

Article

A Hybrid Visual-Based SLAM Architecture: Local Filter-Based SLAM with KeyFrame-Based Global Mapping

Rodrigo Munguia ^{1,*}, Juan-Carlos Trujillo ^{1,†}, Edmundo Guerra ² and Antoni Grau ²

¹ Department of Computer Science (CUCET), University of Guadalajara, Guadalajara 44430, Mexico; juancarlos_max@hotmail.com

² Department of Automatic Control, Technical University of Catalonia UPC, 08034 Barcelona, Spain; edmundo.guerra@upc.edu (E.G.); antoni.grau@upc.edu (A.G.)

* Correspondence: rodrigo.munguia@academicos.udg.mx

† These authors contributed equally to this work.

Abstract: This work presents a hybrid visual-based SLAM architecture that aims to take advantage of the strengths of each of the two main methodologies currently available for implementing visual-based SLAM systems, while at the same time minimizing some of their drawbacks. The main idea is to implement a local SLAM process using a filter-based technique, and enable the tasks of building and maintaining a consistent global map of the environment, including the loop closure problem, to use the processes implemented using optimization-based techniques. Different variants of visual-based SLAM systems can be implemented using the proposed architecture. This work also presents the implementation case of a full monocular-based SLAM system for unmanned aerial vehicles that integrates additional sensory inputs. Experiments using real data obtained from the sensors of a quadrotor are presented to validate the feasibility of the proposed approach.



Citation: Munguia, R.; Trujillo, J.-C.; Guerra, E.; Grau, A. A Hybrid Visual-Based SLAM Architecture: Local Filter-Based SLAM with KeyFrame-Based Global Mapping. *Sensors* **2022**, *22*, 210. <https://doi.org/10.3390/s22010210>

Academic Editor: Jordi Palacín Roca

Received: 8 November 2021

Accepted: 25 December 2021

Published: 29 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Despite the great advances reported during the last years, the Simultaneous Localization and Mapping (SLAM) problem still attracts great attention from the robotics/AI/computer vision research community since it represents a fundamental milestone in the road to developing truly autonomous mobile robots and vehicles. In this context, the visual-based SLAM systems represent an interesting sub-class of SLAM methods due to the inherent characteristics of cameras as sensors. For instance, cameras, in general, provide a lot of information about the robot environment.

Most of the approaches for implementing visual-based SLAM systems, and also visual odometry (VO) systems (e.g., [1,2]), can fall into two broad categories: (i) Filter-based methods and (ii) optimization-based methods. The first class makes use of stochastic filter-based techniques, such as the Kalman Filter [3,4] Extended Kalman Filter [5–9], Unscented Kalman Filter [10–12], Information Filter [13,14], and Particle Filter [15–17], for concurrently estimating the state of the robot as well as the states of the visual features composing the map of the environment. The second class of visual-based SLAM systems, which is also referred to as key-frame-based methods, decouples the robot's localization process from the mapping process and reformulates both problems, the localization, and mapping as optimization problems. Examples of this kind of approach are [18–24].

In our opinion, both approaches present their own advantages and drawbacks. For instance, the filter-based SLAM methods make use of theoretically well-founded stochastic estimation tools coming from the systems and control theory. This facilitates the analysis of important system properties as the observability (e.g., [25–27] or the stability and convergence (e.g., [28,29]), which makes it possible to design SLAM systems based on

sound theoretical foundations. In this sense, optimization-based SLAM methods are in general built following a more heuristically design methodology, and therefore there is a lack of theoretical studies (math proof-type) supporting the operation of these kinds of methods. Moreover, filter-based SLAM methods are very well suited for incorporating aiding information from different sensory sources (e.g., Altimeter, IMU, range sensors, etc.) due to the data fusing nature of stochastic filters. In this sense, optimization-based SLAM methods require in general more ad-hoc solutions for incorporating data from other sensors (e.g., [30]).

On the other hand, perhaps the main drawback of filter-based SLAM methods is related to the fact that its computational requirements scale poorly as the size of the state vector increases when incorporates new map features. Although for small maps (of typically 100 features) the computational requirements are even lower for filter-based methods than those needed for optimization-based methods (see [31]), when it is necessary to build larger maps containing several hundred or even thousands of features, the filter-based methods are unable to maintain a real-time performance using consumer-degree hardware. This is when a clear advantage of the optimization-based SLAM methods becomes evident. Although optimization techniques are more computational power-consuming for small maps, their decoupled localization-mapping architecture and the use of local-optimization strategies make these kinds of methods scale computationally better when the number of map features becomes unmanageable for filter-based methods.

In this work, a new visual-based SLAM hybrid architecture is proposed which aims to take the advantages while at the same time overcoming the drawbacks of both methodologies: the filter-based SLAM methods and the optimization-based SLAM methods. The basic idea is to use each technique for what we consider is better suited according to its strengths: the filter-based technique for implementing a local SLAM process, and the optimization-based technique for building and maintaining a consistent global map of the environment.

In Section 2 the proposed architecture is presented in a detailed manner. Section 3 presents a full monocular-based SLAM system for unmanned aerial vehicles, which integrates altimeter and range measurements, as an implementation case of the proposed architecture. Section 4 presents the experimental results obtained from real data captured from the sensors of a quadrotor. Final remarks are given in Section 5.

2. Proposed Architecture

The proposed visual-based SLAM system is composed of three processes running concurrently: (i) a local SLAM process, (ii) a global mapping process, and (iii) a loop correction process (see the Figure 1).

The local SLAM process implements a filter-based visual-based SLAM system with state vector-size bounded to maintain real-time operation. By itself, this process produces up-to metric scale (world referenced) estimates of both, the camera-robot state, and a local map of features. But in this case, since old features are removed from the vector state, to maintain real-time operation, previously visited areas of the environment can not be recognized, and thus the accumulated position drift can not be corrected by the same process alone. In this sense, the two other processes (global mapping and loop correction) will be dedicated to build and maintain a global and persistent map of the environment as well as correcting the accumulated drift when loops are detected. Some adaptations as the Key-frame selection and the use of anchors from the global map, which will be better explained later, are introduced to the filter-based SLAM method that allows interfacing with the other processes. The local SLAM process can also be conceived as a complex virtual sensor capable to provide 3D odometry information all together with visual and spatial information of the environment.

The global mapping process takes as input the Key-frames produced by the local SLAM process to create and maintain a global and persistent map of the environment. This process runs asynchronously and at a lower operation rate than the local SLAM process.

This process also implements an optimization-based technique as the bundle adjustment to optimize the global map when new Key-frames are available. The map features composing the global map will be called anchors. Besides the (local) state-vector features, the local SLAM process can also make use of the (global) anchors as landmarks to correct its state. Since the global map is built upon the Key-frames produced by the local SLAM, it still will have the same drift as the position estimates of the latter.

The loop correction process is intended to minimize the drift accumulated simultaneously by the global map and the local SLAM, by detecting and closing the trajectory loops. This process runs asynchronously to the other two processes. It takes the current camera frame used by the local SLAM process and tries to associate it with previous key-frames stored by the global mapping process by means of visual descriptors. If a match is founded, indicating that this area of the environment was previously visited, then a corrected position of the camera-robot is computed. The computed camera-robot pose is used to correct the global map drift by means of a global optimization technique as the graph-SLAM, as well it is used to correct the system state estimates of the local SLAM.

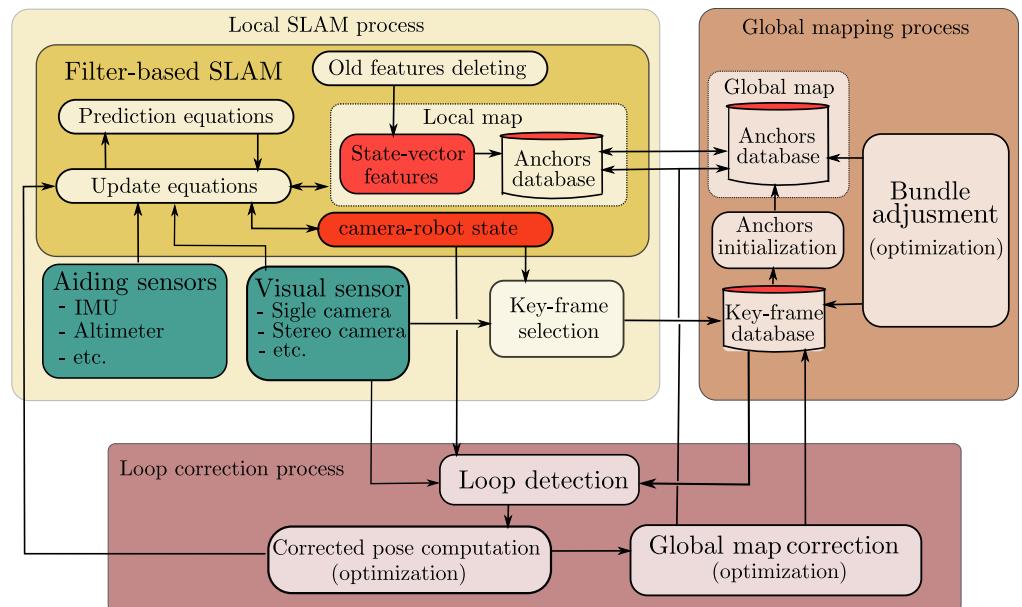


Figure 1. Proposed architecture.

In the following subsections, the proposed architecture will be described in depth: First, in Section 2.1 the local SLAM process is presented, then in Section 2.2 the global mapping process is presented, and finally in Section 2.3 the loop correction process is presented.

2.1. Local SLAM Process

If the following variable-size state vector \mathbf{x} is defined:

$$\mathbf{x} = [\mathbf{x}_r^T \quad \mathbf{y}_1^T \quad \dots \quad \mathbf{y}_n^T]^T \quad (1)$$

which is composed by the state of the camera-robot \mathbf{x}_r and the states of n map features \mathbf{y}_i , $i \in \{1, \dots, n\}$. The i th feature x_i can use any parameterization (e.g., Euclidean, Inverse-depth, etc.). And given:

- (i) a continuous or discrete state-space model of the dynamics of the system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad \text{or} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \quad (2)$$

where \mathbf{u} is the input vector.

(ii) an initialization function of new map features:

$$\mathbf{y}_{new} = \mathbf{f}(\mathbf{x}_r, \mathbf{z}_i) \quad (3)$$

(iii) a series of measurements \mathbf{z} for each sensor with measurement model $\hat{\mathbf{z}}$:

$$\hat{\mathbf{z}} = \mathbf{h}(\mathbf{x}) \quad (4)$$

(iv) knowledge of the initial state \mathbf{x}_0 and the initial covariance matrix \mathbf{P}_0 :

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathbb{E}\{\mathbf{x}_0\} \\ \mathbf{P}_0 &= cov(\mathbf{x}_0, \mathbf{x}_0) = \mathbb{E}\{(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T\} \end{aligned} \quad (5)$$

where $\mathbb{E}\{\cdot\}$ is the expected value.

Then, any general filter-based method (KF, EKF, UKF, Particle Filter, etc.) able to compute an estimate of the vector state $\hat{\mathbf{x}}$

$$\hat{\mathbf{x}} = \mathbb{E}\{\mathbf{x}\} \quad (6)$$

can be used as a basis to implement the local SLAM process. To adapt the filter-based SLAM method to be used in the proposed architecture, the following functionalities must be implemented:

- Deleting of old features. It is well known that the computational cost of filter-based SLAM methods scales poorly as the size of the state vector increases. Therefore, “old” features that are left behind by the movement of the robot-camera must be removed from the vector state \mathbf{x} and covariance matrix \mathbf{P} to maintain a stable computational cost (see [32]).
- Observation of anchors. We will call *anchors* to the map features \mathbf{a}_i whose state and uncertainties are not stored respectively in the state vector \mathbf{x} and the covariance matrix \mathbf{P} . Similar to a map feature, an anchor can be any 3d static point of the environment that can be visually detected and tracked frame to frame (e.g., a corner of a desktop, or the edge of a rock). The difference to a map feature is that the anchor is considered to be a fixed reference for estimating the camera pose. That is, when an anchor \mathbf{a}_i is observed, the state \mathbf{x} of the camera-robot is affected through the filter update, but the state of the anchor remains the same. Thus, the anchors act like “fixed” visual landmarks for the filter-SLAM estimation process. Anchors \mathbf{a}_i can use the same or different parameterization than state vector features \mathbf{x}_i , but a new measurement model $\hat{\mathbf{z}}_a$ (see Equation (7)) that depends on the state of the camera-robot \mathbf{x}_r and the constant parameter \mathbf{a}_i must be defined.

$$\hat{\mathbf{z}}_a = \mathbf{h}_a(\mathbf{x}_r, \mathbf{a}_i) \quad (7)$$

The global map is composed only of anchors, so these will act as an interface between the local SLAM and the global mapping process. For instance, if the global mapping process modifies the position of the anchors observed by the camera, then the state \mathbf{x} of the local SLAM will be affected accordingly.

- Key-frame selection. Key-frames \mathcal{K}_j are frames captured by the camera which are selected regularly among the video stream according to some specific criteria (e.g., spatial [18] or visual [19]). Each j th key-frame \mathcal{K}_j will store the camera-robot state \mathbf{x}_r at the moment that the frame was captured. Hereafter let define \mathbf{x}_{r_j} as the camera-robot state associated with the j th key-frame. After a Key-frame is selected it is sent to the Global mapping module to be processed.
- Position measurement update. To provide an interface with the loop correction process, a position update stage must be implemented to the local SLAM process. In this case, a position measurement model of the following type must be defined:

$$\hat{\mathbf{z}}_p = [x, y, z]^T = \mathbf{h}_p(\mathbf{x}) \quad (8)$$

where $[x, y, z]$ is the position of the camera-robot expressed in the world reference frame. The corrected camera-robot positions computed by the loop correction process, are incorporated into the local SLAM by mean of this update stage.

2.2. Global Mapping Process

First let us define $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_n\}$ as the set of n key-frames generated by the local SLAM process and which are stored by the global mapping process. Also let us define the *global map* $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$ as the set of m anchors stored and processed by the global mapping process.

The following basic capabilities must be implemented by the global mapping process.

- Anchors initialization. As new key-frames arrive from the local SLAM module, new anchors are computed. Each key-frame \mathcal{K}_j can represent a camera view since it has associate visual and spatial information. Therefore, a multiview geometry technique (e.g., [33]) can be used to compute the position of new anchors. For instance, a new subset of anchors \mathcal{A}_{new} can be computed using a stereo-based triangulation technique, where $\mathcal{A}_{new} = f(\mathcal{K}_j, \mathcal{K}_{j-1})$ (see Figure 2).

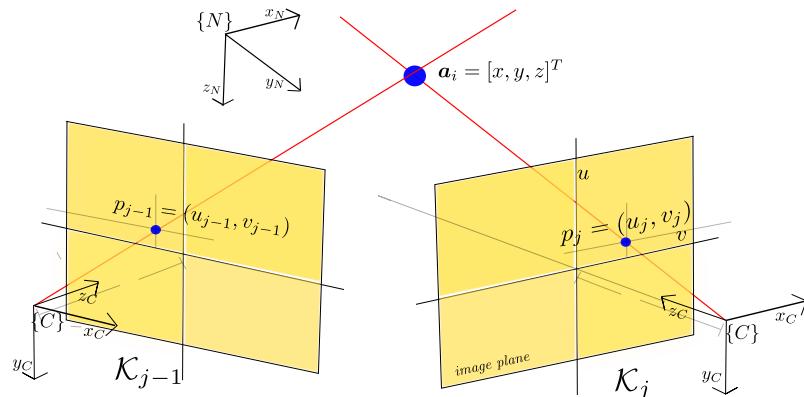


Figure 2. A stereo-based triangulation technique can be used for computing new anchors \mathbf{a}_i . In this case, the camera state information associated with a pair of key-frames $(\mathcal{K}_j, \mathcal{K}_{j-1})$, and the projections (p_j, p_{j-1}) are used to compute the 3d position of anchor \mathbf{a}_i .

- Bundle adjustment. In order to refine the global map, a local bundle adjustment technique is used (e.g., [34]). Assume that n anchors are seen (projected) in m key-frames, and let \mathbf{v}_{ij} be the measured projection of the i th anchor on the j th key-frame. Also let $\mathbf{h}_a(\mathbf{x}_{r_j}, \mathbf{a}_i)$ be the predicted projection of the i th anchor on the j th key-frame, where \mathbf{x}_{r_j} is the camera-robot state associated with the j th key-frame. Then, bundle adjustment minimizes the total reprojection error with respect to n anchors belonging to \mathcal{A} and the m camera states \mathbf{x}_{r_j} associated with the m key-frames contained to \mathcal{K} , or

$$\min_{\mathbf{x}_{r_j}, \mathbf{a}_i} \sum_{i=1}^n \sum_{j=1}^m d(\mathbf{h}_a(\mathbf{x}_{r_j}, \mathbf{a}_i), \mathbf{v}_{ij})^2 \quad (9)$$

where $d(\mathbf{y}, \mathbf{x})$ denotes the Euclidean distance between the image points represented by vectors \mathbf{y} and \mathbf{x} . If the anchor i is not visible in the key-frame j then $d(.,.) = 0$.

The operation rate of the global mapping process is not restricted by the frame-rate operation of the local SLAM process, but still, it is desirable to maintain a reasonable operation rate. When the number of key-frames and the number of anchors increases the computational cost of the bundle adjustment also increases. Therefore in practice, only a subset of the last key-frames are considered in the minimization problem, hence the name of local bundle adjustment.

2.3. Loop Correction Process

This process is intended to detect previously mapped areas of the environment by using visual information to minimize the position drift accumulated by both, the camera-robot state and global map estimates. In this case, the process can be implemented in fact in several ways (e.g., [35–37]). One of these can be for instance to apply a global bundle adjustment when the projection of old anchors, belonging to previously mapped areas, are detected and matched in recent camera frames. However in many cases, due to the great numbers of anchors and key-frames, the minimization problem involving the bundle adjustment can imply a considerable computational cost.

Based on the particularities of the proposed system architecture, the loop correction process can be implemented through the following basic functionalities:

- Loop detection. Let define \mathcal{F} as the current frame captured by the camera, which has the current camera-state \mathbf{x}_{r_F} associated with it. Then, different heuristic criteria can be established, but in general, a loop is detected if enough number of visual features, belonging to \mathcal{F} , are visually matched against some previous (old) key-frame \mathcal{K}_j . Given the above, let defined the matched key-frame as \mathcal{K}_M and its associated camera-robot state as \mathbf{x}_{r_M} .
- Corrected camera-robot pose computation. If a loop is detected, this functionality is intended to compute the corrected (relatively to the previously mapped area) camera-robot position. Let define the subset of anchors \mathbf{a}_i that have a projection $\mathbf{h}_a(\mathbf{x}_{r_M}, \mathbf{a}_i)$ on the key-frame \mathcal{K}_M as $\mathcal{A}_{\mathcal{K}_M}$, where $\mathcal{A}_{\mathcal{K}_M} \subset \mathcal{A}$. Let define the subset of anchors \mathbf{a}_i belonging to $\mathcal{A}_{\mathcal{K}_M}$ that have a measured (matched) projection $\mathbf{h}_a(\mathbf{x}_{r_F}, \mathbf{a}_i)$ on the current frame \mathcal{F} as $\mathcal{A}_{\mathcal{F}}$, where $\mathcal{A}_{\mathcal{F}} \subset \mathcal{A}_{\mathcal{K}_M}$. Moreover, let \mathbf{v}_i be the measured projection $\mathbf{h}_a(\mathbf{x}_{r_F}, \mathbf{a}_i)$ of the i th anchor on the frame \mathcal{F} . Then, the minimization of total reprojection error with respect to n anchors belonging to $\mathcal{A}_{\mathcal{F}}$ and the camera states \mathbf{x}_{r_F} associated with the frame \mathcal{F} , or

$$\min_{\mathbf{x}_{r_F}} \sum_{i=1}^n d(\mathbf{h}_a(\mathbf{x}_{r_F}, \mathbf{a}_i), \mathbf{v}_i)^2 \quad (10)$$

is called Perspective-n-Point (or PnP) problem (see [38]). Now let define the camera-robot state that minimizes the PnP problem as \mathbf{x}_{r_c} . We will also refer to \mathbf{x}_{r_c} as the corrected camera-robot state or corrected pose. The state $\mathbf{x}_{r_c} = [\mathbf{x}_{cp}^T, \mathbf{x}_{co}^T]^T$ is composed by the corrected position \mathbf{x}_{cp} and the corrected orientation \mathbf{x}_{co} of the camera-robot. When a corrected camera-robot pose is computed, a filter update is executed in the local SLAM process, to update the state of the local SLAM with \mathbf{x}_{cp} as position measurement.

- Global map correction. If a corrected camera-robot state \mathbf{x}_{r_c} is available, then a Graph-based SLAM technique (see [39]) can be used to accordingly correct the camera-robot states \mathbf{x}_{r_i} associated with the key-frames contained in \mathcal{K} . Now, let define $\mathbf{x}_g = [\mathbf{x}_{r_1}, \mathbf{x}_{r_2}, \dots, \mathbf{x}_{r_n}]$ as the vector of parameters to be optimized by the graph-based SLAM, where \mathbf{x}_{r_i} describe the pose of node i . Note that in \mathbf{x}_g , one parameter \mathbf{x}_{r_i} correspond to the state \mathbf{x}_{r_M} of key-frame which was matched during the loop detection. Also in \mathbf{x}_g , the last parameter \mathbf{x}_{r_n} correspond to state \mathbf{x}_{r_F} of the camera-robot when the current frame \mathcal{F} was captured. Let \mathbf{z}_{ij} and Ω_{ij} be respectively the mean and the information matrix of a virtual measurement between the node i and the node j . And let $\hat{\mathbf{z}}_{ij}(\mathbf{x}_{r_i}, \mathbf{x}_{r_j})$ be the prediction of a virtual measurement given a configuration of the nodes \mathbf{x}_{r_i} and \mathbf{x}_{r_j} . Finally let $\mathbf{e}_{ij}(\mathbf{x}_{r_i}, \mathbf{x}_{r_j})$ be an error function that computes the difference between the expected measurement $\hat{\mathbf{z}}_{ij}$ and the actual measurement \mathbf{z}_{ij} :

$$\mathbf{e}_{ij}(\mathbf{x}_{r_i}, \mathbf{x}_{r_j}) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_{r_i}, \mathbf{x}_{r_j}) \quad (11)$$

The goal of the graph-based SLAM (see Figure 3) is to found the configuration of nodes \mathbf{x}_g^* that solve the following minimization problem:

$$\min_{\mathbf{x}_g} \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij} \quad (12)$$

where \mathcal{C} is the set of all pairs for which an observation (constraint) \mathbf{z} exist.

For the proposed architecture, two kinds of virtual observation are defined:

- Visual odometry measurements \mathbf{z}_{ij} are defined by the relative transformation between consecutive camera-robot states, so $\mathbf{z}_{ij} = T(\mathbf{x}_{r_i}, \mathbf{x}_{r_j})$, where $j = i + 1$. In this case, there will be $n - 1$ visual odometry measurements linking all the camera-robot states in \mathbf{x}_g . The prediction $\hat{\mathbf{z}}_{ij}(\mathbf{x}_{r_i}, \mathbf{x}_{r_j})$ can be set to zero $\hat{\mathbf{z}}_{ij} = \mathbf{0}$ if not visual odometry model is available (or only by simplicity).
- A single closed-loop measurement, \mathbf{z}_{ij} which is defined by the relative transformation between the corrected camera-robot pose \mathbf{x}_{r_c} and the state of the matched key-frame $\mathbf{x}_{r_j} = \mathbf{x}_{r_M}$, or $\mathbf{z}_{ij} = T(\mathbf{x}_{r_c}, \mathbf{x}_{r_M})$. In this case the prediction $\hat{\mathbf{z}}_{ij}$ is defined by the relative transformation between the current camera-robot pose $\mathbf{x}_{r_n} = \mathbf{x}_{r_F}$ and the state of the matched key-frame $\mathbf{x}_{r_j} = \mathbf{x}_{r_M}$, or $\hat{\mathbf{z}}_{ij} = T(\mathbf{x}_{r_F}, \mathbf{x}_{r_M})$. In this case, for \mathbf{z}_{ij} and $\hat{\mathbf{z}}_{ij}$, $i = n$ since it corresponds to the current camera-robot state.

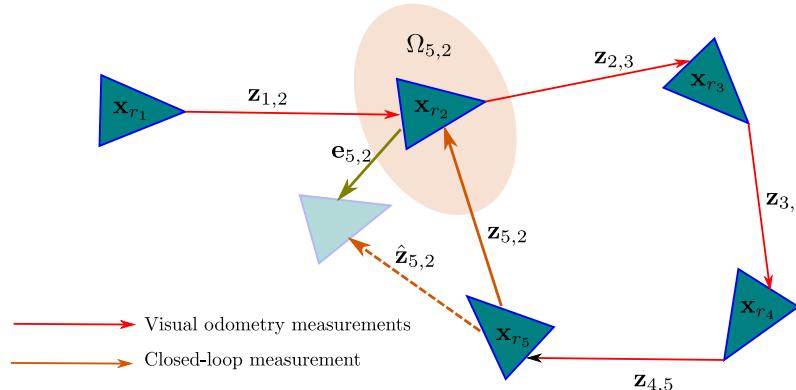


Figure 3. Visual representation of a Graph-based SLAM problem.

Each anchor $\mathbf{a}_i \in \mathcal{A}$ is linked to a specific key-frame $\mathcal{K}_i \in \mathcal{K}$ (typically the first key-frame when the anchor was initialized). If $\mathbf{x}_{r_i} = [\mathbf{x}_p^T, \mathbf{x}_o^T]^T$ and $\mathbf{x}_{r_i}^* = [\mathbf{x}_p^{*T}, \mathbf{x}_o^{*T}]^T$ is respectively the camera-robot state before and after of the graph-based SLAM technique is applied, then, each anchor \mathbf{a}_i position can be corrected with:

$$\mathbf{a}_i = \mathbf{a}_i + (\mathbf{x}_p^* - \mathbf{x}_p) \quad (13)$$

Observe that when a loop is detected, the state of the local SLAM process is corrected directly by updating the state with the corrected pose \mathbf{x}_{cp} , and indirectly by the observation of the corrected anchors \mathbf{a}_i composing the global map.

3. Implementation Case

In this section an implementation case of the visual-SLAM architecture proposed in Section 2 is presented. It is important to recall that the specific system described here does not represent the unique manner to implement the proposed architecture, but only represents a practical validation example among many possibilities.

The system described in this section is a monocular-based SLAM system for MAVs (Micro Aerial Vehicles). Besides the monocular camera, the system includes a barometer for integrating (absolute-referenced) altitude measurements and a range sensor for incorporating information depth. The system provides metrics estimates of the camera-robot state and the map of the environment, and it has loop-closing capabilities.

3.1. Local SLAM

The Local SLAM process is based on the authors' previous work [40]. In this case, an Extended Kalman Filter (EKF) is used for estimating the state of a MAV equipped with a down-facing monocular camera, a barometer, and an ultrasonic range finder as well as for estimating a local map of the environment of the MAV. The camera is mounted over a servo-controlled gimbal which counteracts the changes in the attitude of the MAV. The range sensor is also mounted on the gimbal and parallel aligned with respect to the optical axis of the camera (see Figure 4).

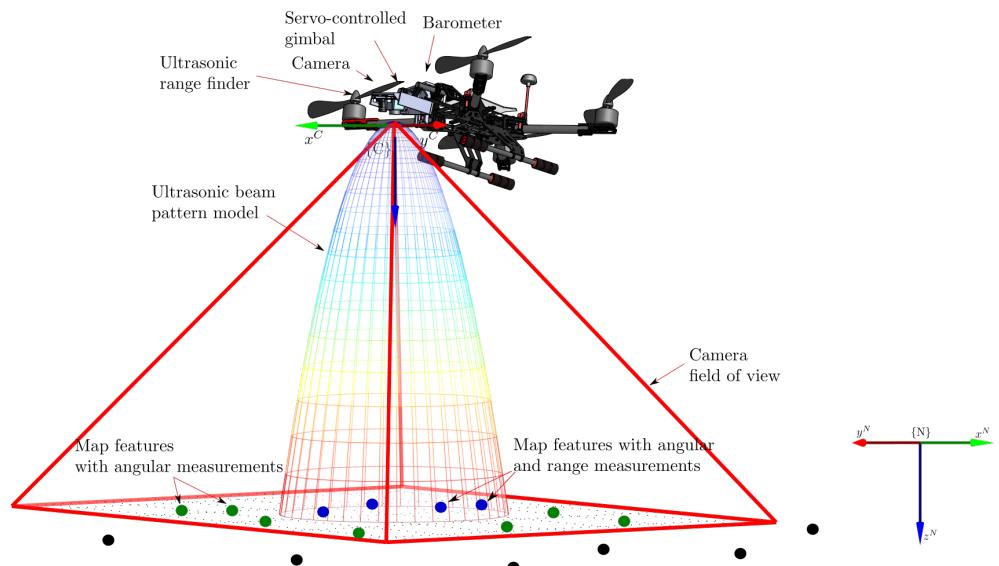


Figure 4. As an implementation case, a monocular-based SLAM system for a MAV equipped with a down-facing monocular camera, a barometer, and an ultrasonic range finder is presented.

Altitude measurements provided by the barometer are integrated for incorporating metric information into the system improving the observability of the metric scale. The range information provided by the ultrasonic sensor is integrated into the system also for improving the observability of the metric scale as well as for improving the robustness of the initialization of map features.

The elements of the state vector \mathbf{x} defined in Equation (1) are:

$$\mathbf{x}_r = [p_x \ p_y \ p_z \ v_x \ v_y \ v_z]^T \quad \mathbf{y}_i = [p_{x_i} \ p_{y_i} \ p_{z_i}]^T \quad (14)$$

where $\mathbf{p}^N = [p_x, p_y, p_z]$ represent the position of the camera-robot expressed in the navigation frame N , $\mathbf{v}^N = [v_x, v_y, v_z]$ represent the velocity of the camera-robot expressed in the navigation frame N . And $[p_{x_i}, p_{y_i}, p_{z_i}]$ represent the position of the i map feature expressed in the navigation frame N . Note that due to the permanent down-facing camera restriction, the problem is simplified to consider only the position estimation of the camera-robot.

The discrete state-space model (Equation (2)) is:

$$\left\{ \begin{array}{l} \mathbf{p}_{k+1}^N = \mathbf{p}_k^N + \mathbf{v}_k^N \Delta t \\ \mathbf{v}_{k+1}^N = \mathbf{v}_k^N + \mathbf{V}^N \\ \mathbf{y}_{1[k+1]} = \mathbf{y}_{1[k]} \\ \vdots \\ \mathbf{y}_{n[k+1]} = \mathbf{y}_{n[k]} \end{array} \right. \quad (15)$$

where Δt is the time step, and $\mathbf{V}^N = \sigma_a^2 \Delta t$ represents an unknown linear velocity impulse with acceleration zero-mean and known-covariance Gaussian processes σ_a .

The model, Equation (3), for initializing new map features is:

$$\mathbf{y}_{new} = \mathbf{p}^N + d\mathbf{R}^{CN} \begin{bmatrix} c_x - u' \\ f \\ c_y - v' \\ f \\ 1 \end{bmatrix}^T \quad (16)$$

$$\text{with } \begin{bmatrix} u' \\ v' \end{bmatrix} = \mathbf{f}_d^{-1}(u, v, k_1, \dots, k_n) \quad (17)$$

where $\mathbf{R}^{CN} = (\mathbf{R}^{NC})^T$, and d is the approximate feature depth computed from the range sensor.

Every time that a range reading is available, new map features are initialized using the next camera frame available. First ORB keypoints [41] are detected on the frame, then a subset of strong keypoints is selected using the methodology proposed in [42]. For each strong keypoint, a new map feature is initialized in the local SLAM system state using model in Equation (16). Map features lying inside the beam pattern of the range sensor are initialized with smaller depth uncertainty than features lying outside of it. For more details about the initialization process see [40]. The corresponding ORB descriptor is stored and associated with each new map feature.

The measurement model $\hat{\mathbf{z}}_{\mathbf{y}_i} = [u, v]^T$ (Equation (4)), that projects a 3D map feature \mathbf{y}_i , with position $[p_{x_i}, p_{y_i}, p_{z_i}]$, to the image coordinates $[u, v]$ of a camera located at $\mathbf{p}^N = [p_x, p_y, p_z]$ is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{f}_d(u', v', k_1, \dots, k_n) \quad \text{with} \quad s \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{33} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \end{bmatrix} \begin{bmatrix} p_{x_i} \\ p_{y_i} \\ p_{z_i} \\ 1 \end{bmatrix} \quad (18)$$

where r_{ij} is the i - j element of the known (by the gimbal assumption) rotation matrix \mathbf{R}^{NC} which allows transforming from the navigation frame N to the camera frame C . Let \mathbf{f}_d the camera distortion model depending on the distortion parameters k_1, \dots, k_n , and let f and (c_x, c_y) be respectively the focal length and the principal point of the camera. In this work, the distortion model described in [43] is used, and the intrinsic parameters of the camera are known by calibration.

Every time a new camera frame is available ORB keypoints-descriptors are computed all over the image. Map features, that have a projection $\hat{\mathbf{z}}_{\mathbf{y}_i}$ over the image, are attempted to be visually matched against the ORB descriptors computed in the current frame, using a FLANN-based matcher [44]. The successful matches represent the visual measurements $\mathbf{z}_{\mathbf{y}_i}$. Moreover, a validation step (e.g., RANSAC) can be added for discarding outliers.

The measurement model for updating altitude measurements obtained from the barometer is simply $\hat{\mathbf{z}}_{p_z} = p_z$. Every time a barometer measurement is available the filter is updated. For more details see [40].

The estimated state $\hat{\mathbf{x}}$ is computed using the typical loop of filter prediction-updates steps defined by the standard EKF-based SLAM methodology (see [45,46]), along with the required adaptations described in Section 2.1:

- Old features deleting. A map feature \mathbf{y}_i is removed from the state vector \mathbf{x} and the covariance matrix \mathbf{P} , when the ratio of unmatched-matched times of a map feature is high, or number of times that it is not considered to be matched is high.
- Observation of anchors. Anchors are parameterized in the same manner as state map features:

$$\mathbf{a}_i = [p_{x_i} \ p_{y_i} \ p_{z_i}]^T \quad (19)$$

and therefore the measurement model $[u, v]^T = \hat{\mathbf{z}}_a$ (7) is similar to (18), but in this case $[p_{x_i}, p_{y_i}, p_{z_i}]$ are fixed parameters and the jacobians does not depend on them. The local SLAM process owns a structure for storing local anchors. A local anchor is one that can be potentially projected into the current camera image-frame. Let \mathcal{A}_L be the

set of anchors belonging to the structure owned by the local SLAM process. Anchors are added to \mathcal{A}_L in two ways:

- The global mapping process copy anchors from \mathcal{A} to \mathcal{A}_L that are visually linked to the current camera frame. This process will be explained in more detail later.
- The map features contained in the state \mathbf{x} , whose position exhibits some good degree of convergence, are removed from the EKF state and transformed into anchors contained in \mathcal{A}_L . In this case, the following simple condition is proposed:

$$\frac{\sqrt{\sigma_{x_i}^2 + \sigma_{y_i}^2 + \sigma_{z_i}^2}}{\|\mathbf{p}^N - \mathbf{y}_i\|} < \lambda \quad (20)$$

If the above criteria is met, then the transformation $\mathbf{y}_i \rightarrow \mathbf{a}_i$ is carried out (simply $\mathbf{a}_i = \mathbf{y}_i$). Where $\sigma_{x_i}^2, \sigma_{y_i}^2, \sigma_{z_i}^2$ represent the variances of of the feature \mathbf{y}_i along each axis taken from the system covariance matrix, and λ is a threshold. In our implementation, a value of $\lambda = 0.1$ was used.

On the other hand, anchors are removed from \mathcal{A}_L in the following cases:

- An anchor is removed from \mathcal{A}_L if similar conditions that those for deleting local map features are accomplished.
- If the loop correction process has detected a loop closing condition and a corrected pose \mathbf{x}_{cp} is available, all the anchors in \mathcal{A}_L are deleted and replaced with visually linked anchors from the corrected global map.

In the same manner as state features, anchors have associated an ORB descriptor. Every time a new camera frame is available the anchors $\mathbf{a}_i \in \mathcal{A}_L$ are projected into the image frame, and they are attempted to be matched in the same manner as the local map features to determine visual measurements of anchors \mathbf{z}_a .

- Key-frame selection. A camera frame is selected as key-frame if two criteria are met:
 - The displacement of the camera-robot is bigger than some threshold t_k depending on the average depth of the n local map features, or

$$\frac{\|\mathbf{P}_k^N - \mathbf{P}_{k-1}^N\|}{\frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \mathbf{P}_k^N\|} > t_k \quad (21)$$

In our implementation, a value of $t_k = 0.15$ was used.

- A minimum number of features (or anchors) were visually matched at that frame. In our implementation, a value of 10 was used for this criteria.
- *Position measurement update*. If the loop correction process has detected a loop closing condition and therefore a corrected pose \mathbf{x}_{cp} is available, the filter is updated in a standard manner with the measurement model (8) and the measurement $\mathbf{z}_p = \mathbf{x}_{cp}$.

3.2. Global Mapping

Several functionalities implemented by the global mapping process and also the loop correction process makes use of a visibility graph (see [19]), that accounts for visual relations between key-frames. Let define the visibility graph \mathcal{V}_g as the symmetric matrix:

$$\mathcal{V}_g = \begin{bmatrix} 0 & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & 0 & w_{23} & \dots & w_{2n} \\ w_{31} & w_{32} & 0 & \dots & w_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & w_{n3} & \dots & w_{nn} \end{bmatrix} \quad (22)$$

where the component $w_{ij} = w_{ji}$ is the number of global map anchors $\mathbf{a}_i \in \mathcal{A}$, that have a projection $[u, v]^T = \hat{\mathbf{z}}_a = \mathbf{h}_a(\mathbf{x}_r, \mathbf{a}_i)$ in both key-frames, \mathcal{K}_i and \mathcal{K}_j . The number of visual

links of the i -key-frame is $w_{\mathcal{K}_i} = \sum_{j=1}^i w_{ij}$. Figure 5 show a visual graph obtained from an actual experiment. \mathcal{K}_1 is the first keyframe. Observe, that if the visual graph is interpreted by rows, from right to left (from recent to older key-frames), it can be inferred when the camera-robot return near to previously mapped areas.

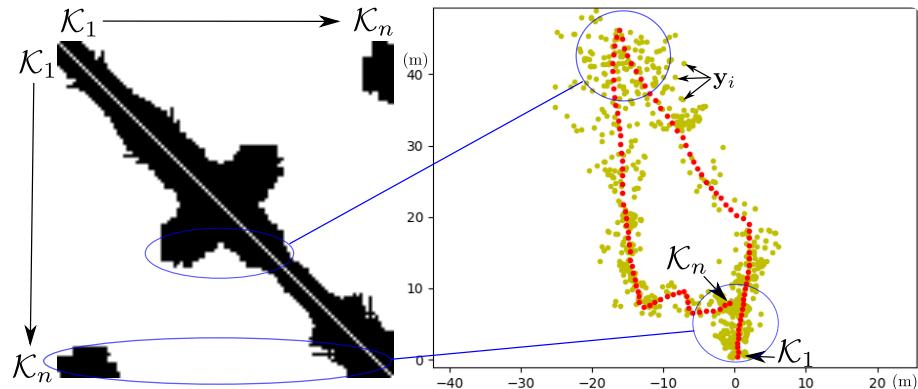


Figure 5. (Left): the plot of a matrix V_g illustrating the visual relations between 110 key-frames taken from an actual experiment (**right plot**). White pixels indicate no visual relations. Black pixels indicate visual relation (value $w_{ij} > 0$). Key-frames are indicated in red, global map anchors are indicated in yellow.

New key-frames \mathcal{K}_j , generated by the local SLAM process, are incorporated into the structure that stores the set of key-frames \mathcal{K} belonging to the global map. When new key-frames are available, the following procedure is executed by the global map process:

- Computing new anchors. If a new key-frame \mathcal{K}_j is available and the number of visual links of the previous key-frame $w_{\mathcal{K}_{j-1}}$ is below a threshold, then new anchors $\mathbf{a}_i \in \mathcal{A}_{new}$ are initialized by triangulation. First, visual matches are searched between the ORB descriptors of key-frames \mathcal{K}_j and \mathcal{K}_{j-1} . Then Outliers are removed using RANSAC. If $[u'_j, v'_j]$ and $[u'_{j-1}, v'_{j-1}]$ are respectively the (undistorted) projection of the anchor \mathbf{a}_i over the key-frames \mathcal{K}_j and \mathcal{K}_{j-1} , then the location of the anchor can be computed by triangulation as follows:

From model (18) we have that:

$$\begin{bmatrix} su' \\ sv' \\ s \end{bmatrix} = \mathbf{K} \begin{bmatrix} p_{x_i} \\ p_{y_i} \\ p_{z_i} \\ 1 \end{bmatrix} \quad \text{with} \quad \mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{33} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \end{bmatrix}$$

where $\mathbf{a}'_i = [p_{x_i}, p_{y_i}, p_{z_i}, 1]^T$, and $[p_{x_i}, p_{y_i}, p_{z_i}]$ is the position of the anchor \mathbf{a}_i expressed in the navigation frame N . Let \mathbf{k}_i^j be the i row of the projection matrix \mathbf{K} of the key-frame j . We have that $su' = \mathbf{k}_1^j \mathbf{a}'_i$, $sv' = \mathbf{k}_2^j \mathbf{a}'_i$, and $s = \mathbf{k}_3^j \mathbf{a}'_i = p_{z_i}$. Substituting the last expression, we have that: $p_{z_i} u' = \mathbf{k}_1^j \mathbf{a}'_i$ and $p_{z_i} v' = \mathbf{k}_2^j \mathbf{a}'_i$. Considering both projections $[u'_j, v'_j]$ and $[u'_{j-1}, v'_{j-1}]$, of the same anchor \mathbf{a}_i , the following linear system can be formed:

$$\begin{cases} p_{z_i} u'_j = \mathbf{k}_1^j \mathbf{a}'_i \\ p_{z_i} v'_j = \mathbf{k}_2^j \mathbf{a}'_i \\ p_{z_i} u'_{j-1} = \mathbf{k}_1^{j-1} \mathbf{a}'_i \\ p_{z_i} v'_{j-1} = \mathbf{k}_2^{j-1} \mathbf{a}'_i \end{cases} \quad (23)$$

The above linear system can be solved for $\mathbf{a}_i = [p_{x_i}, p_{y_i}, p_{z_i}]^T$. New anchors \mathcal{A}_{new} computed by triangulation are added to the global map \mathcal{A} .

- Adding anchors computed by the local SLAM. The anchors computed by the local SLAM process $\mathbf{a}_i \in \mathcal{A}_L$ are added to the global map \mathcal{A} .
- Visibility graph update. Every time a new key-frame \mathcal{K}_j is available, the visibility graph \mathcal{V}_g is updated. In this case, the visual links w_{ij} are updated by taking into account the projections of the new initialized anchors over previous key-frames, and also the projections of old anchors over the new key-frame \mathcal{K}_j .
- Bundle adjustment. First, let define the subset $\mathcal{K}_l \subset \mathcal{K}$ as the subset of m key-frames $\{\mathcal{K}_j, \mathcal{K}_{j-1}, \dots, \mathcal{K}_{j-m-1}\}$ that are visually linked to the most recent key-frame \mathcal{K}_j . A key-frame \mathcal{K}_j is visually linked to another key-frame \mathcal{K}_{j-i} if $w_{(j-i),j} \neq 0$. Moreover, let define the subset $\mathcal{A}_l \subset \mathcal{A}$ as the subset of n anchors $\mathbf{a}_i \in \mathcal{A}$ that have at least three measured projections \mathbf{v}_{ij} over three different key-frames $\mathcal{K}_j \in \mathcal{K}_l$. A measured projection \mathbf{v}_{ij} is determined by visually matching an anchor \mathbf{a}_i , with predicted projection $\mathbf{h}_a(\mathbf{x}_{r_j}, \mathbf{a}_i)$, on the key-frame \mathcal{K}_j using ORB descriptors. The global map is optimized by applying local bundle adjustment (Equation (9)) to anchors $\mathbf{a}_i \in \mathcal{A}_l$ and setting each camera state \mathbf{x}_{r_j} of the key-frames $\mathcal{K}_j \in \mathcal{K}_l$ as fixed parameters.
- Weak anchors deleting. Anchors that can not be matched in at least three key-frames are removed from the global map to maintain only anchors with a good likelihood to be visually matched when the camera-robot revisits previously mapped areas.
- Local map anchors updating. Every time the global map is optimized by the local bundle adjustment the set of anchors $\mathbf{a}_i \in \mathcal{A}_L$ owned by the local SLAM process is updated. In this case, the optimized anchors $\mathbf{a}_i \in \mathcal{A}_l$ replaces their counterparts owned by the local SLAM $\mathbf{a}_i \in \mathcal{A}_L$. Moreover, the new anchors computed (and optimized) by the global SLAM process are added to the local SLAM set \mathcal{A}_L .

3.3. Loop Correction

As it was described in Section 2.3, every time that it is possible, the loop correction process takes the last available frame \mathcal{F} to detect previously mapped areas for correcting the map and camera-robot pose. In this case, the following procedure is carried out:

- Loop detection. If $\mathcal{K}_j \in \mathcal{K}$ is the most recent key-frame, and \mathcal{K}_j^1 is the oldest key-frame visually linked to \mathcal{K}_j , and \mathcal{K}_j^2 is the oldest key-frame visually linked to \mathcal{K}_j^1 , first lets define $\mathcal{K}_o \subset \mathcal{K}$ as the subset of all key-frames not containing the key-frames visually linked to \mathcal{K}_j^1 nor \mathcal{K}_j , or: $\mathcal{K}_o = \{\mathcal{K} \in \mathcal{K} : \mathcal{K} \notin \{\mathcal{K}_j^2, \dots, \mathcal{K}_j^1, \dots, \mathcal{K}_j\}\}$. The ORB descriptors, computed from the current frame \mathcal{F} , are attempted to be matched against the ORB descriptors of key-frames $\mathcal{K} \in \mathcal{K}_o$. RANSAC is applied to remove potential match outliers. Now, let define $\mathcal{K}_m \subset \mathcal{K}_o$ as the subset of consecutive key-frames with at least n number of matches ($n = 15$ is used by the implementation). A potential loop is detected if \mathcal{K}_m contain at least three key-frames. The key-frame $\mathcal{K}_M \in \mathcal{K}_m$ is the key-frame with the highest number of matches (see Section 2.3).
- Camera pose computation. The corrected camera pose \mathbf{x}_{cp} is computed through Equation (10), selecting the anchors $\mathbf{a}_i \in \mathcal{K}_F$ as it is described in Section 2.3, and with $\mathbf{h}_a(\mathbf{x}_{r_F}, \mathbf{a}_i)$ defined by the projection model (18) with $\mathbf{a}_i = [p_{x_i}, p_{y_i}, p_{z_i}]$. The following considerations are taken into account for the minimization:
 - Due to the gimbal assumption, \mathbf{R}^{NC} is set as a known fixed parameter in Equation (10).
 - Due to the integration of the altimeter, p_z (the z-axis location of the camera-robot) is set as a fixed parameter in Equation (10) equal to the current altitude camera-robot position computed by the local SLAM in \mathbf{x}_{r_F} .

It is important to note that the subset \mathcal{K}_F must contain a minimum number of four anchors to compute \mathbf{x}_{cp} , but in practice, to improve robustness, a minimum number of 10 anchors is required in this implementation; otherwise, the loop closure is rejected. Additionally, in this implementation to improve the robustness of the solution of the corrected camera pose an additional test was considered. For this test, the anchors

$\mathbf{a}_i \in \mathcal{K}_{\mathcal{F}}$ are re-projected to the image plane through (10) using the computed \mathbf{x}_{cp} . If the projection of a single anchor \mathbf{a}_i lies outside of the image frame then the loop closure is rejected.

- Global map correction. If a corrected camera pose \mathbf{x}_{cp} is available, then the global map (position of key-frames $\mathcal{K} \in \mathcal{K}$ and anchors $\mathbf{a}_i \in \mathcal{A}$) is corrected through Equations (12) and (13) as it is described in Section 2.3. But in this case, due to that, only the position of the camera-robot must be estimated (gimbal assumption) and that the altitude computed by the local SLAM is taken to be the best estimate (altimeter assumption), the graph SLAM problem is simplified to a 2DOF (x - y) position estimation problem. Therefore, in Equation (12):

$$\hat{\mathbf{z}}_{ij}(\mathbf{x}_{r_i}, \mathbf{x}_{r_j}) = \begin{bmatrix} p_x \\ p_y \end{bmatrix} \quad (24)$$

After a loop closure is carried out, the visibility graph is completely recomputed to reflect the actual visual relationships between key-frames.

- Position update triggering. When a corrected camera pose is available and the global map is rectified, a position update is triggered in the local SLAM process using \mathbf{x}_{cp} as the measurement to correct the local map accordingly to the loop close condition.

3.4. Implementation Notes

The implementation case of the visual-SLAM architecture proposed in Section 2 is written in c++ and makes use of the following libraries: (i) CERES [47] for solving all the minimization problems. (ii) OpenCV [48] for implementing all the image-level processing (e.g., ORB descriptors). (iii) Armadillo [49] for implementing linear algebra operations.

As it was already stated before, the proposed visual-SLAM architecture can be implemented in several manners. Meaning that also the particular implementation case, presented in this section, can be easily extended to incorporate for instance additional sensors aiding. An example of this, could be the integration of GPS measurements to the local SLAM process when they are available. Of course, the SLAM as a pure research problem aims to solve robotic navigation without depending on any external infrastructure (as the GPS). But in practical applications, the system performance will be benefited from the use of any available sensory source information.

4. Experimental Results

A ground application was implemented for capturing and storing the sensors' data obtained from a Bebop 2 quadrotor from Parrot [50] (see Figure 6, right plot). The Bebop 2 is a P7 dual-core CPU and quad-core GPU embedded system running a Linux-based OS with built in Wi-Fi, GPS and camera. It has 8GB internal flash memory, 3350 mAh battery with 25 min flying time. For our purposes, image frames from the frontal camera with a pixel resolution of 320×240 were captured at 30 fps. The Bebop's camera can be set to look downwards. Moreover, altitude measurements produced by the flight controller of the Bebop, and range measurements obtained from the ultrasound sensor were captured at 5 Hz.

In experiments, the quadrotor takeoff from a specific home location, and was manually commanded to follow flight trajectories similar to the one illustrated in Figure 6, where the robot flies away from the home area and eventually returns over there. During each flight, the sensors' data were time-stamped and stored in a data set as it was previously described. The implementation case described in Section 3 was executed in an offline manner using the captured data sets for testing the proposed visual-based SLAM approach. In this case, a major objective was to observe if the proposed SLAM system was able to close the loops and therefore to correct the error drift in both the robot position and the global map by detecting old landmarks belonging to the home area.

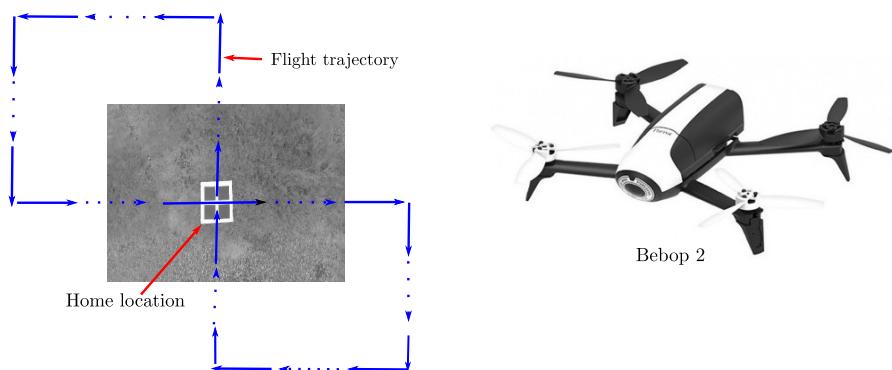


Figure 6. Experimental setup (left plot). UAV used in experiments (right plot).

4.1. Local SLAM

First, let analyze the estimation results obtained by the Local SLAM process. Figure 7 shows a flight trajectory computed solely by the local SLAM process. Observe that the map is always composed of visual features located near the current UAV position since they are removed from the map as the UAV moves away from them to maintain a stable computation time. In this case, it is important to note that Local SLAM can operate completely independently from the other two system process (global map and loop correction) as some kind of visual odometry and local mapping system. Of course, if previous mapped features are removed, then the Local SLAM is unable to recognize previously mapped areas (i.e., close the loop) and therefore unable to minimize the accumulated error drift in estimates. Observe in Plot (c), that by the end of the flight trajectory, the accumulated error x - y position is approximately 5 m. The above by considering that the grid in the computed scene is composed of squares of 1×1 m, and the home location reference measures 0.7×0.7 m. The UAV's actual position at the end of the trajectory has been calculated from knowing the dimension of the home landmark (the four black-square reference), and the intrinsic camera parameters.

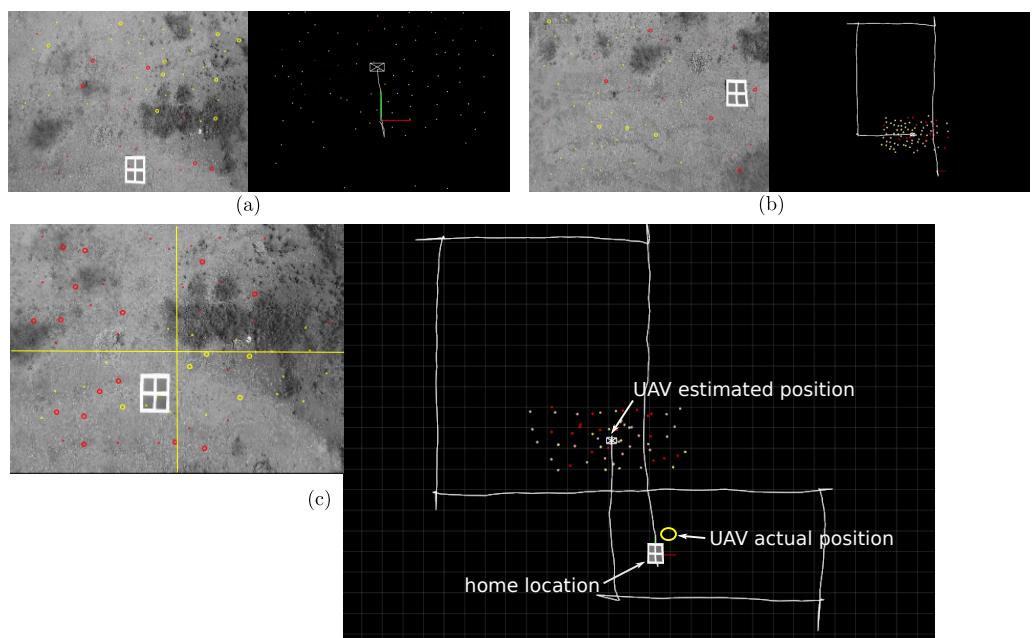


Figure 7. Estimates obtained from the Local SLAM process (aerial x-y view). Plot (a) corresponds to an initial stage of the flight trajectory, Plot (b) corresponds to a middle stage when the UAV is near to complete the first loop, Plot (c) corresponds to a final stage when the UAV has completed a second loop.

Table 1 shows some statistics obtained from the Local SLAM for the flight trajectory illustrated in Figure 7. Experiments were run in a laptop equipped with an Intel i7-6500U

processor with 4 cores running at 2.5 GHz. The actual time duration of the flight trajectory was 86.1 s. Considering the execution times (total and per frame) is evidently that the Local SLAM process can easily meet the real-time requirements with this hardware. In this sense, it is important to note that the Local SLAM is the only process in which the execution time is constrained by the operation rate of the sensors (e.g., camera fps) to accomplish a real-time performance. Moreover, for this experiment, the use of anchors reduced the computation time by around 10 percent, without any significant differences in the estimates by using them.

Table 1. Statistics of the Local SLAM process. In this table, feats:I/D is the relation between the total number of initialized and the deleted EKF features, feat/frame is the average number of features per frame, anchors:I/D is the relation between the total number of initialized and the deleted anchors, time(s)/frame is the average computation time per frame, and total time(s) is the total computation time.

	Feats:I/D	Feat/Frame	Anchors:I/D	Anchors/Frame	Time(s)/Frame	Total Time(s)
No anchors	17,226/17,146	$83.2 \pm 7.03\sigma$	0	0	$0.0118 \pm 0.0025\sigma$	30.77
With anchors	11,163/11,134	$30.49 \pm 9.42\sigma$	1496/1436	$60.4 \pm 11.82\sigma$	$0.0106 \pm 0.0023\sigma$	27.54

4.2. Global Mapping

Figure 8 shows the results obtained for the same flight trajectory described in Figure 7, but in this case when the Global mapping process is taken into account. Observe that since the Global mapping process's main task is only to construct the Global map, the same error drift exhibited by the Local SLAM remains in the estimates.

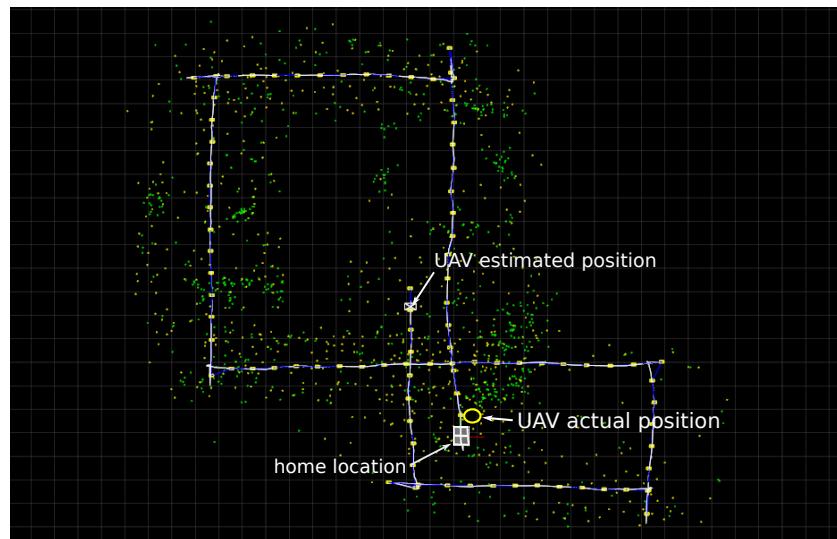


Figure 8. Estimates obtained from the Global Mapping process (aerial x - y view). The Global map is composed of anchors computed by the Global mapping process itself (green dots) and also by anchors computed by the local SLAM process (yellow dots). The location of the Key-frames belonging to the Global map is also highlighted along the UAV trajectory (yellow squares).

Figure 9 shows a lateral view of the estimated global map. Since the orography of the mapped terrain is approximately flat and formed of dirt, grass, and very small bush, it gives a reference for analyzing qualitatively the depth estimates of the Local SLAM (EKF) and the Global mapping (Triangulation plus optimization). In this sense, it can be observed that the anchors computed by the Global mapping process exhibit a higher number of outliers. This result is consistent with the one presented in [31] in which filter-based SLAM methods are compared with optimization-based SLAM methods.

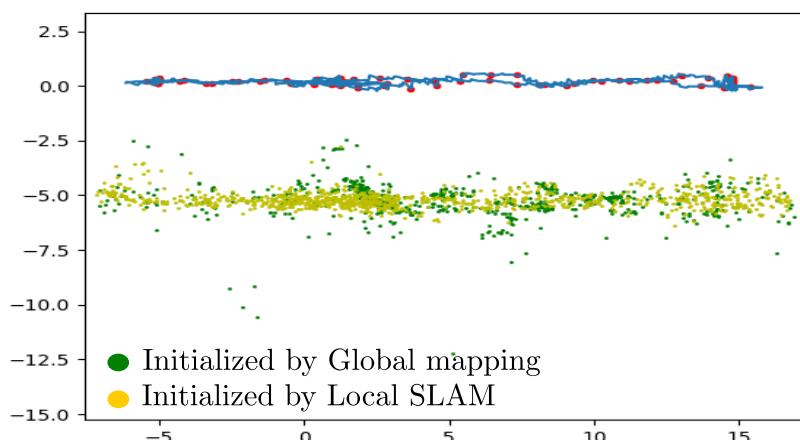


Figure 9. Estimates obtained from the Global Mapping process (lateral z-y view). Anchors computed by the local SLAM process are indicated in yellow. Anchors computed by the global mapping process are indicated in green.

Tables 2 and 3 show the statistics obtained from the global map illustrated in Figure 8. Considering this results, it is clear that the real-time performance of the global mapping process can be easily achieved by the hardware used in experiments. In fact, in future work, the unused computation time could be used for instance to perform a global bundle adjustment of the global map.

Table 2. Statistics of the Global mapping process. In this table, number KF is the total number of Key-frames contained by the global map, anchors:I/D is the relation between the number of initialized and deleted anchors carried out by the global mapping process, anchors:GM is the number of anchors composing the global map that was initialized by the global mapping process, anchors:LS is the number of anchors composing the global map that was initialized by the local SLAM process, anchors:total is the total number of anchors composing the global map.

	Number KF	Anchors:I/D	Anchors:GM	Anchors:LS	Anchors:Total
Global Map	99	2299/1624	675	1008	1683

Table 3. Computation times of the Global mapping process. In this table, n updates is the number of updates carried out by the global mapping process (an update includes the execution of all the steps described in Section 3.2), KF optimized per update is the average number of key-frames optimized by the local bundle adjustment step, Time per update (s) is the execution time per global mapping update, Total time (s) is the total execution time of the global mapping process during the flight trajectory.

	n Updates	KF Optimized per Update	Time per Update (s)	Total Time (s)
Global Map	97	$17.4 \pm 8.1\sigma$	$0.082 \pm 0.032\sigma$	8.001

4.3. Loop Correction

Observing Figures 6–8, it can be seen that the flight trajectory used in experiments has two potential loop detection situations, each one when the UAV passes near over the home location (at the middle and at the end of the trajectory).

Figure 10 shows three examples of the final estimates obtained when the Loop correction process is incorporated into the system. In this case, it is important to note that in our implementation, the detection of new visual features (ORB keypoints) over the images is carried out in a random manner. Therefore every time the algorithm is executed, it produces a slightly different estimated Local SLAM and Global map for the same flight trajectory (observe the Global maps illustrated in Figure 10). For the above reason, the chances of detecting and correcting a loop vary each time the algorithm is executed. For

instance, for a sample of 100 executions of the proposed algorithm over this flight trajectory, the case (a) was obtained 55 times, the case (b) was obtained 15 times, the case (c) was obtained 18 times, and 12 times the algorithm finished without correcting any loop.

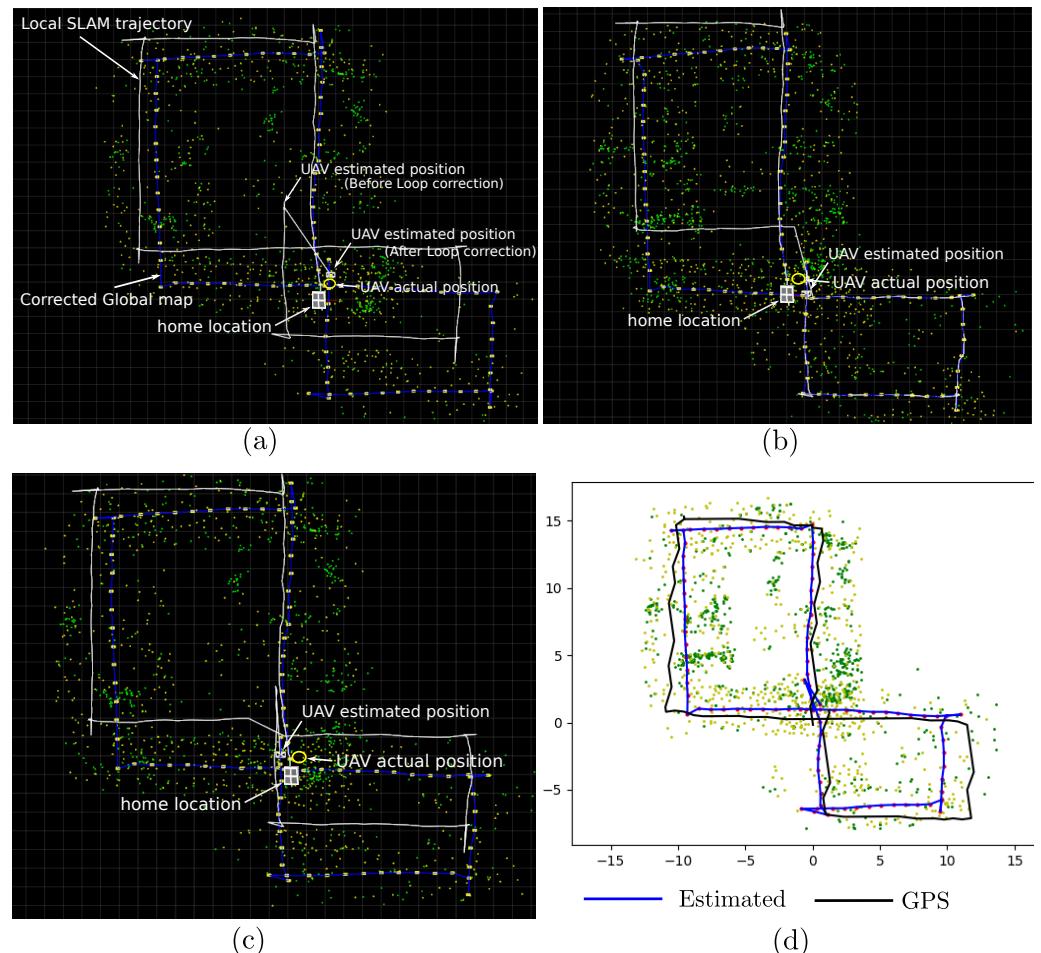


Figure 10. Estimates obtained from the proposed system: Local SLAM + Global mapping + Loop correction (aerial x-y view). Plot (a) shows a case when only the second loop was detected and corrected, Plot (b) shows a case when only the first loop was detected and corrected, and Plot (c) shows a case when both, the first and the second loop was detected and corrected. In experiments, the case (a) occurred more often. Plot (d) shows a comparison between a flight trajectory estimated from the proposed method and the one obtained from GPS.

By comparing the results presented in Figures 8 and 10, it can be observed that every time that the proposed system was able to detect at least one loop, the estimated Global map was considerably improved, and the final error drift in the estimated UAV position was also considerably minimized. Moreover, the correction over the Global map after loop closure can be better appreciated by comparing the Local SLAM trajectory with the final key-frames position: in Figure 8 (without loop closure) both are overlapped, in Figure 10 both differ due to the loop correction over the Global map. In this case, observe that there is a sudden “jump” in the Local SLAM trajectory every time the position of the camera-robot is corrected due to the loop closure. Figure 10 in Plot (d) also shows a comparison between a flight trajectory estimated from the proposed method (after loop-closure) and the one obtained from GPS. In this case, it is important to remark that due to its inherent sources of error, the GPS trajectory should not be taken as a perfect reference for evaluating the actual precision of the estimates. On the other hand, this kind of comparison is still relevant since it is shown that the proposed method is able to provide similar navigation capabilities to

a UAV, but without the use of the GPS, which in the end is one of the major goals of the SLAM methods.

Table 4 shows the average computation time for each step of the Loop correction process. The detection of potential loops is carried out continuously at approximately at 5 Hz. The camera pose computation step is carried out only if a loop is detected and the Global map correction step is carried out only if the corrected camera pose passes the tests described in Section 3.3.

Table 4. Computation times of the Loop correction process. In this table, loop detection is the average execution time needed for detecting potential loops, Camera pose comp is the average execution time needed for computing the corrected camera pose, and Global map correction is the average execution time needed for correcting the global map.

	Loop Detection (s)	Camera Pose Comp. (s)	Global Map Correction (s)
Loop correction	$0.183 \pm 0.161\sigma$	$0.382 \pm 0.021\sigma$	$0.379 \pm 0.010\sigma$

4.4. Comparison with an Optimization-Based Method

Figure 11 (left plot) shows the map and trajectory computed by the well-known ORB-SLAM algorithm [19], when it is run over the first loop of the flight trajectory that was used in previous experiments. For this experiment, the official MATLAB implementation of the ORB-SLAM algorithm, provided by the Computer Vision Toolbox, was used. Since the ORB-SLAM is a purely monocular algorithm (no metric information provided by aiding sensors are considered) the map and camera trajectory is estimated only up to scale.

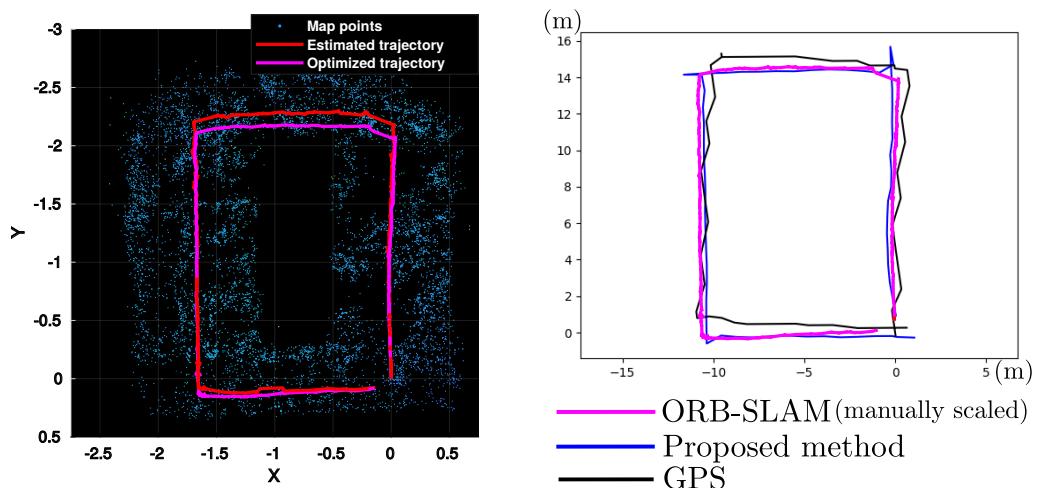


Figure 11. Up to scale map and trajectories computed with ORB-SLAM (left plot). Comparison between the trajectories obtained with (i) ORB-SLAM (manually scaled), (ii) the proposed method, and (iii) GPS (right plot).

For this experiment, to be able to satisfactorily run the ORB-SLAM algorithm over the whole trajectory, several frames had to be manually removed from the original dataset. Most of the removed frames were frames with some degree of blur and correspond to periods of fast camera movements due to sudden changes in the flight trajectory (turns in corners). Without removing those frames the algorithm always crashed due to a low number of visual features being tracked during those periods of fast movements. The above even happen when trying several parameters configurations for tracking an extremely huge number of visual features. Moreover, it is important to note that in this experiment only the first loop of the flight trajectory was used because this implementation of the ORB-SLAM algorithm only allows one close of a loop.

Figure 11 (right plot) shows a comparison between the (optimized after loop closure) trajectory computed by the ORB-SLAM algorithm, the trajectory computed by the proposed

hybrid SLAM method (after loop closure), and the trajectory obtained from GPS. In this case, the ORB-SLAM trajectory was manually scaled to match the metric scale of the other two trajectories.

Table 5 shows some statistics obtained from both methods: the ORB-SLAM and the proposed method. It is very important to consider that the numbers expressed in this table are included only for reference and should not be taken as a direct measurement of the performance of both methods. For instance, MATLAB implementations usually run much slower than C++ implementations. Moreover, there are so many structural differences between methods, that make it difficult to carry out an in-depth comparative study. On the other hand, what this simple experiment suggests is that a visual SLAM system implemented with the hybrid architecture proposed in this work is able to provide similar (but metric-scaled) estimates that those obtained by a state-of-art purely visual-based SLAM method.

Table 5. Statistics of the comparison between the ORB-SLAM and the proposed Hybrid method. In this table, n map feats is the number features/anchors contained by the map, n key-frames is the number of key-frames, and Execution time is overall execution time.

	n Map Feats	n Key-Frames	Execution Time (s)
ORB-SLAM (MATLAB)	8704	274	331.72
Hybrid SLAM (C++)	846	58	20.01

4.5. Other Flight Trajectories

The SLAM system described in Section 3 can not only be applied to simple flight trajectories as the one used in previous experiments. Figure 12 shows the results obtained from two different flight trajectories with a closed-loop. Observe that both trajectories present several changes in direction which in turn corresponds to attitude changes of the drone. Moreover, observe in the right plot that the flight trajectory includes a couple of periods where the drone moves too fast. During those periods, the images captured from the camera are very blurred, and therefore, the tracking of image features becomes unreliable (observe the absence of map features and key-frames in those areas). The proposed method is able to cope with this situation because during short periods where visual information is unavailable, the estimates are computed from the prediction stage of the filter, and the remaining available sensors updates (e.g., altimeter). In both cases, after the loop closure correction, the error in the estimated position is less than 0.5 m. To have a reference for the metric scale of the estimates consider that the (green-red) home axis has a length of one meter.

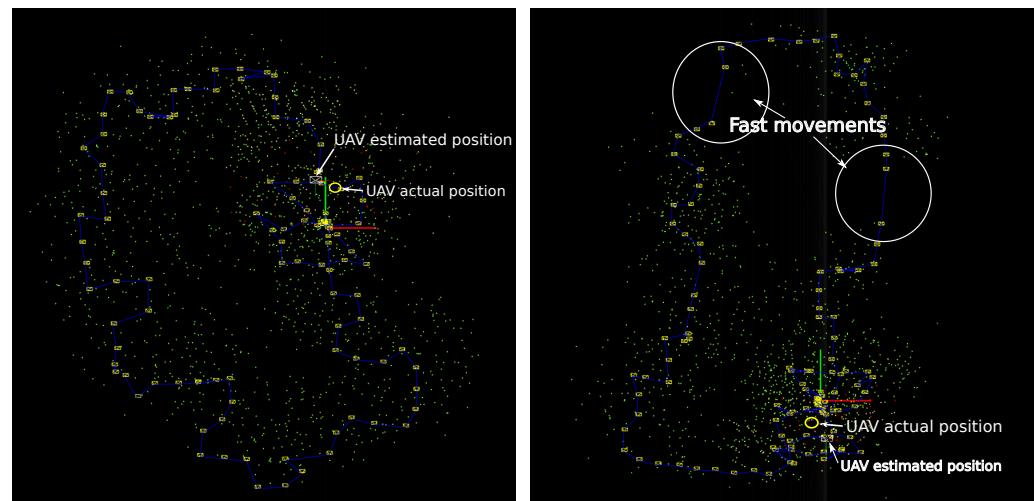


Figure 12. Results obtained from two different flight trajectories with a closed-loop.

5. Conclusions

The experimental results previously presented show that the monocular-based SLAM system for UAVs described in Section 3, which was implemented following the proposed architecture described in Section 2, is able to estimate the state of the robot by using only onboard sensors while at the same time building and maintaining a global map of its environment. The system was able to considerably minimize the error drift in position by detecting closing the trajectory loops. It also was shown that the SLAM system can easily achieve real-time performance using consumer-degree hardware.

It is important to remark again that this monocular-based SLAM system for UAVs represents only a case of many possible ones of implementations of the visual-based hybrid architecture proposed in this work. For instance, a SLAM system for ground vehicles that makes use of an omnidirectional camera or stereo camera as a principal sensor and which uses a Unscented Kalman Filter for implementing the local SLAM process can be also implemented based on the proposed hybrid architecture. In this sense, future work could include development and experimentation with different visual-SLAM applications using the proposed architecture. It could also include testing the performance of the monocular-based SLAM system for UAVs presented in this work in an online context (not running in a data set), and perhaps using the estimated state as the feedback signal of an autonomous flight control system. Moreover, in future work, the path-tracking accuracy of the mobile robot could be evaluated based on the information proposed by the onboard sensors as in [51].

Author Contributions: The contribution of each author is: Conceptualization, R.M. and A.G.; methodology, R.M. and J.-C.T.; software, J.-C.T. and E.G.; validation, R.M., E.G. and A.G.; investigation, J.-C.T. and E.G.; resources, A.G.; writing—original draft preparation, R.M.; writing—review and editing, R.M. and A.G.; visualization, E.G.; supervision, R.M. and A.G.; funding acquisition, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the Spanish Ministry of Science and Innovation project ROCOTRANS (PID2019-106702RB-C21/AEI/10.13039/501100011033).

Acknowledgments: The first author wants to thank his son Roderic Munguia for his contribution to this work as a research assistant.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

- Pan, L.; Cheng, J.; Zhang, Q. UFSM VO: Stereo Odometry Based on Uniformly Feature Selection and Strictly Correspondence Matching. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 4148–4152. [[CrossRef](#)]
- Kostavelis, I.; Boukas, E.; Nalpantidis, L.; Gasteratos, A. Stereo-Based Visual Odometry for Autonomous Robot Navigation. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 21. [[CrossRef](#)]
- Ullah, I.; Su, X.; Zhang, X.; Choi, D. Simultaneous Localization and Mapping Based on Kalman Filter and Extended Kalman Filter. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 2138643. [[CrossRef](#)]
- Kim, P.; Coltin, B.; Kim, H.J. Linear RGB-D SLAM for Planar Environments. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
- Chen, M.; Yang, S.; Yi, X.; Wu, D. Real-time 3D mapping using a 2D laser scanner and IMU-aided visual SLAM. In Proceedings of the 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR), Okinawa, Japan, 14–18 July 2017; pp. 297–302. [[CrossRef](#)]
- Motlagh, H.D.K.; Lotfi, F.; Taghirad, H.D.; Germi, S.B. Position Estimation for Drones based on Visual SLAM and IMU in GPS-denied Environment. In Proceedings of the 2019 7th International Conference on Robotics and Mechatronics (ICRoM), Tehran, Iran, 20–21 November 2019; pp. 120–124. [[CrossRef](#)]
- Zhou, H.; Zou, D.; Pei, L.; Ying, R.; Liu, P.; Yu, W. StructSLAM: Visual SLAM With Building Structure Lines. *IEEE Trans. Veh. Technol.* **2015**, *64*, 1364–1375. [[CrossRef](#)]
- Bloesch, M.; Burri, M.; Omari, S.; Hutter, M.; Siegwart, R. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *Int. J. Robot. Res.* **2017**, *36*, 1053–1072. [[CrossRef](#)]

9. Quan, M.; Piao, S.; Tan, M.; Huang, S.S. Accurate Monocular Visual-Inertial SLAM Using a Map-Assisted EKF Approach. *IEEE Access* **2019**, *7*, 34289–34300. [[CrossRef](#)]
10. Holmes, S.; Klein, G.; Murray, D.W. A Square Root Unscented Kalman Filter for visual monoSLAM. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 3710–3716. [[CrossRef](#)]
11. Lee, S.H. Real-time camera tracking using a particle filter combined with unscented Kalman filters. *J. Electron. Imaging* **2014**, *23*, 013029. [[CrossRef](#)]
12. Tang, M.; Chen, Z.; Yin, F. Robot Tracking in SLAM with Masreliez-Martin Unscented Kalman Filter. *Int. J. Control Autom. Syst.* **2020**, *18*, 2315–2325. [[CrossRef](#)]
13. Eustice, R.M.; Singh, H.; Leonard, J.J.; Walter, M.R. Visually Mapping the RMS Titanic: Conservative Covariance Estimates for SLAM Information Filters. *Int. J. Robot. Res.* **2006**, *25*, 1223–1242. [[CrossRef](#)]
14. Zhou, W.; Valls MirÓ, J.; Dissanayake, G. Information-Efficient 3-D Visual SLAM for Unstructured Domains. *IEEE Trans. Robot.* **2008**, *24*, 1078–1087. [[CrossRef](#)]
15. Celik, K.; Chung, S.J.; Clausman, M.; Somani, A.K. Monocular vision SLAM for indoor aerial vehicles. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 1566–1573. [[CrossRef](#)]
16. Wen, S.; Chen, J.; Lv, X.; Tong, Y. Cooperative simultaneous localization and mapping algorithm based on distributed particle filter. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881418819950. [[CrossRef](#)]
17. Liu, W. Slam algorithm for multi-robot communication in unknown environment based on particle filter. *J. Ambient. Intell. Human Comput.* **2021**. [[CrossRef](#)]
18. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234. [[CrossRef](#)]
19. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [[CrossRef](#)]
20. Concha, A.; Civera, J. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5686–5693. [[CrossRef](#)]
21. Lin, X.; Wang, F.; Guo, L.; Zhang, W. An Automatic Key-Frame Selection Method for Monocular Visual Odometry of Ground Vehicle. *IEEE Access* **2019**, *7*, 70742–70754. [[CrossRef](#)]
22. Zhang, Y.; Xu, X.; Zhang, N.; Lv, Y. A Semantic SLAM System for Catadioptric Panoramic Cameras in Dynamic Environments. *Sensors* **2021**, *21*, 5889. [[CrossRef](#)] [[PubMed](#)]
23. Liao, Z.; Wang, W.; Qi, X.; Zhang, X. RGB-D Object SLAM Using Quadrics for Indoor Environments. *Sensors* **2020**, *20*, 5150. [[CrossRef](#)] [[PubMed](#)]
24. Zhao, Y.; Vela, P.A. Good Feature Selection for Least Squares Pose Optimization in VO/VSLAM. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1183–1189. [[CrossRef](#)]
25. Huang, G.P.; Mourikis, A.I.; Roumeliotis, S.I. Observability-based Rules for Designing Consistent EKF SLAM Estimators. *Int. J. Robot. Res.* **2010**, *29*, 502–528. [[CrossRef](#)]
26. Huang, G.P.; Mourikis, A.I.; Roumeliotis, S.I. A Quadratic-Complexity Observability-Constrained Unscented Kalman Filter for SLAM. *IEEE Trans. Robot.* **2013**, *29*, 1226–1243. [[CrossRef](#)]
27. Lee, S.M.; Jung, J.; Kim, S.; Kim, I.J.; Myung, H. DV-SLAM (Dual-Sensor-Based Vector-Field SLAM) and Observability Analysis. *IEEE Trans. Ind. Electron.* **2015**, *62*, 1101–1112. [[CrossRef](#)]
28. Reif, K.; Gunther, S.; Yaz, E.; Unbehauen, R. Stochastic stability of the discrete-time extended Kalman filter. *IEEE Trans. Autom. Control* **1999**, *44*, 714–728. [[CrossRef](#)]
29. Kluge, S.; Reif, K.; Brokate, M. Stochastic Stability of the Extended Kalman Filter With Intermittent Observations. *IEEE Trans. Autom. Control* **2010**, *55*, 514–518. [[CrossRef](#)]
30. Poulose, A.; Han, D.S. Hybrid Indoor Localization Using IMU Sensors and Smartphone Camera. *Sensors* **2019**, *19*, 5084. [[CrossRef](#)] [[PubMed](#)]
31. Strasdat, H.; Montiel, J.; Davison, A.J. Visual SLAM: Why filter? *Image Vis. Comput.* **2012**, *30*, 65–77. [[CrossRef](#)]
32. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [[CrossRef](#)] [[PubMed](#)]
33. Hartley, R.I.; Sturm, P. Triangulation. *Comput. Vis. Image Underst.* **1997**, *68*, 146–157. [[CrossRef](#)]
34. Triggs, B.; McLauchlan, P.F.; Hartley, R.I.; Fitzgibbon, A.W. Bundle Adjustment—A Modern Synthesis. In *Vision Algorithms: Theory and Practice*; Triggs, B., Zisserman, A., Szeliski, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 298–372.
35. Xia, Y.; Li, J.; Qi, L.; Yu, H.; Dong, J. An Evaluation of Deep Learning in Loop Closure Detection for Visual SLAM. In Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 21–23 June 2017; pp. 85–91. [[CrossRef](#)]
36. Chen, S.; Zhou, B.; Jiang, C.; Xue, W.; Li, Q. A LiDAR/Visual SLAM Backend with Loop Closure Detection and Graph Optimization. *Remote Sens.* **2021**, *13*, 2720. [[CrossRef](#)]

37. Konolige, K.; Agrawal, M. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Trans. Robot.* **2008**, *24*, 1066–1077. [[CrossRef](#)]
38. Wu, Y.; Hu, Z. PnP Problem Revisited. *J. Math. Imaging Vis.* **2006**, *24*, 131–141. [[CrossRef](#)]
39. Grisetti, G.; Kümmerle, R.; Stachniss, C.; Burgard, W. A Tutorial on Graph-Based SLAM. *IEEE Intell. Transp. Syst. Mag.* **2010**, *2*, 31–43. [[CrossRef](#)]
40. Urzua, S.; Munguía, R.; Grau, A. Monocular SLAM System for MAVs Aided with Altitude and Range Measurements: A GPS-free Approach. *J. Intell. Robot. Syst.* **2018**, *94*, 203–217. [[CrossRef](#)]
41. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2564–2571. [[CrossRef](#)]
42. Bailo, O.; Rameau, F.; Joo, K.; Park, J.; Bogdan, O.; Kweon, I.S. Efficient adaptive non-maximal suppression algorithms for homogeneous spatial keypoint distribution. *Pattern Recognit. Lett.* **2018**, *106*, 53–60. [[CrossRef](#)]
43. Bouguet, J. Camera Calibration Toolbox for Matlab. 2008. Available online: http://www.vision.caltech.edu/bouguetj/calib_doc/ (accessed on 23 December 2021).
44. MUJA, M.; Lowe, D. Fast approximate nearest neighbors with automatic algorithm configuration. In Proceedings of the 2009 International Conference on Computer Vision Theory and Applications, Lisboa, Portugal, 5–8 February 2009; pp. 331–340.
45. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [[CrossRef](#)]
46. Bailey, T.; Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robot. Autom. Mag.* **2006**, *13*, 108–117. [[CrossRef](#)]
47. Agarwal, S.; Mierle, K. Ceres Solver. Available online: <http://ceres-solver.org> (accessed on 23 December 2021).
48. Itseez. Open Source Computer Vision Library. 2015. Available online: <https://github.com/itseez/opencv> (accessed on 23 December 2021).
49. Sanderson, C.; Curtin, R. A User-Friendly Hybrid Sparse Matrix Class in C++. In *Mathematical Software—ICMS 2018*; Davenport, J.H., Kauers, M., Labahn, G., Urban, J., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 422–430.
50. Parrot Bebop Drone for Developers. Available online: <https://developer.parrot.com/docs/SDK3/> (accessed on 23 December 2021).
51. Palacin, J.; Rubies, E.; Clotet, E.; Martinez, D. Evaluation of the Path-Tracking Accuracy of a Three-Wheeled Omnidirectional Mobile Robot Designed as a Personal Assistant. *Sensors* **2021**, *7216*, 7216. [[CrossRef](#)] [[PubMed](#)]