FROG

FUN ROBOTIC
OUTDOOR GUIDE

Deliverable D2.1
Robot 6DOF precise localization component

Consortium

UNIVERSITEIT VAN AMSTERDAM (UvA)
YDREAMS - INFORMATICA S.A. (YD)
IDMIND - ENGENHARIA DE SISTEMAS LDA (IDM)
UNIVERSIDAD PABLO DE OLAVIDE (UPO)
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE (ICL)
UNIVERSITY OF TWENTE (UT)

Grant Agreement no. **288235**
Funding Scheme: **STREP**

DOCUMENT INFORMATION

Project

Project acronym:	FROG
Project Full Title:	Fun Robotic Outdoor Guide
Grant agreement no.:	288235
Funding scheme:	STREP
Project start date:	1 October 2011
Project duration:	30 September 2014
Call topic:	ICT-2011.2.1 Cognitive Systems and Robotics (a), (d)
Project web-site:	www.frogrobot.eu

Document

Deliverable number:	D2.1
Deliverable title:	Robot 6DOF precise localization component
Due date of deliverable:	M24 - Sept. 30, 2013
Actual submission date:	October 13, 2013
Editors:	
Authors:	UPO
Reviewers:	All Partners
Participating beneficiaries:	UPO
Work Package no.:	2
Work Package title:	Robot Control, Navigation and Location Based Content
Work Package leader:	UPO
Work Package participants:	YD, IDM, UPO, ICL, UT
Estimated person-months for deliverable:	
Dissemination level:	Public
Nature:	Other
Version:	1.1
Draft/Final	Final
No of pages (including cover):	40
Keywords:	Localization, SLAM

Contents

List of Figures

List of Tables

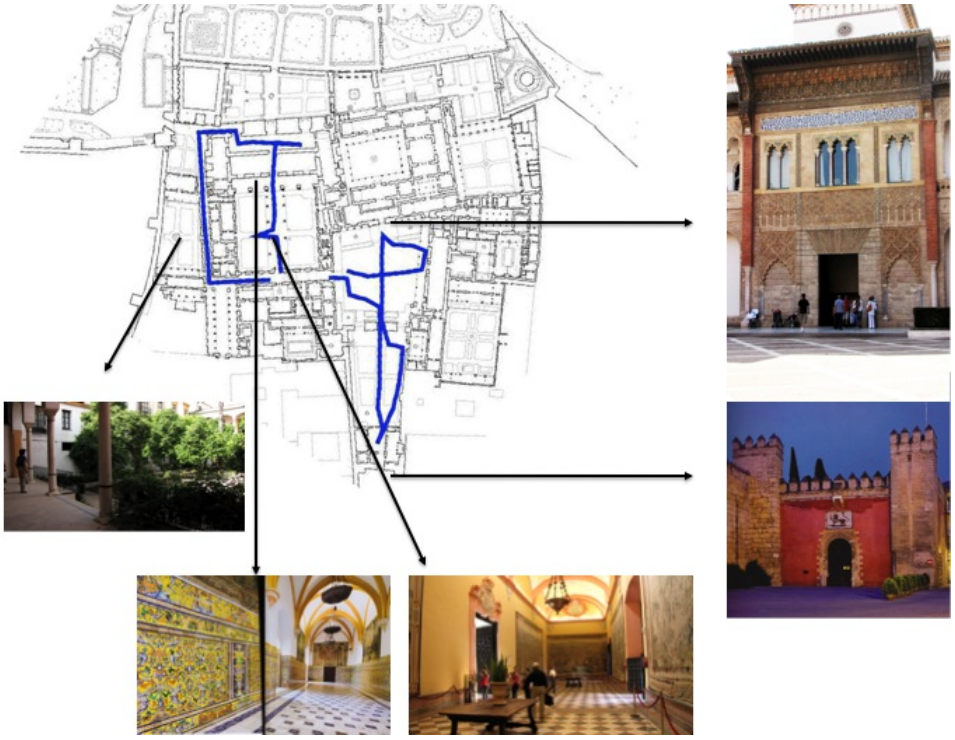

Figure 1: The Royal Alcázar of Seville. This scenario consists of a combination of outdoor and indoor places on man-made structures. The Royal Alcázar is a very crowded scenario.

1 Introduction

The objective of task 2.1 is to provide a precise-enough six degree-of-freedom localization for navigation and augmented reality purposes. Two main localization issues are relevant for the applications considered. One is accuracy, required by the Augmented Reality (AR) application. The second one is robustness, in order to achieve long-term autonomy of the robot.

Within the project, two main scenarios are being considered: the Lisbon Zoo and the Royal Alcázar in Seville (see Figs. 1 and 2). These scenarios have only partial GPS coverage. Therefore, a map-based localization approach will be considered. This requires a first phase to build a map for localization.

Furthermore, the techniques should be applicable whether the robot is deployed in those scenarios or in new scenarios in the future. The current document describes the approaches followed to build the maps required, and to localize the robot within these maps when operating in real-time.

The document is organized as follows. Section 2 will discuss the initial requirements. Then, Section 2.2 will describe broadly the design of the localization system. Then, Section 3 will describe the offline map building procedure, and Section 4 the localization module. The document will end with the initial evaluation performed, describing the methodology and results obtained.

Figure 2: The Lisbon Zoo. This scenario is mostly outdoors, including less structured buildings and roads, slopes, grass and other challenges.

2 Requirements and high-level design

2.1 Requirements

The localization module to be developed within the project should support the navigation as well as the Augmented Reality components of the robot. Both modules require as input the robot pose.

Therefore, a first set of requirements comes from the navigation on the robot in the scenarios of the project. Figures 1 and 2 show some aspects of these scenarios. The Lisbon Zoo is mostly an outdoor scenario, while the Royal Alcázar involves outdoor and indoor places. The size of both scenarios is large, being the part of the Royal Alcázar consisting of about 40,000 square meters, while the Lisbon Zoo is even larger. The routes that are being considered for the guides are of the order of 1 kilometer or more.

From the navigation point of view, the main requirement is the robustness of the localization system, so the mean time between failures of the module is minimized. Localization is a key feature for the rest of the FROG system, and therefore robustness is very important.

The second set of requirements are derived from the AR capabilities (see Fig. 3). AR can be performed with different modalities: for instance, the robot should be able to project onto internal screens virtual overlays of the real images; or it should be able to project or point with a laser over the real scenario. Deliverable D2.4 [6] summarizes the AR component. The AR component requires accuracy and repeatability from the positioning system.

These scenarios and applications are described in D1.1 [4], where also the requirements for the different functional modules of the FROG platform are indicated. From the analysis performed in the document, the target figures initially defined for the robot localization accuracy are 50 cm in position and 3 degrees on orientation.

In this document we will assess whether these values are the correct ones for the application considered and we will evaluate the results of the localization system.

Figure 3: Augmented Reality-based interaction concepts. Different interactions are being considered, such as projections on walls of information related to the location of a Point of Interest (POI), left; projections with a laser pointer or a video projector over the scenario (center and right) or live overlays on video images on the robot belly. These services require data on robot localization, with different requirements on the precision.

2.2 High-level Design

As commented above, the scenarios considered in the project are GPS-denied or the GPS coverage is limited. Therefore, a map-based localization approach is required.

On the other hand, for the applications considered the robot does not need to explore unknown spaces but it should navigate robustly on known scenarios that change slowly with time. It is also possible to deploy the robot before its operation to gather data about the place.

Thus, the approach considered in FROG is to build the maps needed by means of an offline Simultaneous Localization and Mapping (SLAM) phase. Then, these maps are exploited for localization and navigation during the operation of the robots.

In the next sections we will describe the particular design choices for the SLAM and localization algorithms. But, as a general idea, to enhance the robustness of the localization system, features from different sensor modalities will be considered in the map and during the localization phase. This way, and by means of data fusion, we aim to enhance the robustness and accuracy of the robot localization system.

2.3 Robot platform and sensors for localization and SLAM

The FROG robot platform is a four-wheeled platform with a differential drive. Options including the following sensors were considered for localization and SLAM:

- Xsense MTI-G: an Inertial Measurement Unit (IMU), able to provide angular velocities and linear accelerations, as well as a full integrated localization solution if GPS is available
- Front 2D laser range-finder: Hokuyo's UTM-30LX, with 30 meters range
- Rear 2D laser range-finder: Hokuyo's UTM-30LX, with 30 meters range
- Front Stereo Vision Camera: Dalsa Genie-HM1400 XDR (2x): a stereo pair, looking ahead and located at 1.2 m high

The position of the sensors on the robot frame can be seen in Fig. 4. Deliverable D1.3 [5] describes the final position of the sensors, as well as some further details about the robot platform.

The main input for the localization and mapping algorithms will be the IMU, used to estimate velocities; the laser rangefinders; and the output of the vision module (which provides disparity

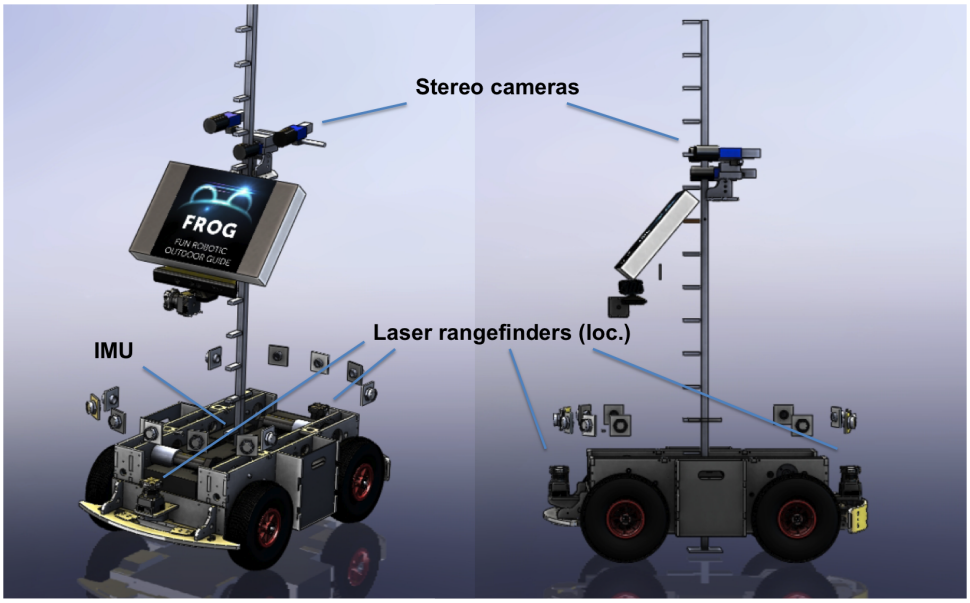

Figure 4: Robot platform and placement of sensors

maps, as well as the rectified images).

3 Map building

As commented above, the scenarios considered require a SLAM solution for building a map for localization. However, the applications considered in FROG allow for an *offline SLAM* solution: the robot can be deployed in the scenario to gather data and a map can be built offline (even though map management will be needed to add future changes in the environment)

The offline SLAM problem, also called full-SLAM problem, consists of obtaining the map and full robot trajectory given all the measurements available. Instead of filtering-based approaches typically used in online settings, such as Kalman or Information Filters [34], the offline full SLAM problem is usually tackled with optimization-based approaches. As will be seen, the full-SLAM problem can be cast as a non-linear least-squares minimization problem, in which the sensorial data provides constraints among the different variables of the problem, typically robot poses and map feature positions.

The non-linear minimization is carried out by the so-called *SLAM backend*. This backend requires an initial estimation of all the variables, as well as the constraints between them, typically encoded as a graph (or hypergraph)[21]. This graph is then provided by the so-called *SLAM frontend*. The frontend also deals with the important problem of data association.

The offline SLAM theory is maturing. Since the seminal work of Lu and Milios [24], many works have been devoted to deal with different issues. For instance, some works are devoted to the development of frontends for building the graph of constraints [14]. Most of the works have been devoted to exploit the structure of the SLAM problem for the minimization problem, as well as for several computational aspects. Different optimization methods, such as conjugate gradient [19], stochastic gradient descent [29] and others have been proposed. [13] takes into consideration that the variables of the problem do not lie on a Euclidean space, but on a manifold, which has implications on the minimization process.

The use of spare matrix factorization for solving the problem is considered in [10]. Online

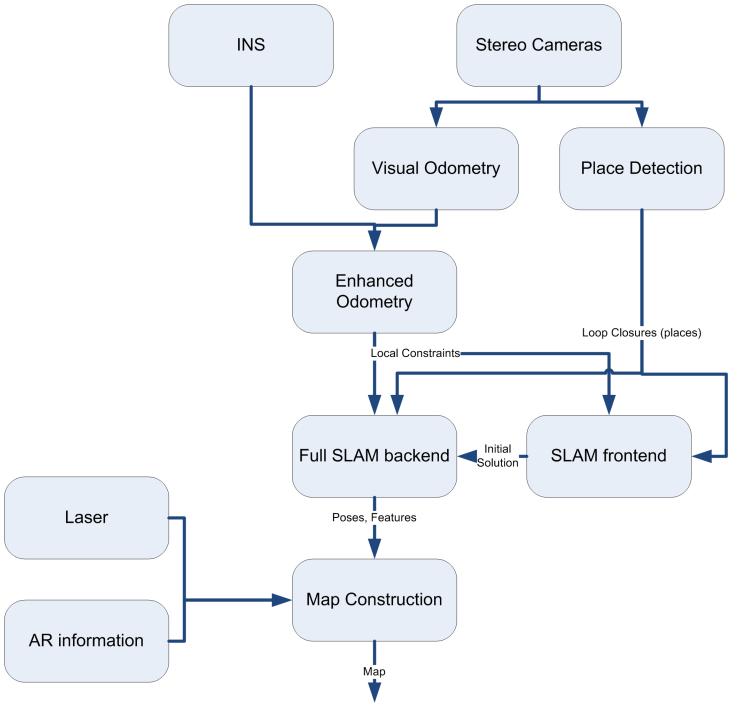

Figure 5: Design of the SLAM system

incremental Smoothing and Mapping is presented in [18, 17], where a full SLAM problem can be incrementally built and solved. These papers also relate the graphical probabilistic models behind SLAM, sparse factorizations of SLAM and sparse linear algebra.

Our approach follows the same scheme of a SLAM backend for solving the full SLAM problem as an optimization problem; and a frontend that provides the objective function to optimize. Figure 5 depicts the main building blocks. The next sections will describe the approaches followed for both cases.

3.1 SLAM frontend

3.1.1 Odometry

Odometry can be defined as the position estimation of a system based on proprioceptive sensors like wheel encoders. Due to the proprioceptive nature of the sensors used for odometry computation, the position estimation tends to diverge through time. Odometry can be also computed based on other interoceptive sensors such as inertial systems. Thus, the MTi-G sensor from Xsense provides us with the angular velocities and linear accelerations, and can provide an integrated INS solution when GPS is available. GPS has not been used because satellite visibility is shadowed most of the time by buildings and trees, and cannot be used indoors. Angular velocities and linear accelerations can be used in order to estimate translations and rotations, they can be used together with wheel speed and steering to improve the robot odometry.

Computer vision can also be used to estimate robot odometry based on frame-to-frame robot translation analysis. Stereo vision has been used for a long time in the robotics community in order to have accurate robot odometry [27]. The visual odometry process using stereo cameras is depicted in Fig. 6. The algorithm is based on the computation of feature matching

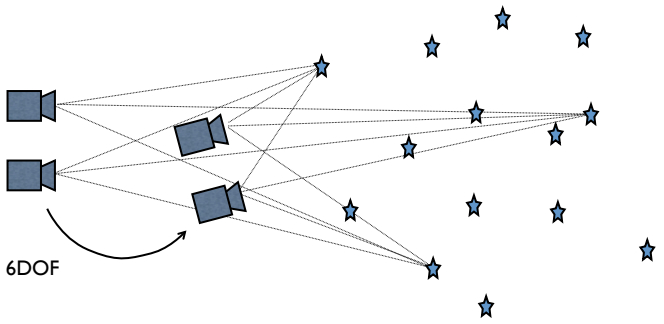

Figure 6: Scheme of the 6DoF visual odometry using stereo cameras. Stars denote 3D objects detected in both cameras.

between a first camera pair and a second pair which is placed at a different position; this matching establishes constraints between the projection and the real position of the features in 3D. If we consider the 3D points as static, the camera motion can be computed as the rotation/translation that allows the alignment of both point clouds (the point clouds in the first and de second camera).

Figure 7 shows a visual odometry experiment carried out at the Royal Alcázar. The setup consisted on a handheld stereo camera connected to a laptop, the camera was moved following the two trajectories (red an blue) depicted in the figure. The starting point of both trajectories is shown by a big star Fig. 7. It can be seen how the blue trajectory is coherent with the corridors and rooms of the map, with small errors in position at the end. On the other hand, the cumulative errors of visual odometry can be easily seen at the end of the red trajectory, where some walls are crossed. In both examples and in general, the more odometry is integrated the bigger are the localization errors.

In general, computer vision odometry tends to be more precise than wheel-based systems because it is not affected by physical factors such as wheel slipage or wheel radii inaccuracies, but at the same time is strongly conditioned by the existence of enough texture in the images captured by the cameras. Then, the best setup for a good odometry is the integration of several sources of information, choosing the one that adapts better in each moment. This way, wheeled odometry will be used when visual odometry fails due to lack of texture or in presence of complex visual scenarios and the MTi-G will be used to have an accurate and stable estimation of the robot roll and pitch. Finally, the robot yaw estimation will be based on the integration of the yaw rate gyroscope of the MTi-G.

3.1.2 Loop closures

As commented, odometry by itself accumulates drift with time. The crucial point of the SLAM phase is to be able to detect when the robot is at the same place, so that the errors of the drifting from the odometry can be reset. This problem is known as loop closure detection in the SLAM.

In the last years, appearance-based methods for place recognition have become one of the

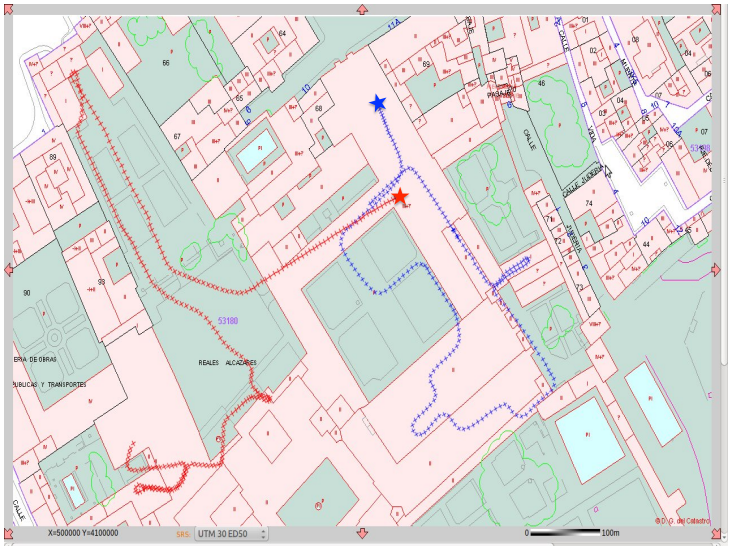

Figure 7: Tests of 6DoF visual odometry using a stereo camera in real experiments at the Royal Alcázar. The estimated position is projected in 2D in order to be compared with the map. Two different trajectories are shown, in red and blue.

main methods for loop closing. These methods are based on image retrieval approaches from the vision community, and, in particular, from *bag-of-words* (BoW) approaches [32]. In these approaches, images are represented by a set of *words* (the bag of words) from a *visual vocabulary*. This representation disregards the geometric information related to the images, but allows for a fast search of similar images in a database.

Typically, the methods begin by building the visual vocabulary to represent the images during an initial training stage. The main steps for building this visual vocabulary from a dataset of images are:

- Compute features for all the frames considered in the training set. Different features can be employed, such as SIFT [23], SURF [1], FAST [30] and others [12]. Typically, the images and features are selected trying to maintain a minimal spatial and temporal separation between images and features.
- The visual dictionary is built by clustering the features, using K-means or another clustering approach. The number of words in the dictionary, as well as the clustering parameters, are important for the performance of the detector (too many words may lead to low abstraction and, therefore, low loop closing capabilities, while too few may lead to high numbers of false positives).

After the vocabulary has been built, new images can be represented by their BoW, by extracting the same kind of features and determining the visual words present on the image. Searching for similar images is reduced to match their BoW representations, which can be carried out very fast [32].

These approaches have been used by the robotics community to solve the loop closing problem in SLAM. Cummins and Newman [8, 9] employ BoW representations of appearance, and combine them with a probabilistic modeling of the generative process of the BoW data from different locations. Other approaches, such as [2], enhance the system by dealing with outliers more robustly, employing more efficient matching and so on. Most of the methods employ a static vocabulary representation that is learnt offline. Recently, methods to build a visual vocabulary have been presented [26].

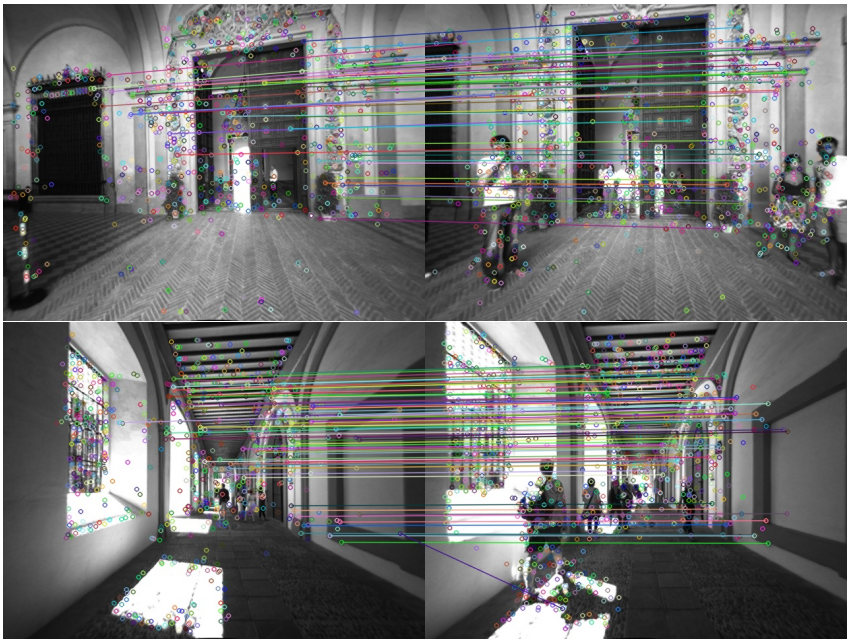

Figure 8: One example of BoW-based loop closure. Left and right images show candidates obtained by the BoW-based matching. Then, SURF features are extracted and matched (also shown). Then, the epipolar constraint is considered to discard outliers. If the number of inliers is over a threshold, the depth of the points is also used to recover the full 6D transformation between the cameras.

In our approach, we use a standard BoW-based method to look for loop closing in SLAM. We will apply the BoW representation as well to augment the metric maps with a view-based appearance map [20] in order to enhance the robustness of the localization system, as it will be described below.

The first attempt to solve the loop closing problem using BoW was to make use of the matching of the current image with the BoW database; however experimental results were far from satisfying our localization constraints in terms of accuracy. BoW may provide false positives and outliers, and they should be filtered out in order to have a reliable loop closing detection. A robust matcher able to detect such false positives has been designed and used in offline mapping and online localization.

The robust matcher takes as input two images matched with BoW and, first, checks if the images are really aligned (they really represent two views of the same scene) and, second, computes a set of SURF matches between both images. The robust matching is divided into the following basic steps:

1. SURF extraction. SURF key-points and descriptors are extracted from both images. These features will be used later to check the level of visual appearance between both images.
2. Visual appearance matching. This step computes the best feature-to-featue matching exclusively based on feature description. This means that no geometric constraints are checked, just feature similarity. This is an important issue because at this point we do not know if the images are two views of the same scene or not. This matching is computed in a “brute force” manner, comparing every feature in one image with all the features in the other image and selecting the most similar one (Euclidean distance between features). Those features with more than one match in the other image are forced to get the closest

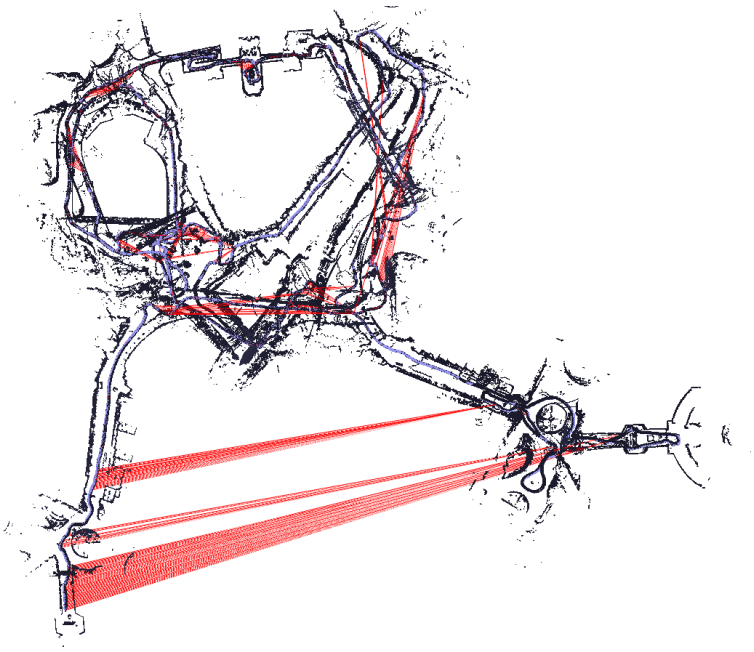

Figure 9: Loop closures obtained at the Lisbon Zoo. Here the typical drift associated to odometry can be also appreciated.

one.

The "brute force" matching is then computed again, but this time searching matches in the other image. At the end of both feature matching steps, it is tested that all features have the same matches in both directions; otherwise the link is erased.

3. Epipolar constraints. The next step will check the epipolar geometry between the pixel position of the previous computed matches. This is a pure geometric constraint and tries to compute the best Fundamental Matrix and the matches that fit it. If the images are different views of the same real object or scene, the relation between the projected 3D points will always be a Fundamental Matrix. Thus, for the pixels $\mathbf{p}_i = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T$ and $\mathbf{p}_j = \begin{bmatrix} u_j v_j 1 \end{bmatrix}^T$, the epipolar constraint is given by:

$$\mathbf{p}_i^T \mathbf{F} \mathbf{p}_j = 0 \tag{1}$$

Matrix $\mathbf{F}$ can be estimated with 8 or more correspondences by using the 8-point algorithm described in [15]. A robust estimation using RANSAC is performed.

4. Matching threshold. If the number of inliers after the estimation of the Fundamental Matrix is higher than a given threshold, then the match is accepted as a loop closure.

For any given pair of images matched by the BoW and validated with the robust matching, we extract and match SURF features (see Fig. 8). Then, the 6 degree-of-freedom transformation between the cameras at both points is computed leveraging the case that we have a stereo pair. By using stereo, the 3D position of the inliers is computed, and the 6D transformation is determined using the same techniques as in Section 3.1.1 for visual odometry. Figure 9 shows an example.

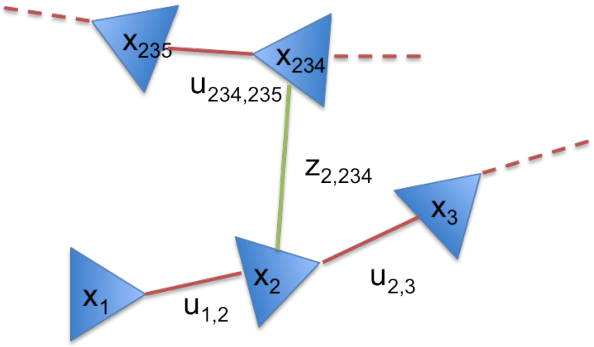

Figure 10: The pose SLAM problem can be represented as a graph of constraints between robot poses. The constraints are given by odometry readings $\mathbf{u}$ and loop closures $\mathbf{z}$

3.2 SLAM backend

Given all the information from the previous methods, the objective of the offline SLAM backend is to obtain the best estimation of the trajectory of the robot, as well as the map that will be used later for the localization of the robot.

In our case, we will consider the so called *pose-SLAM* problem, where only the trajectory of the robot is recovered by the SLAM backend. Thus, the sensorial information, odometry and loop closures, are treated as constraints on the state variables, in this case the robot poses (see Fig. 10). Once the robot trajectory is obtained, the map can be built by integrating the sensorial information from lasers and cameras into the global frame.

Each robot pose $\mathbf{x}_i$ will be represented in 6D by its 3D position and orientation in the space:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{r}_i \\ \mathring{\mathbf{q}}_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ q_{x,i} \\ q_{y,i} \\ q_{z,i} \\ q_{w,i} \end{bmatrix} \tag{2}$$

where the orientation of the robot is represented by a quaternion $\mathring{\mathbf{q}} = \begin{bmatrix} \mathbf{q} & q_w \end{bmatrix}^T$, with $\mathbf{q}$ the vectorial part and q_w the weight.

As commented above, the observations available are the odometry measures (represented by $\mathbf{u}$) and loop closure measurements (represented by $\mathbf{z}$). Every odometry measurement constrains the robot poses between consecutive time instants. This constraint can be represented by the vector function $\mathbf{e}_u(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i,i+1})$ so that $\mathbf{e}_u = \mathbf{0}$ when the values of $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ perfectly satisfy the constraint.

In the same sense, each loop closure measurement $\mathbf{z}_{i,j}$ constrains robot poses $\mathbf{x}_i$ and $\mathbf{x}_j$ represented by $\mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})$. Fig. 10 depicts this view.

The non-linear least squares problem is posed as determining the robot trajectory $\mathbf{x} = \begin{bmatrix} \mathbf{x}_0^T & \dots & \mathbf{x}_i^T & \dots \end{bmatrix}^T$ that minimizes the function:

$$F(\mathbf{x}) = \sum_{u} \mathbf{e}_u(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i,i+1})^T \Omega^u_{i,i+1} \mathbf{e}_u(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i,i+1}) + \sum_{z} \mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})^T \Omega^z_{i,j} \mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j}) \tag{3}$$

$$\mathbf{x}^* = \arg\min F(\mathbf{x}) \tag{4}$$

The quadratic terms are weighted by the matrices $\Omega^u_{i,i+1}$ and $\Omega^z_{i,j}$, which are the information matrices related to the odometric and loop closures measurements respectively, and that therefore weight more those more informative constraints (and dimensions within the constraints).

The solution to this optimization problem is equivalent to estimating the maximum likely robot trajectory in the case of Gaussian likelihood models [35, 17]:

$$p(\mathbf{z}_{i,j}|\mathbf{x}_i, \mathbf{x}_j) \propto \exp(\mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})^T \Omega^z_{i,j} \mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})) \tag{5}$$

Given an initial estimation of $\mathbf{x}$ and all the constraints given by the SLAM frontend, it is possible to obtain a numerical solution of (3) using standard methods like Gauss-Newton or Levenberg-Marquard. One important aspect of the solution is the Jacobian of the error terms with respect to the state $\mathbf{x}$, $\mathbf{J} = \frac{\delta \mathbf{e}}{\delta \mathbf{x}}$. It can easily be seen that this Jacobian presents a very sparse structure for each constraint [21]. For instance,

$$\mathbf{J}_{z,i,j} = \frac{\delta \mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})}{\delta \mathbf{x}} = \begin{bmatrix} \mathbf{0} & \dots & \frac{\delta \mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})}{\delta \mathbf{x}_i} & \mathbf{0} & \dots & \frac{\delta \mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j})}{\delta \mathbf{x}_j} & \mathbf{0} & \dots \end{bmatrix} \tag{6}$$

that is, the Jacobian of the error term associated to a loop closure only has nonzero terms for the robot poses affected.

This allows for efficient ways of solving the problem. In particular, we employ the library *g2o* [21]. This library allows us to express general non-liner minimization problems, as the offline SLAM. Furthermore, it permits us to perform the non-linear minimization on the actual manifold of the constraints when using over-parameretizations. And it can be extended to include new problems, constraints, parameterizations and so on.

3.2.1 Constraints

As described above, the odometric measurements consist of relative displacements between consecutive poses, represented by a translation and orientation $\mathbf{u}_{i,i+1} = \begin{bmatrix} \mathbf{r}_{i,i+1} & \mathring{\mathbf{q}}_{i,i+1} \end{bmatrix}^T$.

Given a robot pose $\mathbf{x}_i$ and the odometric measurement $\mathbf{u}_{i,i+1}$, it is possible to predict the next pose $\mathbf{x}_{i+1}$ as:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \oplus \mathbf{u}_{i,i+1} \tag{7}$$

where $\oplus$ is the usual motion composition operator, considering the quaternion formulation. Then, the odometric constraint can be defined as $\mathbf{e}_u(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i,i+1}) = \mathbf{x}_{i+1} \ominus \mathbf{x}_i \oplus \mathbf{u}_{i,i+1})$, where $\ominus$ is the inverse of the motion composition operator.

The close-loop measurements also provide the relative transformation between distant poses of the robot, in the form $\mathbf{z}_{i,j} = \begin{bmatrix} \mathbf{r}_{i,j} & \mathring{\mathbf{q}}_{i,j} \end{bmatrix}^T$. As above, the constraint related to loop closures is defined as:

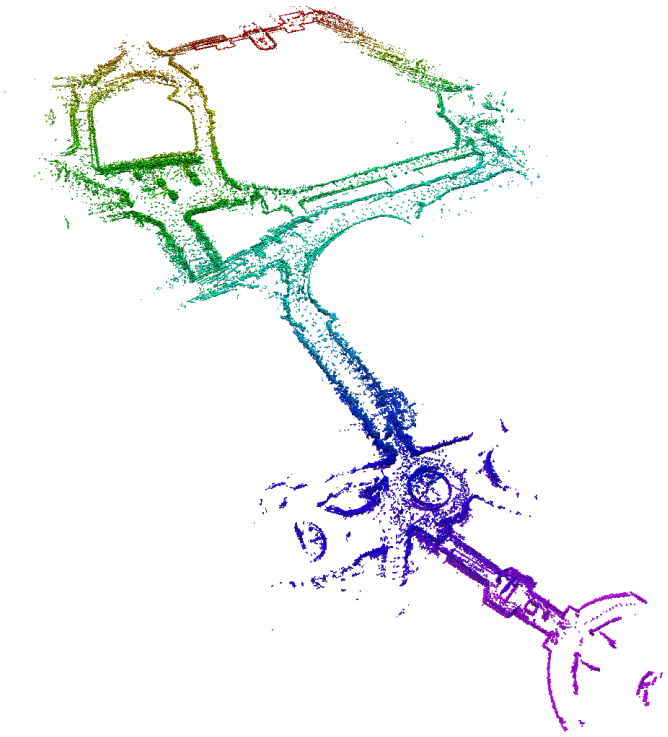

Figure 11: Point cloud obtained from the Lisbon Zoo. The height of the points is color-coded (reddish colors indicating higher ground).

$$\mathbf{e}_z(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{i,j}) = \mathbf{x}_j \ominus (\mathbf{x}_i \oplus \mathbf{z}_{i,j}) \tag{8}$$

3.3 Map building and refinement

After the execution of the previous minimization, an optimal robot trajectory is obtained. This robot trajectory is then used to build a map from the sensor data available. For instance, a 2D or 3D map can be constructed from the laser scans and the stereo vision system. Figure 11 shows the 3D map obtained from the Lisbon Zoo, while Fig. 12 shows the map overlaid on a CAD view of the Zoo.

However, while the robot trajectory is globally consistent, the simple projection of sensorial data (for instance, laser rangefinders, point clouds or stereo data) in the global frame will lead to maps with slight errors, such as fuzzy walls or double walls, as the information from those sensors was not directly considered in the minimization process (this information is not considered within the constraints of Section 3.2.1).

Several approaches have appeared in the last years to cope with this issue. For instance, in [31], the authors consider a model of the laser scans based on line segments (tangents on surfaces), and include the estimation of these line segments within the optimization process, so solving a complete SLAM problem. This approach is called SSA (Sparse Surface Adjustment), in the fashion of Sparse Bundle Adjustment (SBA). This model, however, is more suitable for indoor scenarios. Recently, in [16] the same ideas as in [31] were applied, adding a robust outlier rejection within the optimization process. Nevertheless, the approach is still the same and more suitable to indoor scenarios.

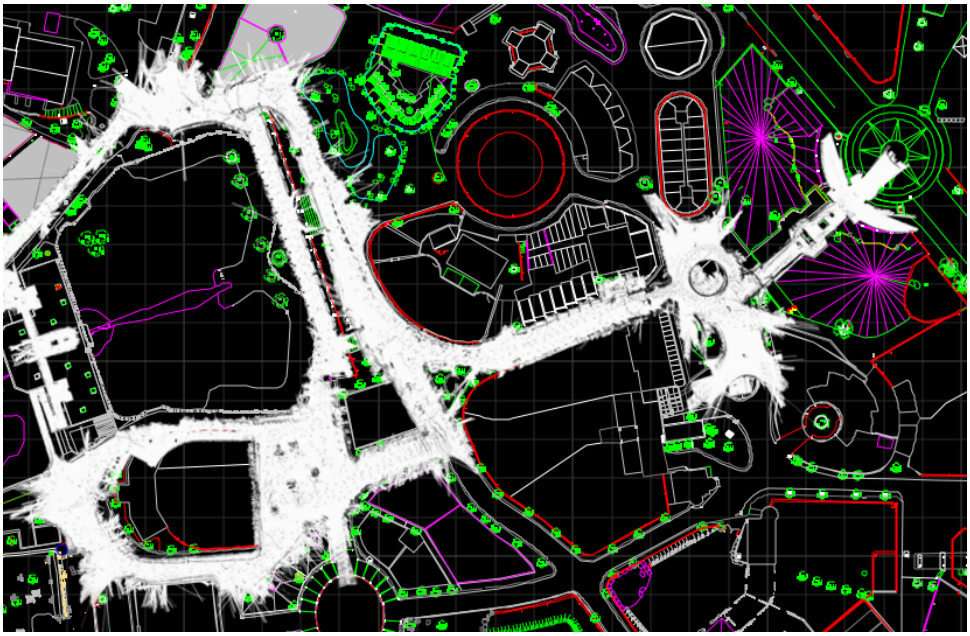

Figure 12: 2D occupancy map of the Lisbon Zoo overlaid on a CAD drawing of the Zoo.

As some of the scenarios in which the FROG robot has to be deployed, such as the Lisbon Zoo, present irregular structures, we have defined a different approach. In this approach, we will not include map features in the SLAM process, but will directly consider the sensorial information from the laser rangefinders to develop new constraints for the pose-SLAM problem that consider the errors on the map being built.

The following steps are carried out:

1. A new set of constraints is obtained by performing scan matching between pairs of laser scans or point clouds. As an initial good solution for the poses of the robot is already available from the initial solution, the scan matching process is performed not only between consecutive robot poses, but also between close poses in space but not in time.
2. The poses are refined by minimizing an error function for these constraints which depends on the quality of the alignment of scans *in the global map*, see below.
3. The initial seed for the minimization is provided by the previous solution

In order to do that, a new measurement function is employed which is directly based on Iterative Closest Point (ICP) techniques. Without loss of generality, we will describe the case for vanilla ICP [38].

Each robot pose $\mathbf{x}_i$ in the graph has associated the set $M_i = \{\mathbf{m}_i^0, \cdots, \mathbf{m}_i^{N(i)}\}$, the 3D points of the scan in the local frame of that pose. For each pair of robot poses that are constrained $\mathbf{x}_i$ and $\mathbf{x}_j$, the current alignment error for the scans is given by:

$$e(\mathbf{x}_i, \mathbf{x}_j, M_i, M_j) = \frac{1}{N} \sum_n (\mathbf{x}_i \oplus \mathbf{m}_i^n - \mathbf{x}_j \oplus \mathbf{m}_j^n)^T (\mathbf{x}_i \oplus \mathbf{m}_i^n - \mathbf{x}_j \oplus \mathbf{m}_j^n) \tag{9}$$

where $\oplus$ is the motion composition operator that transforms a 3D point $\mathbf{m}$ into the global frame. The sum is performed to all points in the two scans that have correct associations according to the scan matching algorithm employed.

Figure 13: Occupancy map of the Royal Alcázar overlaid on a planimetry of the complex.

By considering this error function within the global optimization framework, the poses will be refined to obtain those ones that globally align the point clouds. Within the optimization process, the scan matching is performed at each iteration just to determine the data association between points, so potential new matches are considered at each iteration due to the new relative poses.

3.3.1 Jacobian computation

As commented above, the library employed for minimization applies Gauss-Newton or Levenberg-Marquard to obtain the minimum of (3). These methods use the Jacobian of the error function. This Jacobian can be computed numerically, but as this has to be recomputed in each iteration, it can be sped up by determining an analytical Jacobian.

If we define each term in (9) as $\mathbf{e}^n = \mathbf{x}_i \oplus \mathbf{m}_i^n - \mathbf{x}_j \oplus \mathbf{m}_j^n$, and we stack all of them into a vector $\mathbf{e} = [\mathbf{e}^{0,T}, \cdots, \mathbf{e}^{n,T}]^T$, we get:

$$e(\mathbf{x}_i, \mathbf{x}_j, M_i, M_j) = \frac{1}{N}\mathbf{e}^T\mathbf{e} \tag{10}$$

The Jacobian (6) with respect to the robot poses $\mathbf{x}_i$ and $\mathbf{x}_j$ is:

$$\mathbf{J} = \frac{\partial e(\mathbf{x}_i, \mathbf{x}_j, M_i, M_j)}{\partial \mathbf{x}_i, \mathbf{x}_j} = 2(\bigtriangledown_{\mathbf{x}_i,\mathbf{x}_j}\mathbf{e})^T\mathbf{e} \tag{11}$$

As commented above, in our case each pose $\mathbf{x}_i$ is represented by a translation $\mathbf{r}_i$ and an orientation in 3D represented by a quaternion $\mathring{\mathbf{q}}_i = \begin{bmatrix}\mathbf{q}_i & q_i\end{bmatrix}^T$, with q_i the weight and $\mathbf{q}_i$ the vector part of the quaternion. The motion composition operator $\oplus$ transforms the point $\mathbf{m}_i^n$ into the global frame:

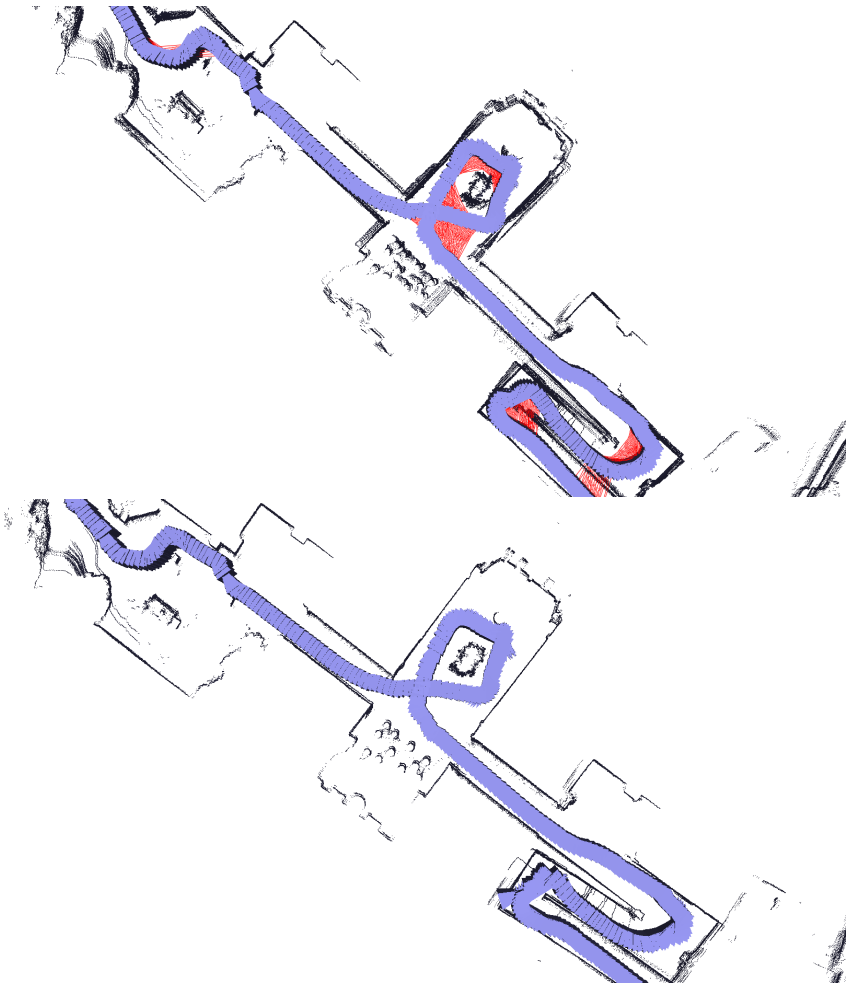

Figure 14: Top: map obtained by projecting the information with the optimized robot pose. The new constraints added are marked in red. Bottom: Map refinement by considering the laser information into the optimization process. In this building-like environment, it can be seen how fuzzy walls are reduced and the details are much more easily appreciated.

$$\mathbf{x}_i \oplus \mathbf{m}_i^n = \mathbf{r}_i + \mathcal{L}_q(\mathbf{m}_i^n) \tag{12}$$

where $\mathcal{L}_q(\mathbf{m}_i^n)$ is the rotation of a vector by using a quaternion representation:

$$\mathcal{L}_q(\mathbf{m}_i^n) = \mathrm{Im}\{\ \mathring{\bar{\mathbf{q}}}_i \cdot (0 + \mathbf{m}_i^n) \cdot \mathring{\mathbf{q}}_i\ \}. \tag{13}$$

where Im is an operator that extracts the vector part of a quaterion and $\cdot$ is in this case the quaternion product:

$$\mathring{\mathbf{p}} \cdot \mathring{\mathbf{q}} = (\ pq - \mathbf{p} \cdot \mathbf{q}\ ,\ p\mathbf{q} + q\mathbf{p} + \mathbf{p} \times \mathbf{q}\ ) \tag{14}$$

which leads to the expression:

$$\mathcal{L}_q(\mathbf{m}_i^n) = \mathbf{m}_i^n + 2q(\mathbf{q} \times \mathbf{m}_i^n) + 2\mathbf{q} \times (\mathbf{q} \times \mathbf{m}_i^n) \tag{15}$$

or expressing the cross product by a skew-symmetric matrix:

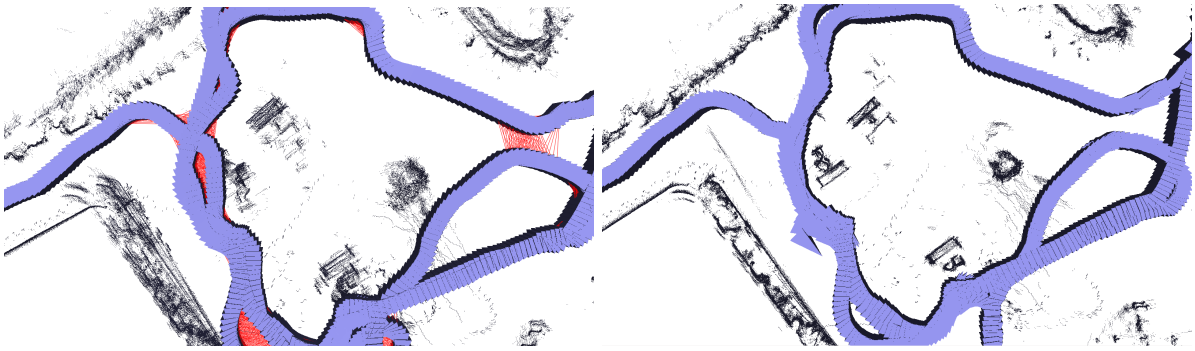

Figure 15: Left: map obtained by projecting the information with the optimized robot pose. Right: Map refinement by considering the laser information into the optimization process. It can be seen how the grass on the top left is correctly aligned. Also, some trees are much better resolved.

$$\mathcal{L}_q(\mathbf{m}_i^n) = \mathbf{m}_i^n + 2q(Skew(\mathbf{q})\mathbf{m}_i^n) + 2Skew(\mathbf{q})Skew(\mathbf{q})\mathbf{m}_i^n \tag{16}$$

Combining (16) into (12) and the definition of $\mathbf{e}^n$ it is possible to compute the Jacobian (6) in closed form. This Jacobian is evaluated at every iteration, considering potential new matches in the point clouds.

3.3.2 Results

As an example, Figs. 14 and 15 describe some results obtained in the Lisbon Zoo maps. Fig. 14 illustrates the case of a building-like structure, and it can be seen how some details are much better appreciated, and some double walls are eliminated.

More relevant is the case of Fig. 15, which has trees and grass and is less structured. In these environments, this approach is more adequate than the ones like [31, 16] which are more tailored to building-like structures.

3.4 Map management

As a result of all the previous procedures, a map is obtained in the form of a graph of refined robot poses. Each robot pose has the associated sensorial data from the lasers and the stereo image pair on board the FROG robot (see Fig. 4).

Using the refined poses, an occupancy grid map is obtained by integrating all the laser measurements into a global frame by using a maximum likelihood mapping algorithm. This map will be used for localization of the robot, see below. Also, the height coordinates of the robot are used to build a height map of the scenario considered.

Furthermore, an appearance map is created. The BoW representation of the images gathered during the mapping process is attached to each node of the pose graph (actually, a subsampled version of this pose graph, with a separation distance between poses of at least 2 meters). This appearance map can be queried at any moment with a given image in order to retrieve similar images and the poses at which those images were gathered. Figure 16 illustrates the idea.

During robot operation, it is possible to add further information to the map, by considering more images gathered at different time instants at the same poses (within a given threshold).

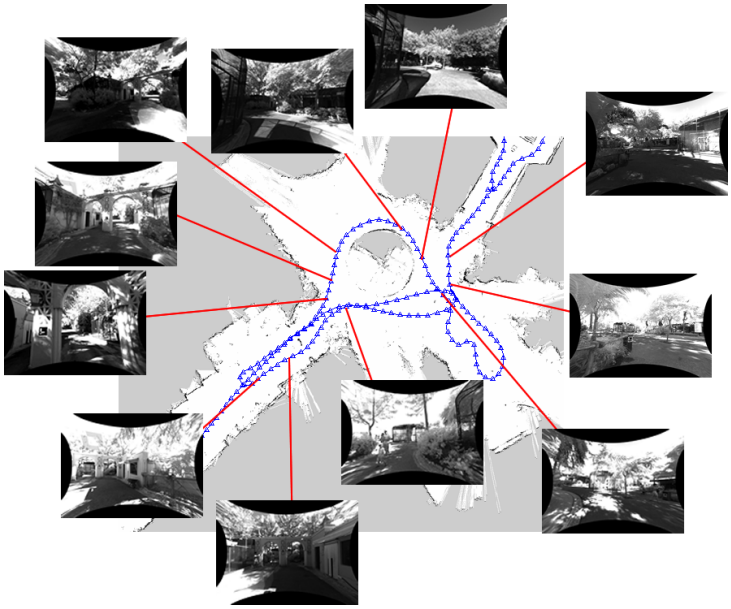

Figure 16: A localization map is built from the optimized pose graph. Furthermore, images are attached to each robot pose, represented by their BoW. This allows fast queries based on appearance.

4 Localization Module

4.1 Design

The localization module should provide the robot pose in 6D to the rest of the FROG robot modules. As indicated in Section 2, a map-based localization approach is employed, and therefore this pose is actually the pose with respect to the map.

This pose of the robot at time instant t, $\mathbf{x}_t$, should be estimated by combining the sensorial input received thus far $\mathbf{z}_{0:t}$ and the map built in the SLAM phase M. A measurement of the certainty of this pose should also be provided. Thus, the output of the module will be a probability distribution on the current pose of the robot $p(\mathbf{x}_t|\mathbf{z}_{0:t}, M)$.

Robot localization is a well-known problem, and a vast amount of work can be found in the literature related to the topic. The main issues related to localization in FROG are accuracy and robustness. The pose of the robot should be accurate enough to allow the robot to point in 3D space for AR interactions. At the same time, it should be robust, and minimize the time between failures.

We employ a Monte Carlo (or particle filter-based) localization approach [33, 37, 22, 28]. Particle filters are very flexible representing arbitrary probability distributions, and allow the fusion of information coming from different sensorial inputs.

Therefore, our distribution probability on the pose of the robot is represented by a set of particles $\langle \mathbf{x}_t^{[i]}, \omega^{[i]} \rangle$ so that

$$p(\mathbf{x}_t|\mathbf{z}_{0:t}, M) \approx \sum_i \omega_t^{[i]} \delta(\mathbf{x}_t^{[i]}) \tag{17}$$

where ω is the importance weight of each particle and $\delta()$ represents a Dirac's delta at the particle's pose.

Algorithm 1 Monte-Carlo Localization Algorithm

$\langle x_t^{[i]}, y_t^{[i]}, \theta_t^{[i]}, \bar{z}_t^{[i]}, \sigma_{z,t}^{[i]}, \gamma, \varphi, \omega^{[i]} \rangle_i^L$ Current state of the filter {Prediction stage}
if Odometric measurement $\mathbf{u}_t = \begin{bmatrix} v & \dot{\theta} & \gamma_{imu} & \varphi_{imu} \end{bmatrix}^T$ **then**
  $\varphi \leftarrow \varphi_{imu}$
  $\gamma \leftarrow \gamma_{imu}$
  for $i = 1$ to L **do**
    $\langle x_{t+1}^{[i]}, y_{t+1}^{[i]}, \theta_{t+1}^{[i]} \rangle \leftarrow$ sample_kinematic_model $(x_t^{[i]}, y_t^{[i]}, \theta_t^{[i]}, \mathbf{u}_t, \Delta t)$
    $\mathbf{v}_g^{[i]} = \mathbf{R}(\gamma, \varphi, \theta^{[i]}) \begin{bmatrix} v & 0 & 0 \end{bmatrix}^T$
    $\hat{z}_{t+1}^{[i]} = \bar{z}_t^{[i]} + \Delta t \mathbf{v}_{g,z}^{[i]}$
    $\hat{\sigma}_{z,t+1}^{[i]2} = \bar{\sigma}_{z,t}^{[i]2} + \sigma^2$
    $\bar{z}_{t+1}^{[i]} = \bar{z}_t^{[i]} - \frac{\hat{\sigma}_{z,t+1}^{[i]2}}{\hat{\sigma}_{z,t+1}^{[i]2} + \sigma_{z,M}^2} (z_t^{[i]} - h_M(x_t^{[i]}, y_t^{[i]}))$
    $\bar{\sigma}_{z,t+1}^{[i]2} = \frac{\hat{\sigma}_{z,t+1}^{[i]2} \sigma_{z,M}^2}{\hat{\sigma}_{z,t+1}^{[i]2} + \sigma_{z,M}^2}$
    $\omega_{t+1}^{[i]} = \omega_t^{(i)} \mathcal{N}(\hat{z}_{t+1}^{[i]}; h_M(x_t^{[i]}, y_t^{[i]}), \hat{\sigma}_{z,t+1}^{[i]2} + \sigma_{z,M}^2)$
  end for
end if
if Laser measurement $\mathbf{z}_t$ **then**
  for $i = 1$ to L **do**
    Compute likelihood $p(\mathbf{z}_t | x_{t+1}^{[i]}, y_{t+1}^{[i]}, \theta_{t+1}^{[i]}, M)$
    Update weight $\omega_{t+1}^{[i]} = p(\mathbf{z}_t | x_{t+1}^{[i]}, y_{t+1}^{[i]}, \theta_{t+1}^{[i]}, M) \omega_t^{(i)}$
  end for
end if
Normalize weights $\{\omega_t^{(i)}\}, i = 1, \ldots, L$
Resample if necessary

The pose of the robot considers 6 degrees of freedom (DOF), and it is represented as indicated in (2), although, for the ease of the explanation, we will represent here the orientation by the roll γ, pitch φ and yaw θ angles, and therefore $\mathbf{x}_t = \begin{bmatrix} x & y & z & \gamma & \varphi & \theta \end{bmatrix}^T$.

However, as we are considering a ground robot, the robot is bound to navigate on the 2D surface of the scenarios considered. Thus, the z coordinate is actually dependent on the x and y coordinates and the map M. Furthermore, the IMU onboard the robot provides a stable solution for the roll γ and pitch φ angles by using internal filters.

Then, in order to reduce the state space required to be covered with the particles, we consider a Rao-Blackwellized filter [11], in which the current 2D pose is tracked by using a particle filter, while the rest of variables are tracked by means of a Kalman filter. The probability distribution function can be factorized as:

$$p(\mathbf{x}_t | \mathbf{z}_{0:t}, M) = \underbrace{p(z, \gamma, \varphi | x, y, \theta, \mathbf{z}_{0:t}, M)}_{Kalman} \underbrace{p(x, y, \theta | \mathbf{z}_{0:t}, M)}_{particles} \tag{18}$$

Furthermore, as commented , the values γ and φ are directly provided by the IMU. Given the values of $x_t^{[i]}, y_t^{[i]}, \theta_t^{[i]}, \gamma, \varphi$, the value of z is estimated by using a Kalman filter. This allows the reduction of the dimension of the space the particles have to sample, which is important for a real-time implementation.

Therefore, our particles are represented by a set $\langle x_t^{[i]}, y_t^{[i]}, \theta_t^{[i]}, \bar{z}_t^{[i]}, \sigma_{z,t}^{[i]}, \gamma, \varphi, \omega^{[i]} \rangle$. These par-

ticles are updated by using the information coming from the odometry and mainly the laser rangefinders of the robot. A brief description of the filter can be found in Algorithm 1. Some of the details will be further described in the following.

4.1.1 Odometry model

The sensors on the FROG platform allow us to obtain the motion of the robot on the robot frame. Wheel encoders provide the velocity on the robot frame v. Furthermore, the XSense IMU provides an estimation of the angular velocity around the robot's Z axis, $\dot{\theta}$, as well as stabilized values for roll and pitch.

Given the linear and angular velocities, the particles are then propagated in line 6 by sampling from a typical probabilistic kinematic model [36, 7].

4.1.2 Height estimation

The height is updated using a Kalman filter. The velocity vector $\mathbf{v} = \begin{bmatrix} v & 0 & 0 \end{bmatrix}^T$ given by the odometry model on the robot frame is rotated into the global frame by using the current orientation estimation $\mathbf{v}_g^{[i]} = \mathbf{R}(\gamma, \varphi, \theta^{[i]})\mathbf{v}$. This is used to predict the next height by integrating this velocity (lines 8 and 9).

The height is then updated in 10 and 11 by considering the height map $h_M(x, y)$ built during the mapping phase (see Figure 11) by discretizing the XY plane and determining the height at every cell. In principle, this map may suffice to determine the height of the robot given its x and y coordinates. However, we integrate the estimation based on the odometry and that on the map in order to smooth the height estimation in case of coarse height maps.

In the height update equation:

$$\bar{z}_{t+1}^{[i]} = \bar{z}_t^{[i]} - \frac{\hat{\sigma}_{z,t+1}^{[i]2}}{\hat{\sigma}_{z,t+1}^{[i]2} + \sigma_{z,M}^2}(z_t^{[i]} - h_M(x_t^{[i]}, y_t^{[i]})) \tag{19}$$

$\sigma_{z,M}^2$ depends on the resolution of the map. If the resolution of the map is fine grained, $\sigma_{z,M}^2$ will be low and the height from the map will dominate. If the resolution is coarse, the fusion with the estimations based on the odometric information will smooth out jumps between cells in the map.

Finally, line 12 updates the weights of the particles by considering the expected height given the height map h_M and the pose of the robot. If the predicted particle gives a value of z that is not consistent with the height map, then the weight of the particle is reduced. This way, the information of the height map is also employed to discard hypotheses.

4.1.3 Map updating and some considerations

The weights of the particles are updated by computing the likelihood of the current sensorial input, considering the information stored in the map and assuming that the pose for a given particle is correct.

Figure 17: Left: an example of the ARUCO markers employed; right: one of the markers deployed at UPO's campus.

We will use the laser rangefinders' information as the main source for this updating. Therefore, from the map database, a 2D localization layer is extracted by using the information from the laser measurements and the robot poses stored in the map, as described in Section 3.4.

Therefore, the localization is actually performed on the 2D surface determined by this map. The x and y coordinates in this map do not exactly match the x and y coordinates in the global frame. If the terrain is mostly flat, or the slopes are small, the errors induced by this approximation are small (for instance, in the Royal Alcázar).

In any case, in the map building process, besides the height map associating a height to any x and y positions, a transformation is obtained relating the x and y coordinates of the localization layer to the global x and y coordinates.

4.1.4 Recovery from errors

Map-based localization, as commented above, is a very well known problem, and impressive results have been presented in the past years in applications like the DARPA Grand Challenges. However, long-term operation of robots in urban-like environments is still a challenge [37]. Even a very impressive demonstration like in the EUROPA FP7 project the robot got lost after 78 minutes traveling when the map features were not actually perceived by the robot [3].

The main strategy we will pursue to augment the robustness of the system is to consider a complete different modality within the localization as a second source of information. As described in Section 3.4, our map is augmented with images, their corresponding bag-of-words representations and features.

Two approaches can be considered in this sense:

- In the sampling of new particles, particles are generated from a distribution derived from the BoW measurements
- Use the BoW-based measurements as a separate fail-safe to detect the divergence of the filter and restart it.

Below we will see an initial analysis of these ideas.

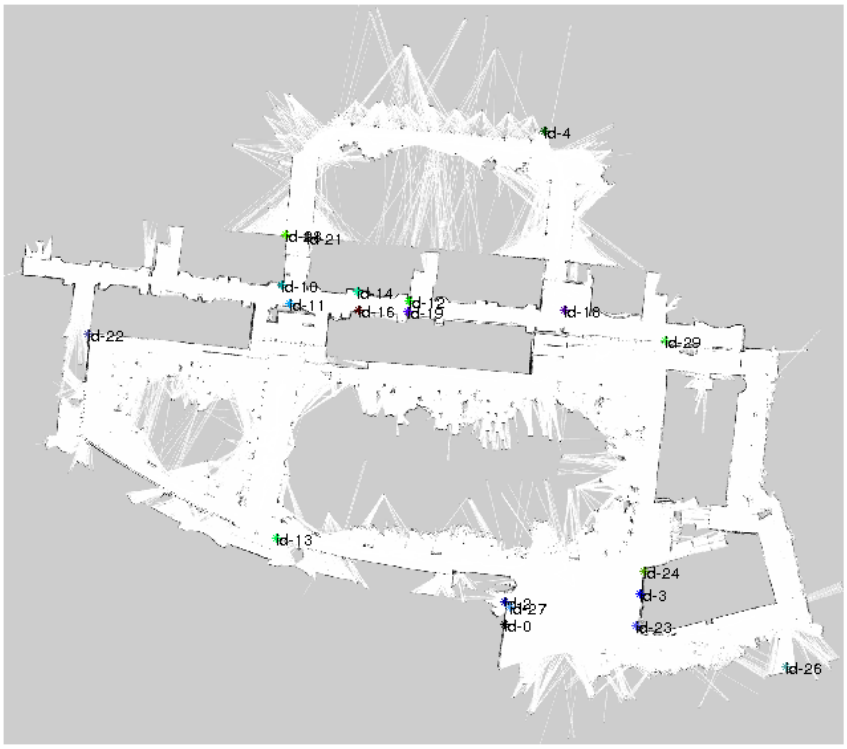

Figure 18: Ground truth positions of the markers at UPO's campus.

5 Benchmarking

The main features of the localization system relevant to the FROG scenarios are accuracy and robustness:

- By accuracy we mean the errors on the pose of the robot on the map. This is relevant for navigation, but particularly relevant for the AR capabilities.
- The robustness is related to the long-term autonomy of the robot, and is given by the time between interventions due to the robot being lost

We have designed different experiments to assess the current capabilities of the system in these two dimensions.

5.1 Accuracy

5.1.1 Experimental setup

We had to check the pointing accuracy of the localization system, as well as its repeatability. In the scenarios considered we do not have full GPS coverage for ground truth. Therefore, in order to benchmark these capabilities, we have designed an experimental setup, based on the deployment of a set of AR markers from the ARUCO library [25] (see. Fig. 17) on the scenarios. These markers are very easily (and robustly) detected on the images, and given their known geometry it is possible to determine their relative position with respect to the camera for calibrated images.

The setup of the experiments is the following:

Table 1: Global error for the different markers and experiments

	Small markers		Big markers	
Experiment	Position Error	Angle error	Position Error	Angle Error
UPO	0.2330	0.0486	0.2590	0.0372
Lisbon	0.4055	0.1235	0.8193	0.1060

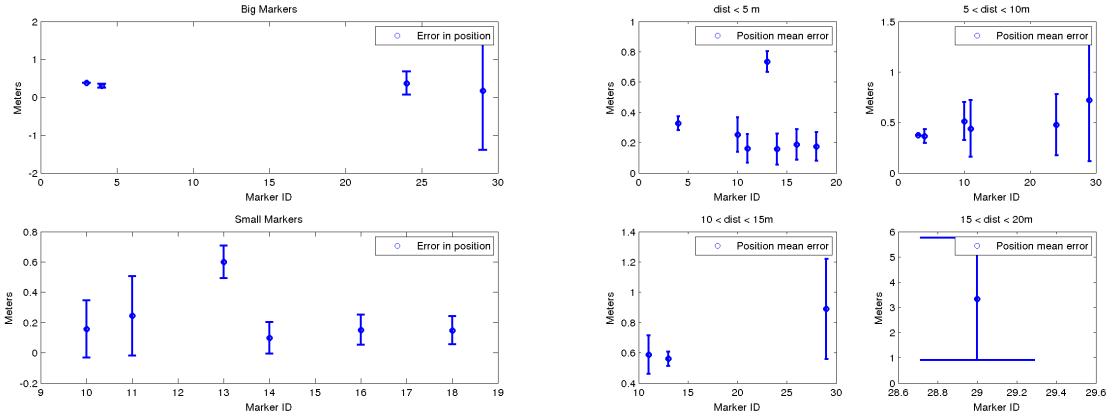

Figure 19: Left: Mean 3D position error and standard deviation per marker (UPO) with respect to the marker size. Right: the same data is organized according to the distance to the marker.

- A set of markers was deployed in the scenarios. They were deployed trying to cover potential AR situations at the Points of Interest[1], and in different configurations. We have deployed markers at the Lisbon Zoo in different experimental sessions. As it was not possible to deploy them in the Royal Alcázar, we also deployed them at UPO's campus, trying to imitate the configuration of the Points of Interest (POIs) in the Royal Alcázar scenario of the project in terms of distance from the robot to these POIs (see Figs. 18 and 26).
- Markers of two different sizes were deployed (the large markers are squares of 0.9 meters on each side, while the smaller ones have 0.25 meters on each side).
- The best position and orientation of all markers in the global frame was computed in the offline SLAM phase described above, by including them into the map. These poses are considered the ground truth.

Given this setup, we analyzed the accuracy of the localization algorithm by running it on new sets of data and obtaining the pose on the global frame of each marker that was seen on the images. Once these poses were computed, we estimated:

- The mean error and variance of the error on the 3D position of the marker with respect to the ground truth.
- The error on the angle estimation on the position of the marker with respect to the ground truth.
- The reprojection error on the image plane for every marker

The markers were not used for anything else in the localization process.

Table 1 shows the global results for all markers by running the localization algorithm using 2000

[1]These Points of Interests are points in which the robot guide will inform the users, or some interaction will take place

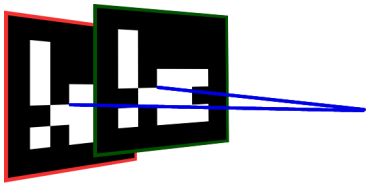

Figure 20: Definition of the angular error: it is defined as the difference in angle between the estimated pose of the marker and the ground truth one in the robot frame.

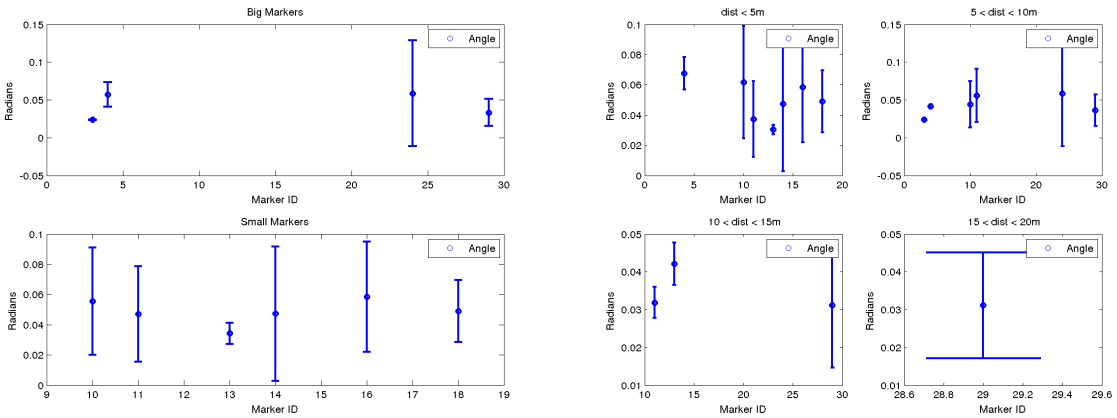

Figure 21: Left: Mean angular Error and standard deviation per marker (UPO) with respect to the marker size. Right: the same data is organized according to the distance to the marker

particles. It can be seen that, in general, the errors are lower in environments containing buildings and more structure, like UPO's campus. In that case, the errors in angle and positioning of the markers are below the figures considered in the initial requirements.

In the Zoo, the errors are larger, of around 5 degrees on the orientation (the positioning errors are the ones on the markers position, not the robot). In the next subsections, these experiments will be analyzed with some more detail.

5.1.2 Experiments at UPO's premises

Figure 19 shows the positioning error of the markers with respect to their size and the distances at which they were observed. As expected, the larger the distance, the higher the error. But in most cases the mean error is below 0.5m.

Figure 21 shows the angular error in pointing towards the markers (see Fig. 20) with respect to their size and the distances at which they were observed. The pointing error is mostly below the 3-degree (0.05 radians) limit.

Another interesting measurement is the error committed in pixels on the reprojection of the markers on the image plane. This is measured by projecting on the image plane a virtual recreation of the marker given its ground truth pose and the estimated pose of the robot by the localization algorithm. This projection is compared with the actual position of the marker on the image plane. Figures 22, 23 and 24 show some qualitative results.

The errors in pixels of these projections can be seen in Fig. 25. The images are of 1400x1024 pixels.

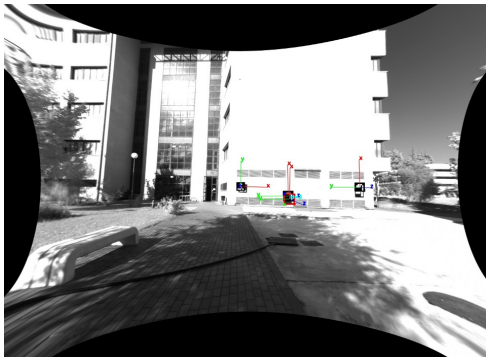

Figure 22: Reprojection results: three large markers seen from more that 20 meters. The overlaid axes correspond to the ground truth reprojection and the estimated pose of the detected markers.

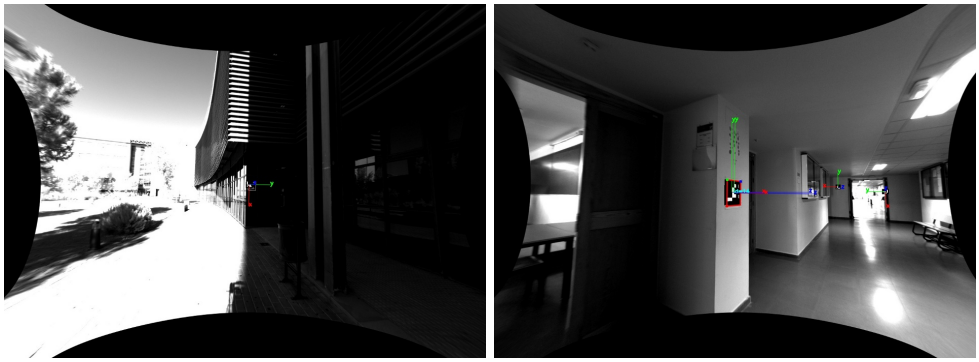

Figure 23: Reprojection results: left, a small marker seen from around 4 meters; right, small marker seen from 2 and a half meters.

5.1.3 Experiments at Lisbon Zoo

The same experiments were carried out at the Lisbon Zoo. The position of the markers is shown in Fig. 26. The positioning errors in the global frame, angular errors and reprojection errors can be seen in Fig. 33.

The main conclusion is that the mean errors were larger (twice in mean) than in the UPO experiments. This is due to the more challenging characteristics of the Zoo, with fewer features for localization. The main component of the error was the angular one (remember that the positioning error shown above is the error on the position of the markers, not the robot position error). Also, in many cases the angular error was below 3 degrees, but in some particular cases was over the value. Therefore, a way to reduce these positioning errors will be described in the next section.

5.1.4 Pose refinement in POIs

As seen in the previous section, even when the position of the robot can be estimated within an envelope of half a meter most of the time, there are particular points, and especially at scenarios like the Zoo, where this accuracy may not be sufficient when the system has to deal with AR contents. Information projection and augmented reality strongly depend on a good localization of the robot with respect to the area in which the information will be projected (or pointed); thus, even if the robot is well localized with respect to the map or the environment,

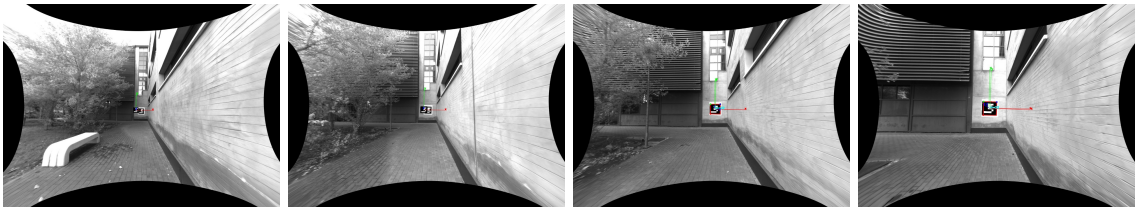

Figure 24: Reprojection results: the sequence shows one of the large markers seen from different distances.

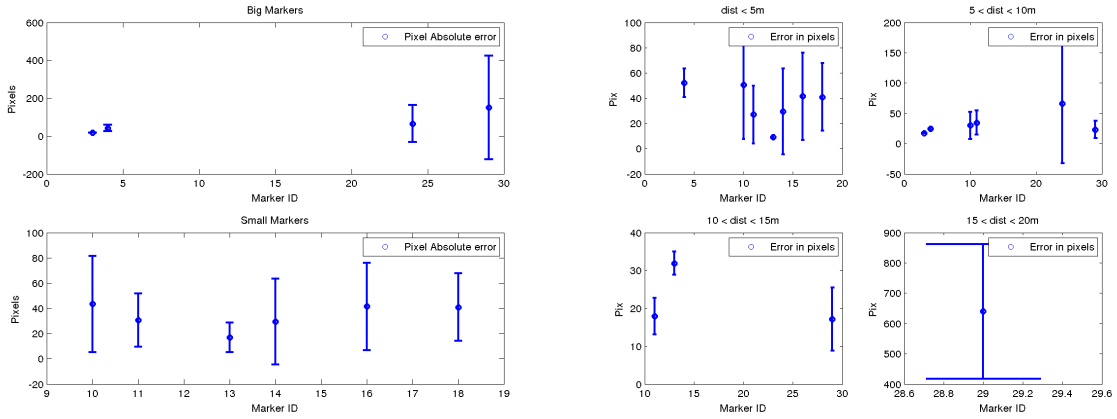

Figure 25: Left: Mean reprojection error in pixels and standard deviation per marker (UPO) with respect to the marker size. Right: the same data is organized according to the distance to the marker.

small map inaccuracies could lead to significant errors in the image projection.

Global maps, as the ones used in this project, must reach a trade-off between global and local alignments. This trade-off normally leads to small local alignment inaccuracies that do not affect global localization, but may affect position estimation with respect particular areas.

In order to minimize this effect as much as possible, we will consider the information available on the augmented view-based map (Section 3.4) for robot pose refinement when it is at a Point of Interest. POIs can be seen as especial waypoints commanded to the robot. The main difference with respect to regular waypoints is that POIs will keep invariant along time because the augmented reality contents must be projected from such positions.

The key idea behind pose refinement in POIs is to introduce into the robot map some visual information directly related to the POI; and to use this information every time the robot reaches the POI position in order to estimate and improve its localization.

The objective is to make this refinement as light as possible from a computational point of view. In this sense, using the bag of words (BoW) seems to be the best option; however, BoW uses a high level representation of the features into the image (clusters) in order to generalize the visual information and make possible visual appearance comparison at low computation cost. However, in this case we are interested in refining the relative pose between the views.

The pose refinement in the POIs algorithm is divided into two parts: the mapping and the pose refinement stages:

- The mapping process consists of including into the robot map (see Section 3.4) images at the poses in which the robot will be commanded when the system requires to visit a particular POI. We will call these images key-frames. Besides their appearance, a set

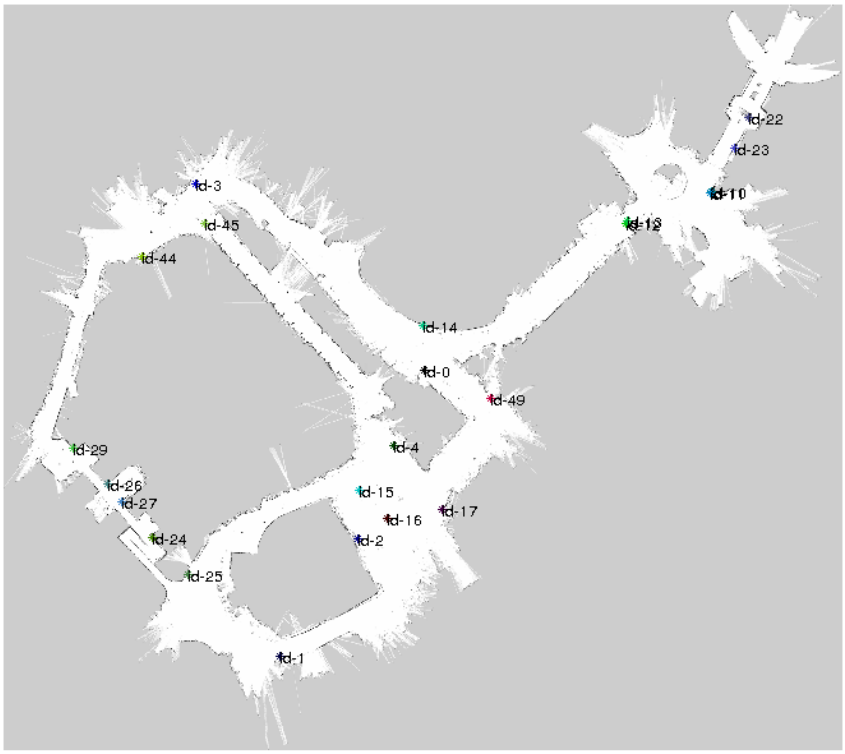

Figure 26: Ground truth positions of the markers at the Lisbon Zoo.

Figure 27: Three different key-frames captured during mapping process in Lisbon. The visual markers in the scene will be used as ground-truth to validate the pose refinement.

of SURF features is extracted from these images; the 3D position of the features is also computed based on the stereo information. These 3D SURF features are then stored into the map associated to the robot pose at the POI. Figure 27 shows some key-frames selected during the mapping process at Lisbon.

- The refinement stage is based on the 3D SURF dataset computed during mapping process. The algorithm will make use of Perspective-n-Point camera pose determination approaches, popularly called PnP methods. The aim of these methods is to determine the relative position between camera and scene from N known correspondences of space control points and their projection into image points. The PnP problem finds its main applications in computer vision and photogrammetry, it is a well known technique for camera pose registration under the assumption of known 3D points.

Solving the matching between current robot image and the key-frame will be based on the robust matching described in Section 3.1.2 for rejecting outliers. Basically, once we know the robot has reached the POI position, an image will be captured and compared with the key-

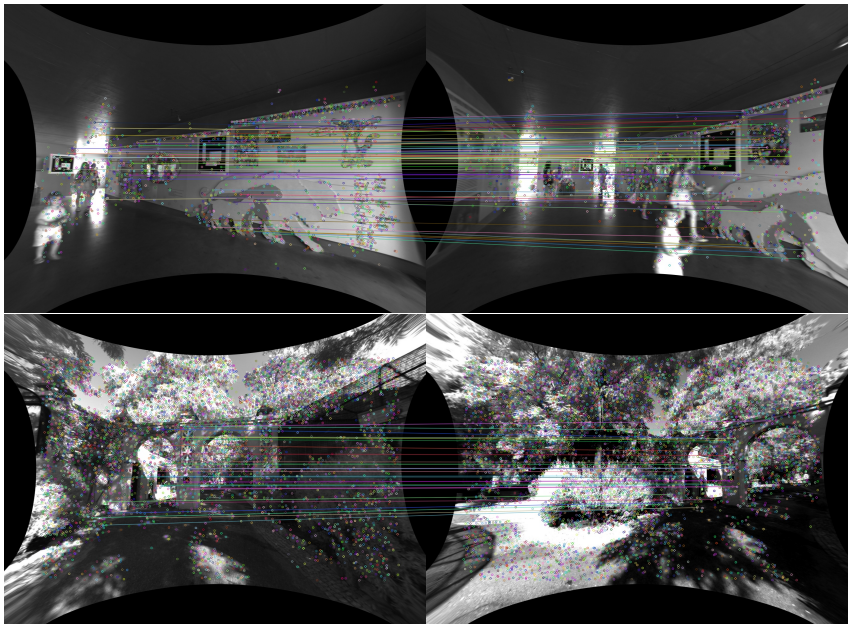

Figure 28: Two examples of feature matching between key-frame and robot image indoors (top) and outdoors (bottom). Colored circles represent detected SURF features and colored straight lines link good matches between images.

frame. For this purpose, SURF features will be extracted from the current image and matched against the SURF features stored in the map for this particular POI. The epipolar constraints together with the double matching check carried out by the robust method presented in this document assure an accurate matching. Figure 28 shows some results of the matching process applied over two different key-frames indoors and outdoors, it can be seen how there are significant orientation changes (about 0.2 rad), occlusions and distance to the key-frame position.

Several experiments with the FROG robot at Lisbon Zoo have been carried out in order to test this refinement. The key-frames for seven potential POIs at the Zoo were recorded during the mapping process and the pose refinement was tested against other datasets also in the Zoo. All these POIs present a visual marker as the ones employed above. These markers are used to compute the relative transformation between the current image and the key-frame, which is then used as ground truth, but they are not needed at all for the algorithm. This is a robust ground truth with small errors (some centimeters) when the camera is close to the marker and acceptable errors (about 10 or 15 cm) when the camera is far from the markers.

The estimated transform was computed based on the method presented in this section and compared against the ground truth. Figure 29 shows the errors committed by our estimation compared with the actual distance from the robot to the desired POI. The algorithm is able to refine the pose of the robot with less than 20 cm of error when the robot is located at less than 1 m. from the intended POI. Considering that the localization algorithm presented in previous sections provides errors in the order of 0.5 m, Fig. 29 shows how the expected errors will be in the order of 0.1 m.

It is worth mentioning that the maximum relative orientation was about 0.2 rad; above this value the robust matcher was not able to deal properly with the features. This drawback can be easily solved by increasing the number of key-frames for a single POI, considering images from very different orientation. This way, the pose refinement algorithm will be run against all the key-frames associated to the POI. This method will increase the computational requirements but

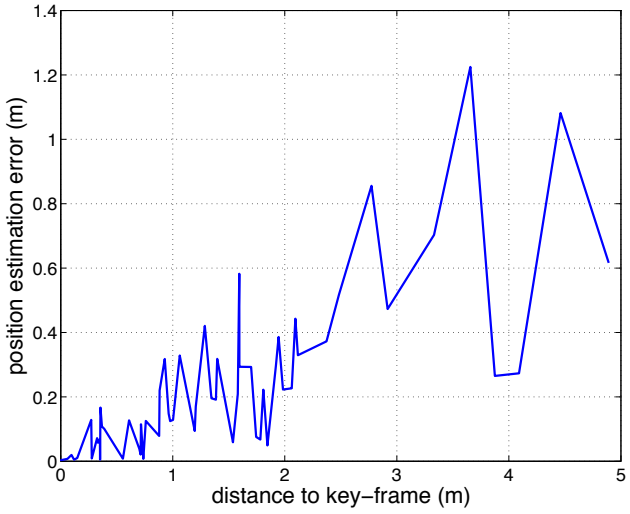

Figure 29: Pose refinement error based on the ground truth provided by the visual markers. The error is plotted with respect the distance to the desired POI, it can be seen how the estimated pose refinement worsens with the distance to the POI and how it is below 0.2m when the robot is at less than 1m of the true POI position.

will also increase the reliability of the method

5.2 Robustness

The second important issue regarding the localization system is robustness: the system should maximize the time between failures, minimizing potential human intervention.

This section will detail some experiments carried out in order to assess the localization approach presented in this document. As previously introduced, the localization module is based on Monte-Carlo Localization (MCL) for normal operation of the robot and a combination of BoW plus robust matching to detect and correct the position of the robot when it gets lost (see Section 4).

Figure 30 presents some localization results using the localization filter purely. The key idea is to analyze how long the localization system can run without getting lost and at the same time if there are areas in the map where the robot gets lost easily. The figure shows the estimated localization (blue solid line) from the MCL in both scenarios UPO and Lisbon Zoo. It can be seen how the position estimation is all time coherent with the map. The robot did not got lost in neither of the experiments, and always was well localized, even in areas with people surrounding the robot.

The good behavior of the localization filter (shown in Fig. 30) is not surprising because it significantly depends on the quality of the map, and the map building approach presented in this document is designed to provide hi-fidelity maps. Further experiments have been carried out in the Royal Alcázar with similar results. Still, longer experiments are required to assess the robustness in the long term. Most of previous experiments have been limited in time. In the next period of the project these experiments will be carried out.

One important issue for enhancing the robustness of localization is the possibility of recovery. The next experiment shows the result of localizing the robot when the initial position is completely unknown. Figure 31 shows the localization when the initial position of the robot

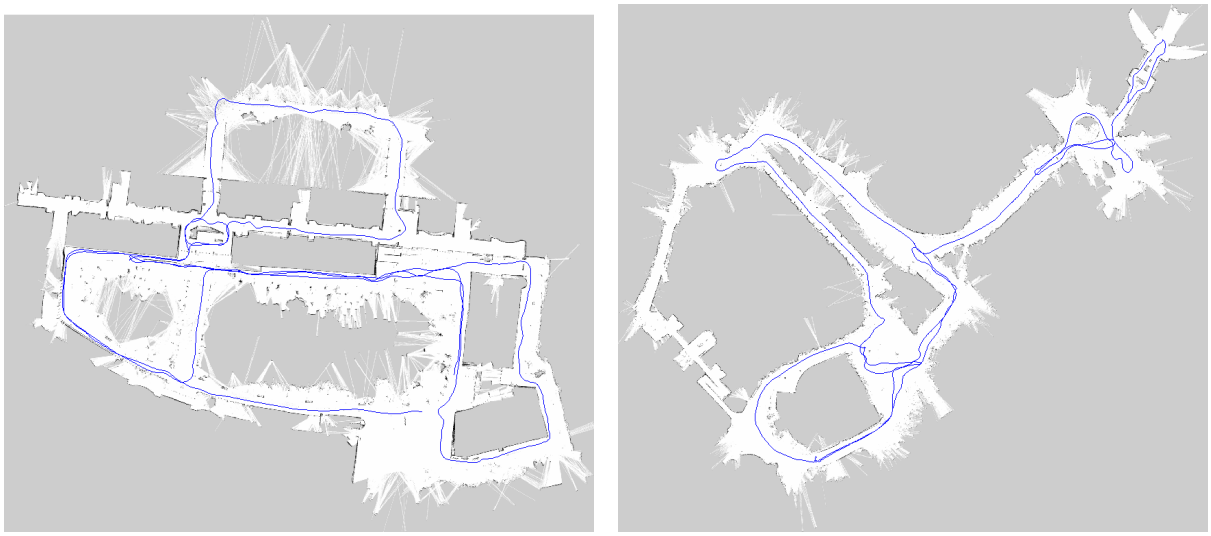

Figure 30: Localization of FROG robot in Lisbon Zoo and UPO. Blue solid line: robot trajectory. Left: Localization at UPO. Right: Localization at Lisbon Zoo. The length of the paths are of the order of 1 km.

is randomly set. In this case, the BoW plus robust matching algorithms are the key in order to properly re-localize the robot at the correct position. It can be seen how the robot tries to localize itself (red solid line) with bad results because the initial position is far from the true one. However, at a certain point the close loop detector is able to reliably identify a previously visited area and re-localize the robot position by injecting particles into the filter. After some filter updates, the estimated robot position converges with the true position (black solid line).

Similarly, we tested the capability of the loop closing detection approach to relocalize the robot after a localization failure. In order to illustrate that, some experiments in which the robot localization was intentionally set to a wrong position are shown in Fig. 32. We can see how the loop closing detection quickly detects the localization error by comparing the images gathered by the robot with the image database attached to the map. Once the loop closing algorithm detects a loop it re-initializes the localization filter to the approximate correct position and the filter is then able to refine the localization based on the laser scans and the map. The figure shows how the system is disturbed six times (red solid lines indicate where the robot pose was artificially moved to another position) and always goes back to the correct position.

6 Conclusions

The deliverable describes the main design decisions for the FROG robot localization module.

An offline pose-SLAM approach is employed for map building. The robot trajectory is recovered by considering all the constraints coming from odometry and loop closures obtained by appearance matching. A second error minimization approach is proposed to refine the maps obtained. The map built is augmented with appearance information.

The main issues of the localization module are described. Furthermore, an analysis of the accuracy and repeatability of the module has been carried out. This analysis indicates that the system provides the required accuracy for environments like the Royal Alcázar. The accuracy is lower in the Lisbon Zoo as it is a less structured scenario. A pose-refinement method has been developed to enhance the relative positioning capabilities for the AR applications.

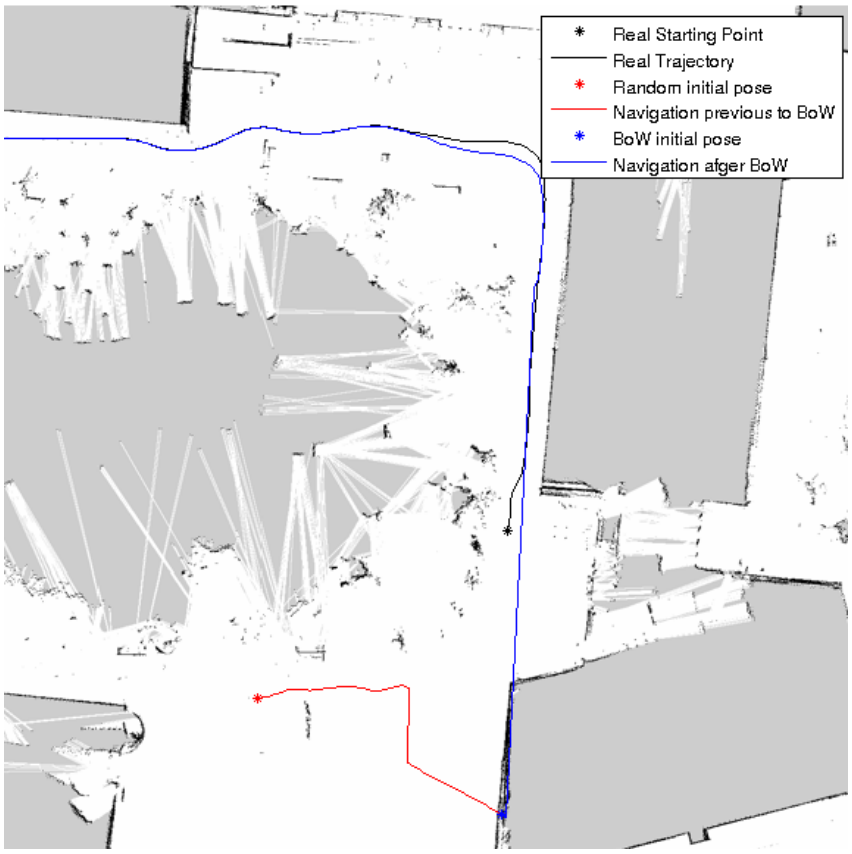

Figure 31: Robot localization at UPO starting at an unknown position (kidnaping problem). Blue solid line: Estimated robot trajectory using MCL+BoW. Black solid line: True position of the robot. Red solid line: MCL localization when robot is started with random position

Initial experiments have been carried out to illustrate the robustness of the localization module, as well as some recovery strategies. However, longer experimental sessions are required to assess the robustness of the system in the long term. In the next period, further experiments explicitly devoted to this will be carried out at the Lisbon Zoo and the Royal Alcázar.

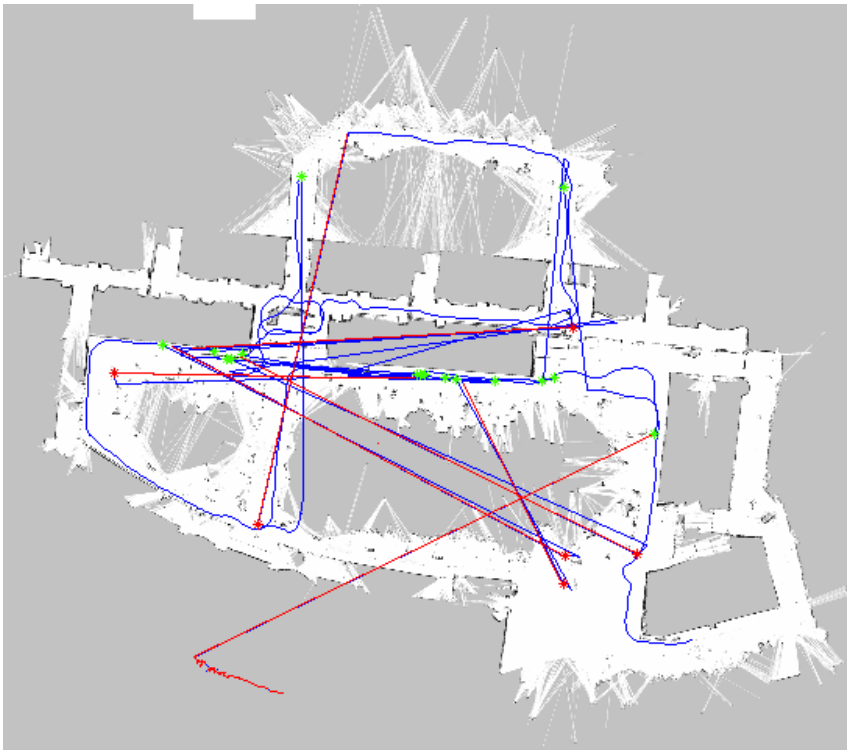

Figure 32: Robot pose recovery after manual error position based on loop closing detection. The initial position of the robot is randomly initialized. Blue solid line: estimated robot trajectory. Red asterisks: Wrong position manually set to the filter in order to force bad localization. Green asterisk: Positions in which the BoW has detected a good match thereby improving the robot localization.

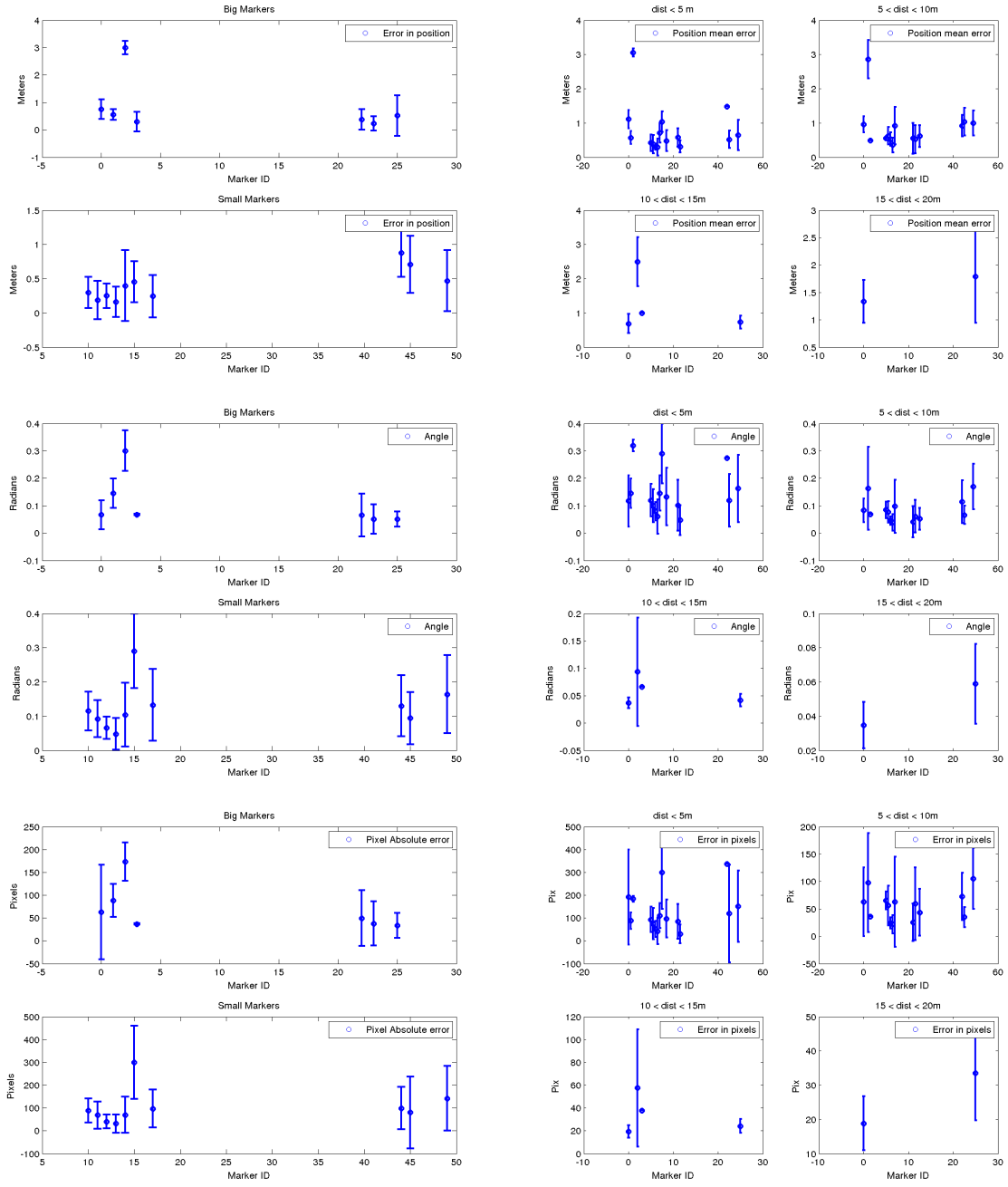

Figure 33: Top: Mean 3D position error and standard deviation per marker (Lisbon Zoo) with respect to the marker size. Middle: Mean angular error and standard deviation per marker (Lisbon Zoo) with respect to the marker size. Bottom: Mean reprojection error in pixels and standard deviation per marker (Lisbon Zoo) with respect to the marker size.

Bibliography

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[2] Tom Botterill, Steven Mills, and Richard Green. Bag-of-words-driven, single-camera simultaneous localization and mapping. *Journal of Field Robotics*, 28(2):204–226, 2011.

[3] Wolfram Burgard. Autonomous Pedestrian-Like Navigation in City Centers. In *ICRA, Workshop on Long Term Autonomy*, 2013.

[4] FROG Consortium. Deliverable D1.1: Functional Requirements, Interaction and Constraints. `http://www.frogrobot.eu/wp-content/uploads/2013/05/FROG-ROBOT-D1.11.pdf`.

[5] FROG Consortium. Deliverable D1.3: Customized Robot Plaftorm. `http://www.frogrobot.eu/wp-content/uploads/2013/04/D1.3_Final.pdf`.

[6] FROG Consortium. Deliverable D2.4: AR Robot Application component. `http://www.frogrobot.eu/wp-content/uploads/2013/04/D2-4-FINAL.pdf`.

[7] P. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Tracts in Advanced Robotics. Springer, 2011.

[8] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[9] Mark Cummins and Paul Newman. Highly Scalable Appearance-Only SLAM - FAB-MAP 2.0. In *Proc. Robotics: Science and Systems, RSS*, 2009.

[10] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research, IJRR*, 25(12):1181–1204, Dec 2006.

[11] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[12] Dorian Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[13] Giorgio Grisetti, Rainer Kuemmerle, Cyrill Stachniss, Udo Frese, and Christoph Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Proc. International Conference on Robotics and Automation, ICRA*, 2010.

[14] J. Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, page 318–325, Monterey, California, 1999.

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[16] Marian Himstedt, Sabrina Keil, Sven Hellbach, and Hans-Joachim B́ohme. A Robust Graph-based Framework for Building Precise Maps from Laser Range Scans. In *ICRA, Workshop on Robust and Multimodal Inference in Factor Graphs*, 2013.

[17] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research, IJRR*, 31:217–236, Feb 2012.

[18] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics, TRO*, 24(6):1365–1378, Dec 2008.

[19] Kurt Konolige. Large-scale map-making. In *Proceedings of the National Conference on AI (AAAI)*, 2004.

[20] Kurt Konolige, James Bowman, J.D. Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit, and Pascal Fua. View-based maps. *The International Journal of Robotics Research*, 29(8):941–957, 2010.

[21] R. Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proc. International Conference on Robotics and Automation, ICRA*, pages 3607–3613. IEEE, 2011.

[22] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[23] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[24] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[25] Rafael Munoz-Salinas. ARUCO: a minimal library for augmented reality applications based on OpenCV. `http://www.uco.es/investiga/grupos/ava/node/26`.

[26] T. Nicosevici and R. Garcia. Automatic visual bag-of-words for online robot navigation and mapping. *Robotics, IEEE Transactions on*, 28(4):886–898, 2012.

[27] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:2006, 2006.

[28] Stephen Nuske, Jonathan Roberts, and Gordon Wyeth. Robust outdoor visual localization using a three-dimensional-edge map. *Journal of Field Robotics*, 26(9):728–756, 2009.

[29] Edwin Olson, John Leonard, and Seth Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. International Conference on Robotics and Automation, ICRA*, pages 2262–2269, 2006.

[30] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[31] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses. In *Proc. International Conference on Robotics and Automation, ICRA*, 2011.

[32] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[33] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[34] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 2004.

[35] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.

[36] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[37] E Trulls, A Corominas Murtra, J Pérez-Ibarz, G Ferrer, D Vasquez, Josep M Mirats-Tur, Centro Cetaqua, and A Sanfeliu. Autonomous navigation for mobile service robots in urban pedestrian environments. *Journal of Field Robotics*, 2011.

[38] Z. Zhang. Iterative Point Matching for Registration of Free-Form Curves and Surfaces. *International Journal of Computer Vision*, 13:119–152, 1994.