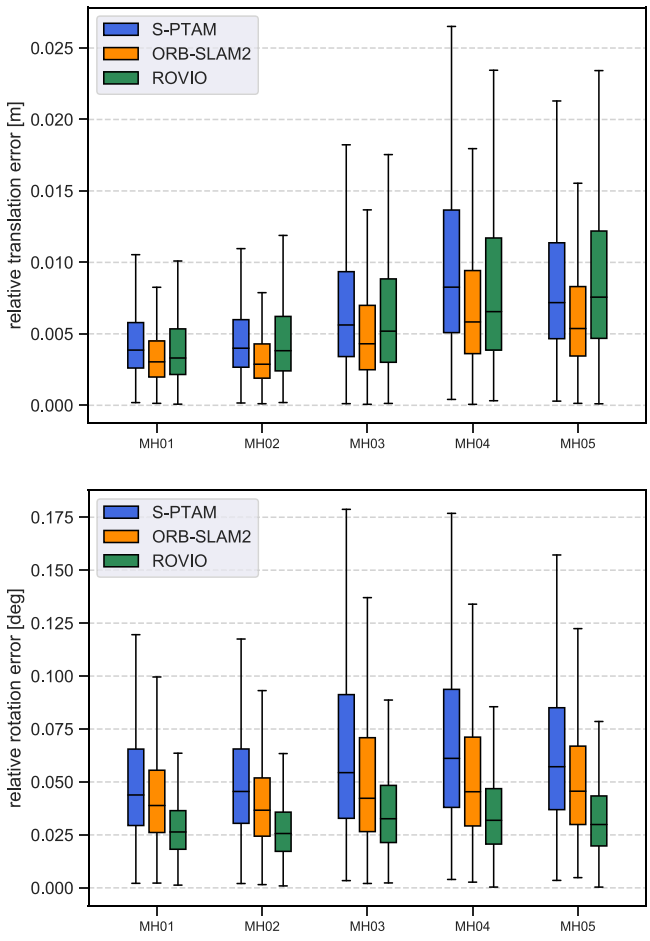


Fig. 9. Frame-to-frame relative errors of all EuRoC MH sequences.

by aligning each estimated pose with its corresponding ground-truth pose and then measuring error on estimating a determined distance along the trajectory. It can be observed that S-PTAM and ORB-SLAM2 outperforms ROVIO where its errors grow with the length of the estimated trajectory segment. The ability to relate map points across the map from different keyframes perspectives allows S-PTAM to produce consistent estimates within larger areas.

Fig. 9 shows competent relative translation and rotation estimates of S-PTAM, although, ROVIO outperforms S-PTAM at estimating relative frame-to-frame motion. While ROVIO does not maintain an extensive map of the environment, it is able to fuse IMU measurements actively obtaining precise relative estimates. These results expose the impact of being able to work with a rich map of the environment. S-PTAM and ORB-SLAM2 sacrifice frame-to-frame motion estimation accuracy in favor of a greater global consistency effectively bounding the estimation drift in long term operation.

4.3. Evaluation of global localization accuracy on odroid XU4

In order to evaluate the global consistency achieved by our system, we present results where the Loop Closing module is activated on the Odroid XU4 platform. Evaluation is performed only against ROVIO since ORB-SLAM2 is not able to work in such limited hardware.

Fig. 10 presents absolute translation and rotation errors comparing estimated trajectory poses against the ground-truth reported poses. The system is able to maintain consistent global

Table 1
Absolute translation errors (RMSE) for all sequences. Errors have been computed after trajectories where aligned with the ground-truth as proposed in [34]. Loop Closing module has been activated for S-PTAM while running on the Odroid XU4. The top performing method on each platform and dataset sequence is highlighted in **bold**.

	Intel Core i7			Odroid XU4	
	S-PTAM	ORB-SLAM2	ROVIO	S-PTAM+LC	ROVIO
MH01	0.054	**0.023**	0.243	**0.089**	0.337
MH02	0.063	**0.024**	0.386	**0.071**	0.393
MH03	0.155	**0.027**	0.260	**0.254**	0.511
MH04	**0.188**	0.260	0.763	**0.292**	0.938
MH05	**0.091**	0.166	0.486	**0.446**	0.822

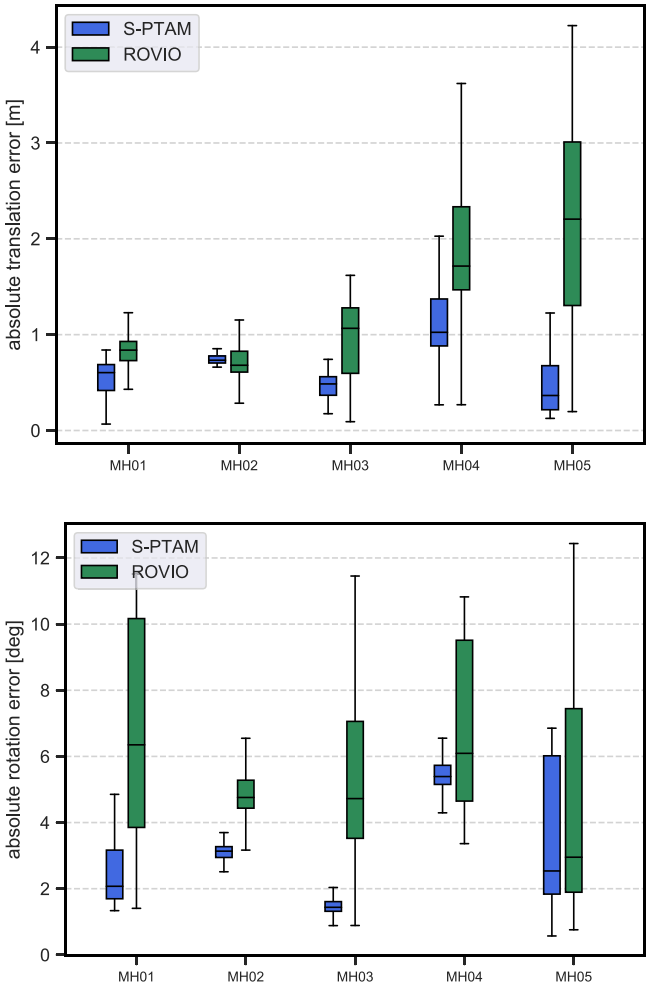


Fig. 10. Absolute translation and rotation estimation errors of all EuRoC MH sequences.

localization within 2 meters of translation error and 6.4 degrees of rotation error on all sequences. ROVIO presents higher absolute translation errors on almost all sequences except on MH02 where it gets better estimates at the beginning of the sequence where rapid motion is present.

Table 1 shows absolute translation errors (RMSE) over trajectories aligned with ground-truth according to the proposed metric in [34]. Scale alignment is applied to ROVIO estimates so that they match ground-truth scale using [35]. ORB-SLAM2 outperforms S-PTAM on sequences MH01, MH02, MH03 on the Intel Core i7 but S-PTAM is able to run with Loop Closing activated on the Odroid XU4 presenting competent results that supersede ROVIOS's.

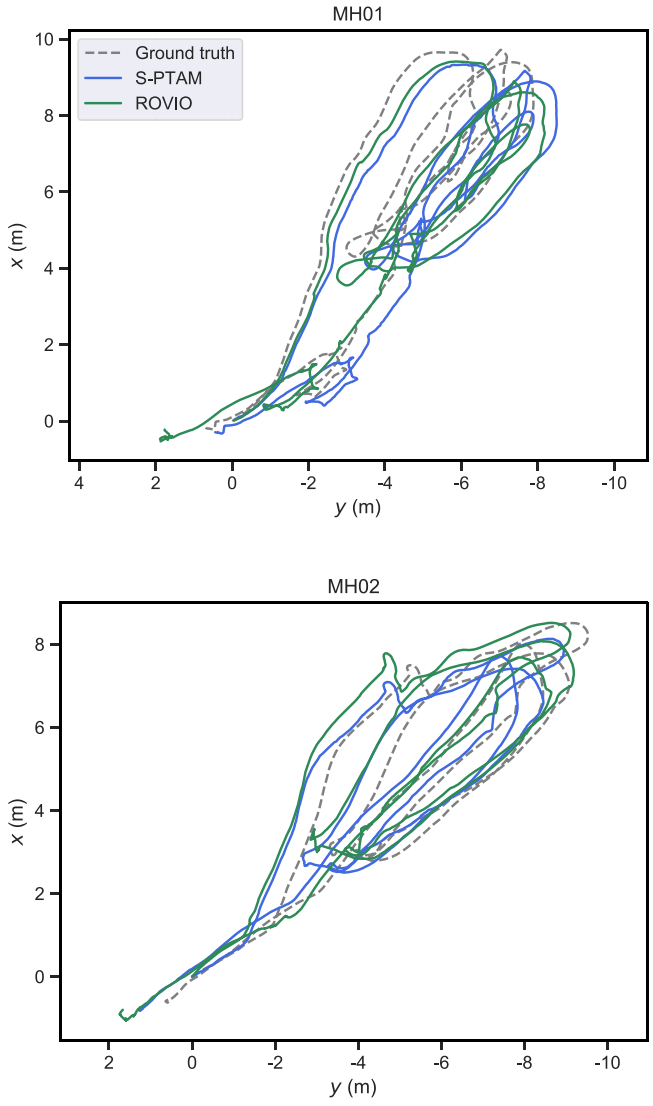


Fig. 11. S-PTAM and ROVIO estimated trajectories over EuRoC sequences MH01 and MH02 running on Odroid XU4.

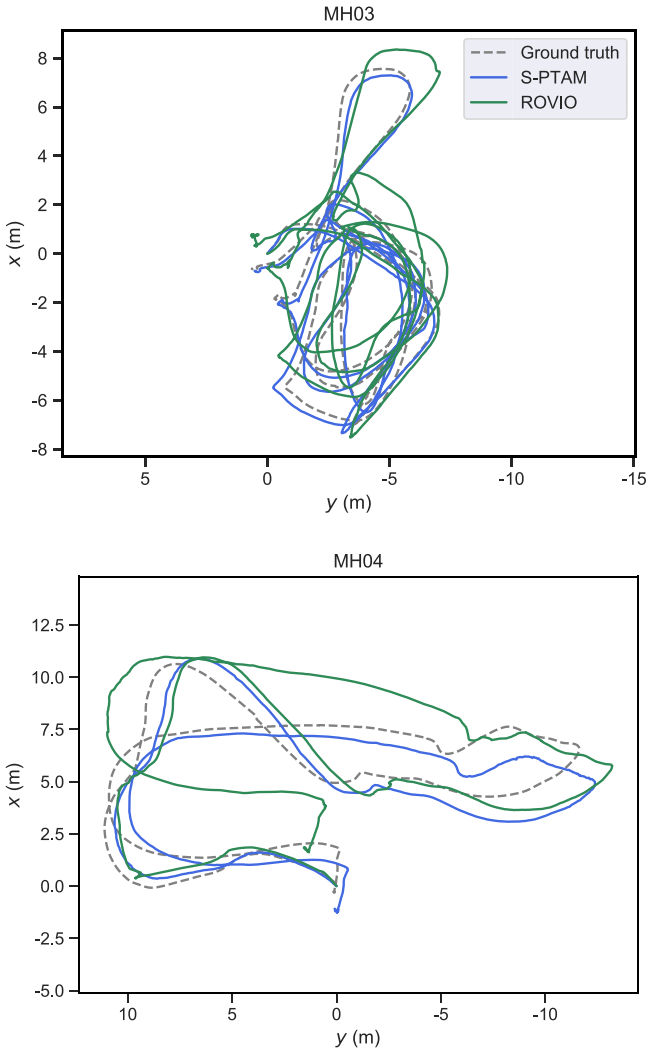


Fig. 12. S-PTAM and ROVIO estimated trajectories over EuRoC sequences MH03 and MH04 running on Odroid XU4.

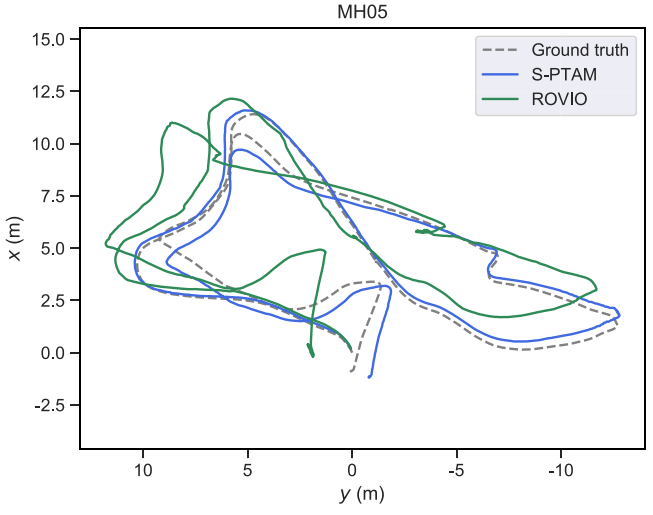


Fig. 13. S-PTAM and ROVIO estimated trajectories over EuRoC sequence MH05 running on Odroid XU4.

Figs. 11–13 exhibits estimated trajectories obtained over all sequences. It can be observe that ROVIO presents higher accumulated drift specially on sequences MH04 and MH05 where estimated trajectory ends several meters away from the ground-truth.

Table 2 exhibits the amount of loops effectively closed on each EuRoC sequence with the processing time required for the map update process. Measured times are divided by map keyframe windows described in Section 3.4.1. Requiring to pause the tracking task only during the safe tracking and mapping window update, by an average of around 50 ms.

5. Conclusions

This work presents covisibility-based point-selection policies in the context of an optimization-based SLAM. This map selection strategy, originally used only during the tracking module, is extended for loop correction, optimization and map update allowing to include a loop closure module. As a result, the computational cost of the overall SLAM system is reduced, but especially the one of the tracking task, which can be bounded as desired. This allows to reach an on-board and real-time execution on resource-constrained platforms, such as those present on MAVs.

Table 2
Performance of loop closing map update tasks on each EuRoC MH sequences running on the Odroid XU4.

Seq.	#Closures	Update stage	Min. (ms)	Avg. (ms)	Max. (ms)
MH01	4	Loop closing window	103.7	159.37	220.5
		Tracking window	9.8	**47.35**	72.4
MH02	2	Loop closing window	14.7	76.35	138.0
		Tracking window	33.0	**43.65**	54.3
MH03	3	Loop closing window	89.7	158.2	246.2
		Tracking window	21.1	**28.5**	33.4
MH04	1	Loop closing window	136.1	136.1	136.1
		Tracking window	18.2	**18.2**	18.2
MH05	6	Loop closing window	133.8	161.1	202.9
		Tracking window	4.9	**33.3**	85.4

In order to prove the feasibility of the proposed approach, we implemented it 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 and ROVIO systems to establish a comparison for performance and accuracy.

Results show the reduction of computational time of the tracking task, which is of significant importance for on-board execution of the system on MAVs. Moreover, it can be seen how the S-PTAM system manages to track the camera pose at rates exceeding most standard cameras when running on a more powerful computer.

Finally, regarding execution on a resource-constrained platform, although the performance is much lower, the results evince that it is still possible to track the camera with rapid and challenging motions while even detecting and closing loops, making sense of maintaining a global map.

Acknowledgments

This research was supported by the UBACyT project No. 20020170100739BA and the PICT project No. 2015-3167.

References

[1] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE Computer Society, Washington, DC, USA, 2007, pp. 1–10, http://dx.doi.org/10.1109/ISMAR.2007.4538852.

[2] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, J. Jacobo Berlles, S-PTAM: Stereo parallel tracking and mapping, Robot. Auton. Syst. 93 (2017) 27–42, http://dx.doi.org/10.1016/j.robot.2017.03.019.

[3] R. Mur-Artal, J.D. Tardós, ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras, IEEE Trans. Robot. 33 (5) (2017) 1255–1262, https://doi.org/10.1109/TRO.2017.2705103.

[4] C. Mei, G. Sibley, P. Newman, Closing loops without places, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 3738–3744, http://dx.doi.org/10.1109/IROS.2010.5652266.

[5] M.A. Nitsche, G.I. Castro, T. Pire, T. Fischer, P. De Cristóforis, Constrained-covisibility marginalization for efficient on-board stereo SLAM, in: 2017 European Conference on Mobile Robots (ECMR), Paris, 2017, pp. 1–6, https://doi.org/10.1109/ECMR.2017.8098655.

[6] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, J.J. Berlles, Stereo parallel tracking and mapping for robot localization, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 1373–1378, http://dx.doi.org/10.1109/IROS.2015.7353546.

[7] M. Sanfourche, V. Vittori, G.L. Besnerais, eVO: A Realtime embedded stereo odometry for mav applications, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 2107–2114, http://dx.doi.org/10.1109/IROS.2013.6696651.

[8] S. Weiss, M.W. Achtelik, S. Lynen, M. Chli, 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, 2012, pp. 957–964, http://dx.doi.org/10.1109/ICRA.2012.6225147.

[9] T. Qin, P. Li, S. Shen, Vins-mono: a robust and versatile monocular visual-inertial state estimator, IEEE Trans. Robot. 34 (4) (2018) 1004–1020, https://doi.org/10.1109/TRO.2018.2853729.

[10] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C.J. Taylor, V. Kumar, Robust stereo visual inertial odometry for fast autonomous flight, IEEE Robot. Autom. Lett. 3 (2) (2018) 965–972, http://dx.doi.org/10.1109/LRA.2018.2793349.

[11] J. Delmerico, D. Scaramuzza, 2018, A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, pp. 2502–2509, https://doi.org/10.1109/ICRA.2018.8460664.

[12] A.I. Mourikis, S.I. Roumeliotis, A multi-state constraint kalman filter for vision-aided inertial navigation, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 3565–3572, http://dx.doi.org/10.1109/ROBOT.2007.364024.

[13] M. Burri, H. Oleynikova, M.W. Achtelik, 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), 2015, pp. 1872–1878, http://dx.doi.org/10.1109/IROS.2015.7353622.

[14] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P.T. Furgale, 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), 2014, pp. 431–437, http://dx.doi.org/10.1109/ICRA.2014.6906892.

[15] M. Bloesch, S. Omari, M. Hutter, R. Siegwart, Robust visual inertial odometry using a direct ekf-based approach, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 298–304, http://dx.doi.org/10.1109/IROS.2015.7353389.

[16] J. Kelly, G.S. Sukhatme, Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration, Int. J. Robot. Res. 30 (1) (2011) 56–79.

[17] E.S. Jones, S. Soatto, Visual-inertial navigation, mapping and localization: a scalable real-time causal approach, Int. J. Robot. Res. 30 (4) (2011) 407–430.

[18] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale, Keyframe-based visual-inertial odometry using nonlinear optimization, Int. J. Robot. Res. 34 (3) (2015) 314–334, http://dx.doi.org/10.1177/0278364914554813.

[19] K. Schauwecker, N. Ke, S. Scherer, A. Zell, Markerless visual control of a quad-rotor micro aerial vehicle by means of on-board stereo processing, in: P. Levi, O. Zweigle, K. Huermann, B. Eckstein (Eds.), Autonomous Mobile Systems 2012, in: Informatik aktuell, Springer Berlin Heidelberg, 2012, pp. 11–20, http://dx.doi.org/10.1007/978-3-642-32217-4_2.

[20] K. Schauwecker, A. Zell, On-board dual-stereo-vision for the navigation of an autonomous mav, J. Intell. Robot. Syst. 74 (1–2) (2014) 1–16, http://dx.doi.org/10.1007/s10846-013-9907-6.

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

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

[23] H. Strasdat, A.J. Davison, J.M.M. Montiel, K. Konolige, Double window optimisation for constant time visual SLAM, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 2352–2359, http://dx.doi.org/10.1109/ICCV.2011.6126517.

[24] K. Konolige, M. Agrawal, Frameslam: From bundle adjustment to real-time visual mapping, IEEE Trans. Robot. 24 (5) (2008) 1066–1077.

[25] D. Gálvez-López, J.D. Tardós, Bags of binary words for fast place recognition in image sequences, IEEE Trans. Robot. 28 (5) (2012) 1188–1197.

[26] R. Mur-Artal, J.D. Tardós, Fast relocalisation and loop closing in keyframe-based slam, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 846–853.

[27] L. Kneip, H. Li, Y. Seo, Upnp: An optimal o(n) solution to the absolute pose problem with universal applicability, in: European Conference on Computer Vision, Springer, 2014, pp. 127–142.

[28] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, in: Readings in computer vision, Elsevier, 1987, pp. 726–740.

[29] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: Computer vision–ECCV 2006, Springer, 2006, pp. 430–443.

[30] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: Binary robust independent elementary features, in: Proceedings of the European Conference on Computer Vision (ECCV), in: Lecture Notes in Computer Science, vol. 6314, Springer Berlin Heidelberg, 2010, pp. 778–792, http://dx.doi.org/10.1007/978-3-642-15561-1_56.

[31] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M.W. Achtelik, R. Siegwart, The euroc micro aerial vehicle datasets, Int. J. Robot. Res. (2016) http://dx.doi.org/10.1177/0278364915620033.

[32] S. Lynen, M.W. Achtelik, S. Weiss, M. Chli, 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, 2013, pp. 3923–3929, http://dx.doi.org/10.1109/IROS.2013.6696917.

[33] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361, http://dx.doi.org/10.1109/CVPR.2012.6248074.
[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of rgb-d slam systems, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 573–580.
[35] S. Umeyama, Least-squares estimation of transformation parameters between two point patterns, IEEE Trans. Pattern Anal. Mach. Intell. (4) (1991) 376–380.

Gastón Castro was born in Buenos Aires, Argentina, in 1989. He received his master degree in Computer Science in 2013 from the University of Buenos Aires, with a master thesis subject on loop closure. He is currently a full time Ph.D. student at the University of Buenos Aires. Teacher Assistant at the same university. His research interests include computer vision, visual SLAM methods, loop closure and robotics.

Matías Nitsche received the Ph.D. degree in Computer Science in 2016 at the University of Buenos Aires, Argentina. He is currently a Research Assistant at the National Council of Scientific and Technological Research (CONICET), Argentina. His research is focused on solving localization and autonomous navigation of Unmanned Aerial Vehicles using stereo-vision and inertial sensors.

Taihú Pire was born in Rosario, Argentina, in 1986. He received the PhD degree in Computer Science from the University of Buenos Aires, Argentina in 2017. He is currently a postdoctoral researcher at the French Argentine International Center for Information and Systems Sciences (CONICET-UNR), Argentina. Currently, his research interests are in develop new Visual SLAM algorithms.

Thomas Fischer was born in Lomé, Togo, in 1988. He received his master degree in Computer Science in 2013 from the University of Buenos Aires. Since then he works as a full time Ph.D. student for CONICET at the Laboratory of Robotics and Embedded Systems from the University of Buenos Aires, and also as a Teacher Assistant at the same university. Currently, his research interests include camera vision, visual SLAM, autonomous real time navigation and exploration, and hexapod platforms.

Pablo De Cristóforis received the Ph.D. degree in Computer Science from the University of Buenos Aires, Argentina in 2013. He is currently a Research Assistant at the National Council of Scientific and Technological Research (CONICET), Argentina. His research interests include autonomous vision-based navigation and exploration, visual SLAM and 3D vision algorithms for mobile robotics.