Experimental Assessment of Different Image Descriptors for Topological Map-Building and Scene Recognition

Emilio García Fidalgo

emilio.garcia@uib.es

Abstract

An autonomous robot needs to perform some tasks in order to be independent. Some of these tasks require that the agent is able to locate itself in its environment. This process is known in mobile robotics as *localization* and it can be achieved by different ways according to the characteristics of the scenario where the robot is working.

One of this ways is that the robot use a world representation called *map*. There exists several methods to represent the agent environment. A widely used option is *topological maps*, where the world is modeled as an abstract graph with nodes which represent locations, and links between them that indicate possible actions to take between two concrete nodes.

The process of construction of these maps is called *mapping*. A number of sensors can be used to this end in order to obtain information from the environment. When a camera is the selected option, visual information needs to be described and managed. In this case, the quality of the mapping and localization processes depends on how images are described.

Within this context, this work introduces, on the one hand, an appearance-based method to create topological maps from a sequence of images; it also defines several measures that permit assessing the performance of different visual descriptors for mapping and localization tasks. On the other hand, this work also comprises a comparison of different view descriptors, involving real cases and different types of scenarios, and using the aforementioned measures as the figures of merit.

As will be seen, the developed framework and the measures exposed in this report can be easily extended and used to test more image descriptors in different environments. This work also shows a first mapping and localization approach in underwater scenarios, not much explored yet.

Key words: Topological Map, Scene Recognition, Localization, Image Descriptor

Contents

1. Introduction

An autonomous vehicle is a robot which is able to operate without receiving remote orders. This kind of vehicles has a lot of applications, especially in scenarios under unsafe conditions, potentially dangerous for humans. However, due to this autonomy, these robots need to handle information from their environment to plan and perform different tasks. An example of these tasks is *localization*.

The localization process is an important problem in mobile robotics. It can be defined as the ability of a robot to locate itself within its environment to achieve tasks. To this end, some applications use external infrastructures like the Global Positioning System (GPS). However, those systems are not available in many places such as indoor, underground and underwater environments, and localization must be solved internally by the robot, using its own sensor information. In these cases, the agent builds its own representation of the world using the available sensors and establishes a mechanism to infer its position in it. The study of this process is called *robot mapping*.

The accuracy of these maps depends on the information needs of the application. There exists three different paradigms for mobile robot mapping:

- Metric maps: This kind of maps represents the world as accurate as possible. They maintain a lot of information about environment details, such as distances, measures, sizes, etc. and they are referenced according to a global coordinate system. The main drawbacks of this approach are the storage needs and the processing time, which makes its use in some real time applications more difficult.

- Topological maps: This approach tries to generate an abstract representation of the world, usually as a graph with nodes and links between them. Nodes represent environment locations with similar features and links are relationships or possible actions to take between the different locations. These maps are simple and compact, and require much less space to be stored than metric maps. However, they are not useful for tasks with accuracy needs, for example obstacle avoidance.

- Hybrid maps: This last paradigm tries to maximize the advantages and minimize the problems of each kind of map alone and combine them in a different mapping technique.

Ultrasonic and laser sensors have been used for years to construct these maps. Nevertheless, recently there has been a significant increase in the number of visual solutions because of the low cost of cameras and the richness of the sensor data provided. Using cameras permits us to obtain a visual representation of the robot's world. To achieve this goal, it is necessary to describe the acquired images and be able to compare these descriptors. Consequently, the quality of the map and the posterior self-localization will directly rely on the method used for visually describing the different environment locations.

Many image detectors and descriptors have been proposed along the years. This work focuses on comparing the performance of some of them to create topological maps and localize images in it. For each one, a graph is created in memory from a

sequence of images according to the differences between consecutive frames. Then, we determine, given a previously unseen image, what is the node it most likely comes from. We also compare these images with a manually generated ground truth graph. This allows us to assess the localization process in a common map for all descriptors, since each image descriptor generates its own representation of the environment and the number of nodes may vary. For every test, we obtain error rates and execution times.

Map building and localization has been widely studied for indoor environments. This is because these scenarios present common regularities and it is easier to extract information from them. However, underwater scenes are most difficult due to the lack of defined structures. Another goal of this work is, thus, to validate the suitability of these descriptors in these unexplored scenarios.

The outline of this report is as follows. Section 2 reviews previous works related to topological mapping, localization and image retrieval systems. Section 3 explains the image feature detectors and descriptors used in this work. Section 4 introduces an abstract model of topological maps and shows how these maps are built. Section 5 presents the experiments and metrics developed to test the efficiency of each descriptor for mapping and localization. Section 6 shows experimental results for outdoor, indoor and underwater environments. Finally, Section 7 concludes the report.

2. Related Work

A large number of solutions for topological map building and image retrieval can be found in the scientific literature. These are areas of active research. The goal of this section is not to make an exhaustive review of the proposed solutions, but to focus on the last 15 years. For an extensive coverage, see [5].

Although most works are based on either topological maps or metric maps, some authors have tried to make hybrid solutions, combining both paradigms in one. Thrun [32] uses an artificial neural network to construct a grid-based model of the environment. On top of the grid representation, topological maps are generated by splitting the metric map into coherent regions. In this work, topological maps are employed for efficient planning while metric maps are used for scene recognition. Sonar sensors are employed as the main source of information.

Winters and Santos-Victor [36] utilize an omnidirectional camera to create a topological map from the environment during a training phase. Nodes are sets of images with common features and links are sequences of consecutive views between two nodes. The large image set obtained is compressed using Principal Component Analysis, resulting in a low dimensional eigenspace from which the robot can determine its global topological position using an appearance-based method. Based on this work, Gaspar *et al* [13] map an indoor environment and emulate insect vision-based navigation capabilities. Similarly, Swets and Weng [30] use also Principal Component Analysis and Multivariate Linear Discriminant Analysis to generate *Most Expressive Features (MEF)* and *Most Discriminant Features (MDF)*, respectively. They experiment with these features for face recognition.

Both global and local image descriptors have been employed along the years for object and scene recognition. The former typically consider the entire image to per-

Figure 1: Rhino.

form the description. The latter are based on local changes between adjacent pixels or regions.

Global representations proposed in the past include responses to banks of filters [34], multi-dimensional receptive field histograms [23] and various forms of color histograms [35]. Local image descriptors comprise different forms of salient image points or image regions: rotationally invariant features [17, 24, 25, 37], Fourier transforms of salient image regions [27], etc.

Some authors have experimented with different descriptor types. For instance, Kosecka *et al* [15] proposed a navigation strategy using gradient orientation histograms as image descriptor. In an exploration phase, a topological map is built by comparing successive frame descriptors. For each node, a set of representative views are computed using Learning Vector Quantization. During the navigation, the current frame's histogram is extracted and compared with each node representatives using the Euclidean distance to determine the most similar location. Kosecka and Yang [14] have also demonstrated the suitability of scale-invariant features for localization tasks. They employ SIFT [17] for this purpose and improve the localization phase using a Hidden Markov Model.

Zivkovic *et al* [38] presented an algorithm for automatically generating hierarchical maps from images. A low-level map is built using SIFT features. Then they cluster nodes to construct a high-level representation. Later, [7] showed a navigation system based on a topological map which used the epipolar geometry to obtain a robust heading estimation.

Museum guidance is one of most tested map-building applications. These robots need to be autonomous in their tasks, such as recognizing people, guiding them through the museum and avoiding obstacles. Because of the growing interest in this type of vehicles, we are going to mention the main contributions in the following. *RHINO* [8] was a robot deployed in the *Deutsches Museum Bonn*, where it guided a hundred of visitors for six days (see Figure 1). Later, the same team constructed an improved version of this robot called *MINERVA* [33]. This prototype was equipped with two cameras and a laser sensor to build a complete map of the environment for the navigation process. More recently, Shen and Hu [26] presented *ATLAS*, a museum guiding robot that combines topological map building and appearance-based matching algorithms for localization. ATLAS also incorporates a human face detection algorithm used to actively approach to new visitors.

Liu *et al* [16] described a new color based descriptor called FACT. They used it to

(a) (b) (c)

Figure 2: (a) Original image; (b), (c) Gradient images in x and y directions, respectively.

create a topological map and navigate through it. This descriptor is based on the fact that, in indoor environments, the important vertical edges (windows, columns, etc.) naturally divide the indoor environment into several meaningful cuts. Consequently, for each cut, the average color value in the U-V space is computed. This U-V average value and the width of the region form the region descriptor. A scene descriptor is formed concatenating each region descriptor in a vector. Scene matching between new scenes and existing nodes was performed computing the 2D euclidean distance between color descriptors and recursively comparing the widths of the regions according to an empirically determined inequality. Every node was characterized with more than one image.

Recent works [10–12] are heading to the *Bag of Words* or *Bag of Visual Words* representation. *Bag of Words* is a concept extracted from text retrieval that was first applied to image search by Sivic and Zisserman [28]. This description quantizes a set of image descriptors against a vocabulary of prototypical descriptors. The vocabulary is learned by clustering descriptors from a set of training data. This quantization process allows a high-dimensional descriptor (for instance SIFT) to be replaced as a single integer. Images can now be represented as a short list of integers specifying which visual words occurred in the image and how often they occurred.

Bacca *et al* [2] have proposed recently an innovative feature management approach for topological map-building and localization, which is based on a human memory model and implements concepts such as *Short-Term Memory (STM)* and *Long-Term memory (LTM)*. Using *Feature Stability Histograms (FSH)*, their method can deal with temporary occlusions and changes in illumination caused by dynamic environments.

To conclude this section, we highlight that: (1) although there exist works that try to compare different image descriptors for various purposes and propose criteria to assess the properties and performance of different features [3, 6, 19, 20], (2) none of them show results for underwater scenarios.

3. Image Description

In this section, the image descriptors that are used in this work are described. The main idea is to explain their essentials and the configuration used in our experiments.

3.1. Gradient Orientation Histograms

This is the only global image descriptor considered in this work. Among others, it was applied by Kosecka *et al* [15] for navigation tasks. Our implementation and use in

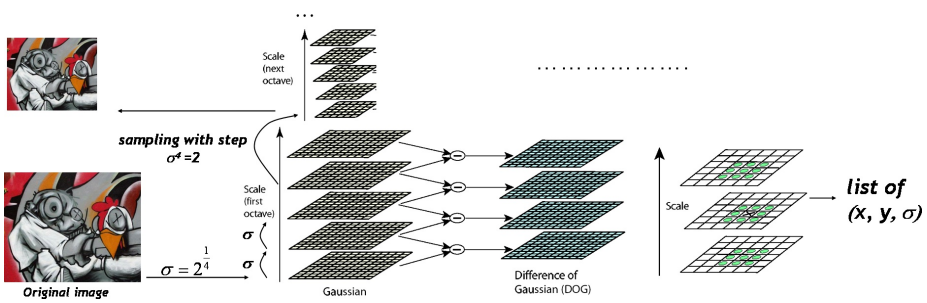


Figure 3: Overview of the DoG scheme, used by SIFT for the key point detection.

our tests is based on that paper. These orientation histograms are computed from the gradient images G_x and G_y, which represent the gray value variations in the x and y directions, respectively.

The edge image is also computed using the Canny detector. To calculate the histogram, we consider only pixels belonging to an edge whose magnitude is in the top 4%. For these egdes, orientation is quantized and accumulated into a 36 bin vector. As a result, we have the gradient orientation histogram, which is normalized for comparison purposes.

In order to obtain better discrimination capability of this representation, we compute the histogram for five sub-images: four quadrants and the central region. The final image descriptor is formed concatenating the five sub-histograms into one. The case of a single histogram for the whole image has also been considered.

We did not employ any external code to use this global descriptor. It has been fully implemented by ourselves, using *C++* language and the *OpenCV*[1] library.

3.2. SIFT

The Scale-Invariant Feature Transform (SIFT) is an algorithm developed by Lowe [17] to detect and describe distinctive key points in images which was originally created for object recognition. These key points are invariant to image rotation and scale and robust across affine distortion, noise and changes in illumination.

To obtain these key points, a scale space is generated convolving the original image with Gaussian kernels at different scales. A set of Difference of Gaussians (DoG) images is obtained subtracting the successive blurred images. Key locations are defined as maxima and minima of the Difference of Gaussians that occur at multiple scales (see Figure 3). Specifically, a DoG image is given by:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L\left(x, y, k_j\sigma\right), \tag{1}$$

where $L(x, y, k\sigma)$ is the convolution of the original image with a Gaussian kernel at scale $k\sigma$:

[1] http://opencv.willowgarage.com

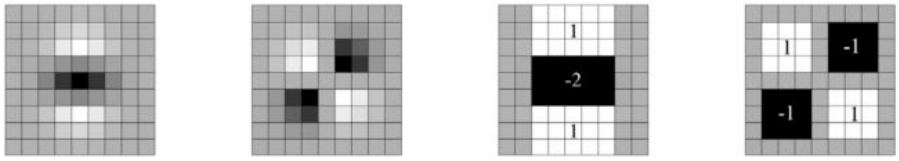


Figure 4: Left to right: the Gaussian second order partial derivatives in y-direction and xy-direction, and SURF approximations using box filters. The gray regions are equal to zero.

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \ . \tag{2}$$

Scale-space extrema detection produces too many candidates, some of which are unstable. Therefore, the next step is to perform a filtering process using the quadratic Taylor expansion of the DoG scale-space function and the eigenvalues of the second-order Hessian matrix, resulting into a reduced set of key locations. This key point detection is combined with a 128 dimensional descriptor, calculated on the basis of gradient orientation histograms of 4×4 subregions around the interest point.

Once the key points have been detected and described, it is important to define an efficient method to match these features in different views. Comparing one by one is not a real solution, due to its inefficiency. Therefore, Lowe used a modification of the k-d tree algorithm called Best-bin-first (BBF) search method that can identify the nearest neighbors with high probability using only a limited amount of computation. The nearest neighbors are defined as the points with minimum Euclidean distance from the given descriptor vector.

In our case, we have tested the original SIFT algorithm. Each image is represented by a set of SIFT descriptors and BBF over a k-d tree is used for feature search. SIFT features and k-d tree implementations employed in this work come from Rob Hess[2].

3.3. SURF

Speeded Up Robust Features is an image detector and descriptor presented by Bay *et al* [4]. It is partly inspired by the SIFT algorithm but outperforms previous solutions in terms of computation time.

The SURF detector is based on the Hessian matrix, due to its good performance. Given a point $\mathbf{x} = (x, y)$ in an image I, the Hessian matrix in x at scale σ is defined as:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}, \tag{3}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} G(\sigma)$ with the image I at point $\mathbf{x}$, and similarly for the rest of terms. The determinant of this matrix is used for selecting the location and the scale.

[2]http://blogs.oregonstate.edu/hess/code/sift/

Denoting the Hessian components by D_{xx}, D_{yy} and D_{xy}, the blob response at location x in the image can be approximated by:

$$det(H_{approx}) = D_{xx}D_{yy} + (0.6D_{xy})^2 \,. \tag{4}$$

These responses are stored in a blob map, and local maxima are detected and refined using the quadratic interpolation.

The Hessian is roughly approximated using a set of box-type filters (see Figure 4). These approximations can be evaluated very fast, and independently of the image size, using integral images:

$$II(x, y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x', y') \,, \tag{5}$$

where $i(x, y)$ is the input image. The 9 x 9 box filters in Figure 4 are approximations for a Gaussian with $\sigma = 1.2$ and represent the finest scale.

Descriptors show how the pixel intensities are distributed within the neighborhood of each feature at different scales. The result is a 64 dimensional vector.

In our experiments, we have used the *OpenCV* SURF and k-d tree implementations. We have limited the amount of features for each image. Only a parameterized number of points with the highest Hessian response have been considered to control the description time per image.

3.4. FAST

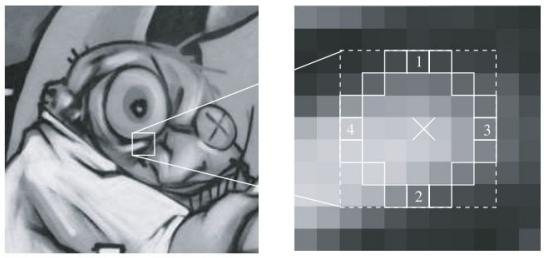


Figure 5: Pixels examined by FAST detector.

Features from Accelerated Segment Test is a corner detector proposed by Rosten and Drummond [22]. It is based on the SUSAN [29] detector.

FAST compares the intensity in a circle of 16 pixels around the candidate point (see Figure 5). Initially pixels 1 and 2 are compared with a threshold, then 3 and 4 as well as the remaining ones at the end. The pixels are classified, according to its intensity, into dark, similar and brighter groups. An image point is a feature if a minimum of pixels can be found on the circle of fixed radius around the point such that these pixels are all brighter or darker than the central point. The feature descriptor consists of a vector containing the intensities of the 16 pixels surrounding the point.

FAST has been reported as 30 times faster than a DoG detector, such as SIFT. However, it is not invariant to scale changes and it depends on a predefined threshold.

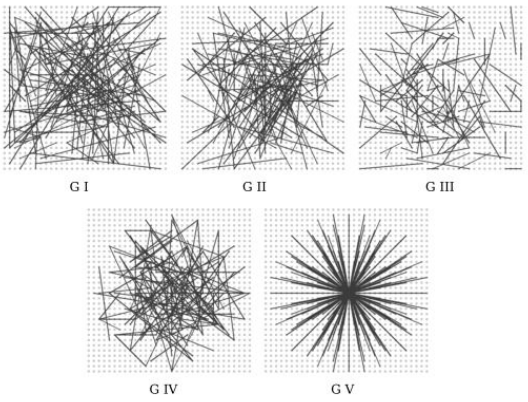


Figure 6: Different spatial arrangements for computing BRIEF: G I: $(x, y) \approx U(-\frac{S}{2}, \frac{S}{2})$; G II: $(x, y) \approx N(0, \frac{1}{25}S^2)$; G III: $x \approx N(0, \frac{1}{25}S^2), y \approx N(x, \frac{1}{100}S^2)$; G IV: (x, y) are randomly sampled from discrete locations of a coarse polar grid introducing a spatial quantization; G V: $\forall i : x = (0, 0)^T$ and y takes all possible values on a coarse polar grid containing the number of desired tests points; the patch size is $S \times S$ pixels and the origin of its coordinate system is located at the center.

We have employed the original Rosten and Drummond implementation, adapted to *C++* language. As before, the number of features is specified to control the image description time. The matching process between two images is performed comparing all the features of the first image with all the features of the second one.

3.5. Star and BRIEF

The remaining cases considered correspond to the combination of different algorithms for detection and description. In this section, we deal with the combination consisting of Star as the feature detector and BRIEF as the feature descriptor.

Star is a feature detector developed by *Willow Garage*[3] and derived from CenSurE (Center Surrounded Extrema) [1]. This last one uses center-surrounded bi-level filters to approximate the Laplacian. The bi-level adjective means that they multiply the image pixel values by either 1 or -1. Examples of filters are shown in Figure 7. The first one is the most faithful to the Laplacian, but hardest to compute. To reduce execution time, CenSurE replaces the two circles with squares (Figure 7d), easier to calculate using integral images. For each pixel in the image, the filter response is calculated for seven scales. Then, a non-maximal suppression is performed over the scale space and weak features are discarded. Finally, unstable points on edges are suppressed examining each feature corner with the Harris measure under a 9x9 window. Meanwhile, Star approximates the Laplacian with 2 overlapping squares: one upright and one 45-degree rotated, and performs a sub-scale interpolation.

Binary Robust Independent Elementary Features (BRIEF) is a simple binary descriptor created by Calonder *et al* [9]. The main goal of BRIEF is to speed up the

[3]http://www.willowgarage.com/

matching process. The descriptor is a binary string, where each bit represents a simple comparison between two points inside a *patch* in the image. A bit is set to 1 if the first point has a higher intensity than the second. In the original work, the authors suggest several point spatial arrangements over a key point centered path (see Figure 6). According to the authors, empirically, better results are obtained with point pairs randomly drawn from uniform or Gaussian distributions of point coordinates. The Hamming distance is used for matching, taking the advantage of the XOR and bit-counting CPU instructions.

As for the implementation, we have used the Star detector included in the *OpenCV* library. For BRIEF, we employed the original code from the authors. The matching process between two images is performed comparing all features of the first image with all features of the second one. This is a very fast process due to the simplicity of the BRIEF descriptor (16 bits).

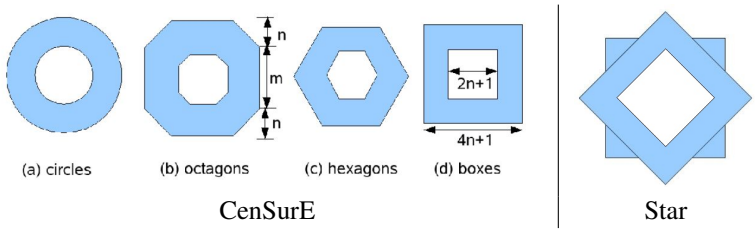


Figure 7: Bi-level filters used by CenSurE and Star

3.6. FAST and BRIEF

In this case, FAST is used to detect corners in the image and these ones are described using BRIEF. All previous considerations for each case are still valid.

3.7. Star and FAST

The last image description tested in this work is again a combination of two previous techniques. Star is used as a key point detector and FAST as a descriptor.

4. Environment Model and Map Construction

This section exposes a general model of topological maps and explains the method used for its construction.

4.1. Environment Model

First of all, it is important to establish what is a topological map for us. In general terms, it is a graph-like representation of the environment, where nodes often represent states in the agent's configuration and links are system actions that take the agent from one state to another. Despite this common use, there is not an agreement about what really are or how they are built. Nevertheless, there have been authors who have tried to establish a general theory of these maps [21].

In our approach, a *node* is a group of ordered images with similar features, which represent a *location* in the environment. For each node, a subset of these images are chosen as node representatives for scene recognition. The election of representatives in the nodes is for computational reasons and to improve the stability of the process. In the scene recognition phase, it would be very slow to compare the query image with each frame in the location. Therefore, it is important to select a set of them, and compare only these ones.

We do not use the above-mentioned concept of *link* and, thus, for now, our matching method does not consider these relationships between nodes to improve the localization process. Finally, for the time being, the map does not contemplate loops. Therefore, if a node is revisited, it appears as a new node in the graph.

Given $I = \{I(1), \ldots, I(m)\}$ as an input sequence of m images, we define our topological map as an ordered sequence of nodes:

$$M = \{N_1, \ldots, N_l\}\,, \tag{6}$$

where l is the number of nodes generated by the mapping process. Consecutive nodes represent locations of the environment the robot has consecutively visited. Particularly, the i-th node is defined as the following 2-tuple:

$$N_i = (J_i, R_i)\,, \tag{7}$$

where J_i is the sorted sequence of images included in the location and R_i the node representatives. Formally:

$$J_i = \{j_{i,1}, \ldots, j_{i,n_i}\} \subseteq \{1, \ldots, m\}\,, \tag{8}$$

$$R_i = \{r_{i,1}, \ldots, r_{i,n_i^*}\} \subseteq J_i\,, n_i^* \leq n_i\,, \tag{9}$$

where each j_i and r_i are indexes of the original sequence of images.

To generate the graph, we need to describe an image and be able to compare two of them. Next sections formalize these concepts.

4.1.1. Image Description

A frame is represented by an *image descriptor*, which changes according to the method used in each case. We denote it with the symbol Δ. In the case of the orientation histograms, the descriptor is defined as follows:

$$\Delta_h(I(i)) = \{h_{i,1}, \ldots, h_{i,j}, \ldots, h_{i,g}\}\,, \tag{10}$$

where each $h_{i,j}$ represents the probability by which the angle $\alpha(j) = j \times \frac{360}{g}$ appears in the image. Observe that, if we are considering five sub-images, the final descriptor is obtained concatenating the respective five histograms into one.

When the image is described by means of local features, the *image descriptor* is defined as:

$$\Delta_f(I(i)) = \{f_{i,1}, \ldots, f_{i,s_i}\}\,, \tag{11}$$

where each f_i is a feature descriptor, different for every case. Every feature descriptor is usually constituted by a sequence of numbers. We assume that there is always available a distance function to determine the dissimilarity between feature descriptors $d_f(f_i, f_j)$.

4.1.2. Comparing Images

In order to compare two frames, we need to define a measure that quantifies how *similar* or *dissimilar* two *image descriptors* are. These measures are called *proximity functions* [31] and they are defined as follows:

$$\rho : \Delta \times \Delta \rightarrow \mathbb{R}\,, \tag{12}$$

where we generically denote by Δ the set of image descriptors produced by a given description method.

This function definition derives in two cases: *dissimilarity* and *similarity* functions. On the one hand, a *dissimilarity function* associates high values to low-similarity descriptors: the less similar they are, the higher is the value. More precisely, it is defined as:

$$\mathrm{dis} : \Delta \times \Delta \rightarrow \mathbb{R}\,, \tag{13}$$

where:

(1) $\exists d_0 \in \mathbb{R} : -\infty < d_0 \leq d(x, y) < +\infty, \forall x, y \in \Delta$

(2) $\mathrm{dis}(x, x) = d_0, \forall x \in \Delta$

(3) $\mathrm{dis}(x, y) = \mathrm{dis}(y, x), \forall x, y \in \Delta$

On the other hand, a *similarity function* associates high values to very similar descriptors: the more similar they are, the larger the value. More precisely, these functions are defined as follows:

$$\mathrm{sim} : \Delta \times \Delta \rightarrow \mathbb{R}\,, \tag{14}$$

where:

(1) $\exists s_0 \in \mathbb{R} : -\infty < s(x, y) \leq s_0 < +\infty, \forall x, y \in \Delta$

(2) $\mathrm{sim}(x, x) = s_0, \forall x \in \Delta$

(3) $\mathrm{sim}(x, y) = \mathrm{sim}(y, x), \forall x, y \in \Delta$

This kind of measures have been used to compare two images in our model. We will discuss about the details later in this document.

Figure 8: Example of representative views in a location: *mean image*, first and last images, respectively.

4.1.3. The Election of Representatives

Another issue that comes from our model definition is how to elect the representatives of a node. For this purpose, we define the *total proximity* of an image in a node as:

$$P(I(i), N_k) = \sum_{j=1}^{n_k} \rho(I(i), I(j)) \,, \tag{15}$$

where n_i is the number of frames of the node N_k. We also define the *mean image* ($\bar{I}_k$) of the location as the image with the minimum or maximum total proximity, depending on the measure used. In case of a *dissimilarity function*, it is defined as follows:

$$\bar{I}_k = \underset{I(j)\,, j \in n_k}{\arg\min} \{\rho(I(j), N_k)\} \,. \tag{16}$$

When a *similarity function* is used as measure, the *mean image* is defined as:

$$\bar{I}_k = \underset{I(j)\,, j \in n_k}{\arg\max} \{\rho(I(j), N_k)\} \,. \tag{17}$$

The *mean image* denotes the frame of a location which is closer to the rest than the other images. We do not define this function as the mean between descriptors because the corresponding descriptor does need to correspond to a real, physically possible image.

The accuracy of the localization process will directly rely on how we select these representative views. In this work, three options have been considered:

- Get only the *mean image*.
- Get the *mean image*, the first and the last images in the node, according to the order of the sequence.
- Get the n images with the best *total proximity* values.

The second option tries to summarize the variability of images inside a node. The third one is the extension of the concept of *mean image* to a subset of node frames, so that this subset comprises those images closest to the rest of images, in the sense of *total proximity* defined before.

4.1.4. Constructing the Map

Once we have defined our model, we are ready to explain the method that has been used to build a topological map. Initially, a sequence of images is taken using a camera. Each frame is described (Δ) and compared (ρ) to the previous one. The result of the proximity function is then evaluated against a threshold, adding the image to the last location in the graph or creating a new node, as appropriate. This threshold depends on the descriptor used for the test and vary for every case.

Despite the obvious visual similarity between consecutive images, it is possible to obtain extreme distances between two of them, due to image noise or during fast turns of the camera. Therefore, we can have nodes with few frames, irrelevant for the localization phase. The next step is, thus, to refine the map, deleting the superfluous nodes comprising a small number of images. An example of this situation is illustrated in figure 9.

For each of the nodes surviving from the purging stage, their representatives are calculated as indicated in section 4.1.3.

Terms view and location will be used interchangeably from now on to refer to nodes of the topological map.

4.1.5. A Brief Note about the Internal Representation of the Maps

The internal representation of a map is the result of taking into account the following:

- First of all, each frame is identified by its index within the input sequence.
- Second, due to the nature of the map construction process, which leads to splitting the input sequence into different segments, a map turns out to be an ordered set of locations/segments.
- Third, due to the frame purging stage, some input frames results to be not assigned to any location in the map.

As a consequence of all the aforementioned, a map is internally stored as a vector of locations, and each location is in turn represented as the vector containing the indexes of the images that have been assigned to the corresponding map node.

This abstract representation is illustrated in Figure 10. First, the number inside the boxes represent the index of the frame in the input sequence and, second, green boxes refer to locations in the graph. In the case depicted in the figure, there are two images not associated with any view.

To finish, notice that using this representation, we know the range of images inside each location and we are able to determine, given a query image, where it should be theoretically classified.

4.2. Proximity Functions

In the previous section we have performed a formal description of our model. This definition includes a set of proximity measures that permits us to compare two image descriptors. In practice, these measures can be implemented in different ways. The main goal of this section is to specify the proximity functions used in this work.

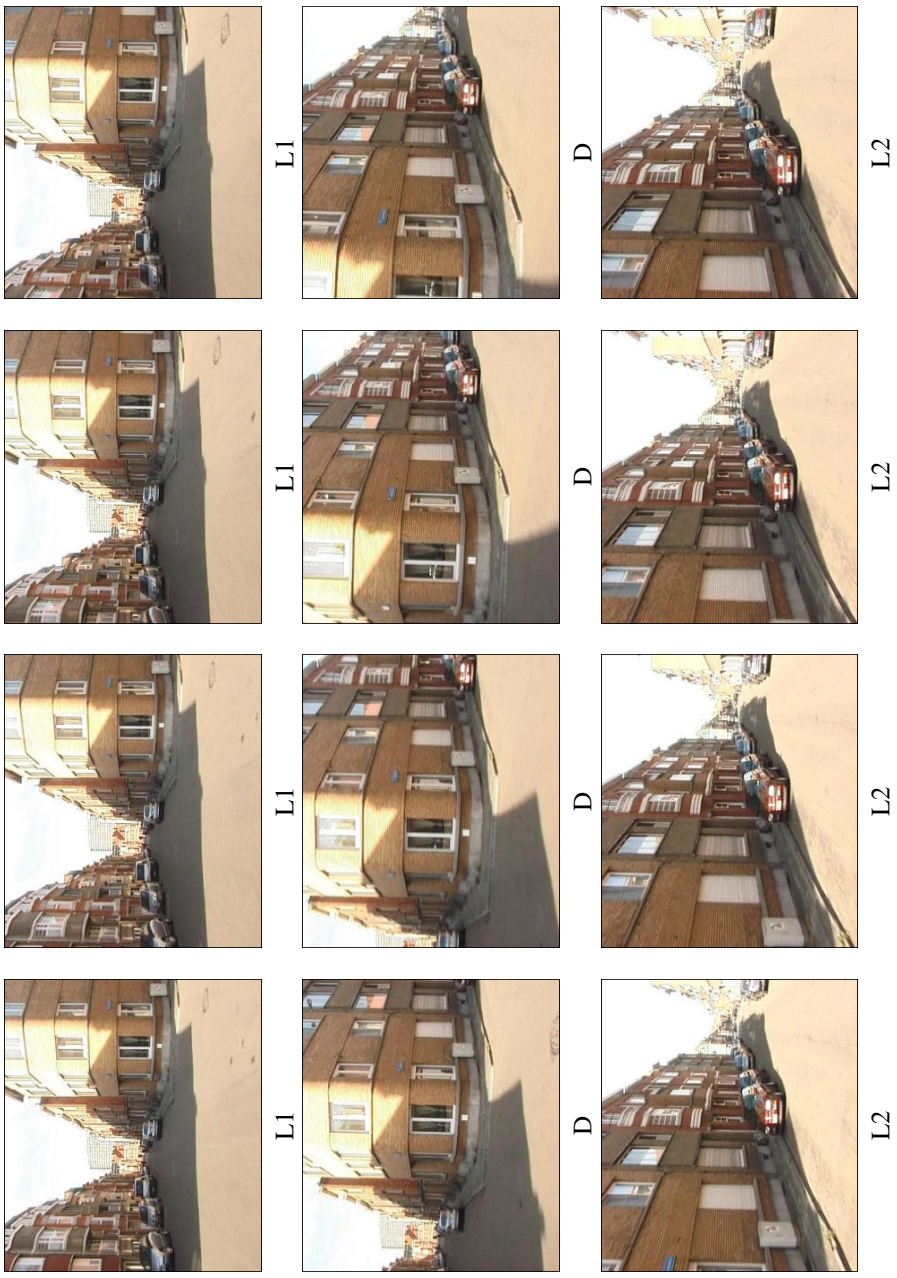


Figure 9: Example of segmentation of an input sequence of images. Below each frame, it is shown the final location assigned. L1 and L2 are locations, while D indicates that the image has been discarded. In this case, the discarded images are due to a right turn of the vehicle.

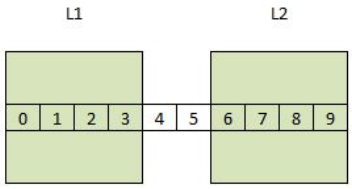


Figure 10: Internal representation of a map.

A dissimilarity function is usually implemented as a distance between descriptors. In the case of orientation histograms, χ^2 is used as empirical distance between two distributions:

$$dis(\Delta_h(I(i)), \Delta_h(I(j))) = \sum_k \frac{(h_{i,k} - h_{j,k})^2}{h_{i,k} + h_{j,k}}, \forall k \in \{1, \ldots, g\}. \tag{18}$$

In case the image is described by a set of local image feature descriptors, we use a similarity function. For this purpose, we employ a measure based on the number of matchings between two image descriptors.

The concept of *matching* varies according to the descriptor used. In the case of SIFT and SURF, we define the set of matchings between two images as follows:

$$\begin{aligned} M(I(i), I(j)) = &\{(f_{i,a}, f_{j,b}) \mid f_{i,a} \in \Delta_f(I(i)), f_{j,b} \in \Delta_f(I(j)), \\ &L_{d,a} = \{d_f(f_{i,a}, f_{j,c}) \mid f_{j,c} \in \Delta_f(I(j))\}, \\ &L_{d,a} \text{ is an ordered set } \mid L_{d,a}(1) \leq L_{d,a}(2) \leq L_{d,a}(3) \ldots, \\ &L_{d,a}(1) \leq 0.6 \times L_{d,a}(2), \\ &L_{d,a}(1) < T\}, \end{aligned} \tag{19}$$

where T is a distance threshold and varies for each kind of feature. The relation between the nearest feature and the second nearest one is an empirical value used in some previous works. More detailed justification about this threshold can be found in [14, 18].

In the case of BRIEF, matchings are defined differently:

$$\begin{aligned} M(I(i), I(j)) = &\{(f_{i,a}, f_{j,b}) \mid f_{i,a} \in \Delta_f(I(i)), f_{j,b} \in \Delta_f(I(j)), \\ &d_f(f_{i,a}, f_{j,b}) = \min\{d_f(f_{i,a}, f_{j,c}) \mid f_{j,c} \in \Delta_f(I(j))\}, \\ &d_f(f_{i,a}, f_{j,b}) = \min\{d_f(f_{i,c}, f_{j,b}) \mid f_{j,c} \in \Delta_f(I(i))\}, \\ &d_f(f_{i,a}, f_{j,b}) < T\}. \end{aligned} \tag{20}$$

Regarding FAST, the matching process do not make use of cross-validation as expressed in equation 20, nor introduces any additional check such as the ratio between the shortest and the second shortest distances of equation 19. This is an order to speed

up the matching as much as possible. Next equation describes formally the procedure implemented:

$$\begin{aligned} M(I(i), I(j)) =& \{(f_{i,a}, f_{j,b}) \mid f_{i,a} \in \Delta_f(I(i)), f_{j,b} \in \Delta_f(I(j)), \\ & d_f(f_{i,a}, f_{j,b}) = \min\{d_f(f_{i,a}, f_{i,c}) \mid f_{i,c} \in \Delta_f(I(j))\}. \end{aligned} \tag{21}$$

Once we have explained what is a matching for us, we are able to define a similarity function as follows:

$$sim(\Delta_f(I(i)), \Delta_f(I(j))) = \frac{|M(I(i), I(j))|}{\left(\frac{s_i + s_j}{2}\right)}, \tag{22}$$

where $|X|$ refers to the cardinal of set X.

This measure is, thus, just the ratio between the number of matchings and the average number of features found in each frame.

5. Experimental Methodology and Performance Metrics

This section describes the tests that have been carried out in order to assess the performance of the different feature detectors and descriptors considered, with regard to the tasks described in the previous sections. To this end, Section 5.1 proposes a test for checking the similarity between generated and reference maps, where the latter have been manually generated by a human using common sense. Later, Section 5.2 evaluates how proficient are the detectors/descriptors regarding the recognition of a view among the full set of frames within a sequence. Finally, Section 5.3 refers to the execution time behaviour in the different cases.

5.1. Checking the Accuracy of a Topological Map

Algorithm 1 describes the way how a topological map is compared to a reference map acting as the ground truth. In our case, the ground truth has been generated manually applying common sense over the set of frames in the test sequence. Figure 9 can be shown as an example of manual partition.

As we previously mentioned, nodes are composed by images that can be visually assigned to the same view. The remaining frames have not been considered, and therefore the GT graph presents discontinuities with regard to the input sequence. A similarity value between the two maps is generated as output.

More precisely, given a reference map GT and a map to assess M, Algorithm 1 determines, for each location in GT, which is the node of M with the largest number of images in common. This number is accumulated to obtain a goodness measure. Remember that there may exist images in GT not classified in any node of M because of the spurious node deletion phase and therefore it does not make sense to find their correspondences in M. These images are discarded when computing the figure of merit corresponding to this performance aspect.

We define the *successful segmentation rate* (SSR) as the percentage of consecutive images of the reference map that have been put together in the same node of the map under assessment. It is formulated as follows:

Algorithm 1 Successful Segmentation Rate (SSR)

Require: Ground truth (GT) and topological map (M) have been generated.

{n_v: Number of coincident frames.}
{f_g: Total number of frames in GT.}
{u_v: Number of unclassified frames.}
$n_v := 0$, $f_g := 0$, $u_v := 0$
for every location L in GT **do**
  for every location i in M **do**
    $C_v[i] := 0$
  end for
  for every image I in L **do**
    $f_g := f_g + 1$
    if notClassified(I, M) **then**
      $u_v := u_v + 1$
    else
      $i :=$ getLocationIndex(I, M) {See algorithm 2 for further information}
      $C_v[i] := C_v[i] + 1$
    end if
  end for
  $n_v := n_v + max\,(C_v)$
end for
return $\left(\frac{n_v}{f_g - u_v}\right) \times 100$

Algorithm 2 *getLocationIndex*(I, M)

Require: M is a generated topological map.
Require: The index of image I in the input sequence is known: i.
Require: The image range of each location is obtained using *min* and *max* functions.

{This function gets the location in map M where an image I has been classified}
{*lmin*: The lowest index of the images classified in a location.}
{*lmax*: The highest index of the images classified in a location.}
$lmin := 0$, $lmax := 0$
for every location L in M **do**
  $lmin := min(L)$
  $lmax := max(L)$
  if $i \geq lmin$ and $i \leq lmax$ **then**
    return L {The index of the location is returned.}
  end if
end for

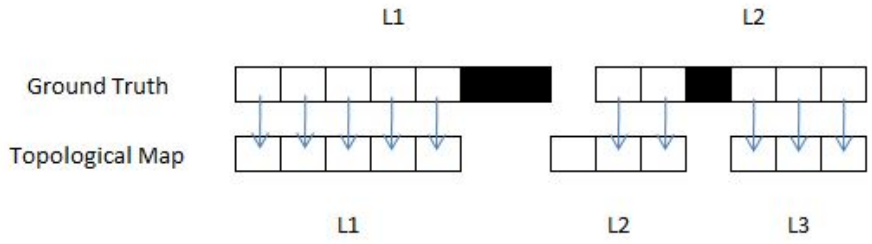


Figure 11: Successful Segmentation Rate Example.

Figure 12: A correct matching. Query image (leftmost image) and the three images chosen as representatives of the location (rightmost images)

$$SSR = \left(\frac{n_v}{f_g - u_v} \right) \times 100\,, \tag{23}$$

where n_v is the number of coincident views, f_g is the total number of frames in the ground truth and u_v the number of discarded views. It is important for us to test that most views in the same ground truth node are classified in the same location in our implementation, because this fact ensures that real locations have not been broken and they keep together in both graphs.

An example of SSR calculation is shown in figure 11. White boxes represent images inside locations, arrows refer to correspondences between graphs and black boxes are the unclassified frames. Most images of the first ground truth location have been classified into the same node in M, therefore we accumulate five images to n_v. The images in the second ground truth location have been assigned to different nodes in the map under assessment. The location of M with the largest number of frames in common in this case is the third one, hence we add 3 ($max\{2, 3\}$) images to n_v. Consequently, the SSR in this situation is:

$$S = \left(\frac{5+3}{13-3} \right) \times 100 = 80\%$$

5.2. View Recognition Assessment

The next phase in this work is oriented to test the quality of each image descriptor recognizing scenes. Given an environment model, we now want to classify new images

as belonging to particular locations and to provide a framework to measure the correctness of this classification. We have developed several tests for these purposes and they are explained in this section.

Algorithm 3 Recognition Error Rate (*RER*)

Require: A generated topological map (M) and a sequence of query images (Q)

1: {i_m: Bad classified images.}
2: {i_t: Total number of query images.}
3: $i_m := 0$, $i_t := 0$
4: **for** every image I in Q **do**
5: $i_t := i_t + 1$
6: {loc_t: The theoretical location of the image according to M.}
7: {loc_a: The location using the visual localization process.}
8: loc_t := getLocationIndex(I, M) {See algorithm 2 for further information.}
9: loc_a := locateImage(I, M) {See algorithm 4 for further information.}
10: **if** loc_t != loc_a **then**
11: $i_m := i_m + 1$
12: **end if**
13: **end for**
14: **return** $\frac{i_m \times 100}{i_t}$

First of all, query images are a subset of the image sequence, or, in other words, form the whole input sequence, we reserve some frames to perform the queries during the experimental assessment, while the rest of frames are used to built the map. In this way, we know which node the corresponding query image should have been assigned to, and we can determine whether the view has been correctly recognized (i.e. the classification has been successful) or not.

Second, remember that the view recognition process starts with the computation of the descriptor of the query image. Next, this descriptor is compared with the descriptors of the representative images of each node. Finally, the query image is assigned to the location that contains the closest frame. If this location matches with the theoretical node, according to the index of the image in the input sequence, we consider it as a correct view recognition. Otherwise, we increment the number of bad classified images in order to obtain a final *recognition error rate* (*RER*) as follows:

$$RER = \frac{i_m \times 100}{i_t}\,, \tag{24}$$

where i_m is the number of bad classified images and i_t is the total number of query images. This process is shown in algorithm 3.

Observe that, since each combination of feature detector and descriptor generates a different set of locations during the construction of the topological map, we can not compare *RER* values directly between descriptors and a common representation of the environment is needed. Due to this reason, this test has been performed against the generated map and against the ground truth. The former permits us to assess the quality of the generated map itself in recognition tasks. The latter allows us to compare each

Algorithm 4 *locateImage*(I. M)

Require: M is a generated topological map.
Require: The descriptor of an image is computed using the *desc* function.
Require: The proximity between two descriptors is computed using the *prox* function.
Require: *rep*(L) returns the representatives of location L.

```
{Returns the closest location of an image I in the map M}
{d_min: The minimum distance between the query and a representative image.}
{loc: The index of the location closest to the query image.}
{Δ(I): Query image descriptor.}
d_min := MAXFLOAT, loc := -1
Δ(I) := desc(I)
for every location L in M do
   for every representative R in rep(L) do
      d_aux := prox(Δ(I), desc(R))
      if d_aux ≤ d_min or d_aux ≥ d_min then
         {It depends on the type of the proximity measure used.}
         d_min := d_aux
         loc := L {The index of the location is stored.}
      end if
   end for
end for
return loc
```

descriptor with the others, using a common environment representation.

Since in our case there can be unclassified images, in this case, we have implemented two possibilities to compute *RER*: (1) avoid using the unclassified images as query images, or (2) consider as a success matching the unclassified image with any of the locations in which the reference node has been split. This last situation is illustrated in figure 13. Ellipses represent locations and numbers indicate the range of images assigned to nodes $L1$ and $L2$. The 20-th image is not included in the image ranges of any of the 2 nodes. Consequently, if the image is classified in the adjacent nodes $L1$ or $L2$, the matching is considered as correct.

In summary, we have performed these variations of the *RER* test in order to contemplate all the possibilities:

- Against the ground truth and discarding unclassified images.
- Against the generated topological map and discarding unclassified images.
- Against the ground truth and considering unclassified images.
- Against the generated topological map and considering unclassified images.

5.3. Temporal Behavior Analysis

The last goal of this work is to determine the image descriptors efficiency in terms of computation time. The requirements on this performance aspect depend on the task

that we need to achieve. For example, a robot that navigates according to a topological map and tries to localize itself in it requires a very fast image descriptor due to the real time needs of the application. However, the topological map construction can be performed in an off-line phase and it is not critical for the agent.

We have defined the following times to evaluate the performance of each descriptor:

- Description time (t_d): It is the time required to compute the descriptor from an image.
- Segmentation time (t_s): It is the time needed to construct the topological map from the sequence of images.
- Query time (t_q): It is the time required to compare a query image with all location representative images and to identify which is the closest one.

For description and query times, *mean*, *maximum* and *minimum* values have been obtained, in order to characterize more accurately the temporal behavior of the aforementioned tasks.

6. Experimental Results

Once our test framework has been defined, this section shows the experimental results that we have obtained in different environments. For each scenario, all the tests have been performed, varying the image description procedure, as well as the way how the representative images of a node are chosen from one experiment to the next. We have considered three kinds of environments: outdoor, indoor and underwater. The corresponding results are described in the following sections.

As was anticipated before, we have selected a subset of the available frames of an image sequence as query images, which have not been used for building the map. More precisely, in all tests, the set of queries have consisted of every fifth frame of the sequence.

In all cases, quantitative results are given as tables withe the following structure:

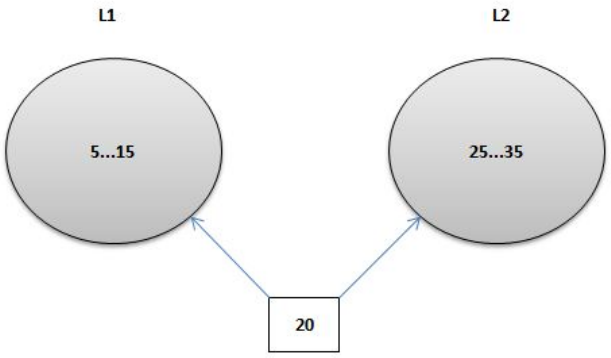


Figure 13: A correct matching considering adjacent nodes.

Figure 14: Examples of outdoor images used.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	16	85.2%	17%	25%	35%	27%
SIFT	21	69.7%	39%	42%	55%	58%
SURF	12	80.9%	41%	52%	58%	54%
FAST	28	74.4%	25%	25%	38%	27%
Star/BRIEF	16	82.7%	54%	63%	57%	68%
FAST/BRIEF	12	85.5%	64%	60%	65%	65%
Star/FAST	19	70.4%	50%	61%	54%	61%

Table 1: Outdoor results where the set of representative images of a node consists of only the *mean image*.

- The L column is the number of locations of the resulting topological map.
- The S column shows the percentage of the SSR test.
- The $R1$ column is the RER for the ground truth discarding unclassified images.
- The $R2$ column is the RER for the topological map discarding unclassified images.
- The $R3$ column is the RER for the ground truth considering unclassified images.
- The $R4$ column is the RER for the topological map considering unclassified images.

The execution time tables show all the values enumerated in section 5.3 in milliseconds.

6.1. Outdoor Environment

The outdoor sequence comprises 1175 images grabbed from a car touring throughout the city of Leuven, Belgium. Visually, one can distinguish roughly 9 locations, being this our ground truth for this scenario. The number of frames associated with individual locations varies between 7 and 90 per location and we consider as a node a segment comprising at least seven images. All images have been resized to 360×288 pixels in order to reduce the computation time. Application parameters for each descriptor have been set to ensure that the number of frames discarded in the construction of the topological map is lower than 80 images.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	16	85.2%	8%	16%	26%	17%
SIFT	21	69.7%	43%	18%	52%	24%
SURF	12	80.9%	26%	40%	37%	40%
FAST	28	74.4%	12%	17%	25%	17%
Star/BRIEF	16	82.7%	59%	62%	64%	71%
FAST/BRIEF	12	85.5%	49%	72%	71%	73%
Star/FAST	19	70.4%	40%	58%	47%	58%

Table 2: Outdoor results where the set of representative images of a node consists of the *mean image*, and the first and the last images of the node.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	16	85.2%	11%	15%	30%	18%
SIFT	21	69.7%	41%	26%	54%	31%
SURF	12	80.9%	31%	47%	48%	47%
FAST	28	74.4%	17%	20%	33%	21%
Star/BRIEF	16	82.7%	42%	46%	52%	55%
FAST/BRIEF	12	85.5%	46%	43%	51%	53%
Star/FAST	19	70.4%	41%	54%	67%	69%

Table 3: Outdoor results where the set of representative images of a node consists of the five most representative images of the node.

	t_d			t_s	t_q		
	Mean	Min	Max		Mean	Min	Max
Orientation Histograms	125	82	199	6	173	170	217
SIFT	646	450	970	91k	110k	3575	127k
SURF	162	113	315	4722	408	306	507
FAST	14	1	43	400	129	18	251
Star/BRIEF	20	7	45	3247	229	73	286
FAST/BRIEF	18	2	39	2572	149	42	192
Star/FAST	25	14	40	1901	175	71	202

Table 4: Computation times for the outdoor in milliseconds.

Figure 15: Examples of indoor images used.

The results for this environment are shown in tables 1 - 4. The SSR indicates that all descriptors create very similar maps, except FAST and SIFT, which generate a lot of locations for the same sequence of images, resulting into a lower SSR value. The segmentation rate is constant for all tables because the criterion for selecting the representative images is only relevant to the recognition process, and it does not affect the map construction. Unlike SIFT, FAST compensates its lower SSR with the fastest description time (t_d), followed by the Star detector, although the latter is not only fast to compute but also invariant to scale changes. As can be observed, SIFT is highly computationally demanding and turns out to be the least compatible with scenarios with real-time needs.

Constructing the topological map, orientation histograms have the best performance in this case. It is because of the simplicity of the dissimilarity function used in this case (χ^2).

According to the results obtained, edge orientation histograms and FAST show the best performance for recognition tasks. Orientation histograms have the lowest RER in most cases. However, FAST is again the fastest image descriptor. SURF is the fastest third, but it does not exhibit good view recognition performance compared with the previous ones. BRIEF turns out to be the worst descriptor, giving a very poor values in recognition and computation times. This last one is a not a good solution according to our experiments.

Regarding the election of node representatives, best recognition results have been obtained using the *mean image*, the first and the last images of each node. This method covers the possible differences in appearance between the beginning and the end of a location. BRIEF is the only descriptor that improves with more views as representatives.

Summing up with this scenario, according to our experiments, edge orientation histograms and FAST seem to yield the best performance in this environment regarding the construction of the topological map and view recognition. The final election will depend on the application needs: FAST has the lowest computation time requirements, while histograms are more accurate recognizing scenes. Star is a very fast detector too, but it can not be used with BRIEF or FAST as a descriptor, due to the bad results of these ones. SIFT and SURF turn out to be, according to the tests performed, less suitable for this kind of scenario.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	18	80.4%	19%	39%	31%	47%
SIFT	18	83.2%	26%	23%	48%	33%
SURF	16	84.1%	7%	29%	37%	33%
FAST	12	93.1%	19%	23%	30%	28%
Star/BRIEF	12	92.01%	60%	72%	67%	72%
FAST/BRIEF	11	89%	57%	70%	61%	71%
Star/FAST	13	91.9%	42%	64%	49%	64%

Table 5: Indoor results where the set of representative images of a node consists of only the *mean image*.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	18	80.4%	17%	43%	30%	45%
SIFT	18	83.2%	19%	21%	35%	27%
SURF	16	84.1%	1%	18%	20%	22%
FAST	12	93.1%	14%	31%	20%	33%
Star/BRIEF	12	92.01%	51%	72%	61%	71%
FAST/BRIEF	11	89%	44%	70%	53%	70%
Star/FAST	13	91.9%	35%	60%	38%	59%

Table 6: Indoor results where the set of representative images of a node consists of the *mean image*, and the first and the last images of the node.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	18	80.45%	18%	38%	26%	40%
SIFT	18	83.2%	22%	22%	48%	32%
SURF	16	84.1%	7%	18%	28%	23%
FAST	12	93.1%	14%	32%	31%	24%
Star/BRIEF	12	92.01%	51%	65%	57%	65%
FAST/BRIEF	11	89%	51%	71%	55%	70%
Star/FAST	13	91.9%	26%	37%	40%	39%

Table 7: Indoor results where the set of representative images of a node consists of the five most representative images of the node.

	t_d			t_s	t_q		
	Mean	Min	Max		Mean	Min	Max
Orientation Histograms	140	87	184	3	163	155	242
SIFT	430	192	697	19026	4073	749	4699
SURF	127	63	232	3069	248	57	314
FAST	4	0	16	203	38	18	109
Star/BRIEF	11	5	18	1420	85	25	147
FAST/BRIEF	5	1	17	237	39	15	105
Star/FAST	22	12	41	991	91	41	117

Table 8: Computation times for the indoor sequence in milliseconds.

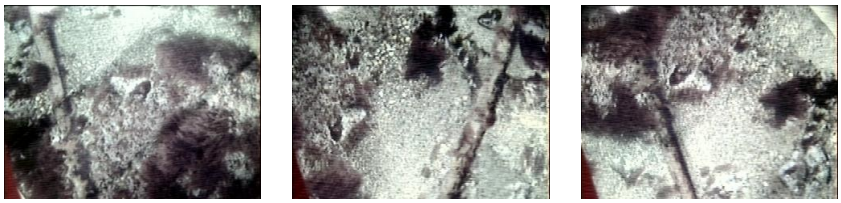


Figure 16: Examples of underwater images used.

6.2. Indoor Environment

The indoor sequence has been recorded during a walk along several corridors of Anselm Turmeda building at UIB. It comprises 832 images of 360×280 pixels. The sequence has been visually split into 9 locations to create the ground truth graph. Each location has a number of images between 7 and 80. As before, nodes with less than 7 images were deleted. At the end, the construction of the topological map does not discard more than 30 images in total.

In this case, the results are shown in tables 5 - 8. As we see, FAST and Star using BRIEF as descriptor have the best SSR values. It is due to the fact that they generate less locations, favoring the localization in the same node of each ground truth image. However, Star/BRIEF should be discarded again because of its bad performance in recognition tasks. FAST and Star result to be the fastest image detectors in this scenario.

We can observe a situation similar to the previous one regarding the time to construct a topological map. Orientation Histograms, as global descriptor, is the fastest followed by FAST. SIFT and SURF turn out to be once again very slow.

The error recognition rate tests show us that the best image descriptor for these environments is SURF, followed by FAST. Again, if the computation time is a requirement, SURF may not be a solution. FAST has less accuracy but it is faster than the other image descriptors and it could be used instead. SIFT is once more not practical for real-time applications. BRIEF is not a good options to be used as the image descriptor. As in the previous section, the best results have been obtained when using the mean, the first and the last views as representatives of each node.

Briefly, our tests show that SURF and FAST are reasonable options to use in indoor environments for recognizing scenes. SURF is the most faithful descriptor in these tasks. However, it is very slow and FAST may be a solution sacrificing accuracy. FAST has also the best SSR creating the topological map and it could be a real possibility. Star is a very fast detector that could be used with another descriptor different from BRIEF or FAST. SIFT has a good performance but it can not be used in scenarios with real-time needs.

6.3. Underwater Environment

The underwater sequence comes from a water tank for testing underwater cable tracking algorithms. There are 250 images of 320×240 pixels that can be divided visually into 5 locations. Each location has between 7 and 50 images. The topological map-building process did not discard more than 25 views.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	5	90.9%	15%	25%	24%	24%
SIFT	8	89.8%	12%	6%	42%	30%
SURF	5	92.1%	0%	9%	6%	8%
FAST	6	91.8%	9%	29%	24%	30%
Star/BRIEF	9	89.1%	81%	77%	74%	78%
FAST/BRIEF	10	72.1%	62%	76%	68%	80%
Star/FAST	12	78.1%	28%	34%	32%	40%

Table 9: Underwater results where the set of representative images of a node consists of only the *mean image*.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	5	90.9%	0%	20%	6%	20%
SIFT	8	89.8%	0%	3%	8%	18%
SURF	5	92.1%	0%	0%	4%	0%
FAST	6	91.8%	0%	16%	12%	16%
Star/BRIEF	9	89.1%	21%	65%	34%	68%
FAST/BRIEF	10	72.1%	37%	71%	46%	72%
Star/FAST	12	78.1%	9%	36%	22%	34%

Table 10: Underwater results where the set of representative images of a node consists of the *mean image*, and the first and the last images of the node.

Descriptor	L	S	R_1	R_2	R_3	R_4
Orientation Histograms	5	90.9%	6%	27%	10%	26%
SIFT	8	89.8%	7%	5%	7%	21%
SURF	5	92.1%	0%	9%	6%	8%
FAST	6	91.8%	6%	25%	18%	26%
Star/BRIEF	9	89.1%	41%	67%	51%	84%
FAST/BRIEF	10	72.1%	52%	86%	68%	89%
Star/FAST	12	78.1%	21%	17%	30%	20%

Table 11: Underwater results where the set of representative images of a node consists of the five most representative images of the node.

	t_d			t_s	t_q		
	Mean	Min	Max		Mean	Min	Max
Orientation Histograms	93	70	122	1	112	104	119
SIFT	630	531	805	24706	29129	2608	40103
SURF	135	113	180	1003	226	208	290
FAST	24	10	66	82	37	26	71
Star/BRIEF	7	4	20	406	70	28	95
FAST/BRIEF	46	12	194	316	90	45	143
Star/FAST	17	10	30	307	84	39	103

Table 12: Computation times for the underwater sequence in milliseconds

These environments are characterized by the absence of defined structures, such as doors, corridors, windows, etc. Due to this reason, it is more difficult to find certain kind of key points, like corners. In this case, a detector less focused on corners, such as SURF, is needed, and FAST results to be not the best option.

The results for this scenario are shown in tables 9 - 12. Again, the SSR test gives us similar values for all the descriptors, except for the FAST/BRIEF case. Specifically, SURF has the best rate building the topological map. The main drawback of this last option is the computation time for describing and querying images. However, its accuracy in the recognition phase is good enough to consider it as a real solution. Star shows shorter times for this sequence, because the FAST algorithm can not determine corners easily and it needs to execute the complete detection process.

We do not observe differences in constructing the topological map with regard to previous cases. On the one hand, Orientation Histograms have the best performance due to the simplicity of the distance calculation. On the other hand, SIFT is the slowest solution. It is important to remark that this is not a critical task and the use of this descriptor depends on the application needs.

The best recognition rates have been obtained again selecting the *mean image*, the first and the last images as representatives of each node. SIFT and SURF are the options with better values for these tasks. As we comment previously, the first one is not usable when there are real-time constraints associated to the task. FAST is the fastest image descriptor for recognizing scenes. However, it exhibits high values of error rates for these cases. Orientation histograms show reasonable values for a short computation time. Again, BRIEF has been shown as a bad solution for scene localization.

In short, in our experiments we have observed that the best option for underwater scenes is SURF. SIFT has also low recognition rates, but it is very slow processing and comparing descriptors. Star is a very fast detector that can not be combined with BRIEF or FAST as descriptors. FAST gets in trouble detecting features in underwater images and it is not a good solution for this case. Another solution could be orientation histograms, due to its good ratio between recognition rates and computation times.

6.4. Results Summary

To conclude, in this section we show graphically the previous results in order to help the reader understand the values obtained. For each environment, we represent only the best case regarding the method to obtain the representative images. As has been shown along the previous sections, this happens for the case in which the set of representative images of a node consists of the *mean image*, and the first and the last images of the location.

Figures 17, 18 and 19 illustrate the results for the outdoor environment. As we saw previously, orientation histograms and FAST are the image descriptors with the best RER values. Any combination with BRIEF has always the worst performance, according to our experiments. Regarding the computation times, FAST and Star are the fastest in description, orientation histograms in segmentation and FAST for queries. The SSR shows very similar values for all cases.

The results for the indoor scenario are shown in figures 20, 21 and 22. In this case, the best values for the RER tests correspond to SURF. Again, BRIEF is not a

good solution as a descriptor. FAST turns out to be the descriptor with the shortest computational times and SSR values.

Figures 23, 24 and 25 show the results for the underwater sequence. Again, SURF presents the best performance in all RER tests. However, computation times are very high, complicating its use in applications with real-time needs. Star is very fast in description tasks, orientation histograms is the best in segmentation and FAST is the fastest descriptor for recognizing scenes. The results for the SSR are very similar again.

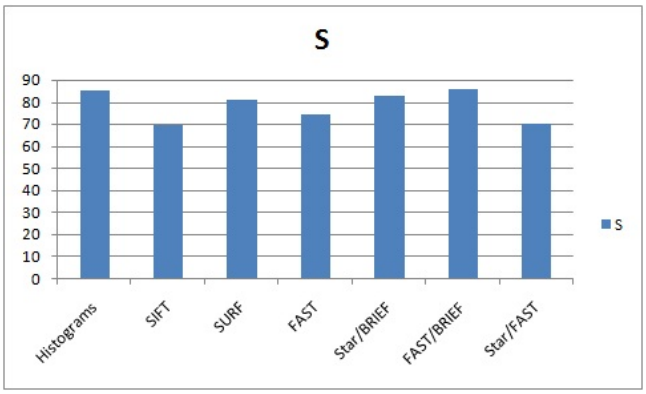


Figure 17: SSR results for the outdoor environment.

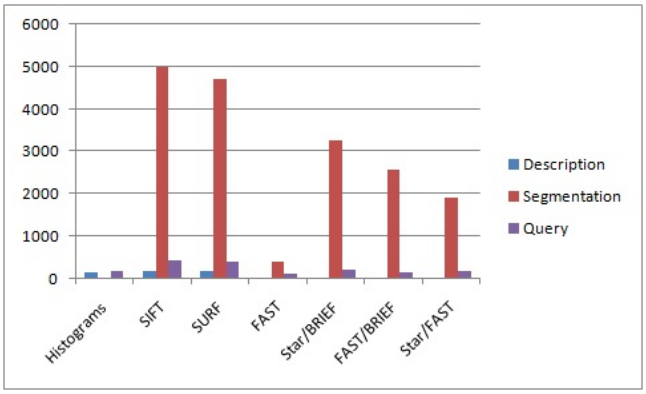


Figure 18: Computation time results for the outdoor environment in milliseconds.

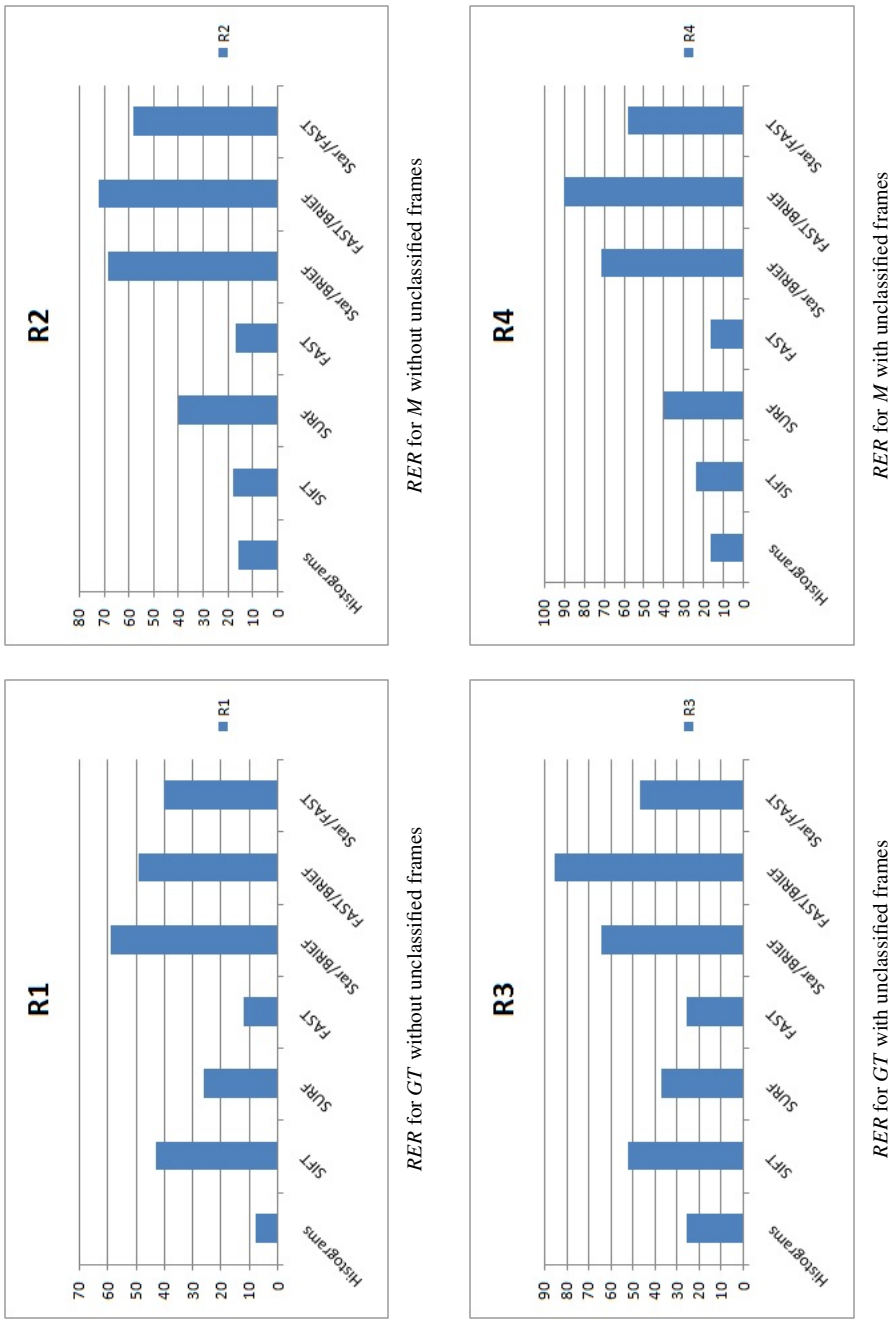


Figure 19: SSR results for the outdoor environment.

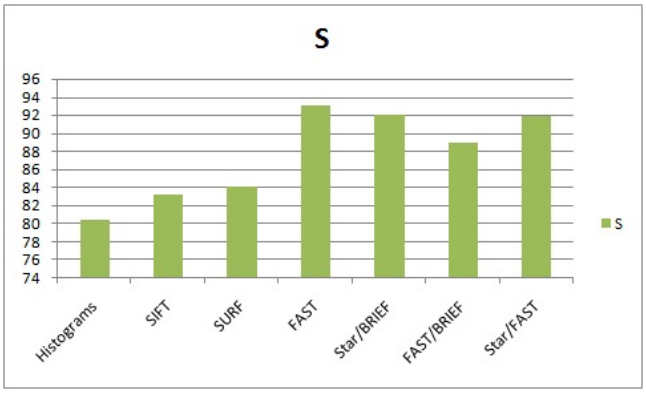


Figure 20: SSR results for the indoor environment.

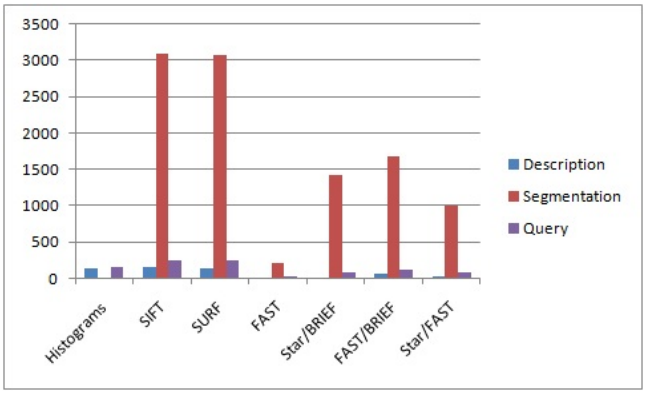
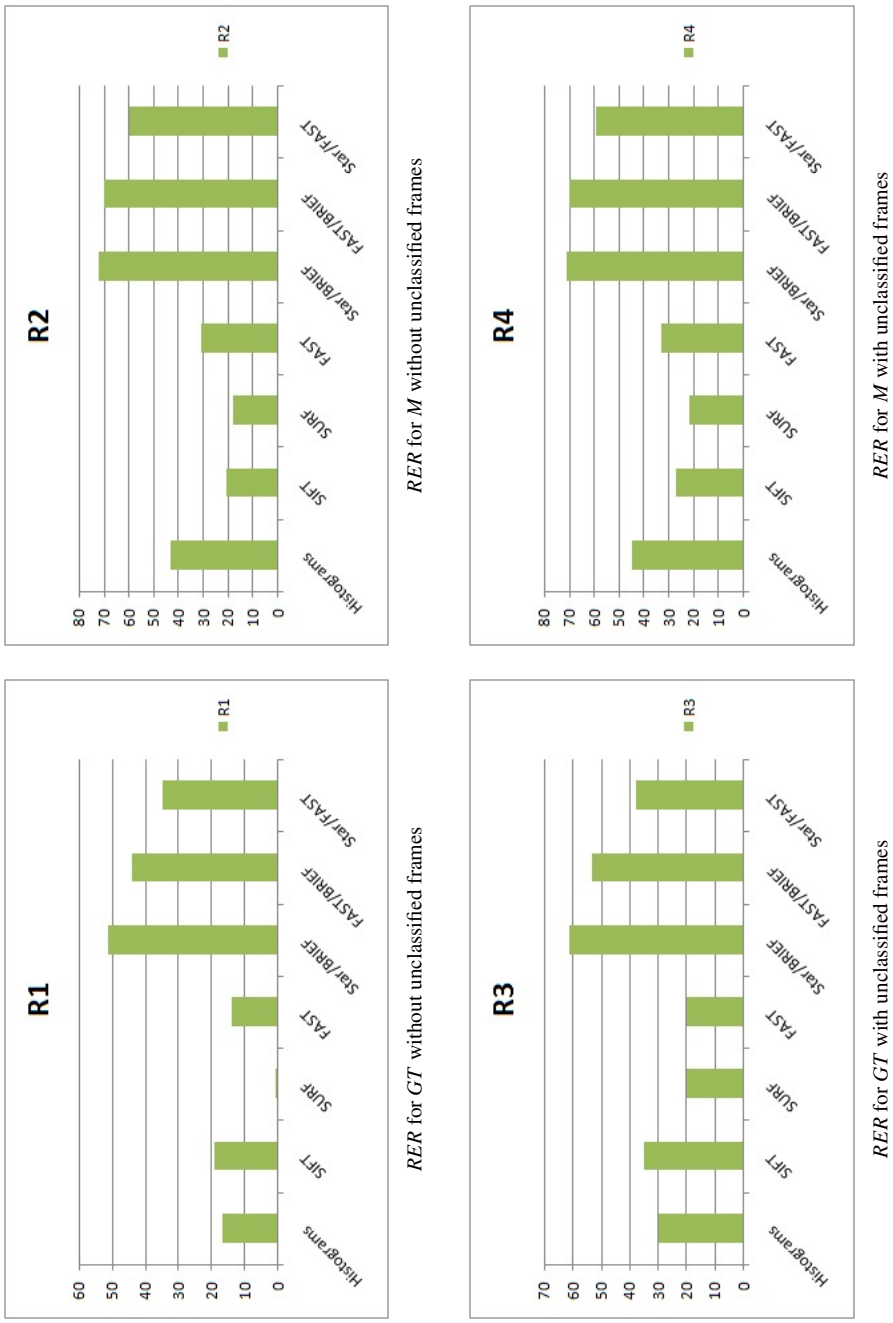


Figure 21: Computation time results for the indoor environment in milliseconds.



RER for M without unclassified frames

RER for M with unclassified frames

RER for GT without unclassified frames

RER for GT with unclassified frames

Figure 22: SSR results for the indoor environment.

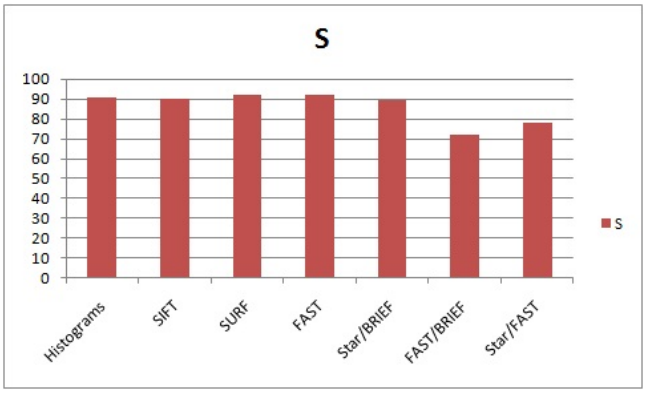


Figure 23: SSR results for the underwater environment.

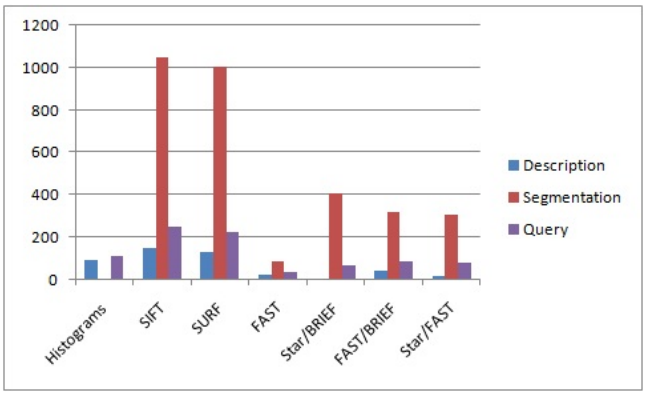


Figure 24: Computation time results for the underwater environment in milliseconds.

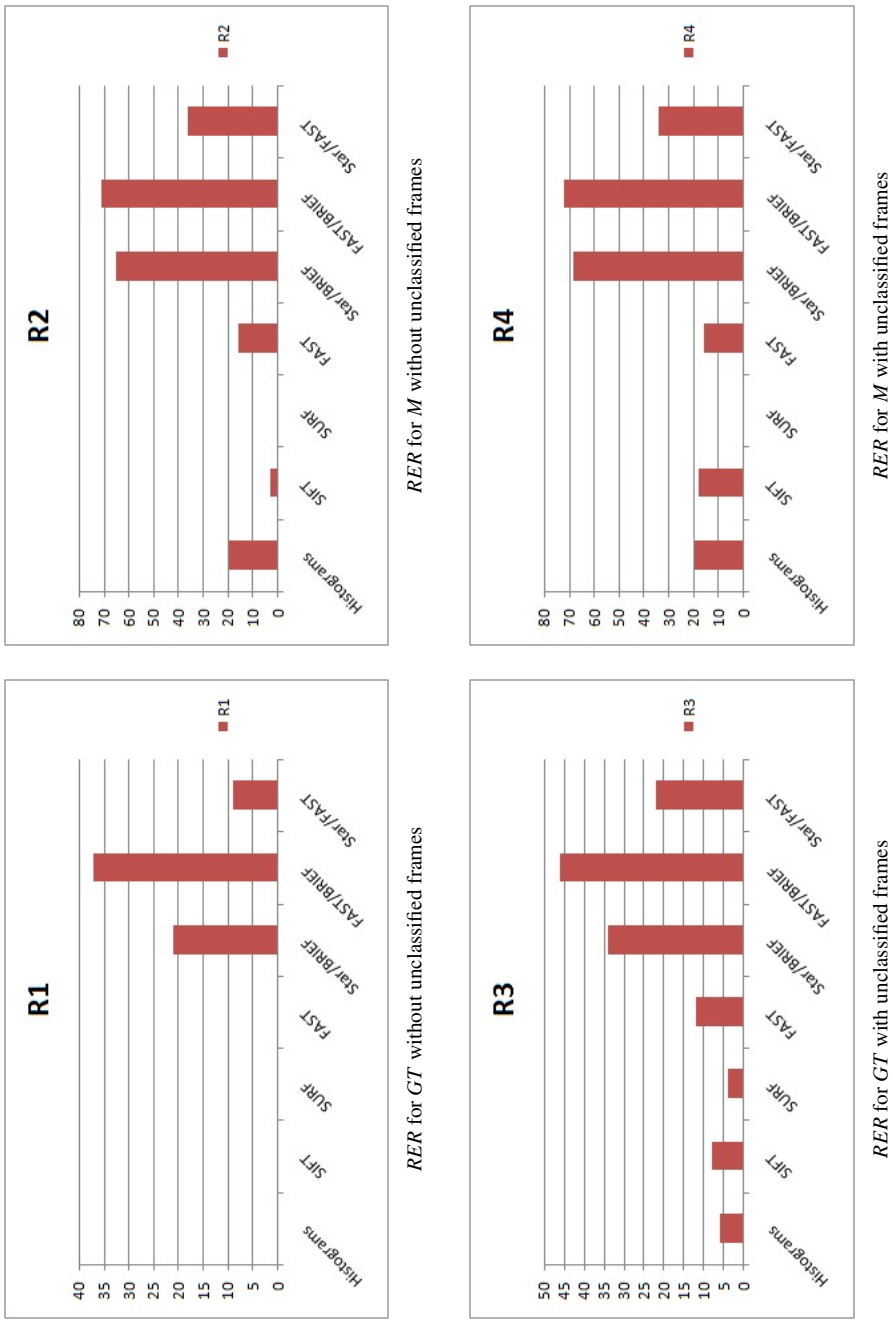


Figure 25: SSR results for the underwater environment.

7. Conclusions and Future Work

In this report we have performed a comparison between different image descriptors for creating topological maps and recognizing previously visited locations. Several measures to evaluate the performance of these descriptors have been proposed and a set of tests has been executed for different environments, such as indoors, outdoors and underwater, in order to check the quality of each one in these scenarios.

Our topological model consists in a segmentation of the input sequence, where every segment is a node of the topological map. Each of these nodes corresponds, thus, to a subset of the original sequence. A subset of these images are in turn selected as representatives of the location and they are used to recognize scenes. We have proposed different methods to choose these representatives. The map is created comparing the descriptors of consecutive images and adding new nodes if the distance between these images is larger than a threshold. The resulting graph is refined deleting locations that have less images than a certain value.

We have defined the *successful segmentation rate* (SSR) concept, which measures the similarity between the models we have adopted for representing the robot environment. When compared with a ground truth map, it can thus be used to check the quality of our representation of the environment. On the other hand, the *recognition error rate* (RER) can be used to evaluate the performance of the descriptors in recognition tasks.

A framework to perform these tests has also been developed and it enables us to add and check new image descriptors easily.

According to our experiments, FAST has been shown as a good option for many environments, resulting in very low computation times in most cases. It is the best option to use outdoors, due to the nature of the images in this scenario, where exist well-defined corners. The main drawback of FAST is that it is not invariant to scale. In this regard, there are different options that are not affected by changes in scale, such as SIFT and SURF. The implementation of SIFT we have made use of leads to very high computation times, thus it is not suitable for applications with real-time needs. SURF has been shown as a good solution for indoor and underwater scenarios, reducing the computation requirements of SIFT.

Regarding Star, it is a very fast key point detector based on *CenSurE*. It has shown a good performance in many scenarios, in most cases better than SIFT or SURF. The combinations detector/descriptor involving Star we have considered in this work have not resulted in good performance.

Finally, orientation histograms have resulted in a good option for outdoor environments and it is the fastest descriptor creating the topological maps.

Our future research will focus on experimenting with other descriptors, improving the performance of some of them, such as Star, that could be a good solution with a correct combination. New image descriptors will also be explored. Another research line to be investigated is the concept of *Bag of Visual Words*, used by some authors in recent years. It could be useful to incorporate this kind of image description to our framework and measure its performance in different environments.

The elected strategy to refine the topological map used in this work is also an early approach. Images in nodes could be added to the closest locations, for instance. We

will research on several ways to improve the representation of our environments for scene recognition.

Currently, our approach does not detect previously visited locations and, when it occurs, a new node is created in the map. It is our intention to explore different solutions to include these loops into the environment representation.

This research work is the beginning of a more complex system, where these maps will be used for navigation tasks. We have started with the experimental evaluation of different image descriptors to represent an environment. We also plan to incorporate temporal information to the model. Next steps will be oriented to navigation path planning over the topological map.

Acknowledgements

I am very thankful to Francesc Bonin-Font and Xisco Bonnin-Pascual for their support, to Maria for her patience and always being by my side and to my supervisor Alberto Ortiz, for his guidance and help.

References

[1] M. Agrawal, K. Konolige, and M. Blas. Censure: Center surround extremas for realtime feature detection and matching. *Computer Vision–ECCV 2008*, pages 102–115, 2008.

[2] B. Bacca, J. Salvi, J. Batlle, and X. Cufi. Appearance-based mapping and localisation using feature stability histograms. *Electronic Letters*, 2010.

[3] M. Ballesta, A. Gil, O.M. Mozos, and O. Reinoso. Local descriptors for visual SLAM. In *Workshop on Robotics and Mathematics (ROBOMAT07), Portugal*, pages 209–215. Citeseer, 2007.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[5] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: a survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.

[6] F. Bonin-Font, A. Ortiz, and G. Oliver. Experimental assessment of different feature tracking strategies for an IPT-based navigation task. *IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2010.

[7] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3927–3932. IEEE, 2007.

[8] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the National Conference on Artificial Intelligence*, pages 11–18. JOHN WILEY & SONS LTD, 1998.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*, September 2010.

[10] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22. Citeseer, 2004.

[11] M. Cummins. *Probabilistic localization and mapping in appearance space*. PhD thesis, University of Oxford, Oxford, United Kingdom, 2009.

[12] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3872–3877. IEEE, 2007.

[13] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *Robotics and Automation, IEEE Transactions on*, 16(6):890–898, 2002.

[14] J. Kosecka and X. Yang. Location recognition and global localization based on scale-invariant keypoints. In *Statistical Learning in Computer Vision, ECCV 2004 Workshop, Prague, Czech Republic*, pages 49–58. Citeseer, 2004.

[15] J. Kosecka, L. Zhou, P. Barber, and Z. Duric. Qualitative image based localization in indoors environments. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2. IEEE, 2003.

[16] M. Liu, D. Scaramuzza, C. Pradalier, R. Siegwart, and Q. Chen. Scene recognition with omnidirectional vision for topological map using lightweight adaptive descriptors. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 116–121. IEEE, 2009.

[17] D. Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. Published by the IEEE Computer Society, 1999.

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[19] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. *International Journal of Computer Vision*, 2005.

[20] AC. Murillo, JJ. Guerrero, and C. Sagues. Topological and metric robot localization through computer vision techniques. *Unifying Perspectives in Computational and Robot Vision*, pages 113–128, 2008.

[21] E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152(1):47–104, 2004.

[22] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[23] B. Schiele and J. Crowley. Object recognition using multidimensional receptive field histograms. *Computer VisionECCV'96*, pages 610–619, 1996.

[24] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5):530–535, 2002.

[25] S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 226–231. IEEE, 2002.

[26] J. Shen and H. Hu. Visual navigation of a museum guide robot. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 9169–9173. IEEE, 2006.

[27] R. Sim and G. Dudek. Learning environmental features for pose estimation. *Image and Vision Computing*, 19(11):733–739, 2001.

[28] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. *International Conference on Computer Vision*, 2003.

[29] S.M. Smith and J.M. Brady. SUSAN: A new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

[30] D.L. Swets and J.J. Weng. Using discriminant eigenfeatures for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):831–836, 2002.

[31] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press, February 2006.

[32] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[33] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, et al. MINERVA: A second-generation museum tour-guide robot. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 1999–2005. IEEE, 2002.

[34] A. Torralba and P. Sinha. Recognizing indoor scenes. *CBCL-202*, 2001.

[35] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. IEEE International Conference on*, volume 2, pages 1023–1029. IEEE, 2002.

[36] N. Winters and J. Santos-Victor. Omni-directional visual navigation. In *Proceedings of the 7th International Symposium on Intelligent Robotics Systems*, pages 109–118. Citeseer, 1999.

[37] J. Wolf, W. Burgard, and H. Burkhardt. Using an image retrieval system for vision-based mobile robot localization. *Image and Video Retrieval*, pages 108–119, 2002.

[38] Z. Zivkovic, B. Bakker, and B. Krose. Hierarchical map building using visual landmarks and geometric constraints. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2480–2485. IEEE, 2005.