Noname manuscript No.
(will be inserted by the editor)

DS-PTAM: Distributed Stereo Parallel Tracking and Mapping SLAM System

Mauro De Croce · Taihú Pire · Federico Bergero

Received: date / Accepted: date

Abstract This paper presents DS-PTAM, a distributed architecture for the S-PTAM stereo SLAM system. This architecture is developed on the ROS framework, separating the localization and mapping tasks into two independent ROS nodes. The DS-PTAM system is ideal for mobile robots with low computing power because it allows to run the localization module on-board and the mapping module —which has a higher computational cost— on a remote base station, relieving the load on the on-board processor. The proposed architecture was implemented based on the original S-PTAM monolithic code and then validated through different experiments on public datasets. The results obtained show the feasibility of the proposed distributed architecture, its correct implementation and the benefits of distributing the computational load on several computers.

Keywords Distributed system · Visual SLAM · Stereo vision

1 Introduction

In order for a mobile robot to navigate its environment, it is essential that it knows its position and orientation (pose) at all times and to have a representation of the environment (map). In robotics these problems are usually addressed simultaneously, and the problem is known as SLAM (*Simultaneous Localization and Mapping*) [5,1,3]. In this type of problem, the need arises to construct a map of the environment (*mapping*), at the same time the robot localize itself in it (*localization*), taking the signals from its sensors or built-in devices.

There are different types of sensors that can be used to solve the SLAM problem: monocular and stereo cameras, sonar, laser, GPS and RGB-D sensors among others. However, each sensor has its limitations, for example, a sonar allows us to detect if there are objects at a certain distance, but it is not possible to determine which objects are involved [13]. GPS sensors (*Global Positioning System*) only work in outdoor environments, and may even fail in places where the signal is not good enough. Laser sensors allow to estimate the depth of the environment with great accuracy, but they are often expensive, not very portable and do not have a long distance range. Finally, RGB-D *Kinect* sensors use an infrared sensor to estimate the depth of the observed scene and are therefore affected by sunlight, which prevents them from functioning properly in outdoor environments [27]. In contrast, cameras have great portability, low cost, can be used in both indoor and outdoor environments, and at the same time provide rich information about the scene being observed. All this makes cameras one of the most widely used sensors to solve the SLAM problem (PTAM [14], ORB-SLAM2 [17], LSD-SLAM [6], ScaViSLAM [22], SVO [9], S-LSD-SLAM [7] and S-PTAM [19]). Camera-based solutions include those using monocular or stereo devices, both allowing a 3D reconstruction of the observed scene, the latter having the advantage of being able to determine the scale of the scene.

Mauro De Croce
FCEIA-UNR
Av. Pellegrini 250 S2000 Rosario, Argentina
E-mail: maurodecroce@gmail.com

Taihú Pire
CIFASIS (CONICET-UNR)
27 de Febrero 210 bis S2000EZP Rosario, Argentina
E-mail: pire@cifasis-conicet.gov.ar

Federico Bergero
CIFASIS (CONICET-UNR)
27 de Febrero 210 bis S2000EZP Rosario, Argentina
E-mail: bergero@cifasis-conicet.gov.ar

On the other hand, SLAM systems have a high computational cost due to the need of processing large amounts of information in real time, such as robot trajectory and map representation that grows along the robots moves. Today, mobile robots have a limited on-board processing power due to the nature of the devices that make them up. Depending on the environment and the task to be performed by the robot, the processing hardware must satisfy weight, size and power consumption restrictions. An excess in any of these quantities could compromise the robot's navigation capability or autonomy. It is therefore appropriate to allocate the processing burden on the basis of different external devices.

This paper proposes a distributed version of the Visual SLAM S-PTAM system in such a way that it is possible to run the localization module (*tracker*) in the robot's on-board processing unit, and separately the map maintenance module (*mapper*) in a remote base station. In this way, the localization module has all the robot's processing capacity, allowing it to process a larger number of frames provided by the camera, and therefore a more precise and robust system is obtained to abrupt robot movements. In addition, the mapping module that demands a high computational cost, when running on a remote station, can be run on a computer with higher computational capacity than those usually found on a mobile robot.

We build on the SLAM system S-PTAM [19] because it is open-source and its performance and accuracy are competitive with other state of the art methods. The proposed system, named DS-PTAM, can distribute the localization and mapping tasks on two computers (on-board and base station). For this purpose, we propose a modular design, based on events communication and distributed memory (instead of shared memory as the original system proposes).

When transitioning to a distributed memory scheme, each module (*tracker* and *mapper*) will have a copy of the map, so that each of them can work independently. For example, the tracker's map is modified after viewing an unknown part of the environment while the mapper's map is modified by improving the internal 3D representation of the environment. We use a communication mechanisms based on ROS message-passing paradigm to maintain the consistency of both maps. To this end, we created message structures that includes only the map modifications and updated poses in order to minimize the number of messages, and thus reducing traffic and the computational cost involved in this synchronization.

The rest of the article is organized as follows: Section 2 presents a complete overview of the state of the art on distributed SLAM systems. Section 3 depicts a general overview of the S-PTAM SLAM system proposed in [19]. Section 4 detailed the distributed stereo-based SLAM system DS-PTAM proposed in this work. In Section 5 a complete evaluation of DS-PTAM is presented. The experiments are carried out on public domain datasets using different computing hardware configurations. Finally, the conclusion and future work are presented in Section 6.

2 Related work

SLAM problems have been a case of study for several decades now, and many works have been carried out in order to obtain solutions based on distributed systems, some of them by means of the use of frameworks developed for this purpose, such as the case of [10] and [21]. The first is aimed to reduce the complexity of communications within the network to facilitate the implementation of distributed SLAM solutions using ROS as the underlying framework. To do this, it takes advantage of the abstraction of messages provided by ROS, as well as its statistics system, which allows it to generate a report on the consumption of network resources used by each execution node. The second work proposes that the data, such as the map and other components, be stored on external servers located in the cloud. This architecture frees the robot from the computational load, which only has to be in charge of tracking, although it does require a connection to Internet by means of WiFi.

Today, the increasing use of Unmanned Aerial Vehicles (UAVs) in various applications has triggered numerous studies on these platforms. As with all mobile robots, airborne vehicles also require SLAM solutions for stand-alone operation. This is the case of [26,15] which propose the use of drones to obtain, through the application of PTAM's paradigm [14], a map in a collaborative manner. On the other hand, taking into account the weight limitation that UAVs can carry in comparison to terrestrial robots, distributed processing takes a fundamental value as it allows both load balancing and computation processing in external devices, relieving the robot of many responsibilities. We found an example of this in [8] where a central station synchronizes multiple vehicles, generating a map for each of them and then joining the overlapping fragments together to obtain a general map.

We can also mention the development proposed in [25], which describes a framework that uses monocular SLAM techniques based on PTAM, to allow multiple airborne vehicles the possibility of avoiding obstacles during the exploration process of the environment. For this development, the ROS framework was used, where each component such as the vehicles' drivers, tracking

and map maintenance functionalities, have been implemented as independent nodes taking advantage of the distributed architecture of ROS, being able to execute them in different processes in order to add scalability to the framework. On the other hand, in [24], a novel distributed object volumetric reconstruction system is introduced using an airborne vehicle that sends data from the images captured by the cameras to a high-volume processing server that will perform the most complex tasks. In addition, it allows the user to make use of a tablet to visualize the results in real time, also providing the possibility of correcting results as the process progresses.

One of the SLAM methods that has had the greatest impact on mobile robotics was PTAM (*Parallel Tracking and Mapping*) [14]. This method poses a new paradigm that applies to Visual SLAM issues in which it is proposed to separate localization and mapping tasks into independent threads by optimizing performance in multi-core processors. In particular, this innovative proposal was used in S-PTAM [19] but addressing the problem of SLAM for large scale environment. It should be noted that previously the paradigms used to find solutions to SLAM problems were based on filters such as *Extended Kalman Filter* [12] and *Particle Filter* [16].

Many of the developments that are part of the state of the art in the area of distributed visual SLAM systems address the problem of collaboration between devices for a more complete mapping of the environment, most of them working on airborne vehicles or mobile devices, such as tablets or cell phones [26, 15, 8, 25, 24]. The present work, instead of tackling the problem of building a more detailed map of the environment, focuses on obtaining a distributed SLAM system capable of localizing a mobile robot in real time. The proposed SLAM system relies on both the robot's on-board processing hardware and other external devices, considering the computing power of each to assign a task commensurate with its processing power.

Preliminary results by the authors were published in a local conference [4]. The present work builds on the cited article extending the results to more datasets and performing a deeper analysis of the experiments.

3 S-PTAM

In this section we introduce the S-PTAM stereo SLAM system on which DS-PTAM is based.

3.1 General Overview

S-PTAM (*Stereo Parallel Tracking and Mapping*) [19] is an open-source method that addresses the problem of SLAM based on stereo vision. It was developed using the ROS framework [20] with the objective of being executed in real time in large environments, obtaining an accurate estimate of the robot's pose while building a map of the surrounding environment.

Following the PTAM approach [14] and in order to harness the power of multi-core processors, S-PTAM divides the tasks of tracking and mapping into two independent execution threads, thus achieving very good performance compared to other current SLAM methods.

At the beginning of its execution, S-PTAM assumes that the stereo camera is located at the origin of world coordinates and without a map of its environment. Then, upon receiving the first stereo frame captured by the camera, the map is initialized by triangulating the extracted features in the left frame and right frame. The triangulated points are called *map points*. Thereafter, the subsequent stereo frames received will be located by estimating the camera pose based on the existing map. A decay-velocity model —based in previous poses— predicts an initial estimate of the camera pose. This estimation is adjusted by projecting the 3D points on the plane of each image and looking for the correspondences with the features extracted in it by comparing each of their descriptors. The obtained correspondences of this procedure are called 2D-3D matches or constraints because they represent the observation of a point in space by the stereo camera. In addition, those stereo frames that observe an unexplored part of the world are selected to be included in the map, adding new map points and new constraints to the map. These frames are called *keyframes*. Then, as the robot advances along its path, it will obtain new stereo images that will be processed with the described method generating an incremental map that will increase in size in each iteration of the process. This functionality is responsibility of the Tracker module.

Running concurrently with Tracker, another execution thread called Mapper is started, which will be in charge of adjusting camera poses and map point, i.e. performing map refinements. These refinements are carried out using the *Bundle Adjustment* method [23]. This thread also looks for new matches between keyframes and map points to increase the constraints between them. In addition, it removes both map points and measurements considered spurious or inconsistent by the Bundle Adjustment method.

3.2 Architecture

Figure 1 shows a diagram of the overall S-PTAM structure, which consists of two main functionalities:

Tracking that receives the information of the points extracted from the stereo images and from them calculates the robot's pose and creates the points that will later be part of the map

Mapper (or Local Mapping) that performs the optimizations on the map (refinement), establishes new connections between new points of the map and keyframes that were previously added and also takes care of removing those points that are considered as outliers.

It is important to note that the Tracking and Local Mapping threads use a shared memory structure that represents the map built during the process and which is accessed and modified by these modules. From now on we will use the terms Tracking and Tracker; and Mapper and Local Mapping indistinctly.

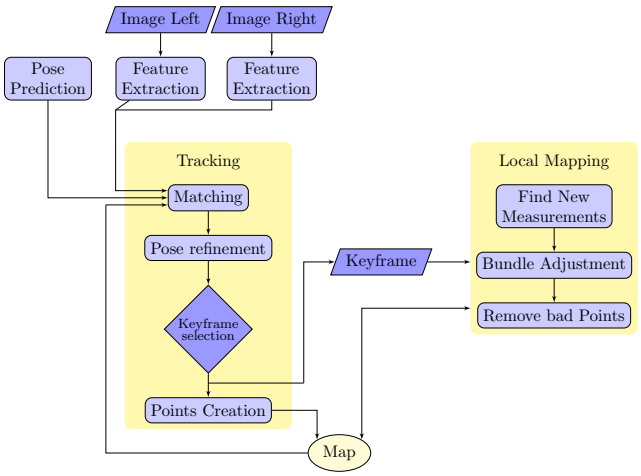


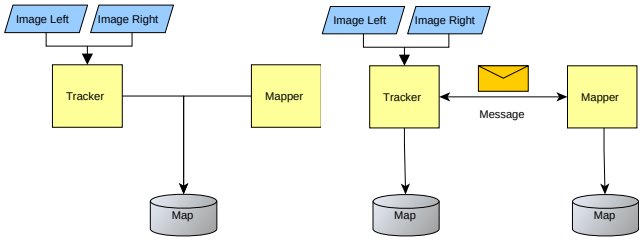
Fig. 1 S-PTAM structure diagram [19].

4 DS-PTAM

In this section we detail the DS-PTAM system, the main contribution of the present work.

4.1 Architecture

As mentioned above, prior to this work, S-PTAM's design made use of shared memory in which there is a single map on which the entire system works. We developed DS-PTAM based on S-PTAM identifying the predominant functionalities and re-designing the architecture so that they are able to be executed on independent ROS nodes. Each of these nodes will interact with others through ROS message-passing system. The comparison between these architectures can be seen in Figure 2. Following a distributed memory scheme implies that each node will have its own copy of the map and it will be desirable that these maps are consistent with each other. To achieve this, each node will have to report the changes that has made to its copy of the map so that the other node can apply those changes to its map. Due to the asynchronous nature of ROS messaging and possible out-of-order or lost of messages some inconsistency could appear. Out of order messages are not a problem if they carry modifications for map sections, since these changes are commutative. Lost messages result in changes not being applied on the destiny map, but successive messages changing the same points will solve this inconsistency since they are not incremental (the messages carry the new state for each map point/keyframe). Finally, conflicting modifications to the same section of the map will result in inconsistency between the two maps. This could be solved using Conflict-Free Replicated Data Structures but is out of the scope of the present work. As we will see, this does not affect the overall accuracy of DS-PTAM.



(a) S-PTAM structure. (b) DS-PTAM structure.

Fig. 2 Comparison between the original S-PTAM architecture and the distributed architecture proposed here. In the distributed architecture both the Tracker and Mapper have an instance of the Map, and the communication between both nodes is done through ROS messages.

In the current S-PTAM design, map elements such as keyframes, map points and measurements can be accessed using memory pointers because they are located in shared memory. On the other hand, in the distributed design proposed in this work a mechanism for identifying objects on different memory spaces is required.

To this end, unique identifiers are assigned to the objects that are part of the map and that are created and modified dynamically during the process. These identifiers will be shared by all nodes and will refer to the same object on the map (each with different memory addresses). To abstract this correspondence behavior,

we created the *Translations class*, which internally consists of a vector of local pointers to objects and methods that allow assigning their values, obtaining the pointed object from a given identifier. The i-th position of this vector holds a pointer to the corresponding element with an id equal to i. Then, having one instance of this class for keyframes and another for map point, both Tracker and Mapper can make reference to a specific object regardless of whether each node has it stored in a different memory space and thus provides a mechanism to communicate to the other node about changes in its attributes in a clear and consistent manner. Figure 3 shows the evolution of the translation objects of each node during execution startup. Additionally, we introduced a message structure for communication between nodes. This structure is based on the ROS standards, which include data types such as integers, boolean, floating, characters, strings, among others. In addition, the structures can recursively hold nested vectors of these primitive types mentioned above.

As mentioned above, DS-PTAM is divided into two ROS nodes: Tracker and Mapper. We now describe their design.

4.2 Tracker

With the goal of achieving a distributed processing, and considering that S-PTAM has a single map, we developed, as a first step, an instance of the map to the Tracker node. In this instance, the functions and methods that allow to add, modify and delete keyframes, map points and measurements to the map are also included.

We implemented the Tracker node in order to be capable of receiving the stereo images coming from the camera of the mobile robot and process them (by extracting local image features) to then create and update its instance of the surrounding environment map. In this way, autonomy and independence of the Tracker node from the Mapper node is achieved, and it can be executed without the need for Mapper to be running simultaneously. This characteristic is one of the main contributions of this work, since until now it was not possible to carry it out with the current monolithic design of S-PTAM. Although, as we will see later, the results will be better when running both nodes concurrently, being able to use the Tracker node autonomously allows us, for example, to run a simplified version of S-PTAM on devices with low computation power.

4.3 Mapper

The other main module of the system (also implemented as a ROS node) is the Mapper. This node is coupled to the Tracker node for complementary work. A map instance is included in the Mapper node that will be continuously adjusted by the Bundle Adjustment optimization method, and each modification will be communicated to Tracker to update its map for consistency among them.

4.4 Nodes communication

The communication between nodes is carried out using message management system provided by ROS, which facilitates this task by means of a *Publisher/Subscriber* mechanism grouped in *topics*. It should be noted that the sender node does not know who the receiver nodes of its message are, i. e. they are unidirectional and asynchronous.

Tracker and Mapper nodes communicate each other using a message structure designed for this purpose that includes information related to map changes. When the Tracker node adds elements (such as keyframes, map points and measurements) it generates a message, including all the necessary information so that the Mapper node can add these objects to its own map. On the other hand, when the Mapper node modifies its map by refining it, it notifies Tracker of these changes, including only relevant information so that Tracker can reflect these changes on its map, and thus avoid a message that contains redundant information and negatively impact the performance of the system. ROS messages are queued internally in the Mapper node and applied sequentially once the node is able to process them. For simplicity of design, it was decided to use the same message structure for both Tracker and Mapper senders. Figure 4 depicts the ROS message used for the communication between Tracker and Mapper nodes.

We implemented a *messages handler* library to facilitate the communication between the nodes, which contains functions that convert elements of DS-PTAM into ROS messages and vice versa. In other words, to create a message, the function will use C++ objects (which represent the elements that DS-PTAM uses internally) as an argument and will return a corresponding message following the structure defined for this purpose. The inverse function will take a message as an argument and return a DS-PTAM object using the appropriate class constructors.

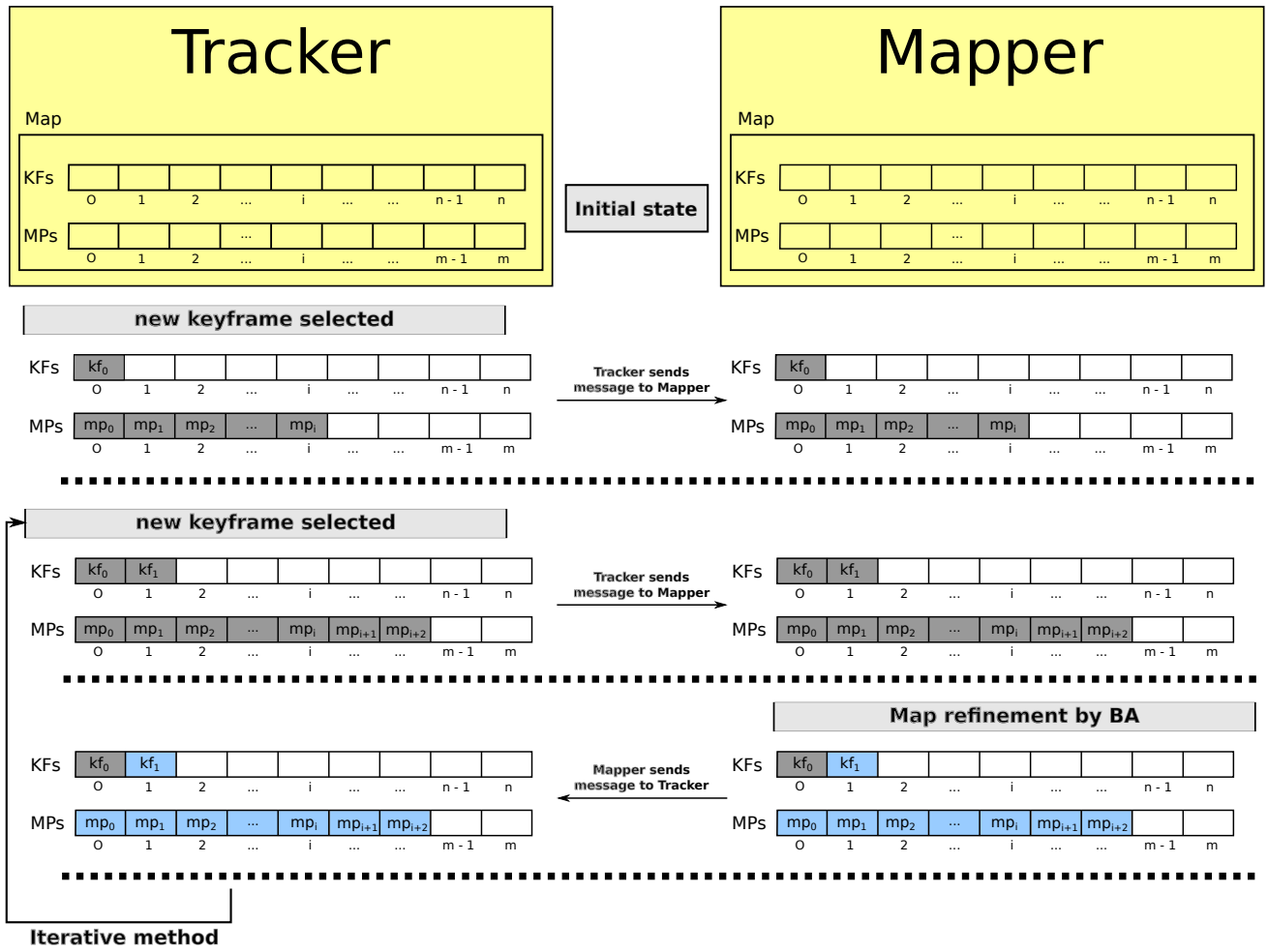


Fig. 3 Changes to translation objects during execution startup.

msg_mapDiff	
msg_KeyFrame[]	KeyFramesToAddorUpdate
msg_MapPoint[]	MapPointsToAddorUpdate
msg_Measurement[]	MeasurementsToAddorUpdate
int64[]	KeyFramesToDelete
int64[]	MapPointsToDelete
msg_Meas_id[]	MeasurementsToDelete

Fig. 4 ROS message used for the communication between Tracker and Mapper nodes.

5 Experiments

In this section we present an exhaustive evaluation of the proposed DS-PTAM system. We assessed it on two real public datasets using different computing hardware configurations.

5.1 Implementation and Benchmark platform

We developed DS-PTAM using the C++ programming language on the ROS Kinetic Kame framework [20]. The operating system used was Ubuntu 16.04 64-bit.

Three types of on-board hardware configuration were used:

- The first used a computer with an Intel Core Quad-core 3.40 GHz processor with 8 GB of RAM. This experiment aims to evaluate the performance of the proposed system on high-performance hardware.
- In the second case, experiments were carried out using hardware with lower computing capacity. In this case, a 3.4 GHz Dual-Core processor and 4 GB RAM was used.
- In the third case, experiments were carried out using hardware with even lower computing capacity to simulate the current on-board processing of mobile vehicles. In this case, a Dual-Core 2.4 GHz processor and 4 GB RAM were used.

In experiments using a remote base station, a Quadcore 1.80 GHz processor with 8 GB RAM running Ubuntu 16.04 64-bit operating system.

Both computers were connected via a wired network to simulate the distribution of computation between the mobile robot and the base station. Although in the common case when running DS-PTAM on a mobile robot, it would be connected by means of a wireless network, it can be noticed that the traffic is not significant since the images are not sent, but only the information of the features extracted in the images along with the modifications made to the map. We measured the traffic bandwidth used by DS-PTAM and it shown that during system initialization the packages transmitted by the Tracker node to the Mapper node have a peak of 2 MB, and then the traffic is stabilized in $\sim$ 250 KB. This behavior comes from the system map initialization where a high number of points are created and has to be transmitted to the Mapper node. Once the system is initialized only newly created points are transmitted, reducing notoriously the use of bandwidth. Similarly, it is possible to observe a peak of 330 KB in the first map refinement message sent by the Mapper node to the Tracker node during the initialization. Once the system was already started the bandwidth is reduced to $\sim$ 40 KB. Again, this behavior is explained because during system initialization several points are refined by the Mapper module, and later on the amount of points to be refined is reduced.

In cases where this type of architecture was not used, only the on-board hardware was used to run the system.

5.2 Datasets

In the experiments we used public domain datasets in the field of mobile robotics to evaluate vision-based SLAM systems. Each dataset has precise information of the robot's status at all times, known as *ground-truth*, which is obtained from different sensors such as GPS and IMU (Inertial Measurement Unit). This information makes possible to compare and evaluate the different SLAM methods.

Firstly, we used the 00 sequence of the KITTI [11] dataset, which consists of a set of stereo images captured by a pair of cameras mounted on a car that travels through an urban environment (outdoor environment). The path driven by the car in this sequence is approximately of 4 km. The images captured by the camera have a resolution of 1344×391 pixels and are obtained at a frame rate of 10 Hz.

In order to evaluate the system in indoor environments, the EuRoC [2] dataset was used. In particular, the *Machine Hall 01* sequence was used, which consists of a set of images taken by a *Micro Aerial Vehicle* (MAV), which travels within an industrial engine room for 182 seconds with a trajectory of 80.6 meters. The images have a resolution of 752×480 pixels and are obtained at a frame rate of 20 Hz. This dataset reflects the ideal case in which DS-PTAM distributed design makes more sense because of the limitations imposed by this type of vehicle make it clear that the equipment for computing mounted on it needs to be reduced.

5.3 DS-PTAM configurations

The DS-PTAM system gives rise to new configurations that were not possible in the S-PTAM original version. Throughout the experiments we will use three configurations namely: **DS-PTAM (Tracker)**, **DS-PTAM (Tracker + Mapper)** and **DS-PTAM**. These configurations are described below:

- **DS-PTAM (Tracker)**: Execution of the Tracker node only.
- **DS-PTAM (Tracker + Mapper)**: Running the Tracker and Mapper nodes on the on-board computer.
- **DS-PTAM**: Running the Tracker and Mapper nodes in a distributed manner.

The first of these configurations corresponds to the cases in which there are no high power computing devices or a base station that can perform the expensive computational task such as map refinement, which is performed by the Mapper node. To carry out the experiment with this configuration, the on-board computer was used, where both the main ROS node (master node) and the Tracker node were executed. It should be noted that for each of the experiments carried out, the dataset was replayed in real-time (emulating the stereo cameras of the robot) on the on-board hardware.

The second configuration is similar to the previous one with the difference that the execution of the Mapper node is incorporated in the on-board computer that executes the main ROS node (master node) and the Tracker node. In this way we will have a refined map and therefore as we will see the localization results will be more accurate. In this case the on-board computer will perform both localization and mapping tasks. It should be noted that in this case the message-passing is carried out as a local memory copy from one process to another.

Finally, the distributed configuration carries out the execution of the main ROS node (master node) and the Tracker node on the on-board computer while the execution of the Mapper node is carried out on the remote

base station. It should be made clear that it is necessary to achieve communication between both computers through the network, and for this purpose ROS provides mechanisms that involve operating system environment variables such as ROS_IP and ROS_MASTER_URI that should be correctly assigned. Specifically, ROS_IP contains the IP address of the equipment that executes a ROS node and that must be identified within the network, on the other hand ROS_MASTER_URI indicates to the ROS nodes the IP address and the listening port of the equipment where the main ROS node (master node) is being executed, this node provides communication with the rest of the nodes and allows the sending of messages between them.

5.4 Results

In order to understand the need to distribute the tasks of tracking and mapping in S-PTAM when it is desired to execute it in a hardware with low computing power, we present in Figure 5 the time comparison between both modules. The figure shows that the mapping module takes about 150 ms to refine the map and this processing time is almost four times longer than the tracking module.

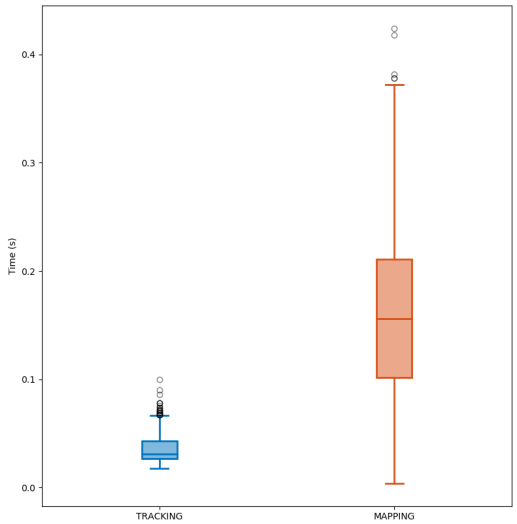


Fig. 5 Computation time boxplots for the tracking and mapping tasks of S-PTAM. Boxes represent interquartile range (IQR), whiskers reach to $-1.5 \times IQR$ and $1.5 \times IQR$, and the points represent data beyond those ranges, considered outliers. The line inside the box represents the median.

Figures 6 and 7 show the comparison between the estimated trajectories of the proposed system for different configurations. The DS-PTAM (Tracker), DS-PTAM (Tracker + Mapper), DS-PTAM and original S-PTAM configurations were used, comparing their trajectories against the ground-truth for each dataset. In

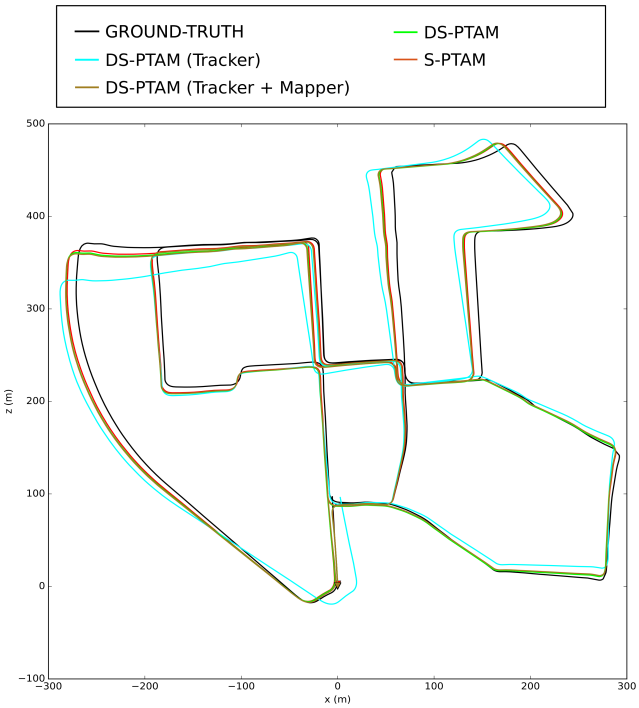


Fig. 6 Comparison of estimated paths for configurations DS-PTAM (Tracker), DS-PTAM (Tracker + Mapper), DS-PTAM and original S-PTAM over the sequence 00 of the KITTI dataset.

both cases ground-truth is represented in black while the light blue color represents the trajectory for the execution of the Tracker node on the on-board computer (this configuration is labeled "DS-PTAM (Tracker)"). The trajectory for the configuration corresponding to the execution of both nodes on the on-board computer, identified as "DS-PTAM (Tracker + Mapper)", is shown in brown. The path for executing original S-PTAM on the on-board computer is shown in red. Finally, shown in green, the distributed execution of the nodes where the Tracker node was executed in the on-board computer while the Mapper node was executed in the remote base station, this configuration is referenced in the figure as "DS-PTAM".

In order to evaluate the proposed system, an analysis of relative and absolute translation and rotation errors is performed. Below are the tables detailing the values of each experiment, in which different configurations were used on the both datasets. In all the tables, r and t are used to depict the rotation and translation errors, respectively. The sub-indexes abs, rel and $RMSE$ refer to absolute, relative and square root errors of the mean quadratic error, respectively. While Max refers to the maximum values obtained for each

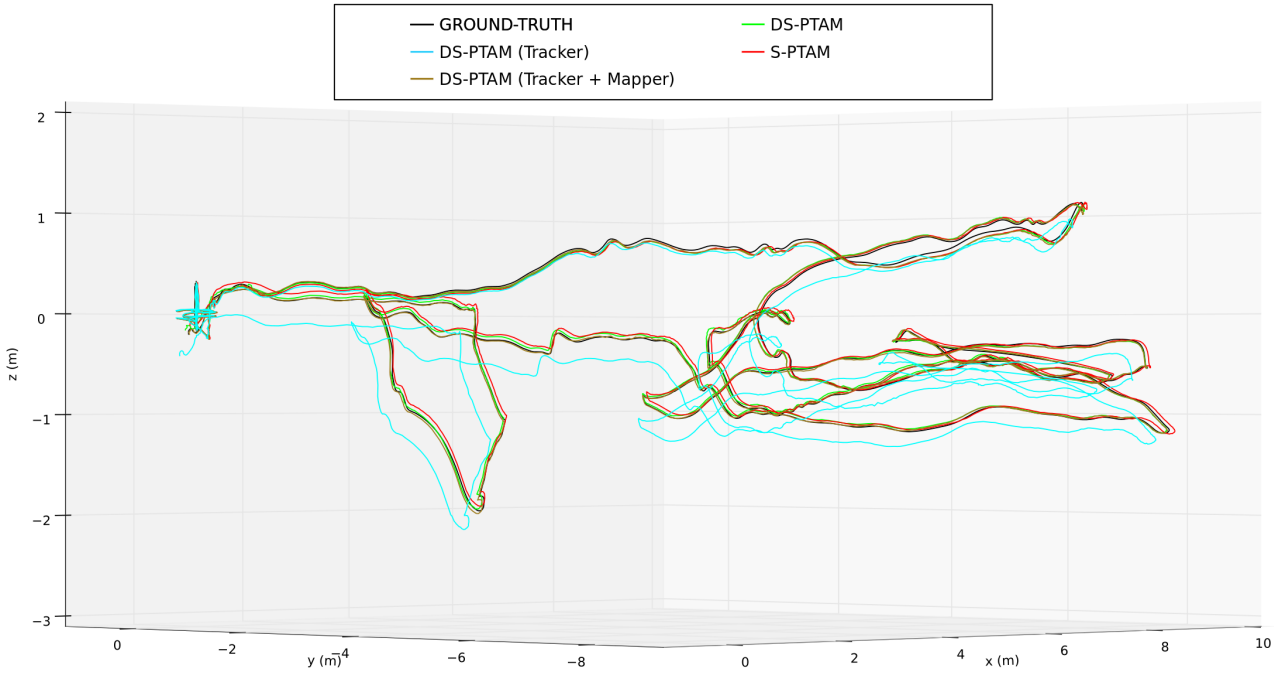


Fig. 7 Comparison of estimated paths for configurations DS-PTAM (Tracker), DS-PTAM (Tracker + Mapper), DS-PTAM and original S-PTAM over the sequence *Machine Hall 01* of the EuRoC dataset.

type of error. The last row of each table shows the average processing times that the localization node takes to process a couple of stereo images for each run. This task is taken as a point of comparison given that the localization must run in real time on a SLAM system. In each of the tables, the minimum values per row are highlighted indicating the best performance and lower error.

In Table 1 results obtained for the KITTI dataset are shown. There we can observe that the maximum absolute errors of translation and rotation for the configuration of the individually executed Tracker node (first column, first and fourth rows) are much higher than the values obtained for the rest of the cases. This behavior is due to the fact that the other configurations have map refinement functionality, resulting in a more precise representation of the environment, implying an improvement in localization. Competitive results are observed for DS-PTAM and S-PTAM, i. e. their behavior is very similar. One can add that DS-PTAM has the advantage of being able to run in a distributed way. In terms of time, the DS-PTAM (Tracker + Mapper) configuration have a higher computational cost, taking longer than all other experiments. This is due to the extra complexity (sending and receiving maps updates, translating ids, etc.) required by the distributed architecture that is not being exploited when running Tracker and Mapper nodes on the on-board computer. On the other hand, the DS-PTAM configuration is the fastest among all the DS-PTAM configurations. In particular it is important to remark the improvement of performance against the DS-PTAM (Tracker). This is because in the distributed configuration, the tracking is performed on an optimized map, making the localization procedure converge faster than on a not optimized map, as the case of the DS-PTAM (Tracker) configuration. This behavior persists on the rest of the experiment depicted below.

The results obtained for experiments on the EuRoC dataset are shown in Table 2. It can be observed that S-PTAM obtains the best results. This is because S-PTAM, working with shared memory, does not have the burden of distributed messaging and data synchronization architecture. It is important to note that the DS-PTAM (Tracker) configuration takes longer than S-PTAM to estimate the localization. Although this behavior is exactly the opposite of the expected behavior, it is because this configuration works on a map that is not being optimized throughout the experiment (due to the absence of the Mapper node) making the localization task take longer. In the last two columns of the table, we also show the results obtained by the state

Table 1 Comparison of errors and average localization times for experiments carried out on the dataset KITTI. The execution was carried out using a 3.40 GHz Quad-Core processor and 8 GB of RAM memory.

	DS-PTAM (Tracker)	DS-PTAM (Tracker + Mapper)	DS-PTAM	S-PTAM
Max t_{abs} (m)	48,028	17,043	16,643	**14,576**
Max t_{rel} (m)	**0,299**	0,312	0,307	0,312
t_{RMSE} (m)	0,030	0,029	0,030	0,029
Max r_{abs} (deg)	11,792	9,026	8,971	**8,789**
Max r_{rel} (deg)	2,197	2,195	**2,186**	2,200
r_{RMSE} (deg)	0,119	0,120	0,119	0,120
Tracking (s)	0,044	0,049	0,044	0,044

of the art system ORB-SLAM2 [17] over the EuRoC dataset. The ORB-SLAM2 system loses localization after a few seconds on real camera frame rate, given the high demanding processing time per frame. To allow ORB-SLAM2 to finish the entire sequence, we ran the system on off-line fashion mode.

Because the previous experiment used a hardware whose computational capacity is much higher than that available to the UAVs nowadays, a new experiment is presented on the EuRoC dataset in which a lower capacity hardware is used. In particular, a Dual-Core 3.4 GHz processor with 4 GB RAM is used to resemble the processing capacity of the mobile robot. A Quad-Core 1.80 GHz processor with 8 GB of RAM is used for the base station.

The results obtained for S-PTAM and for each of the DS-PTAM configurations are shown in Table 3. It can be observed that the performance of DS-PTAM (Tracker + Mapper) and S-PTAM are strongly affected by hardware limitation, while the performance of the DS-PTAM (Tracker) and DS-PTAM configurations is not affected with respect to the results of the previous experiment. It is worth mentioning that DS-PTAM generally achieves the best results as expected.

Note that DS-PTAM (Tracker + Mapper) and S-PTAM are not distributed architectures and thus the Mapper, along with the Tracker, run on the onboard computer, heavily increasing the processing requirements. This increase on the demanding resources is more noticeable in limited hardwares. Effectively, the lack of a high processing CPU makes the systems to do not be able to process every incoming frame, leading to a poor localization and causing the lose of localization.

Finally, an experiment was carried out to run the system using even more limited hardware, specifically a 2.4 GHz Dual-Core processor with 4 GB of RAM to resemble the processing capacity of the mobile robot. The Table 4 shows the values obtained for both localization errors and average times. It is worth noting that for this experiment the configurations corresponding to DS-PTAM (Tracker + Mapper) and S-PTAM did not succeed in completing the localization task, losing the robot localization estimation a few seconds after starting the experiment, and therefore they are not reported in the table. This loss of localization is due to the fact that these configurations require more processing power. This is an expected result that encourages the use of a distributed system when hardware with low computing capacity is available. Finally, it is worth mentioning that both DS-PTAM (Tracker) and DS-PTAM have similar performance.

6 Conclusion

In this paper we present a distributed SLAM system based on S-PTAM (*Stereo Parallel Tracking and Mapping*) [19], which is a method that addresses the problem of SLAM using an stereo camera.

Bearing in mind that S-PTAM divides the localization and mapping tasks into different threads (targeting a multi-core architecture), a new design was proposed that allows the execution of its components in a distributed manner (targeting a cluster architecture). In this way, greater versatility can be achieved when it comes to having different types of hardware with different processing capabilities. This new architecture allows then that a mobile robot can execute part of the processing corresponding to the localization module (*tracking*) onboard and execute the map maintenance module (*mapping*) in an external device that has more processing power, since this task has the highest computational cost. In this way, each module will have the full processing power of the device on which it is executed.

The novel distributed architecture also allows numerous configurations, such as running only the Tracker node, the Tracker node along with the Mapper node (on the same computer or remotely), or even more advanced configurations, such as having more than one Mapper node each analyzing a different section of the map or sharing Mapper nodes among several robots.

Table 2 Comparison of errors and average localization times for experiments carried out on the dataset EuRoC. The execution was carried out using a 3.40 GHz Quad-Core processor and 8 GB of RAM memory.

	DS-PTAM (Tracker)	DS-PTAM (Tracker + Mapper)	DS-PTAM	S-PTAM	ORB SLAM2 (off-line)	ORB SLAM2 (online)
Max t_{abs} (m)	0.340	0.257	0.266	**0.223**	0.255	1.023
Max t_{rel} (m)	0.05	0.05	0.05	0.05	0.02	0.30
t_{RMSE} (m)	0.005	0.005	0.005	**0.004**	0.003	0.008
Max r_{abs} (deg)	4.67	3.40	3.21	**2.85**	3.36	7.19
Max r_{rel} (deg)	1.04	0.84	1.12	**0.82**	0.45	6.00
r_{RMSE} (deg)	0.28	0.28	0.28	**0.22**	0.27	0.31
Tracking (s)	0.0392	0.0423	0.0380	**0.0379**	0.0720	0.0639

Table 3 Comparison of errors and average localization times for experiments carried out on the dataset EuRoC. The execution was carried out using a 3.40 GHz Dual-Core processor and 4 GB of RAM memory.

	DS-PTAM (Tracker)	DS-PTAM (Tracker + Mapper)	DS-PTAM	S-PTAM
Max t_{abs} (m)	0.417	0.718	**0.253**	0.441
Max t_{rel} (m)	**0.059**	0.331	0.061	0.248
t_{RMSE} (m)	0.006	0.021	**0.005**	0.015
Max r_{abs} (deg)	4.708	10.773	**3.259**	9.004
Max r_{rel} (deg)	**1.271**	5.387	1.337	4.992
r_{RMSE} (deg)	0.286	0.383	**0.283**	0.387
Tracking (s)	0.067	0.097	**0.065**	0.096

Table 4 Comparison of errors and average localization times for experiments carried out on the dataset EuRoC. The execution was carried out using a 2.40 GHz Dual-Core processor and 4 GB of RAM memory.

	DS-PTAM (Tracker)	DS-PTAM
Max t_{abs} (m)	0,515	**0,444**
Max t_{rel} (m)	0,303	**0,221**
t_{RMSE} (m)	0,018	**0,012**
Max r_{abs} (deg)	**6,641**	7,398
Max r_{rel} (deg)	**2,209**	4,039
r_{RMSE} (deg)	**0,318**	0,337
Tracking (s)	0.097	**0.089**

Experiments (using public domain datasets) were presented to validate the accuracy and performance of the proposed method. In these experiments, aspects related to the estimated trajectories, the location errors and the time obtained in each of them were evaluated. Based on these results, the benefits of distributing the computational load on several computers are corroborated, mainly in those cases involving mobile vehicles with low computing power, such as UAVs. In the experiments using powerful hardware, the results of the DS-PTAM were not advantageous, therefore new experiments were carried out on a less powerful hardware. The results were much more encouraging, evidencing the benefits of this design and justifying the added complexity of working under a distributed architecture when there is low computing capacity.

As future work it is proposed to include the *Loop Closing* module recently published in [18] to the version of DS-PTAM system here proposed. Additionally experiments should be made on different processor architectures (like ARM) with different processing power and memory. We expect the performance gain obtained by using DS-PTAM (relative to the original S-PTAM) to hold in other architectures. Finally a deep analysis is required on the effects of map growth and map inconsistencies between Tracker and Mapper nodes. Furthermore a study of the energetic efficiency of the whole system is also desired.

Acknowledgements This work is part of the *Development of a weed remotion mobile robot* project at CIFASIS (CONICET-UNR).

References

1. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): part II. IEEE Robotics Automation Magazine **13**(3), 108–117 (2006). DOI 10.1109/MRA.2006.1678144
2. Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W., Siegwart, R.: The EuRoC Micro Aerial Vehicle Datasets. International Journal Robotics Research **35**(10), 1157–1163 (2016). DOI 10.1177/0278364915620033. URL `http://dx.doi.org/10.1177/0278364915620033`
3. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, Present, and Future of Simultaneous Localization and

Mapping: Toward the Robust-Perception Age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016). DOI 10.1109/TRO.2016.2624754

4. De Croce, M., Pire, T., Bergero, F.: Diseño Distribuido de un Sistema de Mapeo y Localización basado en visión para un Robot Móvil. In: Anales de las IX Jornadas Argentinas de Robtica, pp. 116–121 (2017)
5. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. IEEE Robotics Automation Magazine **13**(2), 99–110 (2006). DOI 10.1109/MRA.2006.1638022
6. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (eds.) Proceedings of the European Conference on Computer Vision (ECCV), pp. 834–849. Springer International Publishing, Cham (2014). DOI 10.1007/978-3-319-10605-2_54. URL http://dx.doi.org/10.1007/978-3-319-10605-2_54
7. Engel, J., Stückler, J., Cremers, D.: Large-scale direct slam with stereo cameras. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 1935–1942. IEEE (2015). DOI 10.1109/IROS.2015.7353631. URL http://dx.doi.org/10.1109/IROS.2015.7353631
8. Forster, C., Lynen, S., Kneip, L., Scaramuzza, D.: Collaborative monocular slam with multiple micro aerial vehicles. In: IROS, pp. 3962–3970. IEEE (2013). URL http://dblp.uni-trier.de/db/conf/iros/iros2013.html#ForsterLKS13
9. Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, D.: SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. IEEE Transactions on Robotics **33**(2), 249–265 (2017). DOI 10.1109/TRO.2016.2623335. URL http://dblp.uni-trier.de/db/journals/trob/trob33.html#ForsterZGWS17
10. Gamage, R., Tuceryan, M.: An experimental distributed framework for distributed Simultaneous Localization and Mapping. In: Proceedings of the 2016 IEEE International Conference on Electro Information Technology (EIT), pp. 0665–0667. IEEE (2016). DOI 10.1109/EIT.2016.7535318
11. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision Meets Robotics: The KITTI Dataset. The International Journal of Robotics Research, IJRR **32**(11), 1231–1237 (2013). DOI 10.1177/0278364913491297. URL http://dx.doi.org/10.1177/0278364913491297
12. Jazwinski, A.H.: Stochastic processes and filtering theory. Mathematics in science and engineering. Academic press, New York (1970). URL http://opac.inria.fr/record=b1078357. UKM
13. Kelly, N.: A Guide to Ultrasonic Sensor Set Up and Testing Instructions, Limitations, and Sample Applications (2009). Application Note. Available online: http://www.egr.msu.edu/classes/ece480/capstone/fall09/group05/docs/ece480_dt5_application_note_nkelly.pdf
14. Klein, G., Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 1–10. IEEE Computer Society, Washington, DC, USA (2007). DOI 10.1109/ISMAR.2007.4538852. URL http://dx.doi.org/10.1109/ISMAR.2007.4538852
15. Mahdoui, N., Natalizio, E., Frémont, V.: Multi-UAVs network communication study for distributed visual simultaneous localization and mapping. In: Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), pp. 1–5. IEEE Computer Society (2016). DOI 10.1109/ICCNC.2016.7440564
16. Metropolis, N., Ulam, S.: The Monte Carlo Method. Journal of the American Statistical Association **44**(247), 335–341 (1949). DOI 10.1080/01621459.1949.10483310. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1949.10483310. PMID: 18139350
17. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. CoRR **abs/1610.06475** (2016). URL http://arxiv.org/abs/1610.06475
18. Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J., Jacobo Berlles, J.: S-PTAM: Stereo Parallel Tracking and Mapping. Robotics and Autonomous Systems (RAS) **93**, 27 – 42 (2017). DOI 10.1016/j.robot.2017.03.019
19. Pire, T., Fischer, T., Civera, J., De Cristóforis, P., Berlles, J.J.: Stereo parallel tracking and mapping for robot localization. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1373–1378 (2015)
20. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)
21. Riazuelo, L., Civera, J., Montiel, J.M.M.: C2TAM: A Cloud framework for cooperative tracking and mapping. Robotics and Autonomous Systems **62**(4), 401–413 (2014). DOI https://doi.org/10.1016/j.robot.2013.11.007. URL http://www.sciencedirect.com/science/article/pii/S0921889013002248
22. Strasdat, H., Davison, A.J., Montiel, J.M.M., Konolige, K.: Double window optimisation for constant time visual SLAM. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2352–2359 (2011). DOI 10.1109/ICCV.2011.6126517
23. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle Adjustment — A Modern Synthesis, chap. 21, pp. 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg (2000). DOI 10.1007/3-540-44480-7_21. URL http://dx.doi.org/10.1007/3-540-44480-7_21
24. Wendel, A., Maurer, M., Graber, G., Pock, T., Bischof, H.: Dense reconstruction on-the-fly. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), CVPR '12, pp. 1450–1457. IEEE Computer Society, Washington, DC, USA (2012). DOI 10.1109/CVPR.2012.6247833
25. Williams, R., Konev, B., Coenen, F.: Multi-agent Environment Exploration with AR.Drones, pp. 60–71. Lecture Notes in Computer Science. Springer International Publishing, Cham (2014). DOI 10.1007/978-3-319-10401-0_6. URL https://doi.org/10.1007/978-3-319-10401-0_6
26. Williams, R., Konev, B., Coenen, F.: Scalable distributed collaborative tracking and mapping with Micro Aerial Vehicles. In: Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3092–3097 (2015). DOI 10.1109/IROS.2015.7353804
27. Zanuttigh, P., Marin, G., Dal Mutto, C., Dominio, F., Minto, L., Cortelazzo, G.M.: Operating Principles of Structured Light Depth Cameras, pp. 43–79. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-30973-6_2. URL https://doi.org/10.1007/978-3-319-30973-6_2