ReSLAM: Reusable SLAM with heterogeneous cameras

Francisco J. Romero-Ramirez[1], Rafael Muñoz-Salinas[1,2,*], Manuel J. Marín-Jiménez[1,2], Angel Carmona-Poyato[1,2], Rafael Medina-Carnicer[1,2]

Abstract

State-of-the-art SLAM methods are designed to work only with the type of camera employed to create the map, and little attention has been paid to the reusability of the maps created. In other words, the maps generated by current methods can only be reused with the same camera employed to create them. This paper presents a novel SLAM approach that allows maps generated with one camera to be used by other cameras with different resolutions and optics. Our system allows, for instance, creating highly detailed maps processed off-line with high-end computers, to be reused later by low-powered devices (e.g. a drone or robot) using a different camera. The first map, called base map, can be reused with other cameras and dynamically adapted by creating an augmented map. The principal idea of our method is a bottom-up pyramidal representation of the images that allows us to match keypoints between different camera types seamlessly. The experiments conducted validate our proposal, showing that it outperforms the state-of-the-art approaches, namely ORBSLAM, OpenVSLAM and UcoSLAM.

Keywords: SLAM, Mapping, Localization

1. Introduction

Simultaneous Location and Mapping (SLAM) has experienced important advances in the last years in fields such as robotics and computer vision [1, 2, 3, 4]. Applications that require localisation, such as autonomous vehicles [5, 6], augmented reality [7] and robotic-assisted surgery [8, 9], amongst others, take advantage of current SLAM methods.

Most SLAM methods represent natural features as keypoints that can robustly detect and establish correspondences between images. An appropriate representation of the keypoints is essential, considering that the information captured by the camera suffers alterations over time. However, no current SLAM method (up to our knowledge) is explicitly designed to reuse the map generated using a camera with a different camera (of different optics and resolution). This is an important deterrent of the current SLAM systems from being adopted in real-life applications.

We envision visual SLAM should be used as follows. Imagine we want to provide visual localization capabilities for a particular environment (e.g., an industrial facility, University campus, etc.) to a plethora of heterogeneous autonomous systems (AS) (e.g., autonomous robots, cars, or drones). In general, the computing capabilities of embedded AS are somewhat limited, thus being difficult for them to run in real-time SLAM algorithms. In addition, letting each AS independently create its own map has some risks: a) the SLAM method may fail, creating a wrong map, and thus the AS would get lost; and, b) the maps independently generated by different AS may not match or have a common reference system. This is especially evident in monocular SLAM, where the scale of the generated maps is unpredictable.

A possible approach to solve the problem is creating a *base map* of the environment using a high-resolution camera (see Figure 1). The base map can be processed offline from multiple recordings, ensuring that it is consistent, has no errors, and is properly scaled. Then, a copy of the base map could be loaded on each AS. The AS's map would be dynamically updated as the tracking circumstances demand it. For instance, if the AS navigates in an area already registered on the map, the changes would be minimal. However, the base map would be augmented when the AS moves in unexplored areas. To that end, we need a method able to reuse the base map with any camera installed in the AS.

A straightforward approach for reusing maps would consist of adapting the AS camera images to be like the one employed to create the map. This approach, which has not been explored in the literature (to our knowledge), has a drawback: if the map is created with a high-resolution camera and, then, we want to reuse it with an-

*Corresponding author
Email addresses: `fj.romero@uco.es` (Francisco J. Romero-Ramirez), `in1musar@uco.es` (Rafael Muñoz-Salinas), `mjmarin@uco.es` (Manuel J. Marín-Jiménez), `ma1capoa@uco.es` (Angel Carmona-Poyato), `rmedina@uco.es` (Rafael Medina-Carnicer)

[1]Departamento de Informática y Análisis Numérico, Edificio Einstein. Campus de Rabanales, Universidad de Coŕdoba, 14071, Córdoba, Spain, Tlfn:(+34)957212289

[2]Instituto Maimónides de Investigación en Biomedicina (IMIBIC). Avenida Menéndez Pidal s/n, 14004, Córdoba, Spain, Tlfn:(+34)957213861

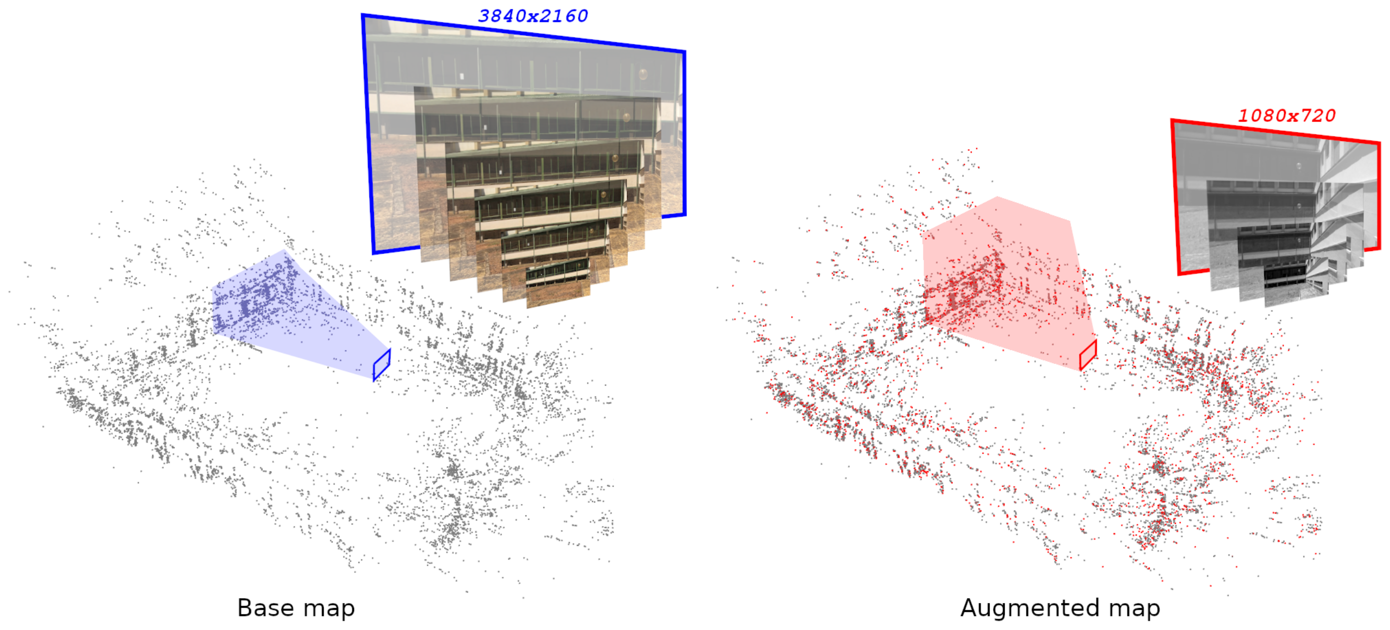


Figure 1: Left image shows the base map generated with a high-resolution 4K camera. Right image shows how the base map is reused with another camera of different resolution and optics, augmented with new information (red points). The image shows how the new camera can relocalize itself in a previously explored environment by using old map points, while also incorporating new points to capture potential environmental variations (such as shadows, objects, etc.), resulting in a more robust tracking system.

other lower resolution camera, we need to upsample the latter to match the former. Then, we artificially increase the computing time required to reuse the map and thus disincentivize the use of high resolutions cameras to create the map.

This work proposes *ReSLAM*, a novel keypoint-based monocular SLAM approach designed to create maps with cameras of different resolutions and optics that can be reused without the limitations previously explained. The key to achieve such a goal is an internal bottom-up pyramidal representation of the camera images, where keypoints at the same level represent regions with similar physical dimensions. Consequently, they can be seamlessly compared, independently of the camera they were created with. In addition, the proposed method allows adapting the computational time by adapting the total number of levels in the pyramid. The fewer pyramid levels employed, the faster the system will run.

The proposed method has been evaluated on new datasets created ad-hoc for this work, since none of the standard SLAM datasets employed in the related literature has been recorded with multiple heterogeneous cameras. A total of six new datasets have been created in different areas of our University Campus, using cameras with different optics and resolutions. The results show that our proposal outperforms the state-of-the-art SLAM methods ORBSLAM [10, 11], OpenVSLAM [4] and UcoSLAM [2] in both speed and accuracy. In addition, the code and datasets are set public for other researchers to use them [3].

The main contributions of this work can be summarized as follows:

1. Development of a keypoint-based monocular SLAM approach that allows the creation and reuse of maps using heterogeneous cameras.
2. Introduction of a novel bottom-up pyramidal representation that allows matching keypoints between different camera types seamlessly.
3. Creation of new datasets designed for this research, featuring recordings from multiple heterogeneous cameras.

The rest of the paper is structured as follows. Section 2 reviews the most related works to this one, while Section 3 provides an overview of the method, introducing its key elements and its mathematical formulation. Section 4 explains our proposal and Section 5 shows the results obtained. Finally, Section 6 draws some conclusions and future works.

2. Related works

Harris and Pike's work [12] is one of the first methods for reconstructing scenes from a single camera. Despite the promising results in the creation of 3D maps from long

[3]`https://www.uco.es/investiga/grupos/ava/portfolio/reslam`

video sequences, the first works lacked methods to establish loop closures and drift corrections, which was partially solved in the work of Davison *et. al.* [13] and Eade and Drummond [14, 15]. While in the previous methods, tracking and mapping are directly linked, the PTAM system [16] separates both tasks into parallel processes, achieving high performance in real-time applications in small environments. Later, Mur-Artal et al. presented ORB-SLAM [1], a feature-based monocular SLAM, which uses ORB features [17] for tracking and relocation. Later, the work is extended with ORB-SLAM2 [10], which allows working with stereo and RGB-D cameras, and finally integrates inertial information in ORB-SLAM3 [11]. In line with the previous work, OpenVSLAM [4] is presented a SLAM framework that has been designed to be usable and scalable for applications based on modeling and mapping.

The significance of the localization process within the broader context of SLAM systems cannot be overstated. Localization plays a crucial role in estimating the camera's pose within the environment, typically achieved by matching visual features between consecutive frames or against an existing map. This information is indispensable for accurate environment mapping and ensuring the consistency of generated maps. Recent advancements in deep learning have shown promising results in enhancing localization accuracy [18].

In the realm of visual localization for SLAM systems, various approaches have been proposed. Feature-based methods, such as SIFT (Scale-Invariant Feature Transform) [19] and ORB (Oriented FAST and Rotated BRIEF) [17], have gained wide usage for detecting and matching visual features between frames. These methods rely on techniques like keypoint extraction and matching to estimate the camera's pose. Notably, the work [20] introduces a Features Combined Binary Descriptor based on Voted Ring-Sampling Pattern (BDVRP), which encodes both intensity and gradient of interest points. Additionally, [21] proposes an unsupervised deep learning method for binary descriptor learning. However, traditional keypoint descriptors face challenges in localization for large-scale environments. To address this, [22] suggests learning semantic-aware local features to improve the robustness of local feature matching. Alternatively, [23] employs depth-maps as a solution.

On the other hand, the camera pose estimation is not in the actual scale when working with monocular cameras. To solve this, SPM-SLAM [24] proposes the use of fiducial markers [25] to solve the localization problem. Fiducial markers are artificial elements placed in the environment that are stable over time, unlike keypoints. Later, the same authors propose UcoSLAM [2], a method capable of combining both keypoints and fiducial markers.

Unlike the systems mentioned above, considered as indirect methods because the localization and optimization process is based on features, methods like LSD-SLAM [26], DSO [27], and LDSO [3] use direct approaches in which the SLAM process is guided by the intensity information of image pixels. These methods require calibration of the photogrammetric properties of the camera, which makes them difficult to use in most of the existing datasets.

Within the SLAM systems, those based on monocular [1, 27, 26] and stereo [28, 10, 2] cameras, are the most common ones. However, over the last few years, new applications have appeared that seek the fusion of information provided by different sensors during navigation [29, 30]. Recent methods based on the use of multiple cameras allow extending monocular SLAM using synchronized cameras [31] which assume synchronized shutters for all of them, or through a set of asynchronous observations [32]. The work of Ragab and Wong [33] presents a SLAM system composed of multiple back-to-back cameras, where each camera separately implements an extended Kalman filter (EKF) that are finally combined to obtain a final optimization. Kaess and Dellaert's work [34], where the cameras are placed in a ring, pursues the same objective.

While SLAM systems create a map of the environment that is optimized over time, most systems do not allow the map to be reused. In the work of Linen *et al.* [35] a first offline stage is proposed where the construction of a 3D map of the environment is carried out using Structure from Motion (SfM), which is later used in a second stage using inertial SLAM localization. The reuse of the map is also treated in the works [36, 11] where the system can locate itself on a pre-built map from both visual and inertial information. However, both for the creation of the map and for relocalization, the same sensor is used.

A technical aspect to highlight about systems that reuse maps is that this can be stored and loaded later in the system. This feature has been recently incorporated into SLAM systems such as OpenVSLAM [4] and UcoSLAM [2], which allow reusable and extensible maps of the environment. However, in both systems, only the same camera can be used for the creation of the map and its subsequent reuse.

In this work, we present ReSLAM, a new method to perform monocular SLAM, allowing combining information from heterogeneous cameras for simultaneous localization and mapping. Our system allows creating high-resolution maps with one camera and later use them with a different camera. At the same time, it allows new map points to be incorporated into the map, extending its initial information.

3. Method overview

This section provides an overview of the proposed SLAM method, introducing its key elements and the notation employed in the paper.

As already indicated, our proposal is a keypoint-based SLAM method working with monocular cameras, although it can be extended to work with stereo and RGB-D cameras. As with most state-of-the-art SLAM systems, a map of the environment is created from a sequence of frames captured with a moving camera. The map contains 3D points corresponding to environment regions seen as keypoints from multiple viewpoints in a set of selected frames called keyframes. The keyframes are selected whenever it is required to expand the map to new regions that are being visited. Periodically, the map is optimized to remove useless map points and improve the 3D accuracy of the map points. The generated map degrades its accuracy with the distance travelled (*drifting problem*). In order to reduce its impact, the system analyzes whether the camera revisits a zone of the map (*relocalization*) and applies a loop closure optimization to remove the drifting error. The system also must be able to solve the so-called *kidnapped robot problem*, which happens when the system, for some reason, loses track of its position in the map. Our method employs a *bag-of-words* (BoW) database of the keyframes that is used for loop-closure detection and relocalization [37].

The steps described above are common to most modern SLAM methods and are employed in ours as well. The key innovation of our approach is that our map can be generated using any pin-hole camera, thanks to the way we represent the camera frames (using a bottom-up pyramidal structure) and how keypoints are matched across them. Our formulation can generate the map seamlessly, adapting to cameras of different resolutions and optics. The rest of the section introduces the key elements required for SLAM and the notation employed in this paper.

3.1. Frames

A camera frame f is represented by the tuple:

$$f = \{\theta, I, \mathcal{I}, \mathcal{K}\} \tag{1}$$

where $\theta \in \mathbb{SE}(3)$ is the pose in the map where the image I was shot and $\mathcal{I} = \{\{I_j\}, j \in \{0, ..., \eta\}\}$ is a pyramid of images obtained by subsampling I. Let us denote $I(\delta), I(\lambda)$ and $I(s)$ the image focal length, optical center and size, respectively, and $I_j(\delta), I_j(\lambda)$ and $I_j(s)$ as the corresponding parameters of each pyramid image. We assume that the camera has no distortion or that it has been removed. Finally, $\mathcal{K}$ represents the set of keypoints of the frame. Please notice that we will use superindices to refer to the elements of a tuple, e.g., θ^f denotes the pose of frame f.

Our pyramid is uniformly built starting from the smallest image I_0 in such a way that its focal length $I_0(\delta)$ is equal to a constant value δ_{min}. The rest of the pyramid levels are created using a constant scale factor $\tau \geq 1$ until the maximum level η is reached, see Figure 2. Then, the parameters of each pyramid level can be calculated as:

$$I_j(q) = I(q) \cdot \frac{\delta_{min} \cdot (\tau)^j}{I(\delta)}, q \in \{\delta, \lambda, s\}, j \geq 0, \tag{2}$$

and the maximum level of the pyramid:

$$\eta = \left\lfloor \frac{\log(\frac{I(\delta)}{\delta_{min}})}{\log(\tau)} \right\rfloor. \tag{3}$$

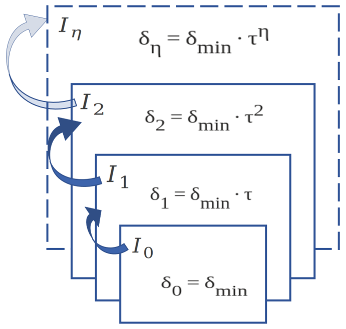

I_η
$\delta_\eta = \delta_{min} \cdot \tau^\eta$
I_2
$\delta_2 = \delta_{min} \cdot \tau^2$
I_1
$\delta_1 = \delta_{min} \cdot \tau$
I_0
$\delta_0 = \delta_{min}$

Figure 2: Pyramid building. The image and camera parameters are scaled from a predefined minimum focal length δ_{min}, and using a scale factor τ.

The way our pyramid is built is one of the keys that makes our method able to work with different types of cameras. Other SLAM approaches use a pyramid of images that is anchored to the focal length of the camera that created the map, making it difficult to use other types of cameras with different focal lengths. Our pyramid, however, can adapt to cameras of different optics and resolutions by setting a fixed set of focal lengths starting from the smallest one δ_{min}. Cameras with large focal lengths will produce more pyramid levels than cameras with small focal lengths.

The critical aspect is that a keypoint at a given level represents a region of the space proportional to the focal length of that level. By analyzing the classic pin-hole camera model, it is easy to derive that, for instance, for a focal length of 1000 pix, a circle of 1 m seen at 10 m of distance will project as a circle of radius 100 pix in the image, independently of the actual image resolution. Consequently, we organize keypoints into pyramid levels of fixed focal lengths to easily compare across cameras independently of their actual optics and resolution.

3.2. Keypoints

Each image of the pyramid $\mathcal{I}$ is processed to detect the set of keypoints $\mathcal{K}$. The total number of keypoints selected $|\mathcal{K}|$ is defined as a function of τ_0, which represents the number of keypoints selected in the lowest pyramid level I_0:

$$|\mathcal{K}| = \sum_{j=0}^{\eta} n_j, \tag{4}$$

where n_j is the number of selected keypoints of each pyramid level j, defined as:

$$n_j = \lfloor \tau_0 \cdot (\tau)^j \rfloor. \tag{5}$$

A keypoint k is represented as the tuple

$$k = \{l, p, d\}, \tag{6}$$

where $l \in \{0, ..., \eta\}$ is the pyramid level where the keypoint was detected and $p \in \mathbb{R}^2$ represents its image coordinates in I_l. Finally, $d = \{(d_1, ..., d_u) \mid d_j \in \{0,1\}, \forall j\}$ is the keypoint descriptor. We are assuming binary descriptors (such as ORB [17]), since they normally present a good balance between speed and repeatability.

3.3. Map definition

Our goal is to create a map of the environment (while navigating it) that can be reused and updated with other cameras. To do so, we define the system map Ω:

$$\Omega = \{\mathcal{O}, \mathcal{F}, \mathcal{G}, \mathcal{B}\}, \tag{7}$$

as a set of structures required to represent the environment. The set $\mathcal{O}$ contains the map points while $\mathcal{F}$ is the subset of selected frames (*keyframes*) used for keypoint triangulation. The graph $\mathcal{G}$ keeps the interconnections between the keyframes, where the edge weights represent the number of map points visible in both keyframes. Finally, $\mathcal{B}$ is a visual place recognition database, represented as a BoW used for relocalization purposes and to detect loop closures [37].

A map point $o \in \mathcal{O}$ represents a three-dimensional region of the environment and is mathematically represented as the tuple:

$$o = \{x, w, \hat{d}, v, \beta\}, \tag{8}$$

where $x \in \mathbb{R}^3$ represents its three-dimensional coordinates in the global reference system, $w \in [0,1]$ is a weight indicating how reliable the point is based on how many times it is observed, $\hat{d}$ is its keypoint descriptor and v is the vector normal to the map point. The parameter v is used to determine whether the map point is visible from a given viewpoint. Finally, the parameter $\beta \in \{0,1\}$ indicates whether the keypoint was added to the map during the map building phase (base map) ($\beta = 0$), or later ($\beta = 1$), possibly using a different camera.

The position x is obtained by triangulating its observations (i.e., keypoints) in multiple keyframes. Let us denote $K(o) = \{(k_{f_j}, d^{k_{f_j}})\}$ as the set of keypoints that the map point o is derived from. Then, the descriptor $\hat{d}$ of the map point o is defined as the most representative descriptor of $K(o)$ as:

$$\hat{d} = \underset{(k_{f_i}, d^{k_{f_i}}) \in K(o)}{\operatorname{argmin}} \sum_{(k_{f_j}, d^{k_{f_j}}) \in K(o)} dist(d^{k_{f_i}}, d^{k_{f_j}}). \tag{9}$$

In essence, $\hat{d}$ is similar to the centroid, i.e., the descriptor from $K(o)$ that minimizes the Hamming distance *dist* to all the other descriptors in $K(o)$.

4. Proposed method

This section provides an explanation of the main steps involved in the creation of the base map and its later use with a different camera (augmented map).

We distinguish between an initial phase in which the base map is created using one camera (sections 4.2-4.5)

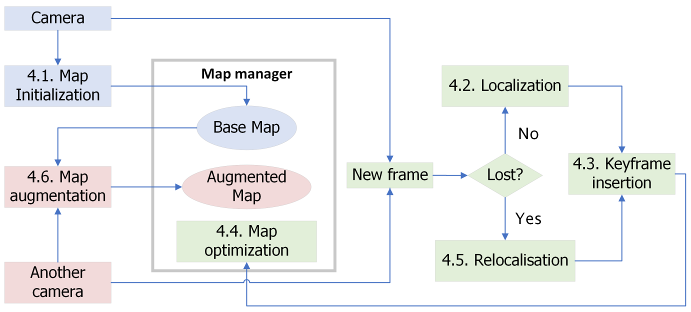


Figure 3: Workflow of the proposed method. The modules used only for the creation of the base map are shown in blue, while the modules for the augmentation of the base map with another camera are in red. In green the modules used for both approaches.

and then a second phase in which the base map is reused by possibly a different camera obtaining what we call the augmented map (section 4.6). We consider the base map to be a stable and robust map that is carefully created to serve as the base for multiple AS to navigate, see Figure 3.

4.1. Map Initialization

The process starts with an empty model Ω, i.e., $\mathcal{O} = \{\varnothing\}$, $\mathcal{F} = \{\varnothing\}$, $\mathcal{G} = \{\varnothing\}$ and $\mathcal{B} = \{\varnothing\}$. The map initialization consists in obtaining an initial pair of keyframes from the frame sequence $\{f_0, \ldots, f_n\}$ such that the relative pose between them can be obtained. In this work, we apply the same approach as in [2] and [10]. We first select f_0 and f_1 and find keypoint matches between them using their descriptors. Although the proposed method is capable of using different types of descriptors (binary or real), ORB is used by default, mainly due to its high performance, although it could be replaced by any other. Then, we calculate the essential and homography matrices and analyze which one explains the scene better. While the homography matrix is exclusively applicable to planar scenes, the essential matrix becomes necessary when the scene is non-planar. To determine the appropriate matrix for a given scenario, the selected method initiates only when it detects sufficient parallax between the frames, and it ensures a robust solution through the application of a set of heuristics [38, 10].

If the number of correctly triangulated matches is at least 50, then f_0 and f_1 are inserted in $\mathcal{F}$ as the first keyframes. The triangulated matches are added as the first map points to $\mathcal{O}$, the graph is $\mathcal{G}$ initialized, and the keypoint descriptors of the frames are added to the BoW database $\mathcal{B}$. Additionally, we assume that the first keyframe f_0 is the center of the map reference system, thus, all map points are referred to it. Also, the pose of the second keyframe θ^{f_1} w.r.t the first frame is known from its homography or essential matrix.

However, if the pose is not good enough, we test again with the frames f_0 and f_2, and so on, until a good solution is found. After a few frames, we replace the role of f_0 with another frame of the sequence and repeat the previous operations.

4.2. Localization

Once the map is initialized, we analyze the subsequent frames of the sequence in order to obtain their poses w.r.t. the initial keyframe (i.e., the map reference system). This is known as the localization problem. To solve it, first we must match map points on the current frame and then calculate the pose by minimizing the reprojection error. In order to speed up the process, the algorithm uses a *reference keyframe* $\hat{f} \in \mathcal{F}$, which is the keyframe with the highest number of map points matched with the previous frame.

The process starts with the input frame f_i that needs to be processed in order to obtain its pose θ^{f_i}. We have the reference keyframe $\hat{f}_{i-1}$ and the previous pose $\theta^{f_{i-1}}$. The first goal is to find matches between the map points and the current frame keypoints that we denote as:

$$\Gamma_{f_i} = \{(f_i, k, o)\}. \tag{10}$$

The tuple (f_i, k, o) indicates that the map point $o \in \mathcal{O}$ projects onto the keypoint k of frame f_i. Please notice that we assign a single keypoint match for each map point. In order to find these matches, we proceed as follows. First, we select the map points that are likely to be observed on the current frame (i.e., the map points that we know project on the reference keyframe) and the keyframes connected to it (using the connection graph $\mathcal{G}$). We filter out map points by analyzing their visibility with the normal vector o^v.

Second, the selected map points are projected onto the frame plane using the pose estimation $\theta^{f_{i-1}}$. For each projected map point, we calculate the pyramid level where it should appear (based on its distance to the frame), and select, in a radius r around the projection, the keypoint k that minimizes the descriptor distance to the map point descriptor $o^{\hat{d}}$. Please consider that a map point represents a circular patch in real space, and its size is determined by the pyramid level at which it is initially detected (the lower the level, the bigger the patch). Thus, based on the distance from the map point to the keyframe, we can estimate the pyramid levels at which it should appear.

As a result of the previous process, we obtain the set of matches Γ_{f_i} employed to estimate the current frame θ^{f_i} by minimizing the reprojection error of the map points over the matched keypoints. The reprojection of x^o on a keyframe f differs slightly from its matched keypoint, and this error can be calculated as:

$$E(f, o, k) = \Psi(\theta^f, I^f(\delta), x^o) - p^k, \tag{11}$$

where Ψ represents the map point projection, as a function of the frame pose θ^f, and the image parameter $I^f(\delta)$.

The pose of the frame can then be obtained by minimizing the sum of the reprojection errors:

$$\theta^{f_i} = \underset{\theta}{\operatorname{argmin}} \sum_{(f_i,k,o)\in\Gamma_{f_i}} H_\alpha(E(f_i, o, k) \cdot \phi_o \cdot E(f_i, o, k)^T), \tag{12}$$

where H_α is the Hubber loss function (employed to reduce the influence of outliers), and ϕ_o, is the information matrix that modulates the importance of each error. We shall define it as:

$$\phi_o = w^o \tau^{l^k} \pi^{\beta^o} M, \tag{13}$$

where M is the 2×2 identity matrix, which is multiplied by a set of weights. First, w^o models the temporal importance of the map point so that points that are observed recurrently are given a higher relevance. Second, τ^{l^k} models the spatial reliability, which decreases for keypoints found in lower levels of the pyramid. And third, π^{β^o} defines the importance of a point as a function of the type of camera it is seen from. The parameter $\pi \in (0, 1]$ gives more importance to map points inserted in the base map (current phase) than to those inserted later. In our experience, this parameter is important because it forces the system to rely on the base map when using other cameras. In particular, it prevents creating alternative maps while navigating known areas, preventing the system from drifting away from the base map.

Once the pose θ^{f_i} is estimated, we analyze if the number of inliers of the optimization is above the threshold τ_r. If so, we assume a correct tracking and select as reference keyframe $\hat{f}_i$ the keyframe of the map with the highest number of map point matches in the frame f_i. If the number of inliers drops below τ_r, the pose is considered unreliable, and the algorithm enters in *lost* mode and will require relocalization in the following frames (see section 4.5).

4.3. Keyframe insertion

Over time, the map needs to be populated with new keyframes in order to account for new areas being explored. When a new keyframe is added to the map, new map points are created and the rest of the map sets are updated as well.

A frame f_i with valid pose (from the previous step) is added to the map if the number of map points matched is below a constant percentage τ_k of the total number of map points matched in the reference keyframe $\hat{f}_i$. If the frame is added, the graph $\mathcal{G}$ is updated indicating the new node and its connections, and also the BoW database $\mathcal{B}$ is updated with the new keyframe. Then, the existing map point observations $K(o)$ are updated, i.e., some map points have new observations. And finally, new map points are added to the map considering the new keyframe.

To add a new keypoint, the following principles are followed. For each keypoint $k \in f_i$, we search correspondences in the neighbor keyframes (according to $\mathcal{G}$). Since the keyframe poses are known, matches can be accurately found applying epipolar restrictions. We only consider possible keypoint matches in the pyramid levels above and below k^l. Matches are triangulated and new map points are added to the map. New map points are considered unstable, and thus, their weight w^o will be low (see Equation 13). A map point becomes reliable when it is seen at least two times in the next three inserted keyframes [1]. As the number of times a map point is seen amongst the keyframes, its weight is increased.

Please notice that our method is designed to operate with heterogeneous cameras. In order to account for that information, new points in this stage are assigned with the values $\beta^o = 0$, indicating that they belong to the base map.

Finally, in order to prevent the map from becoming overpopulated with unnecessary map points, we perform a culling process that removes map points that have not been matched a minimum number of times after its insertion into the map. Also, we remove redundant keyframes, i.e., those with keypoints that match map points already matched in other keyframes at higher levels of the pyramid [2, 10].

4.4. Map optimization

When a keyframe is inserted in the map, the system performs an optimization to jointly refine the keyframe poses and map points by minimizing the map reprojection errors, i.e., a bundle adjustment process. The optimization may be local (i.e., include only the neighbor keyframes to the one just inserted) or global (consider all keyframes). Since global optimization is a very time-consuming process, it is reserved for special cases that really require it, namely loop closures [39]. A loop closure occurs when the system revisits a region of the map. Because of the incremental drift that happens in SLAM systems, it is required a special process to detect this event and to establish matches between map points in order to merge them. When a loop closure is detected (using the BoW set $\mathcal{B}$) a global map optimization is required.

In a global optimization, the poses of all the keyframes $\mathcal{F}, |\mathcal{F}| = n$, and the coordinates of all the map points $\mathcal{O}, |\mathcal{O}| = m$, are refined

$$((\hat{\theta}^{f_1}, \hat{x}^{o_1}), ., (\hat{\theta}^{f_n}, \hat{x}^{o_m})) = \underset{(\theta^f, x^o)}{\operatorname{argmin}} \sum_{f_i \in \mathcal{F}} \sum_{(f_i,k,o) \in \Gamma_{f_i}} H_\alpha(\mathcal{E}(i)), \tag{14}$$

by minimizing the reprojection errors

$$\mathcal{E}(i) = E(f_i, o, k) \cdot \phi_o \cdot E(f_i, o, k)^T,$$

in a similar way to Equation 12. Since this optimization is a slow process (especially as the map grows), it is run by a thread running in parallel.

The local map optimization process is the same as the one described in Equation 14, but using only the keyframes connected to the one just inserted.

4.5. Relocalisation

Relocalisation is the process of finding the pose of a frame f without having prior information. This is required when either the system is in a lost state (because the localisation failed) or when a new camera is used for the first time on the base map. In both cases, the system uses the input frame f to find a set of similar keyframes in the database $\mathcal{B}$. For each keyframe, the associated map points are searched on the frame f and a PnP algorithm (using RANSAC) is applied to find the most likely pose. If the total number of inliers is above 80%, then pose obtained can be considered reliable and the system can enter in a tracking state.

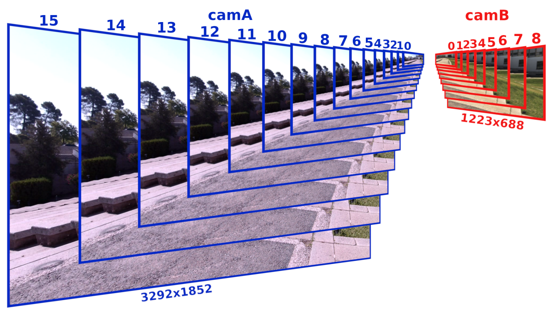
4.6. Map augmentation

The base map generated with our method can be reused by other cameras with different resolutions and optics. The base map is then augmented with keyframes of the new camera and map points created by mixing observations from the base map and the recently added keyframes. From the practical and theoretical point of view, reusing the map simply consists in processing frames f with the new camera and following the workflow explained above. The augmented map allows adapting to the new camera characteristics while being anchored in the base map as much as possible to avoid drifting from it. The new map points added in the augmented map will be assigned with a lower relevance in the optimization process by setting $\beta^o = 1$ (see Equation 13). Nevertheless, the system can create new paths in unexplored areas of the environment using the new camera.

The key to camera compatibility is the image pyramid of fixed focal lengths built in a bottom-up fashion. If one realizes that an image keypoint represents a region of the space with a specific physical dimension, it becomes clear that in our pyramid model, keypoints from different cameras at the same level represent regions with the same dimensions and thus are directly comparable, unlike previous SLAM approaches. Therefore, our method can seamlessly work with images of different cameras.

Let us consider an example where the base map is generated using a high resolution 4K camera (camA) with 3.6 mm lens where the focal length, obtained by calibration, is $I(\delta) = 3594$ pix. Using the base focal $I_0(\delta) = \delta_{min} = 200$ and the scale factor $\tau = 1.2$, we obtain a total of 16 pyramid levels (i.e., $\eta = 15$). Then, let us imagine we want to reuse the base map using a low-resolution 1MP camera (camB) with 2.8 mm lens for which $I(\delta) = 900$ pix. Then, an image pyramid consisting of 9 levels is built for the frames of this camera, i.e., $\eta = 8$. When using the low-resolution camera with the base map, the last seven levels of the pyramid are not employed, it will only use the lowest levels of the pyramid, see Figure 4.

In the previous case, we have considered that the base map is created with a high-resolution camera and the augmented map with a low-resolution one. Although our system can be used the other way around, in our opinion, this



Pyramid levels (j)	Focal lengths $I_j(\delta)$	Image resolutions $I_j(s)^{camA}$	$I_j(s)^{camB}$
0	200	214×120	284×160
1	240	256×144	341×192
2	288	308×173	410×230
3	345	369×208	492×276
4	414	443×249	590×332
5	497	532×299	708×398
6	597	638×359	849×478
7	716	766×431	1019×573
8	859	919×517	1223×688
9	1031	1103×620	–
10	1238	1323×744	–
11	1486	1588×893	–
12	1783	1905×1072	–
13	2139	2286×1286	–
14	2567	2744×1543	–
15	3081	3292×1852	–

Figure 4: Examples of pyramids created with two different cameras. Starting from a minimum focal length of $\delta_{min} = 200$, and considering a scale factor of $\tau = 1.2$, the image pyramids are obtained. Note that the number of levels of the pyramid j, will be determined by the original focal length of the camera, i.e., the larger the image, the more levels the pyramid has.

is the most useful case: the map is generated offline from the recording of a good camera and then used in real-time with a lower-end computer and camera (e.g., a drone or robot). It must be noticed that the computing effort required to run the system is directly proportional to the number of pyramid levels.

5. Experiments and results

This section presents the experiments carried out to validate the proposed method *ReSLAM*. The source code and the datasets employed in this paper are public for research purposes and evaluation[4].

Our method is the only one (up to our knowledge) capable of mixing cameras of different resolutions and optics on the same map, and no public dataset has been found for a proper evaluation. As a consequence, we introduce in this paper six new evaluation datasets, which are described in section 5.1.

[4]`https://www.uco.es/investiga/grupos/ava/portfolio/reslam`

Table 1: Set of parameters of our method and the values employed for experimentation.

Parameter	*Value*	*Description*
δ_{min}	200	Minimum focal length (section 3.1).
τ	1.2	Scale factor (section 3.1).
τ_0	140	Number of features in I_0 (Eq. 5).
π	0.5	Weight of new points (Eq. 13).
τ_r	30	Threshold for valid pose (section 4.2).
τ_k	0.8	Threshold for adding a new keyframe (section 4.3).

All the experiments have been carried out with an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, with 46GB of RAM, using Ubuntu 20.04.2. In addition, the proposed method has several parameters that have been introduced in the previous sections. Table 1 shows the values employed in the experiments. Please, note that the minimum focal length parameter (δ_{min}) is the focal length of the smallest image in the pyramid. The image pyramid is constructed for each input image in the SLAM system and is used for keypoint detection, section 3.1.

To ensure a sufficient number of image pyramid levels, we have set the minimum focal length to 200. This guarantees that the sensor with the smallest focal length ($\delta = 639$) has a minimum of 7 pyramid levels, and the sensor with the largest focal length ($\delta = 2276$) has a maximum of 14 levels.

Our method is compared with three state-of-the-art SLAM methods, namely *ORB-SLAM*[10, 11], *OpenVSLAM*[4] and UcoSLAM[2]. Note that despite the fact that UcoSlam can work using both fiducial markers and keypoints (or a combination of both) only keypoints have been used in our experimentation. On the other hand, although these methods are not specifically designed to reuse a map with different cameras, we have applied the naive approach suggested in section 1, which consists in adapting the new images to match the ones employed to create the map. We will explain this in more detail later.

In any case, comparing two SLAM methods is not a trivial task since we need to evaluate two different measures. On the one hand, we must compare the error in the trajectories of the methods, and on the other hand, the total number of frames actually tracked. To properly compare two methods considering both aspects we are employing the methodology proposed in [2].

The rest of this section is structured as follows. Section 5.1 presents the six new datasets created for this work. Section 5.2 explains the methodology employed to test our method and compare it with the other methods that do not support map reuse. Section 5.3 explains the evaluation measures employed. Finally, Section 5.4 shows the results obtained by the different methods, and Section 5.5 analyses their processing speeds.

5.1. Datasets

Six different datasets have been created for this work (Dataset-00 to Dataset-05), each one of them corresponding to both indoor and outdoor different areas of our

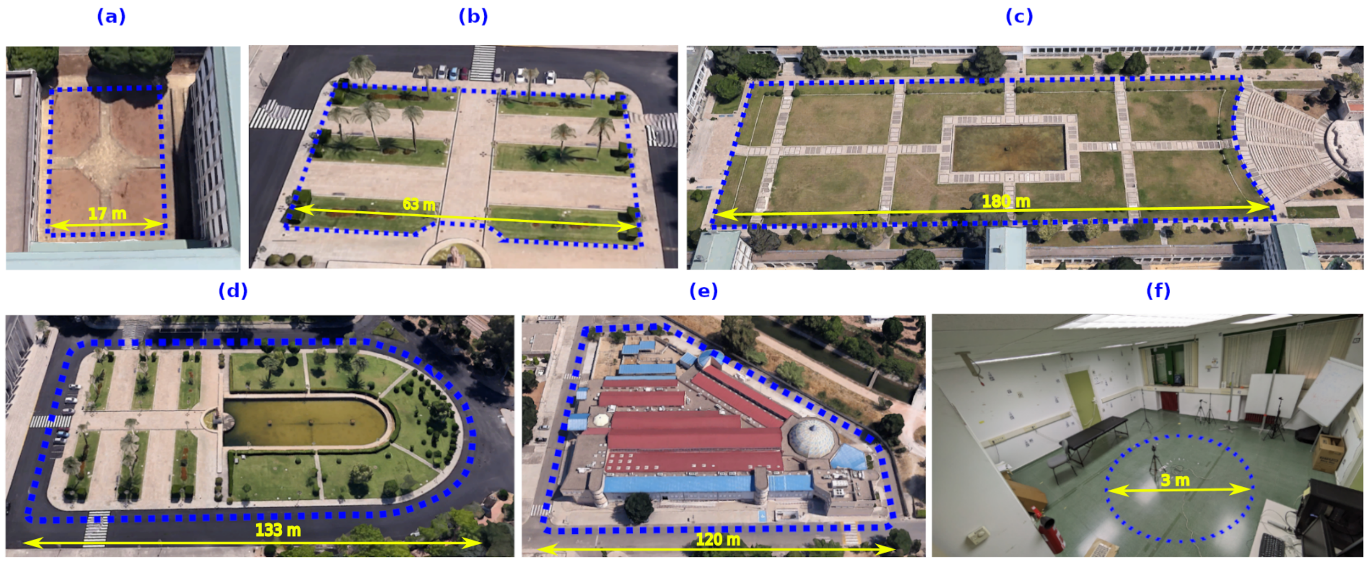


Figure 5: Overview of the trajectories covered in the datasets created. a) Dataset-00, b) Dataset-01, c) Dataset-02, d) Dataset-03, e) Dataset-04, f) Dataset-05.

Table 2: Resolution and focal length of the cameras employed in Dataset-00, Dataset-01, and Dataset-02

	Resolution	Focal length	Model
Camera 0	3840×2160	2276	ELP USB4K02AF-KL100
Camera 1	1920×1080	1458	Logitech HD PRO C920
Camera 2	1920×1080	1800	Iphone SE (1st gen.)
Camera 3	1280×720	639	Intel RealSense D345

Table 3: Resolution and focal length of the sensors employed for Dataset-03 y Dataset-04

	Resolution	Focal length	Model
Camera 4	1920×1080	2228	ELP USB4K02AF-KL100
Camera 5	1920×1080	1472	Logitech HD PRO C920
Camera 6	960×540	976	Iphone SE (1st gen.)
Camera 7	1280×720	754	ELP USB4K02AF-KL100

campus. Figure 5 shows images of the datasets. The datasets cover the following total distances (shown as blue dotted lines in Figure 5): Dataset-00=79.4m, Dataset-01=203.4m, Dataset-02=478.6m, Dataset-03=397.1m, Dataset-04=392.2m and Dataset-05=9.42m. The first three datasets have been recorded using four different cameras with the specifications shown in Table 2. Each sequence was recorded independently from the other by a person walking with the camera in his/her hands. Dataset-03 and Dataset-04 have been recorded using a different set of cameras (Table 3), all mounted on a car. In that case, the four cameras were recording simultaneously. Finally, for Dataset-05 three different cameras have been used (Table 4), recording the walls of a room as it is walked through. In all cases, the sequences were recorded at 30 fps. In total, our datasets consist of twenty different outdoor video sequences (Dataset-00 to Dataset-04), four for each camera and dataset; and six indoor video sequences (Dataset-05), two sequences for each camera. Each sequence begins and ends at the same place in order to be able to perform loop closure.

For each outdoor dataset, a set of control points evenly distributed along each path has been established in order to scale and align the method's estimated trajectories to the ground truth. On the other hand, for the indoor dataset, we employed an Optitrack motion capture system to obtain precise ground truth camera pose.

Table 4: Resolution and focal length of the sensors employed for Dataset-05

	Resolution	Focal length	Model
Camera 8	1280×720	737	ELP USB4K02AF-KL100
Camera 9	1920×1080	1517	ELP USB48MPO1-KAF70
Camera 10	1600×896	1178	Logitech HD PRO C920

5.2. Evaluation methodology

Our goal is to validate the performance of the proposed method a) when the base map is created, and b) when the base map is augmented with a different camera (cross-evaluation). To do so, we have employed the following methodology for each dataset. Each video sequence has been processed obtaining the corresponding base map. Then, for each base map, we cross-evaluate the SLAM system using the rest of the video sequences.

Given that the trajectories obtained lack scale, to evaluate the error between different trajectories, it is necessary to translate and scale each trajectory to the ground truth reference system. For this, the established control points and the poses of the camera have been used in known positions of the trajectory, making use of Horn's method [40].

Then, the Absolute Trajectory Error (ATE) is computed as the RMSE between the positions of the camera in each frame and the ground truth trajectory. However, the ATE can only be computed in those frames where the system provides a valid estimation, i.e., it does not get lost. So, to account for that issue, we also compute for each sequence the percentage of frames the system is able to track, and we call this measure *frames successfully tracked* (FST) as an additional measure of reliability.

While our SLAM method has been specifically designed to use different cameras, this is not the case for ORB-SLAM [10, 11], OpenVSLAM[4] and UcoSLAM[2]. Nevertheless, it is possible to cross-evaluate the video sequences by first creating the map with one camera and then using a second camera by adapting its images to have the same focal length and dimensions as the first one. For example, imagine that we create the map using a 4K camera (3840×2160pix,$I(\delta) = 3594$), and we want to reuse it with a 1MP camera (1280×720pix,$I(\delta) = 900$). If we upsample the 1MP image to 5111×2875pix, the resulting image will have $I(\delta) = 3594$. We can then crop the upsampled image to be 3840×2160pix by discarding the external pixels. Then, the new image can be provided to the SLAM method for cross-evaluation against the first camera. The opposite operation can be also done, using the 1MP camera to create the map and then the 4K camera to cross-evaluate. In this case, the 4K needs to be downsampled to 961×540pix and we will need to add an external border of black pixels to make it 1280×720pix. As one can imagine, this is a suboptimal solution, especially for the case when one needs to upsample, because it will artificially increase the computing time making it inappropriate for low-end computers.

5.3. Evaluation measures

As already indicated, comparing two methods by analyzing only their ATE is not a valid approach. In some cases, a SLAM method fails to track and enters in a lost state. The ATE of the affected frames can not be measured and thus can not be compared. Imagine the extreme case of a SLAM method that is only able to track the first ten frames of a very long video sequence. The ATE computed over these frames is very low because it is at the start of the sequence, and drift is minimal. However, another SLAM method able to track the whole sequence decently will most probably have a higher ATE over the whole set of frames. To avoid these problems, Munoz *et.al.* propose in [2] an evaluation measure that considers both the ATE and the FST to compare two SLAM methods. Below, we provide an overview of it.

Given two methods **m** and **n**, we shall denote E_m^n the ATE of method **m** given the method **n**, which refers to the ATE of the method **m** in the frames tracked by both methods. Likewise, we define E_n^m as the ATE of the method **n** given the method **m**. Also, let us denote T_m and T_n as the total number of frames in which the methods **m** and **n** provide valid tracking results, respectively.

Based on the explanation provided, we define the measure $S_p^s(m,n) \in [0,1]$ for a video sequence s. The measure is calculated according to the following set of rules:

- $S_p^s(m,n) = 1$ if E_n^m is significantly smaller than E_m^n and T_m is significantly larger than T_n.
- $S_p^s(m,n) = 0.5$ if E_n^m is significantly smaller than E_m^n, but the difference between T_m and T_n is not significant.
- $S_p^s(m,n) = 0.5$ if the difference between E_m^n and E_n^m is not significant, but T_m is significantly larger than T_n.
- $S_p^s(m,n) = 0$ otherwise.

These rules can be mathematically formalized as:

$$S_p^s(m,n) = \begin{cases} 1 & (E_m^n - E_n^m) > p \cdot E_n^m \wedge (T_m - T_n) > p \cdot T_m \\ 0.5 & (E_m^n - E_n^m) > p \cdot E_n^m \wedge |T_m - T_n| \leq p \cdot \max(T_m, T_n) \\ 0.5 & |E_m^n - E_n^m| \leq p \cdot \min(E_m^n, E_n^m) \wedge \ (T_m - T_n) > p \cdot T_m \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

The measure analyzes if E_m^n is greater than E_n^m, and if the difference is larger than the threshold $p \cdot \min(E_m^n, E_n^m)$. The threshold avoids infinitesimal differences to be considered relevant and is controlled by the confidence value $p \in (0,1]$. The measure $S_p^s(m,n)$ is 1 if method **m** performs better than method **n** in all the frames of the sequence s, and vice versa if it tends to 0.

Finally, given a set of video sequences $S = \{s\}$,

$$S_p(m,n) = \frac{\sum_{s \in S} S_p^s(m,n) - S_p^s(n,m)}{|S|} \tag{16}$$

evaluates which method performs better over the sequences. The measure $S_p(m,n) \in [-1,1]$ is 1 if **m** is better than **n** in all video sequences, and it tends to -1 in the opposite case.

The measure $S_p(m,n)$ is a fair value indicating which method is better over several datasets and will be employed in this work.

5.4. Evaluation of Methods

This section shows the results obtained for the methods evaluated, namely ReSLAM, ORB-SLAM2, OpenVSlam, and UcoSLAM. The results of ORB-SLAM3 [11] have not been presented in this paper because it has not been able to obtain decent cross-evaluation results in any of the datasets. We believe that is because of its implementation, which does not allow the use of other cameras, even when applying the naive adapting strategy.

Since the measure $S_p(m,n)$ is built on top of the ATE and FST ones, we first present their values for each dataset. Table 6 show the average ATE and FST of each method during the map creation of the maps for each dataset. These values represent how good each method is in creating the base maps. As it can be observed, our

Table 5: Results of $S_p(m,n)$ for each pair of methods compared methods m and n, using different values of the confidence measure p. $S_p(m,n) > 0$ means method m is better than method n.

p	Method m	Method n	Dataset-00	Dataset-01	Dataset-02	Dataset-03	Dataset-04	Dataset-05	All Datasets
0.01	ReSLAM	ORB-SLAM2	0.41	0.50	0.10	0.14	0.04	0.26	0.24
	ReSLAM	OpenVSLAM	0.58	0.79	0.67	0.25	0.42	0.42	0.52
	ReSLAM	UcoSLAM	0.04	0.13	0.04	0.00	0.29	0.08	0.10
	ORB-SLAM2	OpenVSLAM	0.41	0.21	0.43	0.09	0.50	0.46	0.35
	UcoSLAM	ORB-SLAM2	0.23	0.42	-0.05	0.32	-0.33	0.11	0.12
	UcoSLAM	OpenVSLAM	0.67	0.63	0.56	0.25	0.21	0.60	0.49
0.05	ReSLAM	ORB-SLAM2	0.41	0.50	0.10	0.05	0.08	0.30	0.24
	ReSLAM	OpenVSLAM	0.54	0.71	0.61	0.17	0.38	0.39	0.47
	ReSLAM	UcoSLAM	0.04	0.08	-0.04	0.00	0.25	0.12	0.08
	ORB-SLAM2	OpenVSLAM	0.36	0.13	0.36	0.05	0.46	0.39	0.29
	UcoSLAM	ORB-SLAM2	0.27	0.42	-0.05	0.14	-0.29	0.11	0.10
	UcoSLAM	OpenVSLAM	0.58	0.54	0.50	0.13	0.21	0.54	0.42
0.1	ReSLAM	ORB-SLAM2	0.41	0.50	0.10	0.05	0.13	0.22	0.24
	ReSLAM	OpenVSLAM	0.58	0.63	0.56	0.13	0.29	0.35	0.42
	ReSLAM	UcoSLAM	0.00	0.08	-0.08	0.04	0.25	0.12	0.07
	ORB-SLAM2	OpenVSLAM	0.36	0.13	0.29	0.05	0.38	0.28	0.25
	UcoSLAM	ORB-SLAM2	0.27	0.46	0.00	0.09	-0.25	0.06	0.11
	UcoSLAM	OpenVSLAM	0.58	0.50	0.50	0.08	0.13	0.46	0.38
0.25	ReSLAM	ORB-SLAM2	0.27	0.50	0.15	0.09	0.08	0.11	0.20
	ReSLAM	OpenVSLAM	0.58	0.63	0.61	0.08	0.25	0.27	0.40
	ReSLAM	UcoSLAM	0.04	0.17	0.04	0.08	0.21	0.00	0.09
	ORB-SLAM2	OpenVSLAM	0.41	0.17	0.29	0.05	0.38	0.26	0.26
	UcoSLAM	ORB-SLAM2	0.23	0.42	0.05	0.09	-0.17	0.04	0.11
	UcoSLAM	OpenVSLAM	0.54	0.54	0.50	0.00	0.13	0.31	0.34

Table 6: Average Average Absolute Trajectory Error (ATE) and Percentage of Frames Successfully tracked (FST) of the methods evaluated in the creation of the base map across the six datasets.

	ReSLAM		ORB-SLAM2		OpenVSLAM		UcoSLAM	
	ATE	FST	ATE	FST	ATE	FST	ATE	FST
Dataset-00	**0.23**	99.83	0.41	97.51	0.43	99.96	0.49	**100.00**
Dataset-01	0.50	99.87	0.44	60.43	**0.27**	79.26	0.52	**100.00**
Dataset-02	**0.93**	99.30	6.97	94.35	1.65	94.13	1.09	**99.89**
Dataset-03	**1.51**	99.61	6.30	86.36	4.78	99.71	2.83	**99.97**
Dataset-04	**1.35**	95.09	9.31	93.46	2.93	**98.22**	11.55	96.31
Dataset-05	0.15	99.81	0.32	96.23	0.09	97.13	**0.07**	**99.97**
Avrg.	**0.78**	98.92	3.96	88.06	1.69	94.73	2.75	**99.36**

Table 7: Average Average Absolute Trajectory Error (ATE) and Percentage of Frames Successfully tracked (FST) of the methods evaluated using cross-evaluation across the six datasets.

	ReSLAM		ORB-SLAM2		OpenVSLAM		UcoSLAM	
	ATE	FST	ATE	FST	ATE	FST	ATE	FST
Dataset-00	0.29	**99.83**	0.86	88.43	0.38	58.89	**0.27**	99.76
Dataset-01	**0.44**	**99.87**	8.54	81.19	1.88	57.65	0.70	96.39
Dataset-02	**0.85**	90.90	2.31	70.05	9.53	53.64	1.66	**99.35**
Dataset-03	**1.34**	99.81	10.16	89.30	4.25	96.82	5.17	**99.87**
Dataset-04	**1.89**	99.61	2.43	**99.97**	6.80	91.37	13.61	98.57
Dataset-05	**0.08**	95.49	0.14	97.86	0.36	88.20	0.14	**99.42**
Avrg.	**0.82**	97.58	4.07	87.80	3.87	74.43	3.59	**98.89**

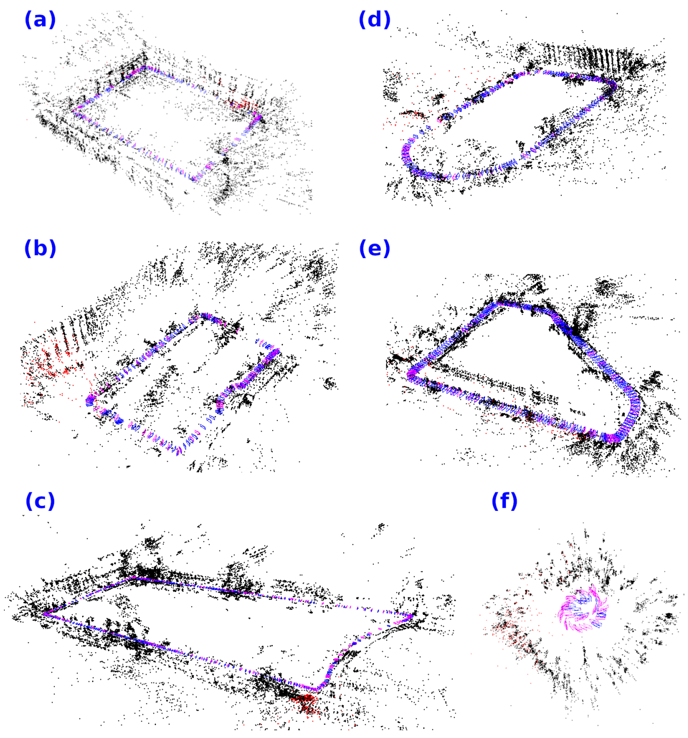


Figure 6: Examples of augmented maps created with our system. (a) Dataset-00, (b) Dataset-01, (c) Dataset-02, (d) Dataset-03, (e) Dataset-04, (f) Dataset-05.

method obtains the lowest ATE in all the datasets, except for Dataset-01 and Dataset-05, where it obtains a value very similar to UcoSLAM. In terms of FST, all methods behave similarly except for ORBSLAM2, which shows the worst performance. Table 7 show the average ATE and FST in cross-evaluation, i.e., using the base map with a different camera. As it can be observed, our method obtains the best ATE results in all cases, with the exception of Dataset-01, where it obtains a value very similar to UcoSLAM. While for Dataset-00, Dataset-01 and Dataset-05, the differences are not very high, they are more evident in the other datasets. Regarding the FST, the proposed method ranks higher in Dataset-00 y Dataset-01. For the other datasets, the FST values obtained are close to the best result, except Dataset02 and Dataset-05, where UcoSLAM obtains better results but at the expense of obtaining a higher ATE.

Figure 6 shows some of the augmented maps generated by our method in the six datasets. The dots represent the three-dimensional points of the map and the blue elements are the keyframes of the base map. The red element represents the keyframes of the augmented map when using a different camera for cross-evaluation. As it can be visually observed, the reconstruction matches the real environment with low errors.

Table 5 shows the values $S_p(m, n)$ comparing all methods using the cross-evaluation data for different confidence values. A positive value indicates that the method $\mathbf{m}$ outperforms method $\mathbf{n}$. As can be observed, it can be said that our method, ReSLAM, is better than the rest of the methods in all datasets. The exception is Dataset-02 where UcoSLAM obtains slightly better results. UcoSLAM is the second-best method.

5.5. *Processing speed*

Table 8: Average processing speed (fps) of each method in creating the map and cross-evaluation using a different camera. The best cases are marked in bold.

	ReSLAM		ORB-SLAM2		OpenVSLAM		UcoSLAM	
	Map	CE	Map	CE	Map	CE	Map	CE
Dataset-00	9.56	**9.13**	2.43	1.22	6.03	2.89	**9.89**	3.99
Dataset-01	**16.98**	**17.94**	4.33	2.71	7.75	3.77	14.66	6.59
Dataset-02	**16.51**	**18.97**	1.40	0.95	7.00	4.28	12.56	5.80
Dataset-03	**19.69**	**24.09**	2.70	1.65	3.20	3.78	8.85	4.19
Dataset-04	**12.25**	**18.84**	2.29	1.28	4.65	2.76	6.76	3.83
Dataset-05	20.61	**22.12**	7.66	4.26	12.46	8.67	**21.01**	18.79

This Section analyzes the processing speeds of our method and compares it with the naive approach. Table 8 shows the processing speed (expressed in frames per second) of the different methods tested. Column *Map* refers to the average fps of a method in creating the base map for a particular dataset, while column *CE* shows the average fps required for cross-evaluation. As can be seen, the proposed method is highly optimized and obtains the highest speed in almost all cases for creating the base map, ORB-SLAM2 being the slowest one.

However, it is in cross-validation where our method really makes an important difference w.r.t. the naive approach. Because of its design, our method can use a different camera without degrading its performance. In our method, using a camera of lower resolution for cross-validating generally results in an increase of the processing speed since the amount of data to be processed is reduced. However, for the other methods, we have to artificially upsample the input image to match the one that created the map.

Table 9 shows the detailed performance of the different methods for each one of the cameras employed in cross-evaluation. In the table, one can better analyze the difference between our method and the rest. The table shows for each camera, the average cross-evaluation fps of our method and the rest of the methods.

It can be seen that our method is at least two times faster than the rest of the methods. While for the other methods, using lower resolution cameras does not increase

Table 9: Average processing speed (fps) of each camera when used in cross-evaluation. The best cases are marked in bold.

Camera (resolution)	ReSLAM	ORB-SLAM2	OpenVSLAM	UcoSLAM
cam6 (960×540)	**21.79**	1.85	3.50	3.74
cam8 (1280×720)	**27.80**	6.23	12.26	26.02
cam7 (1280×720)	**20.80**	0.99	3.39	2.91
cam3 (1280×720)	**20.89**	2.40	3.09	8.40
cam10 (1600×896)	**20.15**	3.81	8.04	17.11
cam5 (1920×1080)	**22.95**	1.20	2.89	3.76
cam4 (1920×1080)	**20.33**	1.81	3.31	5.63
cam9 (1920×1080)	**18.41**	2.69	5.72	13.25
cam2 (1920×1080)	**18.21**	2.11	4.47	4.43
cam1 (1920×1080)	**14.16**	0.56	3.67	4.26
cam0 (3840×2160)	**10.52**	1.18	2.72	4.29
$Avrg(fps)$	**19.64**	2.26	4.82	8.53

the performance, our method is specifically designed to do so without losing accuracy. As previously explained, our proposal is better suited for low-end systems that need to navigate an environment using a pre-created base map.

6. Conclusions and future work

Keypoint-based SLAM systems are a powerful tool for robot navigation. However, they have some drawbacks. First, keypoints change over time and are difficult to detect in low-texture scenarios. Second, when using monocular systems, it is not possible to determine the scale. Third, in general, maps generated with a camera can not be reused with a different camera.

This work has proposed a novel keypoint-based SLAM method for reusing maps created by heterogeneous cameras. Our method, coined ReSLAM, allows creating a map of the environment with a camera that is later reused with another camera of different optics and resolutions. This is the first approach in the literature (up to our knowledge) that addresses that problem. Our proposal's key consists of a bottom-up pyramidal representation of the images that seamlessly match keypoints across different cameras.

ReSLAM has the additional advantage of adapting to the camera resolution, reducing the computing requirements accordingly. This makes our method especially appropriate in scenarios where a base map is created off-line with a high-resolution camera and then reused by a lower-end device.

Since no public datasets have been created (to our knowledge) to evaluate SLAM with heterogeneous cameras, this work has created six new datasets for evaluation purposes. Four different cameras have been employed to record the same trajectory in our university campus area in each dataset.

Our experiments show that our method outperforms state-of-the-art methods in terms of accuracy and speed. Both the datasets and the code are publicly available.

In future work, we will consider creating the base map using a parallel approach. Currently, the base map is created from a single video sequence. However, the video sequence could be split into multiple chunks and processed in parallel and afterward, merge the pieces of the map generated. We also consider extending our code to operate with other sensors such as fisheye, LiDAR, and stereo cameras. Finally, we also consider the use of fiducial markers to allow estimating the real scale with monocular cameras, like UcoSLAM.

Lastly, we believe that the robustness of the base maps can be further enhanced by incorporating fiducial markers throughout the environment. This approach would enable the estimation of scale and facilitate the reuse of the base maps in changing environmental conditions, such as the presence of shadows or objects. Investigating the use of fiducial markers as part of our future work will contribute to improving the adaptability and reliability of our method.

Acknowledgments

This project has been funded under the project PID2019-103871GB-I00 of the Spanish Ministry of Economy, Industry and Competitiveness, and FEDER and Project 1380047-F UCOFEDER-2021 of Andalusia.

References

[1] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, Orb-slam: A versatile and accurate monocular slam system, IEEE Transactions on Robotics 31 (5) (2015) 1147–1163.

[2] R. Muñoz-Salinas, R. Medina-Carnicer, Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers, Pattern Recognition 101 (2020) 107193.

[3] X. Gao, R. Wang, N. Demmel, D. Cremers, Ldso: Direct sparse odometry with loop closure, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018) 2198–2204.

[4] S. Sumikura, M. Shibuya, K. Sakurada, OpenVSLAM: A Versatile Visual SLAM Framework, in: Proceedings of the 27th ACM International Conference on Multimedia, MM '19, ACM, New York, NY, USA, 2019, pp. 2292–2295.

[5] H. Lategahn, A. Geiger, B. Kitt, Visual slam for autonomous ground vehicles, in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 1732–1737.

[6] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (slam) problem, IEEE Transactions on Robotics and Automation 17 (3) (2001) 229–241.

[7] L. Jinyu, Y. Bangbang, C. Danpeng, W. Nan, Z. Guofeng, B. Hujun, Survey and evaluation of monocular visual-inertial slam algorithms for augmented reality, Virtual Reality & Intelligent Hardware 1 (4) (2019) 386–410.

[8] O. G. Grasa, E. Bernal, S. Casado, I. Gil, J. M. M. Montiel, Visual slam for handheld monocular endoscope, IEEE Transactions on Medical Imaging 33 (1) (2014) 135–146.

[9] X. S. Zhou, S. I. Roumeliotis, Multi-robot slam with unknown initial correspondence: The robot rendezvous case, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 1785–1792.

[10] R. Mur-Artal, J. D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Transactions on Robotics 33 (5) (2017) 1255–1262.

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, J. D. Tardós, Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam, IEEE Transactions on Robotics (2021) 1–17.

[12] C. Harris, J. Pike, 3d positional integration from image sequences, Image and Vision Computing 6 (2) (1988) 87–90.

[13] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, Monoslam: Real-time single camera slam, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6) (2007) 1052–1067.

[14] E. Eade, T. Drummond, Scalable monocular slam, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 1, 2006, pp. 469–476.

[15] E. Eade, T. Drummond, Edge landmarks in monocular slam, Image Vis. Comput. 27 (2009) 588–596.

[16] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, in: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 225–234.

[17] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, in: 2011 International Conference on Computer Vision, 2011, pp. 2564–2571.

[18] H. Xiu, Y. Liang, H. Zeng, Q. Li, H. Liu, B. Fan, C. Li, Robust self-supervised monocular visual odometry based on prediction-update pose estimation network, Engineering Applications of Artificial Intelligence 116 (2022) 105481.

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

[20] H. Liu, Q. Zhang, B. Fan, Z. Wang, J. Han, Features combined binary descriptor based on voted ring-sampling pattern, IEEE Transactions on Circuits and Systems for Video Technology 30 (10) (2020) 3675–3687.

[21] B. Fan, H. Liu, H. Zeng, J. Zhang, X. Liu, J. Han, Deep unsupervised binary descriptor learning through locality consistency and self distinctiveness, IEEE Transactions on Multimedia 23 (2021) 2770–2781.

[22] B. Fan, J. Zhou, W. Feng, H. Pu, Y. Yang, Q. Kong, F. Wu, H. Liu, Learning semantic-aware local features for long term visual localization, IEEE Transactions on Image Processing 31 (2022) 4842–4855.

[23] H. Liu, X. Tang, S. Shen, Depth-map completion for large indoor scene reconstruction, Pattern Recognition 99 (2020) 107112.

[24] Spm-slam: Simultaneous localization and mapping with squared planar markers, Pattern Recognition 86 (2019) 156–171.

[25] Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition 47 (6) (2014) 2280–2292.

[26] J. Engel, T. Schöps, D. Cremers, Lsd-slam: Large-scale direct monocular slam, in: ECCV, 2014.

[27] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (3) (2018) 611–625.

[28] J. Engel, J. Stückler, D. Cremers, Large-scale direct slam with stereo cameras, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 1935–1942.

[29] J. Sola, A. Monin, M. Devy, T. Vidal-Calleja, Fusing monocular information in multicamera slam, IEEE Transactions on Robotics 24 (5) (2008) 958–968.

[30] K. Jae-Hean, J. C. Myung, T. C. Byung, Recursive estimation of motion and a scene model with a two-camera system of divergent view, Pattern Recogn. 43 (6) (2010) 2265–2280.

[31] M. J. Tribou, A. Harmat, D. W. L. Wang, I. Sharf, S. L. Waslander, Multi-camera parallel tracking and mapping with non-overlapping fields of view, The International Journal of Robotics Research 34 (2015) 1480 – 1500.

[32] A. J. Yang, C. Cui, I. A. Bârsan, R. Urtasun, S. Wang, Asynchronous multi-view SLAM, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021.

[33] M. E. Ragab, K. H. Wong, Multiple nonoverlapping camera pose estimation, in: 2010 IEEE International Conference on Image Processing, 2010, pp. 3253–3256.

[34] M. Kaess, F. Dellaert, Probabilistic structure matching for visual slam with a multi-camera rig, Computer Vision and Image Understanding 114 (2) (2010) 286–296.

[35] S. Lynen, B. Zeisl, D. Aiger, M. Bosse, J. A. Hesch, M. Pollefeys, R. Siegwart, T. Sattler, Large-scale, real-time visual-inertial localization revisited, Int. J. Robotics Res. 39 (9).

[36] R. Mur-Artal, J. D. Tardós, Visual-inertial monocular slam with map reuse, IEEE Robotics and Automation Letters 2 (2017) 796–803.

[37] D. Galvez-López, J. D. Tardos, Bags of binary words for fast place recognition in image sequences, IEEE Transactions on Robotics 28 (5) (2012) 1188–1197.

[38] P. H. S. Torr, A. W. Fitzgibbon, A. Zisserman, The problem of degeneracy in structure and motion recovery from uncalibrated image sequences, International Journal of Computer Vision 32 (1999) 27–44.

[39] H. M. Strasdat, J. M. M. Montiel, A. J. Davison, Scale drift-aware large scale monocular slam, in: Robotics: Science and Systems, 2010.

[40] B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, J. Opt. Soc. Am. A 4 (4) (1987) 629–642.