Interval Inspired Approach Based on Temporal Sequence Constraints to Place Recognition

Renata Neuland, Fernanda Rodrigues, Diego Pittol, Luc Jaulin, Renan Maffei, Mariana Kolberg, Edson Prestes

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

Interval inspired approach based on temporal sequence constraints to place recognition

Renata Neuland · Fernanda Rodrigues · Diego Pittol · Luc Jaulin · Renan Maffei · Mariana Kolberg · Edson Prestes

Received: date / Accepted: date

Abstract Place recognition is an essential task in many robotics applications. Recognizing if the robot is crossing an already visited place may be used to improve its localization and map estimation. A place recognition strategy must be as accurate as possible, despite the challenges related to environment dynamicity. It should avoid generating false positives since even a few erroneous matches may be enough to cause the degradation of the SLAM process. We propose a novel approach for place recognition inspired by interval analysis theory. Our approach models the known world as a set of intervals based on the robot's observations. The search to determine whether the current robot location is new or known begins as the robot explores its surroundings. Our approach has three main steps. First, it selects a set of nearest neighbors based on the similarity between the current robot observation and the intervals composing the known world. In the second step, our approach uses temporal con-

R. Neuland (ORCID: 0000-0001-6357-1626)
Institute of Informatics, Federal University of Rio Grande do Sul
Caixa postal 15064 - 91.501-970 Porto Alegre - RS - Brazil
E-mail: reneuland@gmail.com

F. Rodrigues (ORCID: 0000-0001-6404-6647)
E-mail: fcsrodrigues@inf.ufrgs.br

D. Pittol (ORCID: 0000-0003-1698-2427)
E-mail: dpittol@inf.ufrgs.br

R. Maffei (ORCID: 0000-0003-0637-9747)
E-mail: rqmaffei@inf.ufrgs.br

M. Kolberg (ORCID: 0000-0002-8117-4402)
E-mail: mariana.kolberg@inf.ufrgs.br

E. Prestes (ORCID: 0000-0003-3691-4253)
E-mail: prestes@inf.ufrgs.br

L. Jaulin (ORCID: 0000-0002-0938-0615)
Lab-STICC, ENSTA-Bretagne, Office M024, 2 rue François Verny, 29806 Brest - France
E-mail: lucjaulin@gmail.com

straints to select one element of the set. And finally, the third step is to sweep the selected interval looking for the query best match. We evaluate our proposal by dealing with visual place recognition using only image information and demonstrate its effectiveness using some challenging public datasets.

Keywords Robotics · Place Recognition · SLAM · Interval analysis

PACS PACS 89.20.Ff Computer science and technology

Mathematics Subject Classification (2010) MSC 68 Computer Science · MSC 68T40 Robotics

1 Introduction

The visual place recognition problem also called loop closure detection, consists of recognizing previously visited places [4]. The ability of identifying already visited places may improve the robot pose estimation and consequently the map quality. Despite challenges related to environment dynamicity, a place recognition strategy must be as accurate as possible. It should avoid the generation of false positives since few erroneous matches are usually enough to cause the degradation of the Simultaneous Localization and Mapping (SLAM) process.

The use of vision as the central sensing modality has shown exciting results for learning and recognizing places [21]. Nevertheless, challenging aspects such as vehicle movements, lighting, and natural or human-made structural changes still require attention. Figure 1 shows examples of challenging situations; each line presents two images from the same place taken at different instants during a day, showing a drastic illumination change.

Any place recognition method must have an environment representation, which is compared to new incoming data to find matches. The goal is to determine if the current incoming data is from a place previously included in its representation, and if so, which one [18].

Several methods have been proposed over the years to deal with this problem. FAB-MAP [6] and SeqSLAM [21] are popular approaches, considered milestones in the state-of-art due to their results in challenging situations. These methods introduced the use of visual words and sequence analysis to the field.

Interval analysis has also been used as a foundation to create algorithms for the loop closure problem. Interval analysis theory [14] lies in the idea of enclosing numbers in intervals. It uses constraints of the problem to reduce the set of candidate solutions. If a candidate satisfies all the constraints, it is part of the solution. For instance, Simon et al.[26] deals with the loop closure problem using this approach. However, as with other interval methods for robotics problems, such as localization and SLAM, the interval represents a measurement obtained by a ranging sensor anchored in the robot's workspace [12][13][23]. To the best of the authors' knowledge, there is no place recognition approach based on intervals analysis that are not created from range sensors.

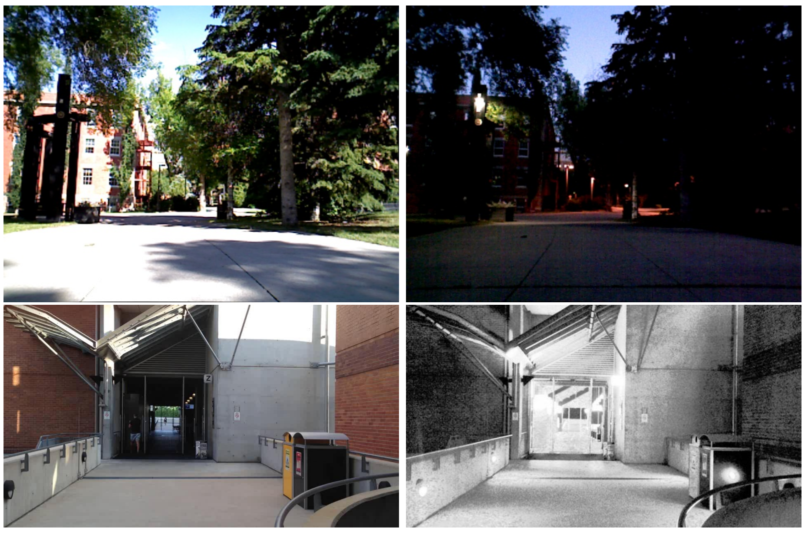


Fig. 1: Examples of pictures from UofA [27] (top) and GPW [10] (bottom) datasets presenting day (left) and evening (right) views of the same place.

In this paper, we propose a novel method for visual place recognition under environmental condition changes using a monocular camera. Our approach is based on interval analysis theory and uses intervals to represent regions of the real-world which are not anchored in the workspace. It creates constraints based on the past searches of the nearest neighbors to reduce the set of possible matching solutions. Our strategy was tested using public datasets and exhibited encouraging results being able to use interval techniques to find matches between sets of images.

The paper is organized as follows. We first present the related work about place recognition methods in Section 2. Section 3 contains our proposal of how to model the known world as a set of intervals. Section 4 presents the visual place recognition method inspired by interval analysis theory using the ideas introduced in Section 3. In Section 5, we evaluate and discuss our method through the analysis of the experiments performed with public datasets. Finally, in Section 6, we conclude and discuss future work.

2 Related work

An essential part of the visual place recognition problem is the scene description. The most common ways to represent a scene are using either a set of local features or the whole image information [18]. Each image can enclose hundreds of local features, which means that describing a scene and finding a match to each feature implies a high computational cost. Besides, using local features under changing conditions can deliver poor results since they are easily missed when images suffer from blur effect. On the other hand, using the whole image information to generate only one descriptor can be faster to describe and search for similar images than using a set of features. However, changes in point-of-view can affect the quality of the global descriptor [18]. It is also possible to use raw images as descriptors [21], once computing the difference between two images is essentially subtracting the corresponding pixels of each other. Yet, this approach can be computationally costly and strongly dependent on the point-of-view. Another kind of approach used to search for features and generate descriptors are the ones based on deep learning [3][17][24]. Using deep learning techniques can be computationally expensive [19]. Thus their suitability may depend on the problems that are being addressed and available resources. Below we discuss some relevant approaches to place recognition in the literature.

FAB-MAP [6] is a probabilistic approach to place recognition based on place appearance. It uses the Bag of Words (BoW) image retrieval system [28] to convert the incoming sensory data into words. It relies on the fact that some words tend to appear and disappear together in specific combinations because they are present in the same objects. However, it needs a training phase to create the BoW vocabulary and does not deal with environmental changes once it is dependent on features similarity. FAB-MAP is suitable for applications at the scale of a few kilometers, but it has a high computational cost. This fact is one of the motivations for the elaboration of a new version called FAB-MAP 2.0 [7], which keeps the essence of its predecessor exploiting its characteristics to reduce computation and memory requirements.

On the other hand, SeqSLAM [21] focuses on environments with perceptual changes, comparing information collected in the same place but taken at different moments during a day, at distinct types of weather conditions, or even at different seasons of the year. In unstructured environments or changing conditions, features may not be reliable enough to provide relevant information to describe precisely a scene. Thus, SeqSLAM considers sequences of images to match places. The authors claim that looking for matches within sequences leads to better results than analyzing only single images.

Fast-SeqSLAM [27] is an enhanced version of the SeqSLAM. The method uses a tree structure to store image descriptors and the nearest neighbor algorithm to speed up the search for matches. Differently from the SeqSLAM that uses image subtraction, it uses HOG descriptors [8], which apply distribution of directions of gradients to represent an image. HOG has been used to describe features or, as shown in Fast-SeqSLAM, whole images. In comparison to SeqSLAM, Fast-SeqSLAM reduces the computational cost keeping the same results.

Talbot et al.[30] proposed the OpenSeqSLAM2.0, an open-source toolbox for visual place recognition based on the SeqSLAM. It uses the same high-level structure presented in the SeqSLAM method but enables variations to some steps of the process. Thus, making it possible to alter some of the configurations of the method.

FAB-MAP and SeqSLAM are the basis for most loop closure methods and have been used for baseline comparison. A place recognition framework to deal with viewpoint changes is proposed by Maffra et al. [19]. It uses an adapted BoW to create a visual vocabulary based on binary features. The method works in parallel to a vision-based SLAM/odometry system and takes advantage of the collected features and 3D information of the environment. Their results are compared with the FAB-MAP 2.0 and DBoW2 [9] methods.

Also inspired by FAB-MAP, a loop closure detection based on BoW with binary features was presented by Huishen et al. [11]. The method uses co-visibility graphs to verify loops and considers the temporal consistency of image sequences.

Bai et al. [3] propose a method to perform place recognition under viewpoint and condition changes exploiting the use of images sequences. This method aims to improve the viewpoint invariance of SeqSLAM using deep learning techniques to the image representation process. Naseer et al. [24] proposed a visual localization method to run over long periods, which is based only on monocular image data and exploits sequence information. Their method uses a semi-dense image descriptor associated with global descriptors generated by neural networks to compare images. The solution proposed is also compared with OpenSeqSLAM.

Uy and Lee [1] proposed the PointNetVLAD, a method based on deep neural networks to deal with place recognition using 3D point-cloud information. It is a combination of the deep networks PointNet [25] and NetVLAD [2]. The method can extract global descriptors from 3D point clouds and use end-to-end training. They also propose functions to reach more discriminative and generalizable global descriptors.

Merrill and Huang [20] introduced an unsupervised deep neural network architecture for loop closure detection. The authors propose to address some recurrent problems associated with the use of convolutional neural networks, like taking too long to extract features, slow querying, and use of training based on a large amount of labeled data. The method learns the scene's geometry represented by the HOG descriptor, which compresses the image information preserving salient features and homography.

In [31], Tsintotas et al. presented a loop closure detection method based on visual words, which converts images descriptors into words, clustering the images in places according to their similarity. At query time, the method also converts the images into visual words and searches for a matching candidate place calculated from a probabilistic function. The next step is to use a voting system to find the match at the chosen place. Furthermore, their method applies geometrical and temporal checks on the matching pair aiming for a performance increasing.

Vysotska and Stachniss [32] proposed a method for visual place recognition that deals with seasonal changes, different weather conditions, and illumination changes. It relies on sequences and can localize the robot in a map composed of multiple images sequences. The method uses a combination of image features that came from a layer of convolutional neural network with a graph-based search.

In cases where computational resources are constrained, either there is not enough data available for training, or in combination with challenging environmental conditions, techniques based on deep learning may not be the most suitable option [30]. Despite the emergence of several techniques based on deep learning, as mentioned before, the traditional visual place recognition techniques are still relevant to the field.

As mentioned previously, interval analysis is another type of approach that can be used to deal with the loop closure problem. An interval method to verify the existence of loops along the uncertain trajectory of a robot is presented by Rohou et al. [26]. More than detecting the occurrence of possible loops the method verifies if a loop occurs whatever the uncertainties associated with the trajectory. It exploits only proprioceptive measurements for proving the existence of loops, which is demonstrated by experimenting using autonomous underwater vehicles. The method uses the idea that classical intervals of reals can be extended to trajectories by means of tubes. It also uses the measurements from common sensors to build a tube from this data by computing a piecewise linear interpolation. So, the interval represents a measurement anchored in the robot's workspace, which can pose a limitation when no ranging sensor is used. As we will see later, our proposal does not have the requirement of representing the environment regions anchored in the real world.

3 Novel interval-based modeling to incoming data

This section presents concepts and operations from interval analysis as well as our proposal of how modeling the known world using interval theory.

3.1 Interval background

Intervals are useful to represent uncertain values. They may enclose the uncertainties related to values in a compact representation, which may be manipulated using Interval Arithmetic to propagate and reduce the associated uncertainties. Some interval operations necessary for the understanding of the method proposed in Section 4 are presented. Other interval operations can be seen in [14].

Considering the operator $\diamond \in \{+, -\}$ and the intervals $[x]$, $[y] \in \mathbb{IR}$, where $\mathbb{IR}$ is the set of all possible intervals of real numbers.

$$[x] \diamond [y] \overset{\text{def}}{=} \{x \diamond y \mid x \in [x],\ y \in [y]\}.$$

For instance,

$$[x^-; x^+] + [y^-; y^+] = [x^- + y^-; x^+ + y^+].$$

It is important to highlight that a scalar number can be treated as a punctual interval. For instance, number 2 can be represented by the interval $[2; 2]$.

The intersection operation gains special attention in the interval analysis context [22]. Given two intervals $[x], [y] \in \mathbb{IR}$, the intersection between them is defined by

$$[x] \cap [y] \overset{\text{def}}{=} \{x \mid x \in [x] \text{ and } x \in [y]\}.$$

This operation helps to find inconsistencies in sets of data. Besides, it narrows the interval results, reducing uncertainties. Interval approaches are known for generating guaranteed solution sets, which is true when the problem model and error bounds are known. However, occasionally it is not possible to predict all noises of the system, and when this happens, the constraints lead us to an empty solution. In this case, we can accept that not all intervals have an overlap and adjust the number of intervals intersecting according to the needs. This is called q-relaxed intersection [5]. Given a set of intervals

$$[x]_i \in \mathbb{IR},\ 1 \leq i \leq n$$

the q-relaxed intersection denoted by

$$\bigcap^{\{q\}} [x]_i \quad \text{for } 1 \leq i \leq n,$$

results in a set of all $x \in \mathbb{R}$ which belong to all $[x]_i$, except q at most.

3.2 Seeing the known world through an interval-based perspective

In our approach, we consider a robot in a workspace $\mathbb{W}$ modeled as a Euclidian space $\mathbb{R}^3$. The robot moves along a path defined as a continuous function

$$\tau : [0, 1] \rightarrow \mathbb{W},$$

with $\tau(0)$ representing the robot's initial configuration, and $\tau(1)$ representing the goal, or last configuration of the path assumed by the robot [15][16].

The robot goes through the path collecting information about its surroundings, which is represented by the observations set $\mathbb{U}$. We may sort the set $\mathbb{U}$ by the order in which the observations appeared along the path, like a timestamp. Thus, all observations are mapped to a natural number using the following function

$$o : \mathbb{U} \rightarrow \mathbb{N}.$$

Using this function we define the set of labels $\mathbb{O}$ as

$$\mathbb{O} = \big\{o(u) \mid u \in \mathbb{U}\big\},$$

and the o-distance between two observation as

$$\rho : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{N}.$$

For instance, given the observations u, $v \in \mathbb{U}$, the o-distance between them is

$$\rho(u, v) = |o(u) - o(v)|.$$

Note that the o-distance is related to the order that the robot collected the information; it does not consider the information contained in the observation itself.

When we do consider the information that observations contain, we can measure the similarity between them, which represents how much alike two observations are. The similarity function is defined by

$$\alpha : \mathbb{U} \times \mathbb{U} \to [0, 1],$$

where α represents a generic similarity function that depends on the kind of information used. When working with laser or sonar is common to use the sum of absolute differences. If the observations are images, a common approach is to use the hamming distance between descriptors to compute the similarity.

The definitions of o-distance and similarity analyzed together may indicate that:

– the observations $u, v \in \mathbb{U}$ are from contiguous points of the path, if

$$\alpha(u, v) \geq \theta_\alpha^0 \quad \wedge \quad \rho(u, v) \leq \theta_\rho^0,$$

where θ_α^0 and θ_ρ^0 are predefined thresholds used to indicate minimum similarity and maximum o-distance between observations collected in contiguous points, respectively.

– the observation $u \in \mathbb{U}$ was collected during a revisit when there is at least one $v \in \mathbb{U}$ where

$$o(u) > o(v) \ \wedge \ \alpha(u, v) \geq \theta_\alpha^0 \ \wedge \ \ \rho(u, v) \geq \theta_\rho^1,$$

given that θ_ρ^1 is a predefined threshold used to indicate the minimum o-distance between observations during a revisit.

– the robot is not moving, or it is in some symmetric region at the moment it collects a set of observations $\mathbb{U}' \subseteq \mathbb{U}$, that for all pairs $u, v \in \mathbb{U}'$ the following condition holds

$$\rho(u, v) < |\mathbb{U}'| \quad \wedge \quad \alpha(u, v) \geq \theta_\alpha^0,$$

where the order labels of the elements of $\mathbb{U}$ are inherited by $\mathbb{U}'$.

Observations collected in contiguous points of the path usually share a large amount of information. Thus, we can join them to represent a more significant region, creating a simplified version of the environment. Analyzing the collected information based on the o-distance and similarity functions, we can represent portions of the path through intervals.

Given two observations $u, v \in \mathbb{U}$, where $u \neq v$, there is a path between the positions where u and v were collected. Even in the case of u and v have been collected consecutively, there is a multitude of possible observations between these points that were not collected and are not in $\mathbb{U}$. However, an interval can still represent this part of the world. Considering, for instance, the interval $[x]$:

$$[x] = [o(u); o(v)].$$

Setting up an interval, according to the previous definitions, shall respect some conditions as enumerated as follows. Each interval $[x]$ has an anchor element $A_{[x]}$ used to define whether an observation is in $[x]$ or not. $A_{[x]}$ may be, for instance, the first element of the interval. Thus, $o(u) \in [x]$ if:

1. $u \in \mathbb{U}$;
2. $\alpha(u, A_{[x]}) \geq \theta^1_\alpha$, i.e. the similarity measure between $A_{[x]}$ and u must be higher than a threshold θ^1_α;
3. $\rho(u, A_{[x]}) \leq \theta^2_\rho$, i.e. the o-distance measure between $A_{[x]}$ and u must be smaller than a threshold θ^2_ρ;
4. the elements of $\mathbb{U}$, which have order labels between $o(A_{[x]})$ and $o(u)$ are in $[x]$.

After creating a new interval, it is necessary to represent the information enclosed by it. So, each interval $[x]$ has a global representation $G_{[x]}$ of all observations it contains. $G_{[x]}$ can be equal to $A_{[x]}$. However, we propose a different structure to be used as set of intervals global representation in the next section. Table 1 summarizes the thresholds presented in this session.

Table 1: Thresholds

Symbol	Meaning
θ^0_α	indicates the minimum similarity between two observations collected in contiguous points
θ^1_α	indicates the minimum similarity between an interval anchor and all element of its interval
θ^0_ρ	indicates the maximum o-distance between two observations to be considered as contiguous points
θ^1_ρ	indicates the minimum o-distance between two observations to be considered as a revisit
θ^2_ρ	indicates the maximum o-distance between an interval anchor and all elements of its interval

4 Interval inspired approach for place recognition

We propose a novel approach to deal with the place recognition problem using interval analysis theory. The presented scenario uses images as observations and is based on two sets of images - a reference and a query - from two traversals of the same environment at different times and under different conditions [21]. Our method uses these sets to identify matches and determines if the robot is crossing an already visited place.

4.1 Dealing with images

First, we need to deal with the images collected by the robot. The following definitions about how our method deals with the images are valid to both reference and query sets. When using images as the main source of information it is common to apply description methods to generate a more compact representation. Among the available methods, we adopted the Local Difference Binary (LDB) descriptor [33], which was developed to be more robust, more distinctive, and faster to compute than other state-of-art binary descriptors. LDB can be used to describe feature patches or whole images, and we have chosen it due to its low memory requirements and fast comparison process based on Hamming distance. However, our method may support any binary descriptor.

LDB method divides the area of interest into a grid and computes the intensity and the gradient of each grid's cells. After that, it performs binary tests comparing the intensity and gradients of the cells among each other. The method uses multiple granularities of grids, capturing the scene structure with different levels of details. The final descriptor is composed of the concatenation of the descriptors from all level grids.

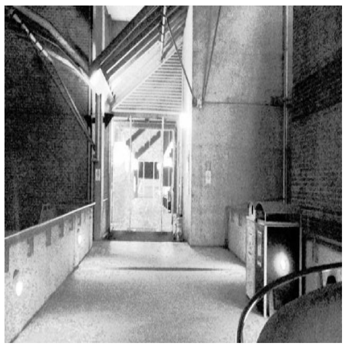
We opted to divide the images into four parts, where each quadrant generates one descriptor to minimize problems related to partial occlusions. Figure 2 shows the steps to generate an image descriptor. First, the robot collects an image, Figure 2a, after, the image is split in four patches, one to each quadrant, Figure 2b, and for each patch a LDB descriptor of 346 bits is computed, Figure 2c. Finally, we concatenate the descriptors in one single descriptor that represents the whole image, Figure 2d.

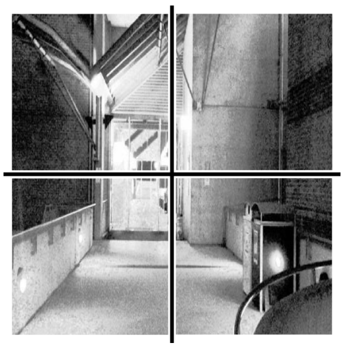
4.2 Modeling the known world as a set of intervals

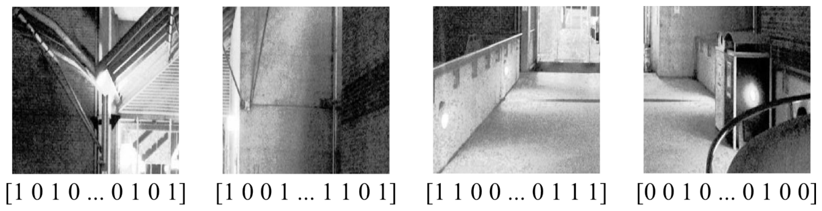
We based our approach on the idea of modeling the known world as a set of intervals, as shown in Section 3. All we know about the environment are the observations collected by the robot, which in our experiments are images. From now on, we are not dealing with raw images, but their descriptors, as presented in Section 4.1.

Our method encloses images in intervals according to the similarity of the environment regions. Figure 3 illustrates the intervals creation, considering the presented images came from the reference set. Each of them has a label from the set $\mathbb{O}$ according to their order. Supposing the images labeled from 0 to 2 respect the four conditions presented in Section 3.2, the method creates the interval $[0; 2]$ to represent that region of the environment. Notice that while the robot is collecting the images, the intervals set is updated.

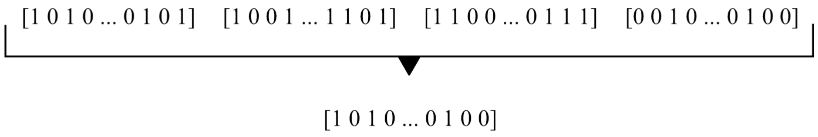
The reference set of images is the robot's known world. Our method represents it as a set of intervals by adapting the **four conditions** presented previously to the given scenario as follows.


(a) Original image


(b) Splitting image on 4 patches



(c) Describing patches



(d) Concatenating the patch descriptors into one

Fig. 2: Steps to describe an image.

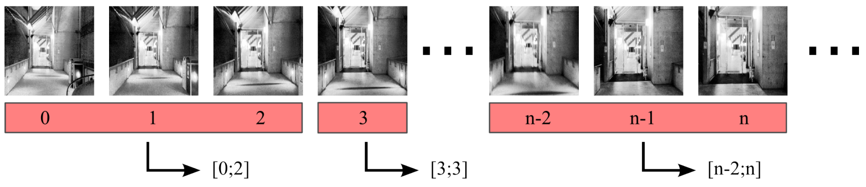


Fig. 3: Generating intervals to represent the robot observations.

In agreement with the **first condition**, all images to be included in an interval are from the reference set. According to the **second condition**, an incoming image can be represented by an interval $[x]$ only when the similarity between it and the interval anchor $A_{[x]}$ is higher than a threshold θ^1_α. This similarity is measured based on the hamming distance between the descriptors generated by the anchor and the incoming image. We consider the first image represented by the interval as $A_{[x]}$, and θ^1_α as a dynamic value computed during the execution. It represents the average similarity between all consecutively captured images, except in case of the difference is not significant. By not significant, we mean similarities higher than 90%, i.e., there is a strong indication that the robot is not moving. The method updates the average similarity during each new iteration and uses this value to represent the θ^1_α threshold.

The images from the reference and query sets are sorted based on the order they were taken. This information helps the method to keep the third and fourth conditions valid. The **third condition** is used to limit the interval width, which can be unlimited by not forcing a threshold θ^2_ρ lower than the reference set size. In practice, this means that to a dataset with 100 images and $\theta^2_\rho = 100$ if the images have high similarity among all of them, our method may create just one interval $[0; 99]$ to represent it. This may happen in long corridors that look the same at all points of the path, which is an open challenge that we currently do not address.

Finally, about the **fourth condition**, each new image is either part of the last created interval or the first image of the new one. After an interval $[x]$ is created, it is necessary to generate a global representation of the information contained by it, called $G_{[x]}$. It is possible to use the anchor $A_{[x]}$ to represent the interval information. However, we propose a new structure that takes into account all elements of the interval.

Figure 4 shows an example of how the new structure is created. The intervals in Figure 4 are only examples to facilitate the computation. Considering an interval $[x] = [7; 10]$ that is associated with four images, thus, to four descriptors. The global representation of $[x]$, denoted by $G_{[x]} = (G^M_{[x]}, G^F_{[x]})$, is a pair of vectors representing majority and frequency. Where the first vector $G^M_{[x]}$ (majority) represents the most frequent values to each position in the interval's descriptors and the second one $G^F_{[x]}$ (frequency) stores the frequency that they appear. In the example, the first position of all descriptor vectors is 0, so the first position of the vectors of $G^M_{[x]}$ and $G^F_{[x]}$ are 0 and 1. Zero because it is the most frequent among the first positions of the descriptor vectors, and one represents a frequency of 100%. In the case of a tie, the method uses 0 associated with a frequency of 50%, as seen in the second position of the vectors.

Interval $[x]$	**Descriptors in** $[x]$	**Global representation** $G_{[x]}$
$[x] = [7; 10]$	[0 1 1 1 ... 1 1 1 0]	$G^M_{[x]} = [0 \quad 0 \quad 1 \quad 0 \quad \cdots \quad 1 \quad 0 \quad 0 \quad 0]$
	[0 0 1 0 ... 1 0 0 0]	$G^F_{[x]} = [1 \quad 0.5 \quad 0.75 \quad 0.75 \quad \cdots \quad 1 \quad 0.75 \quad 0.5 \quad 1]$
	[0 0 1 0 ... 1 0 0 0]	
	[0 1 0 0 ... 1 0 1 0]	

Fig. 4: Representing an interval $[x]$ through $G_{[x]}$.

Based on all these concepts, the method transforms the reference set into a set of intervals. This set is essential to the searching for matches presented in the next section.

4.3 Searching for matches

Streams of incoming images are the only information we have, i.e., no proprioceptive sensors are available. Thus, we match known images (reference set) to the new ones collected during the robot navigation (query set). Even though we do not use or compute the robot orientation, our method can deal with a slight orientation change when searching for matches. Figure 5 presents a flow scheme of the proposed method. Until now, we described the first part (light green), dealing with the reference set of images and creating the set intervals that represent the known world. From now on, we will focus on the second part (light blue) dealing with the query images and the process of finding their correspondent matching on the reference set. Note that the image description process is common to both part, and the set of intervals created in the first part is used in the second part.

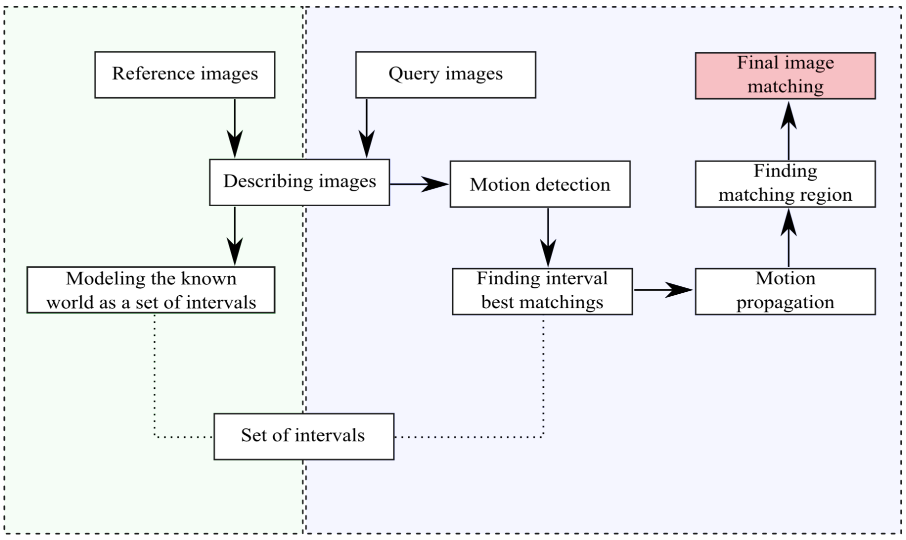


Fig. 5: Method flow scheme.

After receiving and describing an image from the query set the method executes five steps. First, it defines if the robot was moving or not and stores this information for later use. Second, it searches on the set of intervals (presented in Section 4.2) for the best match, which are also stored to be used in the next steps. On the third step, the method uses the computed motion set, based on the first step to propagate the set of best intervals of each iteration.

The fourth step is in charge of finding a matching region to the query image among the propagated intervals. Based on this region, the fifth step defines an image matching from the reference set to the query image. More details about each of these steps are presented as follows.

4.3.1 Step one, robot motion detection

The information we use about the robot motion is binary, i.e., we assume either the robot is moving (1) or not (0). We keep the information about the average similarity between consecutive images from the query set. We say that the robot is not moving if the last four iterations do not present a similarity between the images lower than 95% of the computed average. Summing up, not detecting significant changes on the robot sensing is an indication that the robot is not moving. The method stores the motion information in a set $\mathbb{M}$. The maximum size of this set is defined by w-1. In the case of $|\mathbb{M}| = w - 1$ and new information needs to be stored, the oldest information in $\mathbb{M}$ is discarded.

4.3.2 Step two, finding interval best matches

Our method uses the descriptor of the current query image to search for k interval nearest neighbors. These intervals represent regions of the environment that are most probably to contain the place where the query image was taken, Figure 6 illustrates this process supposing $k = 4$. Given a observation u, the four nearest neighbors more similar to u are $[a]$, $[b]$, $[c]$ and $[d]$.

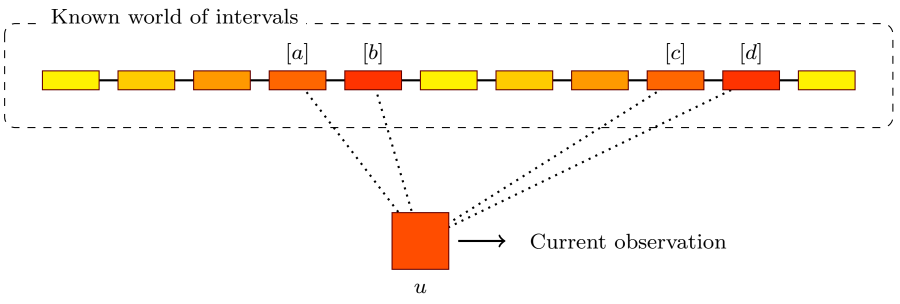


Fig. 6: Selecting the k intervals most similar to the current observation.

The method creates a set $\mathbb{V} = \{v_1, v_2, ..., v_k\}$ of nearest neighbors at each iteration. These sets are kept in a set $\mathbb{Y} = \{\mathbb{V}_1, \mathbb{V}_2, ..., \mathbb{V}_w\}$, which has the maximum size limited by w. If $|\mathbb{Y}| = w$, when new information needs to be stored, the oldest $\mathbb{V} \in \mathbb{Y}$ is discarded. This information will be essential to compute the next steps.

As cited above, the method needs to be able to compare intervals. So we need to define a similarity function β to determine how similar an interval is from an upcoming observation:

$$\beta : \mathbb{U} \times \mathbb{U} \to [0;1].$$

Notice that to perform this comparison it is necessary to transform the upcoming observation into a punctual interval. In this case, the global representation of the created interval will be based on the descriptor of the only observation it contains. Then, the global representation of this interval may be compared with the global representation of any other interval. For our proposal, we define the function β as follows:

$$\beta = \eta \sum_i r(i)$$

where η is a normalizing constant and r is given by:

$$r(i) = \begin{cases} G^F_{[x]}(i) + G^F_{[y]}(i) & \text{if } \; G^M_{[x]}(i) = G^M_{[y]}(i), \\ 2 - G^F_{[x]}(i) - G^F_{[y]}(i) & \text{Otherwise.} \end{cases}$$

An example of how to compute the function β may be seen in Figures 7 and 8. Considering two intervals $[x]$ and $[y]$ represented by $G_{[x]}$ and $G_{[y]}$ respectively. Our method sweeps through the first vector (majority), comparing the values of each index. If the values of an index are equal, the method sums their frequencies' values (Figure 7).

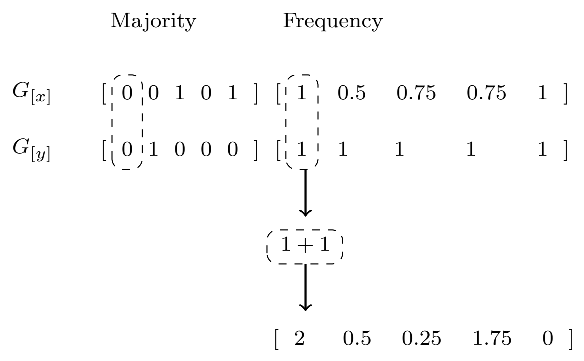


Fig. 7: Computing β. Case 1: entries with the same value.

On the other hand, if the values in an index are different, the method sums the result of 1 minus the frequency (highlighted in Figure 8). So, even though being different, their frequencies contribute to the computation.

Finally, the function sums and normalizes these partial results. Thus, the similarity value between $[x]$ and $[y]$ of the presented example is

$$\frac{2 + 0.5 + 0.25 + 1.75 + 0}{10} = 0.45.$$

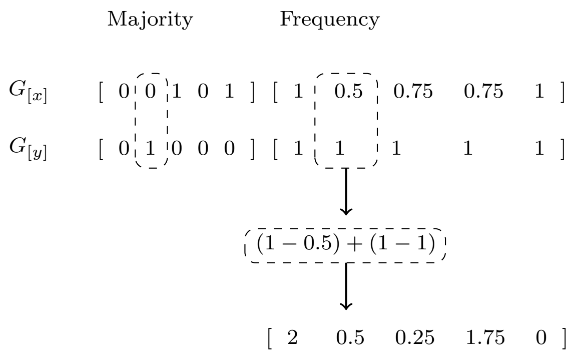

Majority
Frequency
$G_{[x]}$ [0 0 1 0 1] [1 0.5 0.75 0.75 1]
$G_{[y]}$ [0 1 0 0 0] [1 1 1 1 1]
(1 − 0.5) + (1 − 1)
[2 0.5 0.25 1.75 0]

Fig. 8: Computing β. Case 2: entries with different values.

4.3.3 Step three, robot motion propagation

In this step, each interval in $\mathbb{Y}$ is added with a given δ, computed according to values in set $\mathbb{M}$. The resulting intervals are included in a set $\mathbb{C}$ of candidates to the current iteration, as shown below. For all $\mathbb{V}_i \in \mathbb{Y}$,

$$\mathbb{C} = \mathbb{C} \bigcup \Big\{ [x] + \delta_i \mid [x] \in \mathbb{V}_i \Big\}$$

where

$$\delta_i = \sum_{j=i}^{|\mathbb{M}|} m_j.$$

Notice that $[x]$ encloses references to the order/index of the images. Creating the set $\mathbb{C}$ is an important step to incorporate to the intervals the robot motion. Figure 9 presents an example of how to propagate the intervals in $\mathbb{Y} = \{\mathbb{V}_1, \mathbb{V}_2, \mathbb{V}_3, \mathbb{V}_4\}$, considering $w = 4$, $k = 2$ and a set $\mathbb{M} = \{0, 1, 1\}$. At each iteration (represented by gray rectangles), k intervals are selected and propagated according to the robot motion. Iteration $t - 3$ do not consider previous candidates since $w = 4$. But a pair of candidates $\mathbb{V}_1$ is selected. At iteration $t - 2$, no robot motion was detected, and two new candidates are selected. From iteration $t - 2$ to iteration $t - 1$, the robot moves and the previous selected candidates ($\mathbb{V}_1$ and $\mathbb{V}_2$) are moved accordingly. Still at $t - 1$, new candidates are selected. At iteration t, all candidates ($\mathbb{V}_1$, $\mathbb{V}_2$, and $\mathbb{V}_3$) are moved given the detected robot motion, and new candidates are selected. Thus, the method creates the current set $\mathbb{C}$, represented in Figure 10.

4.3.4 Step four, finding a matching interval

Now we need to find the most similar interval to the current image query. Assuming that set $\mathbb{C}$ contains the solution, it is the input to the q-relaxed intersection method presented in Section 3, where the q parameter is the smallest

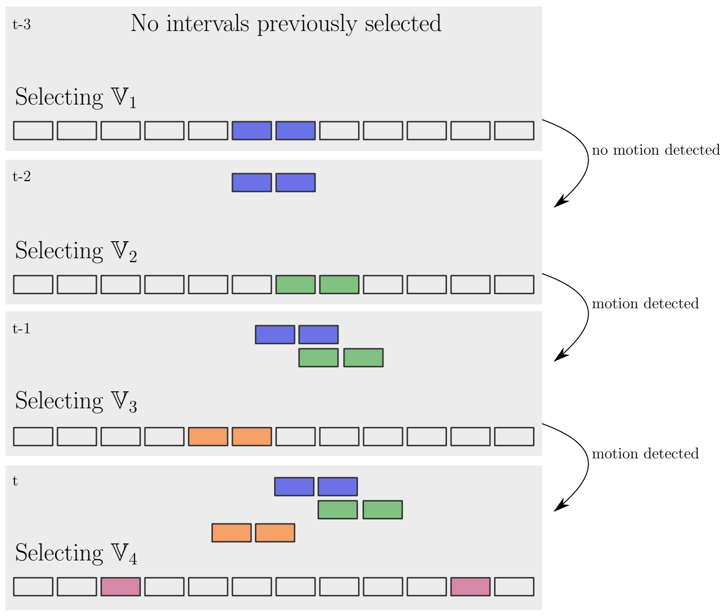


Fig. 9: Intervals propagation to create set $\mathbb{C}$.

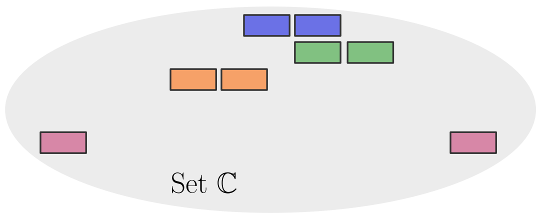


Fig. 10: Intervals in set $\mathbb{C}$ at the end of iteration t.

value that returns a non-empty solution. The result of the relaxed intersection defines the current matching interval.

4.3.5 Step five, finding image matching

After the q-relaxed intersection process, the method has an matching interval that represents a region of the environment. Looking for a more precise result, our method sweeps the images represented by the interval computing the similarity between each of them and the query image using the α function. The weighted average, based on these similarities, defines the final image matching.

5 Experiments

This section presents and discusses the experiments performed. It includes the presentation of the datasets, the methods parametrization, and a comparison between our method and OpenSeqSLAM2.0 [30].

5.1 Datasets

We selected three public datasets which present environment appearance changes and slight changes in point-of-view to evaluate our approach. They also have groundtruth information available, and all images are originally sorted based on the order they were collected.

1. **GPW** – Garden Points Walking dataset [10], from the Queensland University of Technology, Brisbane, Australia. The dataset has three traversals. The images were collected at three different times using a handheld camera. Figure 11 presents samples of each traversal, each column shows a different place, first and second lines show images collected under sunlight, holding the camera in the right and left hand, respectively. The third line shows images collected at night, holding the camera in the right hand. Each traversal contains 200 images. For simplification purposes, we will call the traversals GPW-DR (day-right), GPW-DL (day-left), and GPW-NR (night-right).

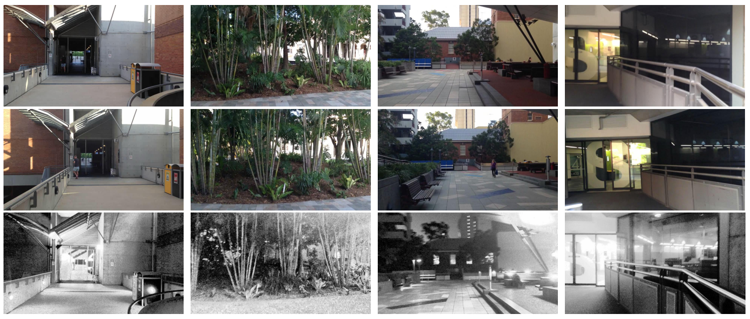


Fig. 11: GPW dataset sample. Traversal GPW-DR on the first line, GPW-DL on the second, and GPW-NR on the third.

2. **UofA** – University of Alberta dataset [27], from the University of Alberta, Edmonton, Canada. The dataset was collected at two different times of the day using a Husky robot. One traversal was made under sunlight and the other during the evening. Thus, we have two sets of 645 images from the same places under different conditions. Samples of the sets are presented in Figure 12, four distinct places (columns) at two different times of the day (lines). For simplification purposes, we will call the sets generated on each traversal UofA-D (day) and UofA-E (evening).

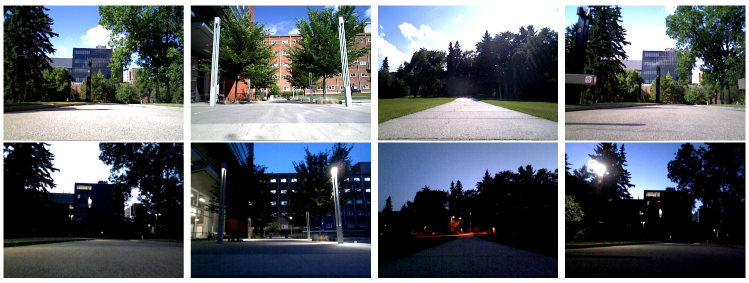


Fig. 12: UofA dataset. Traversal UofA-D on the first line and UofA-E on the second. Each column shows a different place.

3. **NORD** – Nordland dataset [29] contains four videos of a 728km train ride in northern Norway. Each video was taken in a different season, capturing the appearance differences over time from the same point of view. In our experiments, we use a sample of the winter, summer, and spring traversals, 1 frame every 10 seconds removing the parts with no perceptual movement. After sampling, each traversal contains 3052 images. Figure 13 presents some of these images, where each column shows a different place and each line one traversal, winter, spring, and summer.

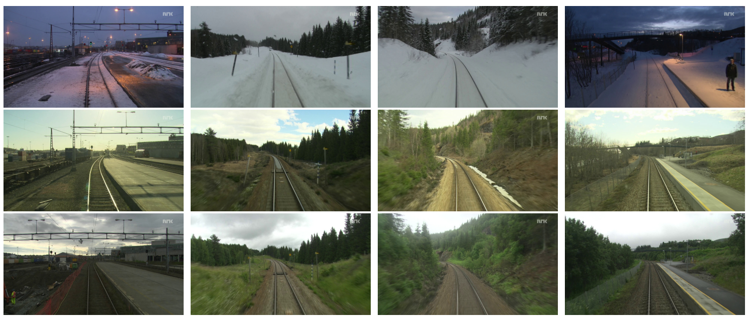
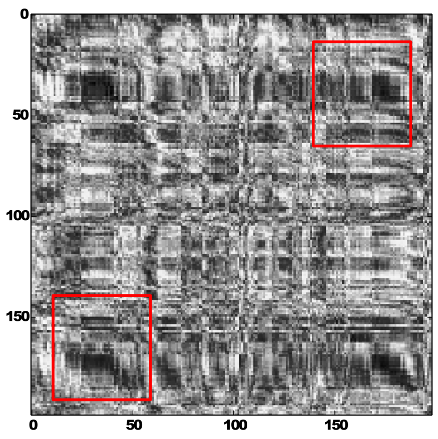


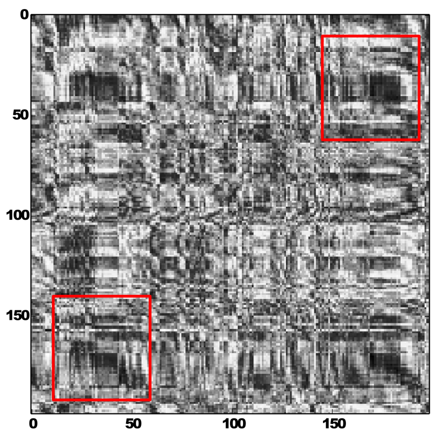
Fig. 13: Nordland dataset sample. Traversal winter on the first line, spring on the second, and summer on the third.

Figure 14 shows the heatmaps comparing the images from the selected datasets using the LDB descriptor and hamming distance. The grayscale represents the similarity level between the traversals, where the darkest points represent higher similarities than the light ones. The groundtruth in all cases is a straight line from the top left to the bottom right.

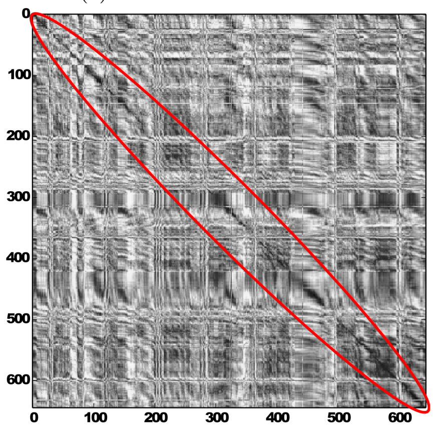
Figure 14a shows the heatmap based on the GPW-DR as a reference and the GPW-NR as a query. Figure 14b presents the heatmap of the distances between the GPW-DL and GPW-NR. In both heatmaps, we can see dark



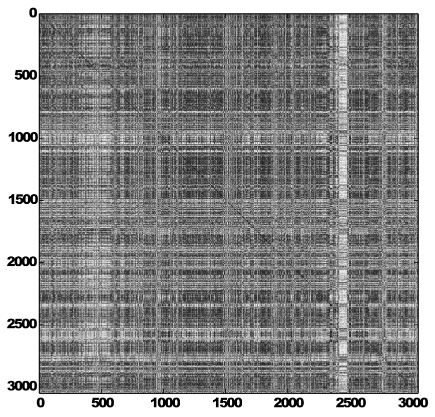
(a) GPW-DR vs. GPW-NR



(b) GPW-DL vs. GPW-NR



(c) UofA-D vs. UofA-E



(d) Nordland - Spring vs. Winter

Fig. 14: Heatmap of descriptors distance.

points on the top-right and bottom-left corners (highlighted in red squares). Despite the high similarity, those regions are not true matches.

The heatmap in Figure 14c presents the distances between the UofA-D and UofA-E images. The correct matching diagonal is easier to perceive here than on the other heatmaps. We can see that it is not simple to detect the correct matches based only on the descriptors' similarities, all maps present points with high similarity out of the expected region. Gaps in the groundtruth diagonal are also common. Even if the images are from approximately the same place and angle, they may present low similarities. Possible causes are the new shadows from illumination changes or partial occlusions due to people and vehicles dynamism on the environment.

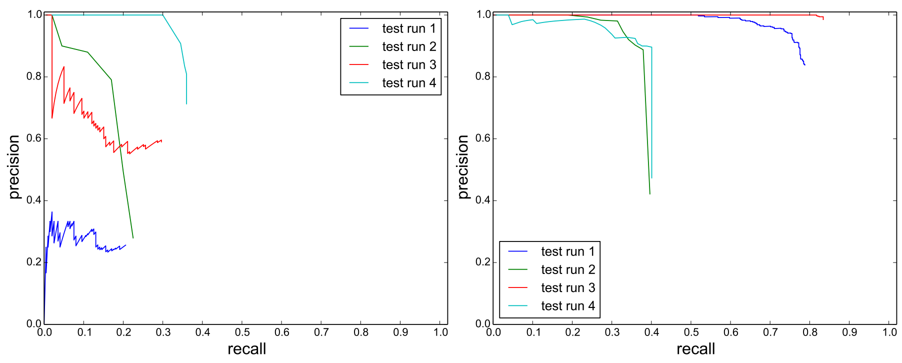
Figure 14d shows the heatmap of spring vs. winter traversals from the NORD dataset. Despite the extreme perceptual differences due to the natural changes between seasons, we can see an almost perfect diagonal where is the groundtruth. However, it is noticeable that lots of regions over the heatmap indicate high similarities in erroneous places.

5.2 Parametrization

We compare the proposed approach to OpenSeqSLAM2.0 [30] results. Usually, the method's parameters have a high impact on the results. Based on this, and trying to find the best results of OpenSeqSLAM2.0 using the aforementioned datasets, we varied some parameters and followed the guidelines presented in the original paper. Their main parameters are R_{window}, d_s, $searchMethod$, and $matchingMethod$. The R_{window} value was fixed around 2% of the traversal size as recommended by the authors, i.e., 13 to the UofA and 4 to GPW and 61 to Nordland. The original paper shows that the options to set the $matchingMethod$ have similar performance, so we fixed it to use the score thresholding method.

We tested some variation of the parameters $searchMethod$ and d_s according to the limits presented on the origin paper. The $searchMethod$ parameter is strongly related to the dataset, we tested the *cone* and the *trajectory-based* search. More information about each one can be seen in [30].

The d_s parameter affects the method performance and may increase the execution time. The authors tested d_s values between 2 and 40, and their results suggested that the higher the d_s values, the better is the produced result. However, using high values as d_s will deprecate the method time performance. Given the number of images of the Nordland dataset traversals, we kept the $d_s = 20$ as suggested by Talbot et al. [30].



(a) GPW-DR vs. GPW-NR (b) UofA-D vs. UofA-E

Fig. 15: OpenSeqSLAM2.0 test runs to define best configurations. Parameters on Table 2.

Figure 15 shows the precision-recall (PR) of the main test runs we performed with OpenSeqSLAM2.0 using GPW and UofA datasets. The difference among them are the *searchMethod* used and the values for the parameter d_s, Table 2 summarizes it. As expected, the higher the d_s, the better were the results. So we use $d_s = 100$ despite the performance dropping. Otherwise, the OpenSeqSLAM2.0 did not generate competitive results to some of the tested datasets. It is worth to notice that the GPW (Figure 15a) obtain better results using the cone search method and the UofA (Figure 15b) using the trajectory search method. Excepts for the GPW-DL vs. GPW-DR, where all precision-recall curves presented pour results and the best one were obtained using the trajectory search method, more information can be seen in the next section.

Table 2: OpenSeqSLAM2.0 test runs parameters from Figure 15.

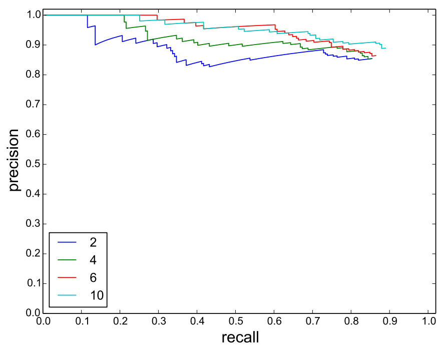
	Search Method	d_s
— test run 1	*trajectory*	40
— test run 2	*cone*	40
— test run 3	*trajectory*	100
— test run 4	*cone*	100

Our method has two main parameters that need to be defined, the number of interval candidates per iteration (k), and the window size (w) of past iterations taken into account to compute the current matching. So, we present as follows some variation of our main parameters and their impact on the obtained result.

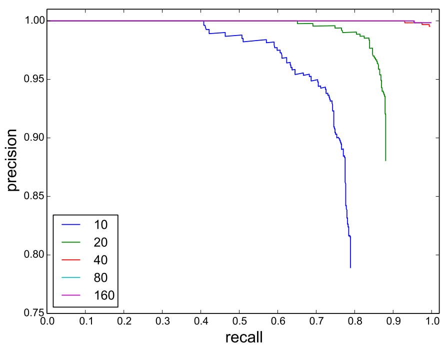
Figure 16a shows the effect in the precision-recall curves of different values for k. The experiments were done using 2, 4, 6, and 10 as values to k. This parameter depends on the quality of the image descriptor. Let's consider one observation and a list of candidates to match with this observation according to their similarity. When we sort this list by the computed similarity score, the best match may not be the correct one. This is why our method considers a set (size k) of best matches and not only the best one. We can see in Figure 16a that the highest k does not present the best result. This happens because computing more intervals also include more uncertainties.

Figure 16b shows the effect in the precision-recall curves of varying w. This parameter defines how many past iterations will be considered to compute the current iteration. The method strategy is to choose intervals based on how many times they were selected as good matching. A sequence of good matches indicates that those are possibly the best match. However, as well as the k parameter, the definition of a high value to w can improve the results, taking into account more iterations, or include more uncertainties into the computation.

The tests using the UofA dataset both day vs. evening and evening vs. day used the same parameters. Despite the light condition changes, there was a minimal change in point-of-view. Thus, both variations were able to use the same values to obtain the results presented in Section 5.3. This situation



(a) k parameter variation using GPW dataset.



(b) w parameter variation using UofA dataset.

Fig. 16: Precision-recall behavior of the proposed method parameter variation.

was different using the GPW dataset. The changes in point-of-view associated with the light conditions seem to require a more expensive combination, raising both the values of past iterations considered and the number of candidates on each iteration.

In general, we observe that using a w value around 100 generated competitive results. It is worth highlighting that our method analyses past iterations. Thus the first iterations are less reliable given the little amount of information collected.

About the k parameter, we note that using only one candidate is not enough. This parameter is strongly affected by the dataset. In the case of similar regions distributed along the robot path, most of them must be represented by some candidate, thus, increasing the chances of choosing the correct matching as fast as possible when a distinguishable region appears.

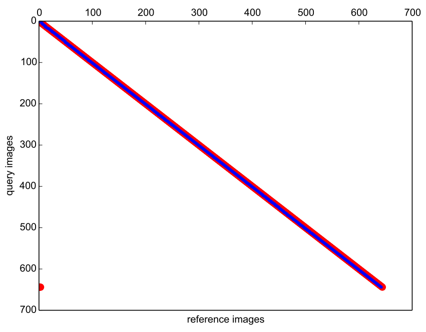
Besides the behavior presented in Figures 16a and 16b, these parameters also affect the method performance. We need to keep in mind that all selected interval during the defined window will be processed. This means kw intervals being processed during each iteration. Thus, we need to consider the time necessary to do it. A promising path to follow in future works is to search for possible adjusts on the values of w and k in run time. Table 3 presents the parameters values used during the experiments.

5.3 Comparison

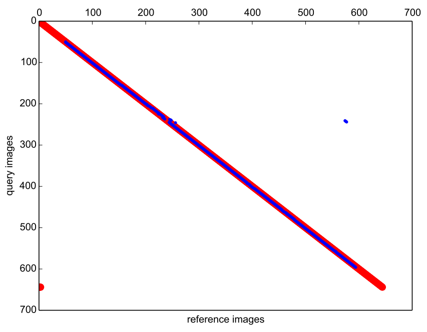
In this section, we present the results obtained by our method and their comparison with the OpenSeqSLAM2.0. The method's parameters are presented in Table 3. Figure 17 shows the matches found using UofA dataset for our method (left) and OpenSeqSLAM2.0 (right). The red dots represent the ground truth, and the blue dots are the selected matches. The dataset has two sets of images from different traversals, which we tested as reference and query using both

Table 3: Parameters.

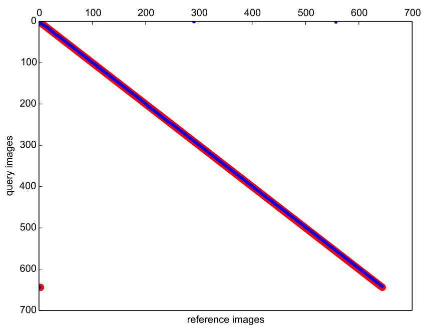
	UofA	**GPW**		**NORD**
		DR vs. NR	DL vs. NR	
		Ours		
w	100	100	200	100
k	6	6	20	6
		OpenSeqSLAM2.0		
R_{window}	13	4	4	61
d_s	100	100	100	20
$searchMethod$	trajectory	cone	trajectory	trajectory
$matchingMethod$	thresholding			



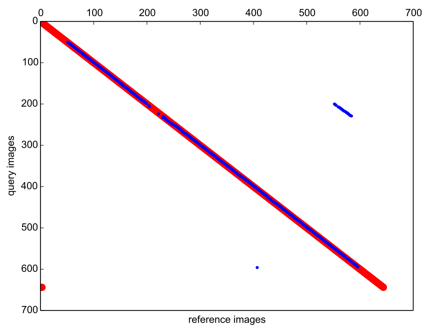
(a) Ours: UofA-D vs. UofA-E



(b) OpenSeqSLAM2.0: UofA-D vs. UofA-E



(c) Ours: UofA-E vs. UofA-D

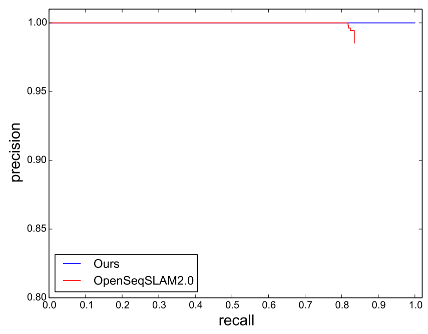


(d) OpenSeqSLAM2.0: UofA-E vs. UofA-D

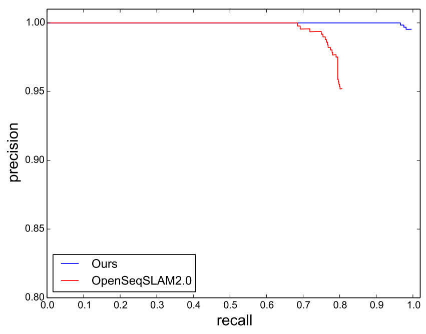
Fig. 17: UofA dataset. Groundtruth and matches are depicted in red and blue, respectively.

methods. The first line presents the day images as reference and the evening images as query, and the opposite can be seen on the second line.

Despite the changes between the first and second trajectories, both methods show fitting results, i.e., an almost perfect line over the groundtruth. We can see a lack of matches at the graphics beginning (top-left) and ending (bottom-right) when using the OpenSeqSLAM2.0 (Figures 17b and 17d). This



(a) UofA-D vs. UofA-E



(b) UofA-E vs. UofA-D

Fig. 18: Precision-recall comparison between OpenSeqSLAM2.0 e our method using UofA dataset.

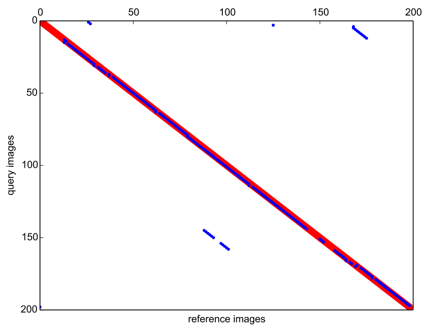
is caused by the way the method uses the d_s parameter. When $d_s = 100$, it is not able to give the first and last 50 matches.

Figure 18 presents the precision-recall curve of our method and OpenSeqSLAM2.0 using the UofA dataset. On the left (Figure 18a) are the curves using UofA-D vs. UofA-E traversals. Our method achieved the perfect score with a 100% of recall at 100% of precision. The OpenSeqSLAM2.0 presented 82% of recall at 100% of precision. On the right (Figure 18b) are presented the curves using UofA-E vs. daUofA-D traversals. OpenSeqSLAM2.0 shows a 68% of recall at 100% of precision, while our method kept the total precision at 96%.

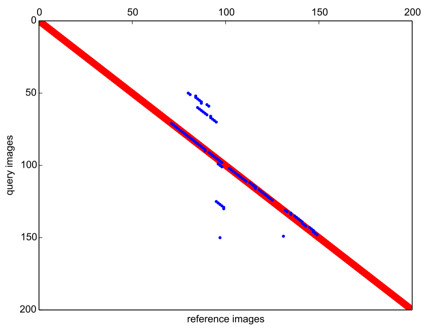
Figure 19 shows the matches obtained by our method (left) and OpenSeqSLAM2.0 (right) using GPW dataset. The first line presents the GPW-DR vs. GPW-NR, and the second line presents the GPW-DL vs. GPW-NR. The red dots represent the groundtruth, and the blue dots are the selected matches. GPW is a challenging dataset, one of the sets was taken using the night mode generating images with differences such as color scale, shadows, and light reflection. Besides, using the GPW-DL vs. GPW-NR we also deal with changes in point-of-view. Considering Figure 19, we can see that our method found more matches closer to the groundtruth than OpenSeqSLAM2.0.

The precision-recall curves using GPW dataset are presented in Figure 20. Both methods obtained worst results than the results using the UofA dataset, this dataset proved to be more challenging. Considering the GPW-DR vs. GPW-NR our method presented 30% of recall at 100% of precision, the same result obtained by OpenSeqSLAM2.0. However, our method kept the precision higher than 85% until the end. Using the GPW-DL vs. GPW-NR our method presented 23% of recall at 100% of precision, OpenSeqSLAM2.0 achieves its higher peak at 45% of precision.

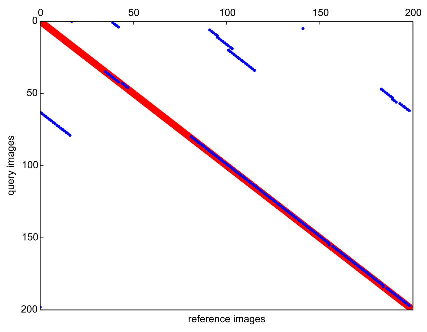
Figure 21 presents the matches of the experiment using the Nordland dataset. Groundtruth and matches are depicted in red and blue, respectively. In the first line, we can see the results using the spring and winter traversals.



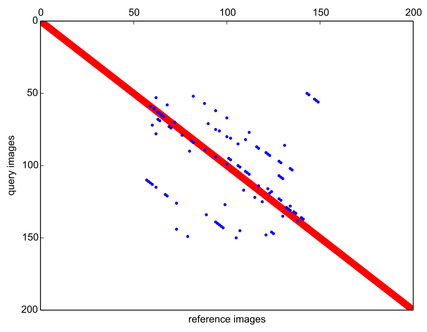
(a) Ours: DR vs. NR



(b) OpenSeqSLAM2.0: DR vs. NR

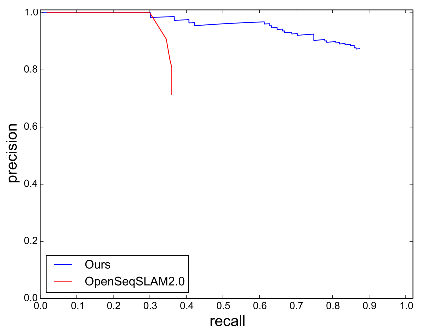


(c) Ours: DL vs. NR

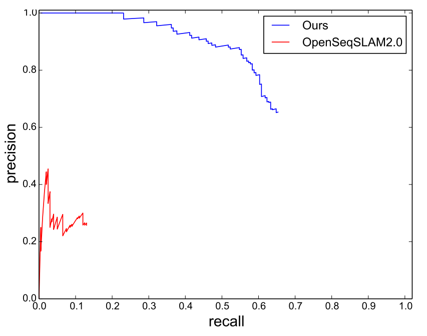


(d) OpenSeqSLAM2.0: DL vs. NR

Fig. 19: GPW dataset. Groundtruth and matches are depicted in red and blue, respectively.



(a) GPW-DR vs. GPW-NR



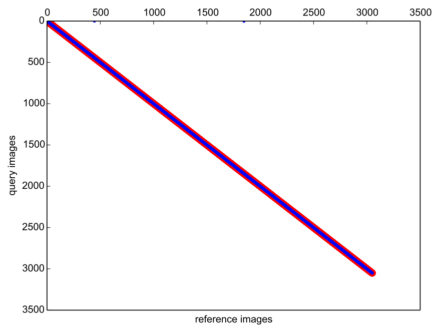
(b) GPW-DL vs. GPW-NR

Fig. 20: Precision-recall comparison between OpenSeqSLAM2.0 e our method using GPW dataset.

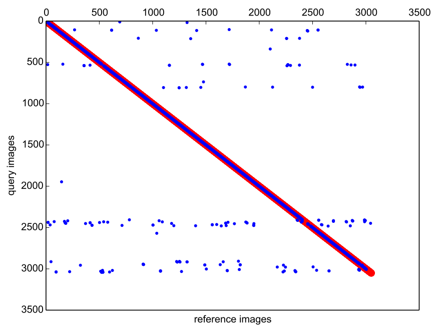
And in the second line are the summer and winter traversals. Both methods are most of the time following the groundtruth. Despite some incorrect matches, the general result is promising, as we can see on the PR curves in Figure 22, where both experiments presented PR curves with high precision values.

Figure 22a presents the PR curves of our method and OpenSeqSLAM2.0 using the spring vs. winter traversals. Our method shows a 90% of recall at 100% of precision and keeps more than 99% of precision until the end. OpenSeqSLAM2.0 shows 20% of recall at 100% of precision, keeping the precision higher than 99% until 79% of recall too.

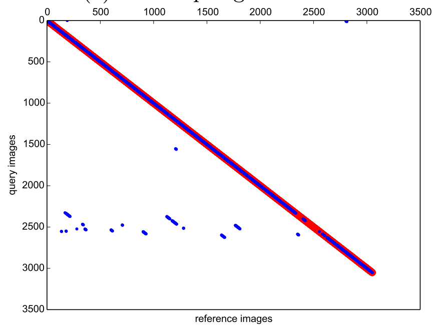
The summer vs. winter PR curves, in Figure 22b, shows our method reaching 52% of recall at 100% of precision, but keeping the precision higher than 99% until 79% of recall. Besides, it was never below 90% of precision. In comparison, the OpenSeqSLAM2.0 shows a 50% of recall at 100% of precision and keeps more than 99% of precision until 85% of recall.



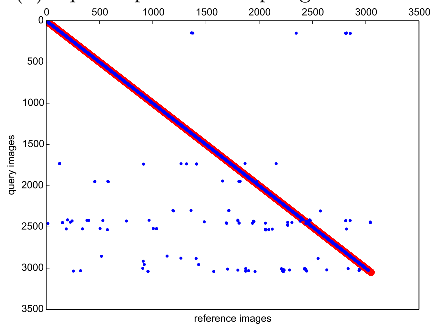
(a) Ours: Spring vs. Winter



(b) OpenSeqSLAM2.0: Spring vs. Winter

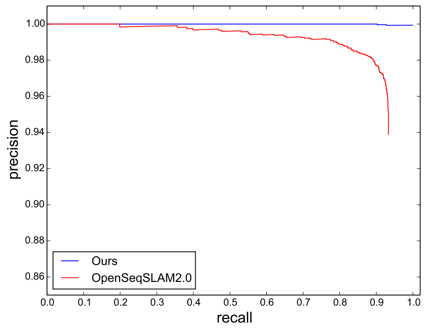


(c) Ours: Summer vs. Winter

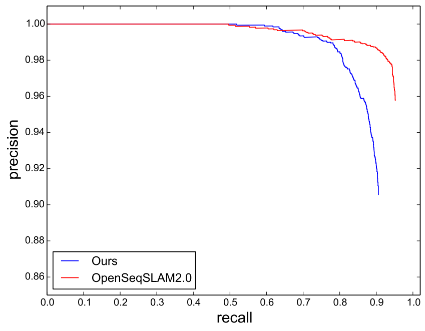


(d) OpenSeqSLAM2.0: Summer vs. Winter

Fig. 21: Nordland dataset. Groundtruth and matches are depicted in red and blue, respectively.



(a) Spring vs. Winter



(b) Summer vs. Winter

Fig. 22: Precision-recall comparison between OpenSeqSLAM2.0 e our method using Nordland dataset.

5.4 Computational cost

The computational cost of visual place recognition methods may limit their suitability for some applications. The main issue of our approach in this regard is to define the size of the set of intervals to be processed in each iteration. This size is defined by the input parameters k and w discussed on Section 5.2, i.e., kw.

Table 4 presents a high level comparison of the time spent to run the different configurations presented on Section 5.2. The computer used to perform all experiments was a laptop with 8GRAM and an Intel Core i7 processor. About our method, the time presented includes the process of loading the precomputed images descriptors. It does not take into account the image loading process.

We use the OpenSeqSLAM2.0 public code and include a timer starting exactly before calling the $OpenSeqSLAMRun()$ and stopping the timer right after to define the time spent by the method. It does not include the loading configuration. Computing the difference matrix and the processes associated with it are huge time-consuming for OpenSeqSLAM. As aforementioned, we use a considerably higher d_s parameter to improve the results. Thus, a dropping performance is expected.

Table 4: Approximate time spent running (mm:ss)

	GPW-DR vs. GPW-NR	GPW-DL vs. GPW-NR	UofA	Nordland
Ours	00:03	00:04	00:22	12:18
OpenSeqSLAM2.0	00:52	00:41	15:25	31:14

Taking everything into account, we can see that the proposed approach is fast, considering the values in Table 4, and presents competitive results as shown in the experiments on Section 5.3.

6 Conclusions

We propose a visual place recognition method inspired by interval theory using a monocular camera. The already existing interval approaches for the robotics problem uses intervals to represent a measurement anchored in the robot workspace. The novelty of our method is to model the known world as a set of intervals not anchored in the workspace. Apart from that, it relies on a relaxed set of constraints based on the search of nearest neighbors from past iterations.

The method shows a high success rate finding matches even in datasets with significant perceptual changes. We optimize the search for matches by computing many steps based purely on the interval information, i.e., using the order in which the observations were taken, without image comparison. Furthermore, we presented a different way to use LDB as a global descriptor, which enhanced the image retrieval rate.

The tested datasets provide one image per second, and our method is capable of producing a matching result in much less time in an average computer (Intel Core i7 - 8GiB RAM). Our approach obtained, in few seconds, $\approx 30\%$ of recall at 100% of precision, keeping the precision higher than 80% until the maximum recall using a challenging dataset GPW-DR vs. GPW-NR. Besides, in a more favorable dataset (UofA) it was able to provide a 100% recall at 100% of precision.

Nordland dataset presents the chance of performing a long term visual place recognition. It is a widely diffused dataset that contains significant visual changes among its traversals. Our method was able to achieve 90% of recall at 100% of precision and keeps more than 99% of precision until the maximum recall using the spring vs. winter traversals.

Besides, our approach does not use information corresponding to the future of the current query to compute a match. These indicate the potential for online operation tests.

Also, as future work, we intend to improve the motion estimation to work with datasets with high variability in speed. Another enhancement would be to calibrate the method parameters during the execution, using the accumulated information to define the amount of processing necessary to match each query.

Acknowledgements This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and CNPq.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Angelina Uy, M., Hee Lee, G.: Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
2. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: Cnn architecture for weakly supervised place recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
3. Bai, D., Wang, C., Zhang, B., Yi, X., Yang, X.: Sequence searching with cnn features for robust and fast visual place recognition. Computers & Graphics **70**, 270–280 (2018). DOI 10.1016/j.cag.2017.07.019. URL `https://doi.org/10.1016/j.cag.2017.07.019`
4. Bampis, L., Amanatiadis, A., Gasteratos, A.: Fast loop-closure detection using visual-word-vectors from image sequences. The International Journal of Robotics Research p. 0278364917740639 (2017)
5. Brefort, Q., Jaulin, L., Ceberio, M., Kreinovich, V.: Towards fast and reliable localization of an underwater object: an interval approach. Journal of Uncertain Systems **9** (2015)
6. Cummins, M., Newman, P.: Fab-map: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research **27**(6), 647–665 (2008)
7. Cummins, M., Newman, P.: Appearance-only slam at large scale with fab-map 2.0. The International Journal of Robotics Research **30**(9), 1100–1123 (2011)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition (CVPR), Conference on, vol. 1, pp. 886–893. IEEE, IEEE (2005). DOI 10.1109/cvpr.2005.177. URL `https://doi.org/10.1109/cvpr.2005.177`
9. Galvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics **28**(5), 1188–1197 (2012)
10. Glover, A.: Datasets (2014). Data retrieved from Robotics@QUT, `https://wiki.qut.edu.au/display/cyphy/Datasets`
11. Huishen, Z., Ling, X., Huan, Y., Liujun, W.: An improved bag of words method for appearance based visual loop closure detection. In: 2018 Chinese Control And Decision Conference (CCDC), pp. 5682–5687. IEEE (2018). DOI 10.1109/CCDC.2018.8408123. URL `https://doi.org/10.1109/CCDC.2018.8408123`
12. Jaulin, L.: Range-only slam with occupancy maps: A set-membership approach. IEEE Transactions on Robotics **27**(5), 1004–1010 (2011)
13. Jaulin, L.: Range-only slam with indistinguishable landmarks; a constraint programming approach. Constraints **21**(4), 557–576 (2016)
14. Jaulin, L., Kieffer, M., Didrit, O., Walter, É.: Applied Interval Analysis. Springer, London (2001). DOI 10.1007/978-1-4471-0249-6. URL `https://doi.org/10.1007/978-1-4471-0249-6`
15. Latombe, J.C.: Robot Motion Planning. Kluwer Academic, New York (1991)
16. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006). DOI 10.1017/cbo9780511546877. URL `https://doi.org/10.1017/cbo9780511546877`
17. Lin, S., Cheng, R., Wang, K., Yang, K.: Visual localizer: Outdoor localization based on convnet descriptor and global optimization for visually impaired pedestrians. Sensors **18**(8), 2476 (2018). URL `https://doi.org/10.3390/s18082476`
18. Lowry, S., Sünderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P., Milford, M.J.: Visual place recognition: A survey. IEEE Transactions on Robotics **32**(1), 1–19 (2016)
19. Maffra, F., Chen, Z., Chli, M.: Viewpoint-tolerant place recognition combining 2d and 3d information for uav navigation. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2542–2549. IEEE (2018). DOI 10.1109/ICRA.2018.8460786. URL `https://doi.org/10.1109/ICRA.2018.8460786`
20. Merrill, N., Huang, G.: Lightweight unsupervised deep loop closure. arXiv preprint arXiv:1805.07703 (2018)
21. Milford, M.J., Wyeth, G.F.: Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In: Robotics and Automation (ICRA), International Conference on, pp. 1643–1649. IEEE (2012). URL `https://ieeexplore.ieee.org/abstract/document/6224623/`

22. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to interval analysis, vol. 110. Siam, Philadelphia (2009)
23. Mustafa, M., Stancu, A., Delanoue, N., Codres, E.: Guaranteed slam—an interval approach. Robotics and Autonomous Systems **100**, 160–170 (2018)
24. Naseer, T., Burgard, W., Stachniss, C.: Robust visual localization across seasons. IEEE Transactions on Robotics **34**(2), 289–302 (2018). DOI 10.1109/TRO.2017.2788045. URL `https://doi.org/10.1109/TRO.2017.2788045`
25. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
26. Rohou, S., Franek, P., Aubry, C., Jaulin, L.: Proving the existence of loops in robot trajectories. The International Journal of Robotics Research **37**(12), 1500–1516 (2018)
27. Siam, S.M., Zhang, H.: Fast-seqslam: A fast appearance based place recognition algorithm. In: Robotics and Automation (ICRA), International Conference on, pp. 5702–5708. IEEE (2017). URL `https://ieeexplore.ieee.org/abstract/document/7989671/`
28. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Computer Vision (ICCV), International Conference on, p. 1470. IEEE (2003). URL `https://members.loria.fr/MOBerger/Enseignement/Master2/Exposes/sivic03.pdf`
29. Sünderhauf, N., Neubert, P., Protzel, P.: Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. In: Proceedings of Workshop on Long-Term Autonomy, International Conference on Robotics and Automation (ICRA), p. 2013. IEEE (2013). URL `https://nikosuenderhauf.github.io/assets/papers/openseqslam.pdf`
30. Talbot, B., Garg, S., Milford, M.: Openseqslam2. 0: An open source toolbox for visual place recognition under changing conditions. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7758–7765. IEEE (2018)
31. Tsintotas, K.A., Bampis, L., Gasteratos, A.: Assigning visual words to places for loop closure detection. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2018). DOI 10.1109/icra.2018.8461146. URL `https://doi.org/10.1109/icra.2018.8461146`
32. Vysotska, O., Stachniss, C.: Effective visual place recognition using multi-sequence maps. IEEE Robotics and Automation Letters **4**(2), 1730–1736 (2019). DOI 10.1109/lra.2019.2897160. URL `https://doi.org/10.1109/lra.2019.2897160`
33. Yang, X., Cheng, K.T.: LDB: An ultra-fast feature for scalable augmented reality on mobile devices. In: 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 49–57. IEEE (2012). DOI 10.1109/ismar.2012.6402537. URL `https://doi.org/10.1109/ismar.2012.6402537`