Change detection using weighted features for image-based localization

Derner, Erik; Gomez, Clara; Hernandez, Alejandra C.; Barber, Ramon; Babuška, Robert

DOI
10.1016/j.robot.2020.103676

Publication date
2021

Document Version
Accepted author manuscript

Published in
Robotics and Autonomous Systems

Citation (APA)
Derner, E., Gomez, C., Hernandez, A. C., Barber, R., & Babuška, R. (2021). Change detection using weighted features for image-based localization. *Robotics and Autonomous Systems*, *135*, Article 103676. https://doi.org/10.1016/j.robot.2020.103676

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Change Detection Using Weighted Features for Image-Based Localization

Erik Derner[a,b], Clara Gomez[c], Alejandra C. Hernandez[c], Ramon Barber[c], Robert Babuška[a,d]

Corresponding author: Erik Derner, erik.derner@cvut.cz

[a] *Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic*
[b] *Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic*
[c] *Robotics Lab, Department of Systems Engineering and Automation, Carlos III University of Madrid, Spain*
[d] *Cognitive Robotics, Delft University of Technology, The Netherlands*

Abstract

Autonomous mobile robots are becoming increasingly important in many industrial and domestic environments. Dealing with unforeseen situations is a difficult problem that must be tackled to achieve long-term robot autonomy. In vision-based localization and navigation methods, one of the major issues is the scene dynamics. The autonomous operation of the robot may become unreliable if the changes occurring in dynamic environments are not detected and managed. Moving chairs, opening and closing doors or windows, replacing objects and other changes make many conventional methods fail. To deal with these challenges, we present a novel method for change detection based on weighted local visual features. The core idea of the algorithm is to distinguish the valuable information in stable regions of the scene from the potentially misleading information in the regions that are changing. We evaluate the change detection algorithm in a visual localization framework based on feature matching by performing a series of long-term localization experiments in various real-world environments. The results show that the change detection method yields an improvement in the localization accuracy, compared to the baseline method without change detection. In addition, an experimental evaluation on a public long-term localization data set with more than 10 000 images reveals that the proposed method outperforms two alternative localization methods on images recorded several months after the initial mapping.

Keywords: mobile robotics, image-based localization, change detection, long-term autonomy

1. Introduction

Deployment of autonomous mobile robots in industrial and domestic environments is challenging due to the dynamics of these environments. Advanced methods are needed to perform localization and navigation precisely and reliably, despite the changes occurring in the environment.

In this work, we present a novel approach to change detection based on local features and their descriptors. A robot equipped with a camera moves through its environment and detects changes that have occurred with respect to an initial mapping session. Once the robot detects a change, it captures it by decreasing weights assigned to the corresponding features stored in the environment representation. At the same time, stable features that have been detected during subsequent visits to the same place are given higher importance through increasing their weights.

Examples of changes that we consider in our work are moving chairs and items on tables, pictures on computer or TV screens, changing contents of whiteboards and notice boards, opening or closing doors, adjusting blinds in the windows, etc. These changes occur every day in various industrial, domestic, and office environments, as illustrated in Figure 1.

The change detection method can be used for robot localization based on place detection, which can be further used to perform more complex tasks such as navigation. It allows the robot to recognize its surroundings more reliably and therefore to perform these tasks more precisely. The advantages of the proposed method are its short learning time, low demand for computational resources, its data efficiency and the low requirements on the sensor hardware.

Article accepted to *Robotics and Autonomous Systems*. https://doi.org/10.1016/j.robot.2020.103676

Figure 1: Examples of changes that can be detected by the proposed algorithm. Handling these changes appropriately allows for more accurate localization.

The paper is organized as follows. Section 2 presents the related research in the field of change detection and localization in dynamic environments. A baseline visual localization framework is introduced in Section 3 and the proposed change detection method in Section 4, along with its incorporation into the localization framework. Sections 5 and 6 describe the experimental evaluation. The conclusions and future work are given in Section 7.

2. Related Work

Dynamically changing environments present a challenge in most robotic navigation contexts. In order to perform stable localization and path-planning, robots must take into account these changes [1]. Change detection and localization in dynamic environments has attracted the interest of many authors and various approaches have been proposed, as surveyed for example in [2] and [3].

Most change detection algorithms are based on object detection and tracking in long-term operation [4–7]. In [4], a robot patrols an indoor environment and detects movable objects by change detection and temporal reasoning. The objective is to determine how many movable objects are there in the environment and to track their position. The Rao-Blackwellized particle filter and the expectation-maximization algorithm are used to track the objects and to learn the parameters of the environment dynamics. In [5], a service robot is deployed in various indoor environments and a hierarchical map of the environment is maintained that takes into account the changes in the object positions by comparing current object detections to the mapped ones. In [7], the change detection problem is treated through reasoning about observations. Observations are classified considering long-term, short-term, and dynamic features, which correspond to mapped static objects, unmapped static objects, and unmapped dynamic objects, respectively. Short-term features produce local adjustments to the belief about the trajectory of the robot, while long-term features generate global adjustments.

Other works detect changes via correspondences between robot views or images [8–16]. Full RGB-D views are used in [8] to build a map of the robot world. Changes between successive views are computed to discover the objects (moved areas) and to learn them. Similarly, in [10], a Truncated Signed Distance Function (TSDF) grid and a 3D reconstruction of the environment are maintained. New observations are aligned with the previous ones and included in the new reconstruction. The new reconstruction is compared to the previous one in order to identify dynamic

clusters between both reconstructions. Image views are used in [11] to detect changes using Gaussian Mixture Models (GMMs). As GMMs have long computational times, Vertical Surface Normal Histograms provide main plane areas that are discarded in the search for changes. Change detection is accomplished as the difference in the Gaussians generated for two images.

Pointclouds from a LiDAR are compared to an octree-based occupancy map in [12] to obtain a set of changes. Change candidates are computed using the Mahalanobis distance and filtered to eliminate outliers. Authors in [13] proposed a 2D LiDAR-based framework for long-term indoor localization on prior floor plans. The system combines graph-based mapping techniques and Bayes filtering to detect significant changes in the environment. The authors use an ICP-based scan matching to determine the probability that a LiDAR scan related to a trajectory pose corresponds to the currently observable environment. This probability is used to improve the trajectory estimation through the update of the previous nodes.

An approach for mapping and localization dealing with sensor inaccuracies and dynamic environments was presented in [14]. The method based on Extended Kalman Filtering incorporates the measurements of landmark strength in the map. The landmarks between the map and the scene are matched in each step and the landmark strength is updated. Weak landmarks are pruned, while newly observed landmarks are added. A system of visual mapping using an input from a stereo camera was presented in [15]. The method builds and continually updates a metric map of views, represented as a graph. In [16], a method for life-long visual localization using binary sequences from images is proposed. The approach is based on using sequences of images instead of single images for recognizing places. Features are extracted using global LDP descriptors to obtain the binary codes of each image. These binary descriptors are efficiently matched by computing the Hamming distance.

Change detection has also been broadly studied for outdoor environments [17–19]. Structural change detection from street-view images is performed in [17]. Multisensor fusion SLAM, deep deconvolution networks and fast 3D reconstruction are used to determine the changing regions between pairs of images. In [18], a Bayesian filter is proposed to model feature persistence of road and traffic elements. Single-feature and neighboring-feature information are used to detect changes in feature-based maps and estimate feature persistence. Many works have been devoted to overcoming seasonal changes for outdoor environment navigation [20–23]. In [20], HOG features and deep convolutional networks are used to compare and match the newly acquired image with a database of images independently of the weather and seasonal conditions. The approach presented in [21] compares different variants of SIFT and SURF feature detectors in the frame of an appearance-based topological localization on panoramic images capturing seasonal changes. Visual words from local features have been used in [24] and [25] to determine the best candidate image given a query image. Visual words are built from the co-occurrence of SIFT features at each location at different times of the day or year. Spatial words (spatial relations between features) are used to verify candidate visual words.

Only a limited number of works have been devoted to long-term localization based on local features in indoor environments [26–29]. In [26], SURF features are applied to training images in order to learn a new descriptor that is more robust to changes in both indoor and outdoor environments. The work [27] presents an approach based on experiences. An observation is compared with all stored experiences and matched against the most similar one.

The approaches [28] and [29] are inspired by the Atkinson and Shiffrin human memory model [30]. They adopt the concepts of short-term memory (STM) and long-term memory (LTM) to deal with changing environments in long-term localization. In [28], the STM and LTM are represented as finite state machines. Each new feature gets first to the STM and needs to be repetitively detected to be moved to LTM. Features are removed from STM and LTM if they are not detected in successive visits to the same place. In [29], the authors introduce a Feature Stability Histogram (FSH) that registers local feature stability over time through a voting scheme. Both methods employ their own mechanisms to update the respective feature containers. In [28], only LTM is used for matching. In [29], both LTM and STM is used for matching within the FSH and the feature strength is considered. Both approaches use omnidirectional images for experiments, even though the algorithms do not seem to be specifically dependent on that type of images.

Many of the aforementioned algorithms need computationally demanding learning processes and a vast amount of training data [26] or heavy maintenance of 3D map reconstructions [8, 17], while other methods build on spatio-temporal relations [24, 25]. On the contrary, the approach proposed in this paper relies on local feature detection and matching, which allows it to run in real time on low-cost hardware platforms. It does not require any data-hungry, computationally challenging learning stage to build an initial map of the environment first and then continuously update it during the long-term operation of the robot. No further assumptions need to be made regarding periodic repetitiveness of changes, similar speed of the robot following given trajectories, etc.

Among the two related methods [28] and [29], the proposed method is closest to [28] thanks to a comparable relative simplicity of the approach. However, we do not use the short-term and long-term memory concepts. Instead, we assign weights to the features that capture their stability and therefore also their importance. The feature weights are updated proportionally to the similarity of the feature descriptors, which grants a smoother way of capturing the feature significance than [28].

3. Visual Localization Framework

The change detection method proposed in this paper can be used for localization or place detection with various algorithms based on local features. In this section, we present a visual localization framework that will serve as a baseline and that will be later extended with the proposed change detection method.

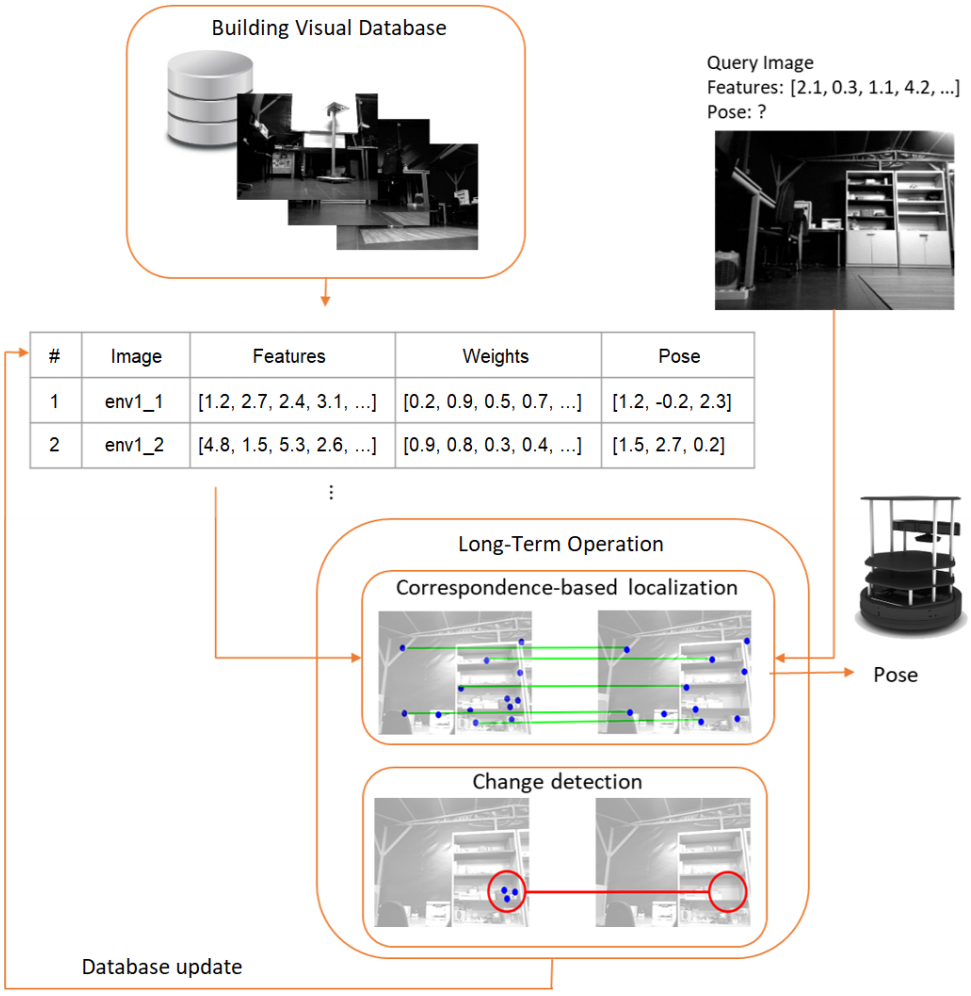


Figure 2: Overview of the visual localization framework. The long-term localization uses a previously built visual database. Query images from the robot are matched against the visual database and the closest match determines the pose of the robot. The change detection module monitors the changes in the matched images and updates the visual database when it detects a difference.

An overview of the method is presented in Figure 2. For now, we assume that the change detection module is not active and we introduce the baseline localization framework.

At first, a robot equipped with a camera builds a discrete representation of the environment in the form of a *visual database*, where images are stored together with the robot pose. Note that a robot featuring a self-localization

capability in a static environment needs to be employed to build the visual database. The location of the robot can be estimated using e.g. wheel encoders, laser rangefinder readings, inertial measurements, visual odometry, or a combination of these sources.

During the long-term deployment, the robot continuously localizes itself in the environment, based solely on the visual information by using feature matching against the previously built visual database. The matching procedure will be described in detail in Section 3.3.

3.1. Building the Visual Database

A mobile robot equipped with a camera moves around the environment and records images together with the corresponding robot poses. The visual database consists of records r_i, $i = 1, \ldots, N$, which have the following structure:

- a grayscale image I_i, captured by the camera mounted on the robot,
- a set of features F_i detected in the image I_i, where $F_i = (f_i^1, f_i^2, \ldots, f_i^{m_i})$,
- a set of descriptors D_i of the features F_i, where $D_i = (d_i^1, d_i^2, ..., d_i^{m_i})$ and the indices $1, 2, \ldots, m_i$ match the descriptors with the corresponding features,
- a set of weights W_i of the features F_i, where $W_i = (w_i^1, w_i^2, \ldots, w_i^{m_i})$ and the indices $1, 2, \ldots, m_i$ match the weights with the corresponding features,
- the coordinates $c_i = (x_i, y_i, \varphi_i)$ representing the pose of the robot.

A robust feature detector and descriptor need to be employed to detect the features and calculate their compact representation. The feature detector and descriptor can be selected by the user. As suggested in [28] and [29], we have chosen Speeded-Up Robust Features (SURF) [31]. To verify this choice, an experimental evaluation of different features is given in Section 6.5.

3.2. Weighted Features

The weights W_i of features F_i are introduced to capture the importance of individual features in each database record r_i. All weights are in the range $w_i^j \in [0, 1]$. The weights are initialized at 0.5 for all features that have not been assigned any weight so far.

In the baseline localization method, the weights retain their initial values throughout the algorithm runtime. When updating the visual database based on change detection, the weights capture the stability and therefore also the reliability of the feature – the features with the highest weights are the most stable and reliable ones.

The weights capture the information whether the image patch represented by the feature (and its descriptor) is still present in the scene. This is also why the initial weight of a feature is set to the half of the interval of possible values – at first, it is not known whether the feature is stable (important) or not. The weight update process through change detection will be detailed in Section 4.

3.3. Correspondence-Based Localization

Using the previously constructed visual database, the robot can localize itself using a real-time feed of images from its camera that we refer to as the *query images*. The following steps are performed to localize the robot:

1. Capture a grayscale image I_q – the query image.
2. Run a feature detector and descriptor on the query image. The set of descriptors on the query image I_q is denoted as $D_\mathrm{q} = \{d_\mathrm{q}^1, d_\mathrm{q}^2, \ldots, d_\mathrm{q}^{m_\mathrm{q}}\}$.
3. Match the set of descriptors D_q found in the query image against the sets of descriptors D_i corresponding to each database record r_i using a standard matching algorithm [32].
4. Report the pose of the robot as the pose c_{i^*} stored with the database record r_{i^*}, which achieved the highest weighted correspondences ratio among all records r_i in the database.

The index i^* of the database record r_{i^*} with the highest weighted correspondences ratio is determined by the following equation:

$$i^* = \underset{i}{\operatorname{argmax}} \left(\frac{\sum_{j=1}^{m_i} p_{\mathrm{q},i}^j w_i^j}{\sum_{j=1}^{m_i} w_i^j} \right), \tag{1}$$

where the binary variable $p_{\mathrm{q},i}^j$ captures whether the feature f_i^j from F_i appears in the tentative matches between the query image I_q and the database image I_i. Tentative matches refer to a preliminary matching of descriptors [32] that can be verified through a model estimation method such as RANSAC [33]. This means that we are searching for a database record that has the largest proportion of feature descriptors matched with the feature descriptors found in the query image, giving more importance to the features with higher weights.

We chose to keep the baseline localization method as simple as possible to clearly demonstrate the impact of using weighted features within change detection. However, the proposed method can be plugged into more sophisticated methods as well. For instance, even though we use linear search for simplicity, more advanced techniques can be employed, such as k-d trees to store feature descriptors, allowing for a substantial speed-up for large databases [28]. Other approaches, e.g. visual vocabulary [34] or hierarchical k-means [35], can be used to further improve the performance.

4. Change Detection Method

We propose a method for change detection that improves the long-term autonomy of mobile robots through maintaining an accurate, up-to-date representation of the environment. The essence of the method is in learning the scene regions that are stable, distinguishing them from regions that change. This task is performed through feature-based change detection and results in a representation robust to changes in the environment. In the following text, we present the change detection method and show how it extends the baseline localization framework described in Section 3.

4.1. Detecting Changes

The change detection algorithm is based on the comparison of feature descriptors. We define a similarity measure between two descriptors d and d' based on their Euclidean distance:

$$s(d, d') = \frac{1}{1 + \|d - d'\|_2}. \tag{2}$$

This definition follows a standard approach to convert a distance measure to a similarity measure [36]. The similarity measure takes values between 0 and 1 with higher values indicating more similar descriptors. Using Euclidean distance for comparing SURF descriptors is suggested in [31] as one of the standard options.

The outline of the change detection algorithm is as follows:

1. Based on the pairs of tentative correspondences found by the matching algorithm, use MSAC [37] to estimate the transformation between the query image I_q and the best-match database image I_{i^*}.
2. Transform the positions of the features F_{i^*} in the best-match database image I_{i^*} to the coordinate frame of the query image I_q, yielding a set of transformed features $\bar{F}_{i^*}$.
3. Calculate the descriptors $\bar{D}_{i^*} = \{\bar{d}_{i^*}^1, \bar{d}_{i^*}^2, \ldots, \bar{d}_{i^*}^{m_{i^*}}\}$ of the transformed features $\bar{F}_{i^*}$ in the query image I_q.
4. Calculate the similarity s_j between the descriptors $d_{i^*}^j$ corresponding to the features $f_{i^*}^j$ in the database image I_{i^*} and the descriptors $\bar{d}_{i^*}^j$ of their projections $\bar{f}_{i^*}^j$ in the query image I_q.

To calculate s_j, the similarity measure (2) is adapted to the following form:

$$s_j = \frac{1}{1 + \left\| d_{i^*}^j - \bar{d}_{i^*}^j \right\|_2} \text{ for } j = 1, \ldots, m_{i^*}. \tag{3}$$

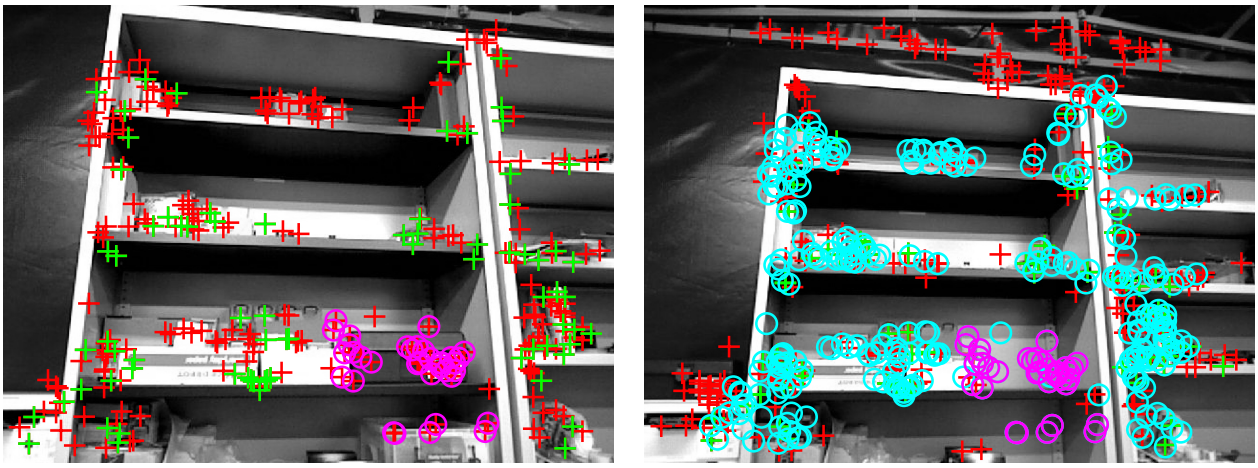
The change detection is therefore based on computing the similarity measure between the descriptors $d_{i^*}^j$ calculated on the best-match database image I_{i^*} and their transformed counterparts $\bar{d}_{i^*}^j$ calculated on the query image I_q. Note that this is different from using the features F_q and their descriptors D_q detected in the query image for comparison.

4.2. Weights Update

The weights of the features in the best-match database image are updated using the similarity values s_j. The weights are updated proportionally to the similarity between the descriptors $d^j_{i^*}$ calculated on the best-match database image I_{i^*} and their transformations $\bar{d}^j_{i^*}$ calculated on the query image I_q:

$$\forall j \in \{1, \ldots, m_{i^*}\} : w^j_{i^*} \leftarrow \min(C s_j w^j_{i^*}, 1) \tag{4}$$

As the similarity measure s_j takes values between 0 and 1, we choose $C = 2$ so that the weight remains constant if the similarity s_j is equal to 0.5.



(a) (b)

Figure 3: A database image (a) and a query image with one object missing on the third shelf from the top (b). The crosses represent the features in both images. Tentative correspondences found by the matching algorithm are shown in green. The cyan circles show the transformations of the features from the database image to the query image. The magenta circles show the features that were identified as a change.

Figure 3 shows a scene on which we illustrate the principle of the change detection algorithm. An item, e.g. a toolbox, has been removed from one of the shelves after building the visual database. The change detection algorithm transforms the features from the database image to the query image and calculates their SURF descriptors. They are then compared to the corresponding SURF descriptors in the database image. Since the descriptors in the region of the toolbox have low similarity, their weights are decreased. Note that as a by-product, some unstable features may have their weights decreased as well.

4.3. Long-Term Operation

The localization framework presented in Section 3 can now be extended by change detection with feature weight updates. An overview of the long-term localization with the change detection module is shown in Figure 2. In the long-term operation, the robot continuously localizes itself in the environment and maintains its visual database up to date by incorporating changes detected in the environment.

An important requirement that makes the change detection method efficient is that wrong matches (localization failures) need to be avoided as much as possible, because incorporating false-positive changes in wrongly matched database records decreases the quality of the visual database. To avoid this, we introduce three conditions that serve as a *confidence criterion*: a spatial condition, a temporal condition, and a sufficient number of correspondences. Only if all three conditions are met, the change detection module is run and the visual database is updated. In this way, the chance of incorporating false change detections on wrongly matched images is minimized.

4.3.1. Spatial Condition

The spatial condition relates the 'physically' closest database records (pose-wise) and the closest matches found by the localization algorithm (descriptor-wise), with reference to the best-match record. It exploits the assumption that images taken from nearby positions have some common features. Simply put, it is met only if the poses of the most similar matches are not too far from each other, taking into account the density of the database records. The spatial condition is evaluated through the following procedure:

1. Determine n_s most similar matches in the visual database for the query image, where n_s is a user-specified parameter. Sort them in descending order of the weighted correspondences ratio, which is calculated as the argument of (1).
2. Calculate the Euclidean distances e_i, $i = 2, \ldots, n_s$, between the best match and the successive $(n_s - 1)$ most similar matches. For this purpose, the distances are calculated in the (x, y)-space of the robot (omitting the angle φ).
3. Calculate the mean Euclidean distance m_s:

$$m_s = \frac{1}{n_s - 1} \sum_{i=2}^{n_s} e_i \,. \tag{5}$$

4. Similarly as in Step 2, calculate the Euclidean distances e'_j between the best match and $(n_r - 1)$ database records closest to the best match in the (x, y)-space of the robot, where n_r is a user-specified parameter.
5. Calculate the reference mean Euclidean distance m_r:

$$m_r = \frac{1}{n_r - 1} \sum_{j=2}^{n_r} e'_j \,. \tag{6}$$

6. Compare m_s with m_r. If $m_s < m_r$, the poses of the most similar matches are close enough to each other and the spatial condition is met.

Typically, the number of closest reference records n_r is set to be several times larger (e.g. 5×) than n_s. Note that the reference means m_r can be pre-calculated offline for all database records for efficiency.

4.3.2. Temporal Condition

The temporal condition verifies whether the distance between the current location and the previous location is smaller than a given threshold δ. It takes into account the physical limitations of the robot and discards unreliable matches in cases when the robot appears to have moved further than it possibly could. The temporal condition is tested only if the spatial condition was met both for the current and for the previous image, which allows for recovery after a localization failure.

4.3.3. Sufficient Number of Correspondences

As described in Step 1 of the change detection algorithm in Section 4.1, we use MSAC [33] to estimate the transformation between the query image and the best-match database image. In this way, we also obtain the information on which features are considered inliers and which are outliers. A small number of inlier correspondences indicates that the match is not very reliable. Therefore, if the number of corresponding feature pairs classified as inliers is smaller than a given threshold θ, the weights are not updated.

4.4. Limitations

The proposed method adjusts the importance of features in the database images through their weights. This approach relates all subsequent visits of a place to the first mapping session. If the appearance of a place changes (almost) completely, there are none or only a very few features that can be matched between the current view and the appearance of the place captured during the visual database creation. In such cases, the query image cannot be matched correctly. For instance, if applied to outdoor environments, the method is in general not able to deal with seasonal changes. Nevertheless, unlike other authors [28, 29], we have decided not to include feature addition to the

algorithm. The motivation behind this design choice is that new features might come, for instance, from a temporary occlusion or a severe change of lighting conditions. Distinguishing such situations from permanent changes usually means introducing more parameters to the method. Our aim is to keep the long-lasting features from the initial scanning, as we presume that the main structures in most of the scenes remain stable over long periods of time.

As we only use single camera images and no information about the depth is needed, we are limited to the use of planar perspective feature projection (Section 4.1). The projection works well if the viewpoints (robot poses) in the visual database are similar to those in the query images, which is the case in the data sets used in the experiments. If the robot gets too far from the records present in the visual database, the viewpoint changes significantly and the reprojection of features does not work well. As a consequence, even though it would be possible to update also the adjacent images in the visual database with the detected changes, we do not perform this update as the benefit would be suppressed by the errors incurred by the reprojection.

While the algorithm allows for adding new images to the database at any point, new database records need to come with the information about the pose of the robot, see Section 3. Such information may not be available to the robot performing the long-term operation, as the only sensor it needs to have in our setting is a camera.

5. Evaluation on Our Data Sets

For the experimental evaluation, we have chosen TurtleBot 2 as a widely used and affordable platform, equipped with an RGB-D camera and an on-board computer with a performance comparable to standard laptops. Note that even though the camera is capable of recording depth images, the depth information is not used in our method, as we only use grayscale images. We have used two TurtleBot robots to evaluate the robustness of the method using two different cameras in various environments:

1. TurtleBot 2 equipped with a camera Asus Xtion PRO LIVE, used in experiments at the Robotics Lab, Carlos III University in Madrid, Spain;
2. TurtleBot 2 equipped with a camera Orbbec Astra Pro, used in experiments at the Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic.

In both cases, we attached an additional structure to the robot to fix the camera at a higher position, see Figure 4. The pose of the robot was captured through odometry based on wheel encoders. We used floor markers and manual correction of the measured poses to ensure sufficient ground truth precision.

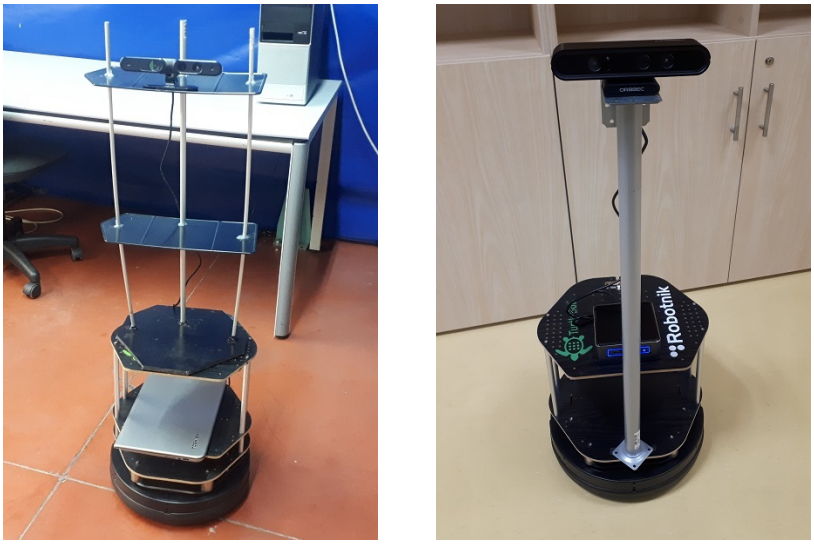
5.1. Data Sets

While several data sets featuring city streets, seasonal changes and other outdoor environments are available [21], the number of available long-term indoor data sets is limited. With the exception of the STRANDS Witham Wharf data set [38], which we evaluate in Section 6, there are, to the best of our knowledge, no public data sets capturing indoor dynamic environments which would be well suited for the type of changes that we focus on in this paper. To allow for a detailed analysis and better insight into the method's performance, we have recorded our own data sets from four indoor environments at the Leganés campus of the Carlos III University in Madrid and at the Czech Institute of Informatics, Robotics, and Cybernetics in Prague.

The data sets in each of the environments – *Lab*, *Classroom*, *Hall*, and *Office* – consist of multiple sequences. The sequences were recorded on different days and at different times of the day, capturing various changes in the environment (moving chairs and items on the desks, changing the picture on computer screens, opening and closing window blinds, etc.). We have also recorded new sequences after eight months in the *Lab*, *Classroom*, and *Hall* environments to evaluate the long-term performance of the method on a longer time span.

5.2. Experimental Setup

At first, we have constructed a visual database for each environment. Our visual databases are sparser than the query sequences to make the database images distinctive and avoid multiple records from the same place. Therefore, the visual databases typically contain fewer images than the query sequences. The playback of the recorded query sequences served as a stream of query images in real time. Each query image was matched with the most similar image in the visual database and the pose associated with the most similar database image was returned as the pose



(a) (b)

Figure 4: Mobile robots TurtleBot 2 used in the experiments: (a) at the Carlos III University in Madrid, (b) at the Czech Technical University in Prague.

of the robot. If the confidence conditions (Section 4.3) were met, the feature weights of the best match record in the visual database were updated based on the detected changes.

In all experiments, we have used the following default configuration of the confidence criterion: the number of closest samples was set to $n_s = 2$ and the number of reference samples $n_r = 10$. The temporal difference tolerance was set to $\delta = 0.5\,\mathrm{m}$ and the minimum number of correspondences was $\theta = 10$. We have empirically evaluated that the default values worked well in all performed experiments. However, they may be adjusted for improved performance on data sets with substantially different properties. Optimization and further analysis of the parameters will be performed as a part of our future work.

We have compared the root-mean-squared (RMS) localization errors on the query sequences matched against the original visual databases and against the visual databases that have been updated by executing localization with change detection on the query sequences. For all images in all query sequences, a maximum of 20 % of the image area has changed with respect to the visual database and with respect to the other query sequences.

The localization errors are calculated for each pair of a query image and the matched database image as an ℓ^2 norm (Euclidean distance) between the ground truth robot pose of the query image and the robot pose stored with the matched database image. The robot orientation angle is also included in the robot pose vector and it is wrapped to yield a maximal difference of π rad. Note that the units of the localization RMS error are not meters, as the pose vector combines values in meters and radians.

5.3. Results

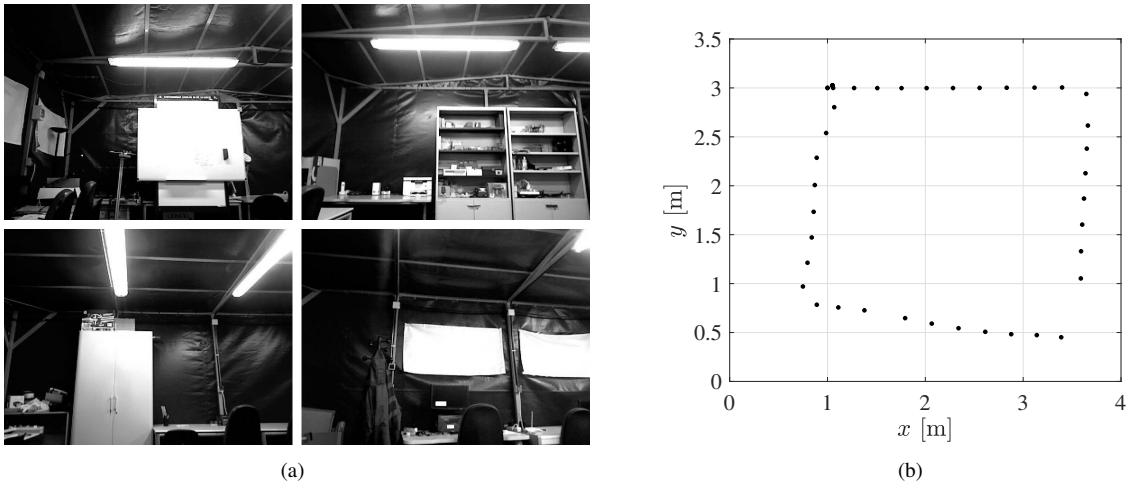
In this section, we present the results of the method on four indoor environments: *Lab*, *Classroom*, *Hall*, and *Office*. For each of them, we have randomly chosen different combinations of the query sequences for updating the visual database to demonstrate the method performance in various scenarios.

5.3.1. Lab Environment

The *Lab* data set was recorded at the Carlos III University in Madrid. The changes in the environment that appear in this data set include moving up to 3 chairs, altering the content on the whole area of the whiteboard, moving up to 10 objects on desks, and moving up to 5 items on the shelves of the cabinets.

First, we have created the visual database L-DB for a trajectory of an approximately square shape, see Figure 5. We have recorded three other sequences, L-Q1, L-Q2, and L-Q3, which serve as the query sequences. The sequence

L-Q4 was recorded eight months later. Table 1 summarizes the properties of the sequences and Table 2 presents the results.



(a) (b)

Figure 5: *Lab* environment: (a) examples of images, (b) the trajectory travelled.

Table 1: Properties of the image sequences used in the Lab environment.

Image sequence	**Number of images**	**Distance travelled**
L-DB (database)	41	10.0 m
L-Q1 (query)	89	10.5 m
L-Q2 (query)	85	10.2 m
L-Q3 (query)	84	10.0 m
L-Q4 (query)	97	10.4 m

Table 2: Localization RMS errors on different query sequences in the Lab environment. Sequences evaluated on a database updated with changes are shown in bold.

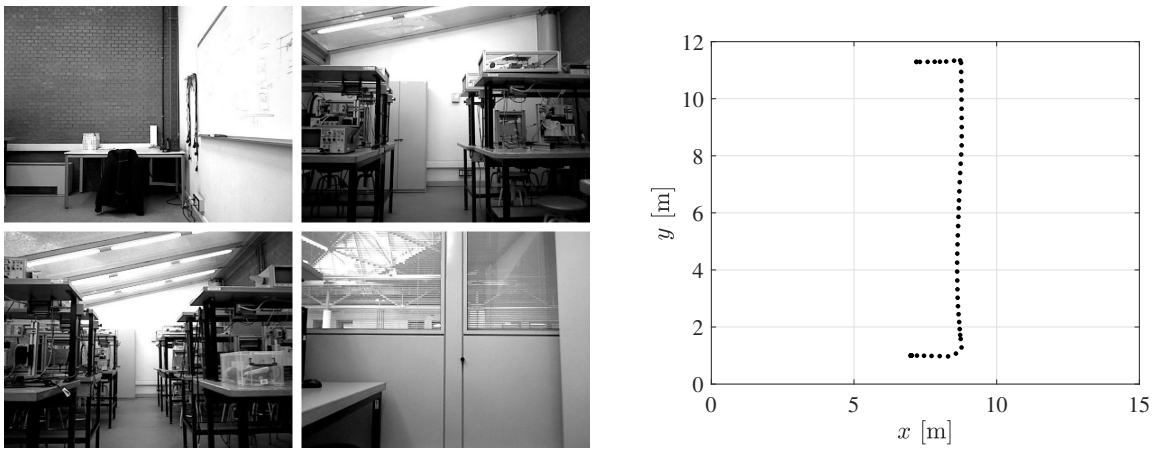
Visual database	**Updated on**	**Query sequence**	**Localization RMS error**
L-DB	–	L-Q3	0.56
L-DB	**L-Q1**	**L-Q3**	**0.46**
L-DB	**L-Q1, L-Q2**	**L-Q3**	**0.41**
L-DB	–	L-Q4	1.25
L-DB	**L-Q1**	**L-Q4**	**1.20**

The localization accuracy on the query sequence L-Q3 improves when the visual database L-DB is updated by changes detected on the sequence L-Q1. If the visual database is afterward updated also on the sequence L-Q2, the localization accuracy on the query sequence L-Q3 is further improved.

The number of changes that have occurred in the environment during the long time span before recording the new query sequence L-Q4 was significant. Despite that, the localization RMS error has improved on the query sequence L-Q4 when it was evaluated on the visual database L-DB updated by the sequence L-Q1, compared to evaluating it on the original visual database L-DB. The sequence L-Q1 captures changes such as moved chairs and objects on tables, which helps to build a more robust representation of the environment. The stable elements in the environment are assigned higher weights, which allows for a more accurate localization even after a long period of time.

5.3.2. Classroom Environment

The *Classroom* data set, featuring e.g. desks with items being moved on them, a whiteboard with changing content, and adjustable window blinds, was also recorded at the Carlos III University in Madrid. We have first created the visual database C-DB, see Figure 6. Two query sequences, C-Q1 and C-Q2, were recorded several days after the visual database C-DB, and another query sequence, C-Q3, was recorded eight months later. Table 3 summarizes the properties of the sequences and Table 4 presents the results.



(a) (b)

Figure 6: *Classroom* environment: (a) examples of images, (b) the trajectory travelled.

Table 3: Properties of the image sequences used in the Classroom environment.

Image sequence	**Number of images**	**Distance travelled**
C-DB (database)	62	13.5 m
C-Q1 (query)	100	13.6 m
C-Q2 (query)	101	13.3 m
C-Q3 (query)	117	13.3 m

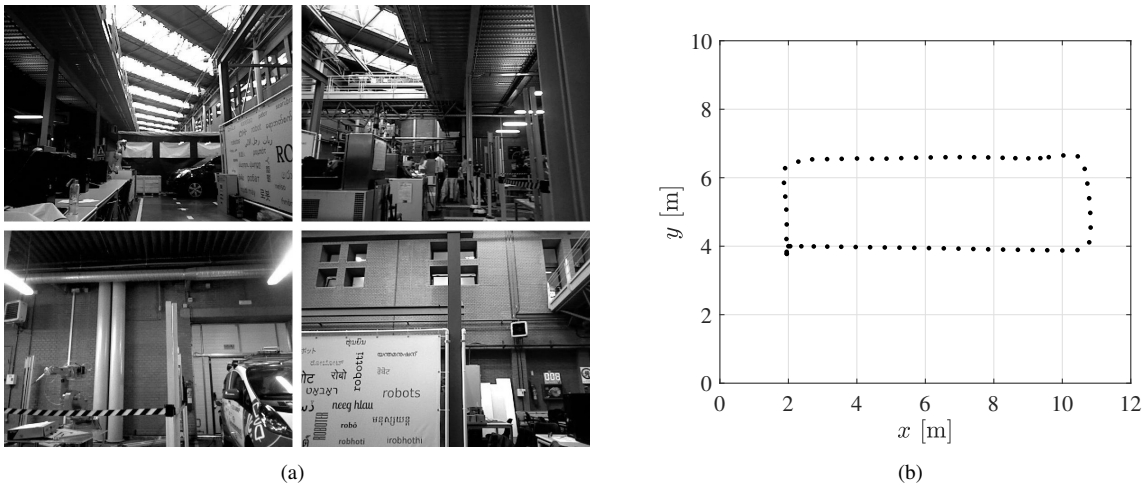
Table 4: Localization RMS errors on different query sequences in the Classroom environment. Sequences evaluated on a database updated with changes are shown in bold.

Visual database	**Updated on**	**Query sequence**	**Localization RMS error**
C-DB	–	C-Q2	0.39
C-DB	**C-Q1**	**C-Q2**	**0.39**
C-DB	–	C-Q3	1.02
C-DB	**C-Q1**	**C-Q3**	**0.96**

The localization task was more challenging in the *Classroom* environment. In the case of the query sequence C-Q2 evaluated on the visual database C-DB updated on C-Q1, the localization accuracy remains the same as for the evaluation on the original C-DB. However, for the new query sequence C-Q3, updating the visual database helps to decrease the localization error.

5.3.3. Hall Environment

The *Hall* data set, resembling an industrial environment, was also recorded at the Carlos III University in Madrid. We have first created the visual database H-DB for a rectangular trajectory, see Figure 7. We have recorded two other sequences, H-Q1 and H-Q2, which serve as the query sequences. Eight months later, we have added another sequence H-Q3. Table 5 summarizes the properties of the sequences and Table 6 presents the results.



(a) (b)

Figure 7: *Hall* environment: (a) examples of images, (b) the trajectory travelled.

Table 5: Properties of the image sequences used in the Hall environment.

Image sequence	**Number of images**	**Distance travelled**
H-DB (database)	58	22.5 m
H-Q1 (query)	67	7.1 m
H-Q2 (query)	59	6.9 m
H-Q3 (query)	53	6.2 m

Taking into account the spacing between consecutive database images (see Figure 7b), the localization results using the initial visual database are already accurate, leaving little space for improvement. Nevertheless, an improvement was achieved in both cases – for the query sequence H-Q2 evaluated on the visual database H-DB updated with H-Q1 and for the query sequence H-Q1 evaluated on the visual dabatase H-DB updated with H-Q2. Within the 8 months, the *Hall* environment has undergone substantial changes. The visual database H-DB updated with any of the two sequences H-Q1 or H-Q2 allows for a more accurate localization on the new query sequence H-Q3.

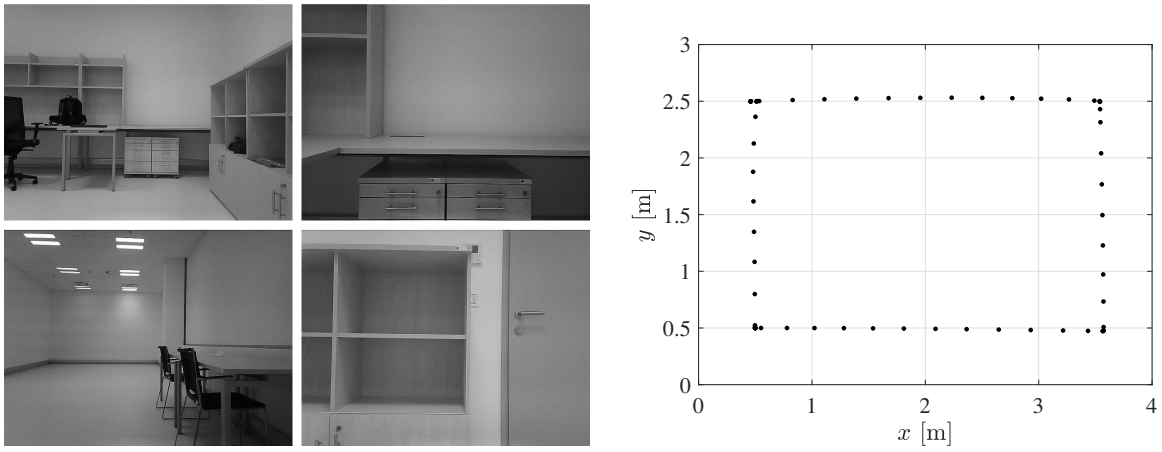
5.3.4. Office Environment

The *Office* data set was recorded at the Czech Technical University in Prague. The office undergoes changes such as opening and closing cabinet doors, changing positions of the chairs, or replacing items on the desks. We have first created the visual database O-DB for a rectangular trajectory, see Figure 8. We have recorded four other sequences, O-Q1, O-Q2, O-Q3, and O-Q4, which serve as the query sequences. Table 7 summarizes the properties of the sequences and Table 8 presents the results.

The *Office* data set proves to be more difficult in terms of precise localization. The environment contains large uniform and textureless areas and also repetitive instances of identical objects, which makes the feature-based localization more challenging. Nevertheless, the change detection method results in improved localization accuracy.

Table 6: Localization RMS errors on different query sequences in the Hall environment. Sequences evaluated on a database updated with changes are shown in bold.

Visual database	Updated on	Query sequence	Localization RMS error
H-DB	–	H-Q1	0.20
H-DB	**H-Q2**	**H-Q1**	**0.19**
H-DB	–	H-Q2	0.16
H-DB	**H-Q1**	**H-Q2**	**0.15**
H-DB	–	H-Q3	1.80
H-DB	**H-Q1**	**H-Q3**	**1.64**
H-DB	**H-Q2**	**H-Q3**	**1.65**



(a) (b)

Figure 8: *Office* environment: (a) examples of images, (b) the trajectory travelled.

Table 7: Properties of the image sequences used in the Office environment.

Image sequence	Number of images	Distance travelled
O-DB (database)	67	10.2 m
O-Q1 (query)	119	9.7 m
O-Q2 (query)	116	10.2 m
O-Q3 (query)	116	10.3 m
O-Q4 (query)	111	10.1 m

Table 8: Localization RMS errors on different query sequences in the Office environment. Sequences evaluated on a database updated with changes are shown in bold.

Visual database	Updated on	Query sequence	Localization RMS error
O-DB	–	O-Q1	0.74
O-DB	**O-Q2**	**O-Q1**	**0.72**
O-DB	–	O-Q2	0.93
O-DB	**O-Q3**	**O-Q2**	**0.90**
O-DB	–	O-Q3	0.94
O-DB	**O-Q1**	**O-Q3**	**0.90**
O-DB	–	O-Q4	0.97
O-DB	**O-Q2**	**O-Q4**	**0.91**

5.4. Discussion

The change detection method reduces the localization (pose estimation) RMS error by 27 % in the *Lab* environment and by 9 % in the *Hall* environment. These two environments are rich in distinctive features, which allows for a better performance of the feature-based localization method. Therefore, localization can also better benefit from the change detection method. The other two environments, *Classroom* and *Office*, have shown to be more difficult for localization. However, the change detection method could still improve the localization accuracy by 6 % in both cases. The change detection method has proven to work even when dealing with a long time span in the case of all three environments *Lab*, *Classroom*, and *Hall*, where we have recorded new sequences several months after the first experiments.

The method runs in real time. In our experiments, we were capturing an image every second. The processing time of a single query image is approximately 250 ms on a standard computer (CPU Intel Core i7-4610M @ 3.0 GHz, 16 GB RAM) for visual databases of less than 100 images. Methods allowing speed-up on larger databases can be employed, as discussed in Section 3.3. Such extensions are beyond the scope of this paper.

6. Evaluation on Public Data Set

In addition to the experiments with our own data sets, we evaluate the method on a public data set Witham Wharf from the STRANDS project [38]. The data set is intended for RGB-D localization in changing environments and therefore, it suits well a long-term indoor localization experiment with our method. Furthermore, we use the data set to compare the performance of our method to two alternative methods for feature-based localization [28, 39]. Finally, we analyze the impact of using different feature detectors and descriptors within our method.

6.1. Data Set Overview

The data set contains images captured by a mobile robot every 10 minutes at eight locations in an open-plan office. It is divided into a training set recorded over the period of one week and three test sets, which were recorded on three different days throughout the following months. The training set consists of 1008 images for each of the eight locations, while each test set contains 144 images for each location. We omit the available depth information and convert the RGB images to grayscale images, as in Section 5.

To form the visual database, we take the first image for each location from the training set, see Figure 9. All the remaining images from the training set are used to build the query sequence S-Q1. Each test set corresponds to one of the query sequences S-Q2, S-Q3, and S-Q4, see Table 9.

Figure 9: Initial images (S-DB) taken from each of the eight locations in the STRANDS Witham Wharf data set.

Table 9: Summary of the STRANDS Witham Wharf data set properties.

Image sequence	Number of images	Date of recording
S-DB (database)	8	10 November 2013
S-Q1 (query)	8056	10–16 November 2013
S-Q2 (query)	1152	17 November 2013
S-Q3 (query)	1152	2 February 2014
S-Q4 (query)	1152	14 December 2014

6.2. Method Performance

This experiment compares the baseline localization algorithm, where the change detection module is disabled, with localization using change detection. As the data set consists of eight discrete locations rather than images from trajectories (like in Section 5), we use a different evaluation metric than for our data sets. We calculate the localization accuracy as the fraction of images that were assigned to the correct ground truth location among all evaluated images. The nature of the data set also does not allow for the use of the spatial and temporal conditions in the confidence criterion. The confidence criterion therefore relies on the minimum number of correspondences, which was set to $\theta = 10$.

The localization algorithm with change detection takes S-DB as the visual database and runs first on the long query sequence S-Q1 (training set). Then, retaining the feature weights after processing S-Q1, it is executed on the query sequences S-Q2, S-Q3, and S-Q4 (test sequences). The baseline localization without change detection is evaluated independently on all four query sequences. The results are presented in Table 10 and examples of matched places for query images from one place are shown in Figure 10.

Table 10: Localization accuracy of the proposed method on the query sequences from the STRANDS Witham Wharf data set using the SURF features. Better results are shown in bold.

Visual database	Updated on	Query sequence	Change detection	Localization accuracy
S-DB	–	S-Q1	off	64.15 %
S-DB	**–**	**S-Q1**	**on**	**66.37 %**
S-DB	–	S-Q2	off	58.68 %
S-DB	**S-Q1**	**S-Q2**	**on**	**60.16 %**
S-DB	–	S-Q3	off	60.76 %
S-DB	**S-Q1**	**S-Q3**	**on**	**62.67 %**
S-DB	–	S-Q4	off	47.83 %
S-DB	**S-Q1**	**S-Q4**	**on**	**50.61 %**

Note that the robot records the images 24 hours and therefore, a substantial portion of the images, which were taken at night, is completely dark. Such images do not contain any features and therefore cannot be matched correctly. To ensure a fair evaluation, we have not excluded any images from the data set.

The sequences are processed in batches of eight images, one from each place. Once the images from the same time frame are processed, the algorithm moves to the next time frame. If the change detection module is enabled, the weights are updated after processing every image. Already on the query sequence S-Q1, during ongoing change detection, a higher localization performance is achieved with the change detection module enabled. For the test sequences S-Q2, S-Q3, and S-Q4, feature weights pre-trained by processing S-Q1 are used as a starting point and they continue to update throughout the experiment. Also on all the test sequences, the localization is more accurate when the change detection module is enabled.

6.3. Adaptive Appearance-Based Map

An alternative approach to the proposed method is the Adaptive Appearance-Based Map (AABM) for long-term localization, presented in [28]. The algorithm introduces two types of feature storage: the short-term memory (STM)

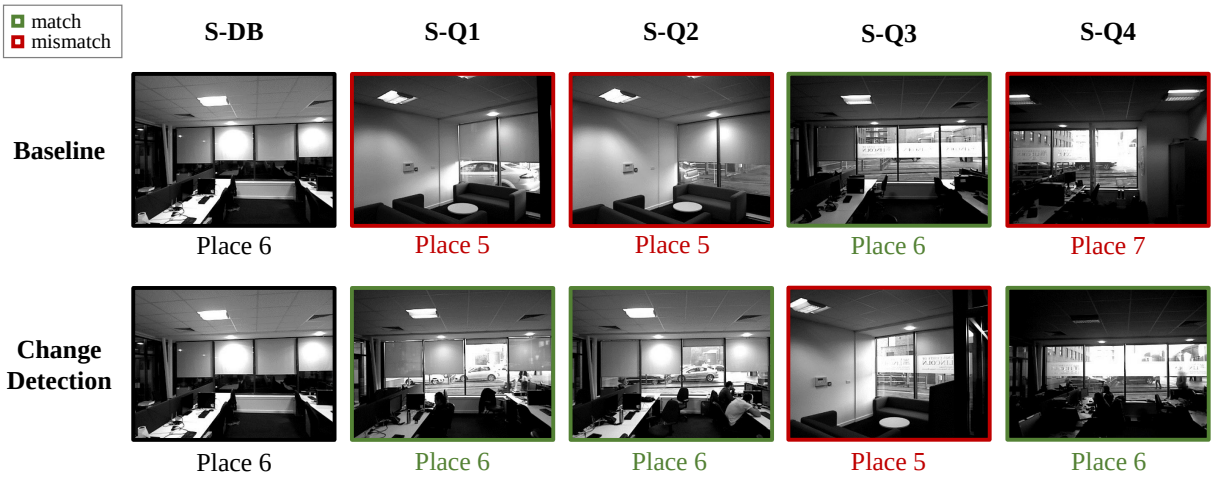


Figure 10: Examples of images matched by the baseline method and using the change detection. The figure shows successful matches (S-Q1, S-Q2, S-Q4) as well as a localization failure (S-Q3) when using the change detection method.

and the long-term memory (LTM). While LTM is used for feature matching, STM serves as a waiting list for newly discovered features captured at the same place. Both types of memory are adaptively updated based on the features observed in the query images. For a detailed description of the method, please refer to [28].

We evaluate AABM in the same way as our method (Section 6.2). To configure the method, we adopt the parameters used in the experiments reported in [28]. As the authors do not suggest how to tune the parameters for specific data sets and since both experiments presented in [28] share similar properties with the Witham Wharf data set, we have executed the evaluation in the two configurations used in the paper. The first one uses STM with 4 stages and LTM with 5 stages, while the second one is configured with a 2-stage STM and a 4-stage LTM. We used the same baseline localization method as in our other experiments, which is similar to the *static map* described in [28]. The results are reported in Table 11.

Table 11: Localization accuracy of [28] on the query sequences from the STRANDS Witham Wharf data set. Performance better than our method (Table 10) is denoted in bold.

Base images	Pre-trained on	Query sequence	STM stages	LTM stages	Localization accuracy
S-DB	**–**	**S-Q1**	**2**	**4**	**66.47 %**
S-DB	–	S-Q1	4	5	55.50 %
S-DB	**S-Q1**	**S-Q2**	**2**	**4**	**61.89 %**
S-DB	S-Q1	S-Q2	4	5	55.38 %
S-DB	S-Q1	S-Q3	2	4	57.29 %
S-DB	S-Q1	S-Q3	4	5	49.05 %
S-DB	S-Q1	S-Q4	2	4	43.14 %
S-DB	S-Q1	S-Q4	4	5	45.05 %

To ensure a fair comparison, the STM and LTM update (denoted *rehearsal* and *recall* in [28]) was performed only if the number of matching features between the query image and the best-match LTM was at least $\theta = 10$, same as in our method. Without this modification, the LTM would be cleared after observing several successive fully dark images and the method would stop working.

First, we discuss the performance of AABM using 2 STM stages and 4 LTM stages. The method [28] shows a comparable to slightly better performance w.r.t. our method (Table 10) on images recorded in the period of the first eight days (0.10 % for S-Q1 and 1.73 % for S-Q2). However, the performance drops substantially for images captured

three months later (5.38 % on S-Q3) and one year later (7.47 % on S-Q4). For the configuration of 4 STM stages and 5 LTM stages, our change detection method outperforms AABM on all query sequences.

The results indicate that the performance of AABM varies depending on the number of STM and LTM stages. The advantage of our method is that its performance is not particularly dependent on the choice of parameters – we did not have to change the configuration of the method in any of the performed experiments. Apart from the setting of the parameters, the difference in the performance can be explained by the way the features are processed. Our method accentuates the features from the initial scanning that remain the same and reduces the impact of those that have changed by updating their weights. By contrast, AABM completely drops LTM features that are not present for several frames and needs to re-learn them again, passing them first through STM. Therefore, feature weighting in our method provides a better way to capture feature importance.

6.4. Localization Using FAB-MAP

This section presents an evaluation of a general localization method that does not update the database. We chose the Fast Appearance-Based Mapping [39], abbreviated as FAB-MAP, which is a popular method for localization and mapping.

FAB-MAP is a probabilistic approach based on Chow Liu trees [40]. The method learns a generative model of bag-of-words observations [34]. The requirement for a learned model (*vocabulary*) represents the main difference from our method and AABM. However, FAB-MAP is expected to perform well even with a model trained on different data, which diminishes the issue with the training overhead. The authors show that the method is able to deal with scene changes, which is applicable to our experimental scenario.

We have used a publicly available implementation[1] of the FAB-MAP algorithm with the Indoor Environments vocabulary[2]. We have adapted several parameters in the default configuration of the method to match our evaluation scenario on the STRANDS Witham Wharf data set. The SURF blob response threshold was changed to 4.0, as suggested in the documentation of the vocabulary used. The prior probability of being at a new place was set to zero, because we require each image to be assigned to one of the existing locations, due to the nature of the Witham Wharf data set. Finally, the position prior model was set to uniform as the images in the Witham Wharf data set do not come from a continuous robot trajectory, but they are recorded at individual locations.

Same as in Sections 6.2 and 6.3, we have used S-DB as the base environment representation and S-Q1 to S-Q4 as the query sequences, see Table 9. Each query image was assigned to one of the eight locations in S-DB. The overall accuracy for each query sequence is reported in Table 12.

Table 12: Comparison of the localization accuracy of FAB-MAP [39] and the proposed change detection method on the query sequences from the STRANDS Witham Wharf data set. The better performing method is denoted in bold.

Base images	Query sequence	Method	Localization accuracy
S-DB	**S-Q1**	**FAB-MAP**	**67.42 %**
S-DB	S-Q1	Our method	66.37 %
S-DB	S-Q2	FAB-MAP	59.64 %
S-DB	**S-Q2**	**Our method**	**60.16** %
S-DB	S-Q3	FAB-MAP	57.73 %
S-DB	**S-Q3**	**Our method**	**62.67** %
S-DB	S-Q4	FAB-MAP	46.53 %
S-DB	**S-Q4**	**Our method**	**50.61** %

The results show that FAB-MAP, being a more sophisticated technique, expectably outperforms our method, and also AABM, on the initial query sequence featuring a small number of changes. However, from the performance on the following sequences, we conclude that our method performs better on data recorded several months later after the initial scanning, similarly as in comparison with AABM discussed in Section 6.3. The proposed method with change

[1] http://www.robots.ox.ac.uk/~mjc/Software.htm

[2] http://www.robots.ox.ac.uk/~mjc/FabMap_Release/IndoorVocab_10k.zip

detection is better by 4.94 % on S-Q3 and by 4.08 % on S-Q4 than FAB-MAP. Therefore, this again confirms the benefits of change detection for images recorded after a longer period of time has passed from the initial scanning, where more changes are present in the environment.

6.5. Feature Types

As discussed in Section 3.1, the proposed method can be used with various feature detectors and descriptors, according to the choice of the user. In this experiment, we compare SURF [31], ORB [41], and BRISK features [42]. In all cases, we use the same method for feature detection and for descriptor calculation.

The ORB and BRISK features have binary descriptors. In contrast to the SURF descriptors, which can be compared by using the Euclidean distance, these descriptors are compared through the Hamming distance [41, 42]. The ORB and BRISK descriptor distances are normalized to the range of the SURF descriptor distances for consistency.

We have evaluated the method in an identical scenario as in Section 6.2. The baseline algorithm without change detection is run independently on all four query sequences without any prior weight updates. The change detection is executed first on the sequence S-Q1 and then the updated visual database is used as an entry point to evaluate the sequences S-Q2, S-Q3, and S-Q4. The results are presented in Table 13 for the ORB features and in Table 14 for the BRISK features.

Table 13: Localization accuracy of the proposed method on the query sequences from the STRANDS Witham Wharf data set using the ORB features. Better results are shown in bold.

Visual database	Updated on	Query sequence	Change detection	Localization accuracy
S-DB	–	S-Q1	off	56.24 %
S-DB	**–**	**S-Q1**	**on**	**56.65 %**
S-DB	–	S-Q2	off	52.26 %
S-DB	**S-Q1**	**S-Q2**	**on**	**53.73 %**
S-DB	–	S-Q3	off	48.26 %
S-DB	**S-Q1**	**S-Q3**	**on**	**48.96 %**
S-DB	**–**	**S-Q4**	**off**	**37.59 %**
S-DB	S-Q1	S-Q4	on	37.50 %

Table 14: Localization accuracy of the proposed method on the query sequences from the STRANDS Witham Wharf data set using the BRISK features. Better results are shown in bold.

Visual database	Updated on	Query sequence	Change detection	Localization accuracy
S-DB	–	S-Q1	off	61.35 %
S-DB	**–**	**S-Q1**	**on**	**62.18 %**
S-DB	–	S-Q2	off	54.60 %
S-DB	**S-Q1**	**S-Q2**	**on**	**55.82 %**
S-DB	–	S-Q3	off	47.57 %
S-DB	**S-Q1**	**S-Q3**	**on**	**50.43 %**
S-DB	–	S-Q4	off	40.36 %
S-DB	**S-Q1**	**S-Q4**	**on**	**42.45 %**

The results show that an improvement with respect to the baseline is achieved by using the change detection method with alternative feature types in almost all cases, with the only exception of the query sequence S-Q4 evaluated using the ORB features. However, in that case, the localization accuracy is low, both without and with the change detection. The SURF features (Table 10) achieve the best localization accuracy on all query sequences. The average computation times per query image for implementation in MATLAB using the Computer Vision Toolbox on a standard computer[3] are 163 ms for the SURF features, 363 ms for the ORB features, and 880 ms for the BRISK features. The

[3]CPU Intel Core i7-4610M @ 3.0 GHz, 16 GB RAM

algorithm therefore runs in the shortest time using the SURF features. Note that in all cases, we used the default parameters of the localization method, of the detector, and of the descriptor. We did not perform any parameter tuning.

6.6. Discussion

The evaluation on the STRANDS Witham Wharf data set has demonstrated that the proposed method is able to achieve improved localization accuracy over the baseline and outperform the alternative methods [28, 39] in a long-term scenario. The performance of these methods could possibly be improved by extensive parameter tuning, but the authors do not provide any guidelines for setting the parameters. The strength of our method is therefore the small number of parameters and the robustness of the method to their setting, as demonstrated by using the same configuration for substantially different data sets.

The change detection method allows for the use of different feature types. SURF features yield the best results both in the localization accuracy and in the processing time.

The Witham Wharf data set contains images assigned to the same place taken from slightly different viewpoints, as well as under substantial day/night illumination changes. Such situations increase the risk of false matches and subsequent wrong feature updates. The experimental results show that the proposed method yields improved performance over the baseline despite these challenges.

7. Conclusions

We have proposed a method for change detection based on the comparison of local visual features and we have shown how the change detection method can be incorporated into a localization framework. The representation of the environment in the form of a visual database is continuously adapted as the robot moves through its area of operation. We have introduced feature weights to capture the importance of each feature. The weights are updated based on the feature descriptor similarity.

We have used a three-component confidence criterion to decrease the risk of impairing the visual database by introducing changes from wrongly matched images. The experimental evaluation on four different environments has shown that the change detection algorithm allows to capture the dynamics of the environment and leads to more accurate localization.

An evaluation on the Witham Wharf data set with thousands of images provided an insight into the method performance in additional experiments. The proposed method outperforms the baseline localization method without change detection and also two alternative approaches, AABM and FAB-MAP, on the majority of the query sequences.

In our future work, we are planning to perform a long-term experiment by recording multiple large-scale sequences over a long time span and to further improve the algorithm by evaluating other feature weight update methods. Using a precise localization based on data from additional sensors or from a motion capture system would improve the accuracy in the visual database building stage. This would reduce the reprojection errors in image matching and allow for propagating the detected changes also to the neighboring records, improving the overall localization accuracy.

Another possible line of future research would be to incorporate the semantic information into the change detection algorithm, e.g., by applying an object detection method to determine the changes in the scenes based on object occurrence.

Acknowledgments

This work was supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000470) and by the Grant Agency of the Czech Technical University in Prague, grant no. SGS19/174/OHK3/3T/13. This research has also received funding from HEROITEA: Heterogeneous Intelligent Multi-Robot Team for Assistance of Elderly People (RTI2018-095599-B-C21), funded by Spanish Ministerio de Economia y Competitividad, and the RoboCity2030 – DIH-CM project (S2018/NMT-4331, RoboCity2030 – Madrid Robotics Digital Innovation Hub).

References

[1] R. Alterovitz, S. Koenig, M. Likhachev, Robot planning in the real world: Research challenges and opportunities, AI Magazine 37 (2) (2016) 76–84.

[2] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, M. J. Milford, Visual place recognition: A survey, IEEE Transactions on Robotics 32 (1) (2015) 1–19.

[3] E. Garcia-Fidalgo, A. Ortiz, Vision-based topological mapping and localization methods: A survey, Robotics and Autonomous Systems 64 (2015) 1–20.

[4] N. Bore, P. Jensfelt, J. Folkesson, Multiple object detection, tracking and long-term dynamics learning in large 3D maps, arXiv preprint arXiv:1801.09292 (2018).

[5] L. Kunze, H. Karaoguz, J. Young, F. Jovan, J. Folkesson, P. Jensfelt, N. Hawes, SOMA: A framework for understanding change in everyday environments using semantic object maps, AAAI, 2018.

[6] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, et al., The STRANDS project: Long-term autonomy in everyday environments, IEEE Robotics & Automation Magazine 24 (3) (2017) 146–156.

[7] J. Biswas, M. Veloso, Episodic non-Markov localization: Reasoning about short-term and long-term features, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 3969–3974.

[8] R. Finman, T. Whelan, M. Kaess, J. J. Leonard, Toward lifelong object segmentation from change detection in dense RGB-D maps, in: 2013 European Conference on Mobile Robots, IEEE, 2013, pp. 178–185.

[9] B. Bescós, J. M. Fácil, J. Civera, J. Neira, DynSLAM: Tracking, mapping and inpainting in dynamic scenes, arXiv preprint arXiv:1806.05620 (2018).

[10] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, C. Cadena, TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery, in: 2017 IEEE International Conference on Robotics and automation (ICRA), IEEE, 2017, pp. 5237–5244.

[11] P. Drews, L. J. Manso, S. da Silva Filho, P. Núñez, Improving change detection using Vertical Surface Normal Histograms and Gaussian Mixture Models in structured environments, in: 2013 16th International Conference on Advanced Robotics (ICAR), IEEE, 2013, pp. 1–7.

[12] L. Wellhausen, R. Dubé, A. Gawel, R. Siegwart, C. Cadena, Reliable real-time change detection and mapping for 3D LiDARs, in: 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), IEEE, 2017, pp. 81–87.

[13] F. Boniardi, T. Caselitz, R. Kümmerle, W. Burgard, A pose graph-based localization system for long-term navigation in CAD floor plans, Robotics and Autonomous Systems 112 (2019) 84–97.

[14] J. Andrade-Cetto, A. Sanfeliu, Concurrent map building and localization on indoor dynamic environments, International Journal of Pattern Recognition and Artificial Intelligence 16 (03) (2002) 361–374.

[15] K. Konolige, J. Bowman, Towards lifelong visual maps, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2009, pp. 1156–1163.

[16] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, E. Romera, Towards life-long visual localization using an efficient matching of binary sequences from images, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 6328–6335.

[17] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, R. Gherardi, Street-view change detection with deconvolutional networks, Autonomous Robots 42 (7) (2018) 1301–1322.

[18] F. Nobre, C. Heckman, P. Ozog, R. W. Wolcott, J. M. Walls, Online probabilistic change detection in feature-based maps, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 1–9.

[19] B. Neuman, B. Sofman, A. Stentz, J. A. Bagnell, Segmentation-based online change detection for mobile robots, in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 5427–5434.

[20] T. Naseer, W. Burgard, C. Stachniss, Robust visual localization across seasons, IEEE Transactions on Robotics 34 (2) (2018) 289–302.

[21] C. Valgren, A. J. Lilienthal, SIFT, SURF and seasons: Long-term outdoor localization using local features, in: 3rd European Conference on Mobile Robots, ECMR'07, 2007, pp. 253–258.

[22] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, T. Duckett, Recurrent-OctoMap: Learning state-based map refinement for long-term semantic mapping with 3-D-Lidar data, IEEE Robotics and Automation Letters 3 (4) (2018) 3749–3756.

[23] Z. Chen, L. Liu, I. Sa, Z. Ge, M. Chli, Learning context flexible attention model for long-term visual place recognition, IEEE Robotics and Automation Letters 3 (4) (2018) 4015–4022.

[24] E. Johns, G.-Z. Yang, Feature co-occurrence maps: Appearance-based localisation throughout the day, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 3212–3218.

[25] E. Johns, G.-Z. Yang, Generative methods for long-term place recognition in dynamic scenes, International Journal of Computer Vision 106 (3) (2014) 297–314.

[26] N. Carlevaris-Bianco, R. M. Eustice, Learning visual feature descriptors for dynamic lighting conditions, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 2769–2776.

[27] W. Churchill, P. Newman, Experience-based navigation for long-term localisation, The International Journal of Robotics Research 32 (14) (2013) 1645–1661.

[28] F. Dayoub, T. Duckett, An adaptive appearance-based map for long-term topological localization of mobile robots, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2008, pp. 3364–3369.

[29] B. Bacca, J. Salvi, X. Cufí, Appearance-based mapping and localization for mobile robots using a feature stability histogram, Robotics and Autonomous Systems 59 (10) (2011) 840–857.

[30] R. C. Atkinson, R. M. Shiffrin, Human memory: A proposed system and its control processes, Vol. 2 of Psychology of Learning and Motivation, Academic Press, 1968, pp. 89–195. doi:https://doi.org/10.1016/S0079-7421(08)60422-3.

[31] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Computer Vision and Image Understanding 110 (3) (2008) 346–359.

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

[33] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395. doi:10.1145/358669.358692.
URL https://doi.org/10.1145/358669.358692
[34] Sivic, Zisserman, Video Google: A text retrieval approach to object matching in videos, in: Proceedings Ninth IEEE International Conference on Computer Vision, 2003, pp. 1470–1477 vol.2.
[35] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 2, IEEE, 2006, pp. 2161–2168.
[36] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer Science & Business Media, 2013.
[37] P. Torr, A. Zisserman, MLESAC: A new robust estimator with application to estimating image geometry, Computer Vision and Image Understanding 78 (1) (2000) 138–156.
[38] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, M. Hanheide, Long-term topological localization for service robots in dynamic environments using spectral maps, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014. doi:10.1109/IROS.2014.6943205.
[39] M. Cummins, P. Newman, FAB-MAP: Probabilistic localization and mapping in the space of appearance, The International Journal of Robotics Research 27 (6) (2008) 647–665. doi:10.1177/0278364908090961.
[40] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory 14 (3) (1968) 462–467.
[41] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An efficient alternative to SIFT or SURF, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2564–2571.
[42] S. Leutenegger, M. Chli, R. Y. Siegwart, BRISK: Binary robust invariant scalable keypoints, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2548–2555.