Object-based pose graph for dynamic indoor environments

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

DOI
10.1109/LRA.2020.3007402

Publication date
2020

Document Version
Accepted author manuscript

Published in
IEEE Robotics and Automation Letters

Object-Based Pose Graph for Dynamic Indoor Environments

Clara Gomez[1], Alejandra C. Hernandez[1], Erik Derner[2,3], Ramon Barber[1], and Robert Babuška[2,4]

***Abstract*—Relying on static representations of the environment limits the use of mapping methods in most real-world tasks. Real-world environments are dynamic and undergo changes that need to be handled through map adaptation.**

In this work, an object-based pose graph is proposed to solve the problem of mapping in indoor dynamic environments with mobile robots. In contrast to state-of-the art methods where binary classifications between movable and static objects are used, we propose a new method to capture the probability of different objects over time. Object probability represents how likely it is to find a specific object in its previous location and it gives a quantification of how movable specific objects are. In addition, grouping object probabilities according to object class allows us to evaluate the movability of different object classes.

We validate our object-based pose graph in real-world dynamic environments. Results in mapping and map adaptation with a real robot show efficient map maintenance through several mapping sessions and results in object classification according to movability show an improvement compared to binary classification.

***Index Terms*—Mapping, Dynamics, Service Robots.**

I. INTRODUCTION

REAL-WORLD dynamic environments undergo huge changes in short periods of time. Changes present a major challenge for mobile robots as maps rapidly become outdated. This is a problem that needs to be solved in order to enable mobile robots to autonomously operate in real-world environments. Our goal is to solve this problem by building a map that adapts during subsequent mapping sessions and classifies the elements of the environment according to their movability.

The dynamics of an environment can be classified as high dynamics, changes that occur while the robot is present in the environment (i.e. a person walking or a car driving through a road), or low dynamics, changes that happen while the robot is elsewhere or not sensing (i.e. returning to a room after any rearrangement of the furniture or daily register of cars in a car park). We propose a solution for the low dynamics in which the robot will register the changes between different mapping sessions using an object-based pose graph. Object-based pose graphs have been extensively used in recent years obtaining very good results for static environments [1], [2]. Low-dynamic environments have been mainly addressed through feature-based representations [3], [4], [5] and less research has focused on object-based approaches.

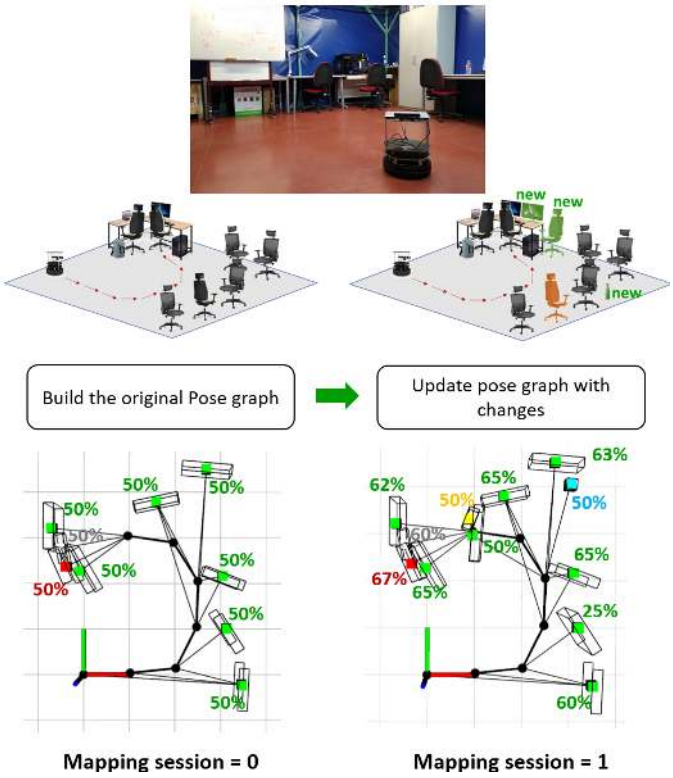

Fig. 1. Proof of concept of the dynamic mapping method in an indoor environment. First, the original object-based pose graph is built (7 chairs, 1 computer and 1 backpack were mapped). For the adaptation of the map, one chair was moved and a bottle and a screen were added. The map adapted to the changes adding the new objects and increasing or decreasing the probabilities of the old ones.

In this paper, we propose to build an object-based pose graph and maintain it during several mapping sessions. The probability of an object being present in the environment will be calculated for each session. To the best of our knowledge, probability-based approaches have been used to model the dynamics when maps are created from features but not from objects. Object probability allows us to infer the movability and classify different objects based on the long-term experience of the robot.

The proposed mapping method involves object detection and representation, building the initial pose graph and adapting

Manuscript received: February, 25, 2020; Revised April, 15, 2020; Accepted June, 12, 2020.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments.

[1]Robotics Lab, University Carlos III of Madrid, Spain. {clgomezb, alejhern, rbarber}@ing.uc3m.es

[2]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Czech Republic. {erik.derner, robert.babuska}@cvut.cz

[3] Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic.

[4] Cognitive Robotics, Delft University of Technology, The Netherlands.

Digital Object Identifier (DOI): see top of this page.

the pose graph to the changes in the environment. The main pipeline and an operation example is shown in Figure 1, where two mapping sessions are included and the map reflects the changes as inclusion of new objects and variation of the object probabilities. Our main contributions are:

- The design of a novel method to maintain object-based pose graphs in dynamic environments.
- A function to capture how static or movable an object is according to the experience of the robot.
- A fully autonomous dynamic-environment mapping system that infers the movability of different object classes and improves object classification according to their movability.

This paper is organized as follows: Section 2 surveys the relevant related work on mapping in low-dynamic environments. Section 3 presents the building stones of the method proposed, namely object detection and cuboid generation. Sections 4 and 5 describe the proposed mapping method and the adaptation of the map to changes, respectively. Finally, an experimental evaluation of the method and the conclusions drawn from this work are presented in Sections 6 and 7.

II. RELATED WORK

Methods that adapt to the changes occurring in real-world dynamic environments have received much attention from the robotics community. Most low-dynamics approaches are based on features extracted from laser scans, images or point cloud information. Some authors propose a binary classification of these features just considering if they are stable or not. In [6], a pose graph is updated to remove the scans that no longer match the environment. In such a way, the resulting map is built with the scans that belong to static objects. Similarly, in [7], a grid map, initialized with the architectonic map, is augmented with the features that persist in time.

Other authors found binary classification very limited and defined methods to measure the object movability or stability of the features. The Feature Stability Histogram (FSH) was proposed in [3], where image features are gathered for each node of a topological map. Over time, a voting scheme is used to register the local feature stability and the resulting map is built with the most stable features. Along the same line, the work presented in [4] applies a ranking function that estimates how likely a landmark is observable under the current situation. Top-ranked landmarks are stored for the resulting map. A method for grid maps in which the belief about the occupancy of a cell is represented with Hidden Markov Models is presented in [5]. The resulting map includes the change probability for each cell, which is a novelty in contrast to the aforementioned works. In [6], [7], [3], mapping is only carried out for the most stable features, which is a limitation as the information that could be inferred from the dynamic objects is overlooked. In [8], another solution is proposed based on maintaining different representations following a memory model. The sensory memory stores the most current information. An attention mechanism selects which information is moved to the Short term memory. And finally, through rehearsal, static information can be committed to the Long term memory.

Other works focus on maintaining the most updated version of the environment by having a record of the static and current dynamic elements. In [9], a pose graph based on point clouds uses matching techniques to accumulate the aligned data and remove the outdated ones. A more sophisticated approach is introduced in [10], where a pose graph is maintained using the belief of scan matches given a certain robot position and observations. A pose is removed if its belief drops below a tolerance value. These approaches, although being a solution for localization, neglect prior situations and forget about the former presence of elements and their locations. Some works solve such issues by maintaining multiple representations. The work presented in [11] keeps maps from different experiences that are evaluated simultaneously selecting the most adequate one for the current situation or creating a new one. Similarly, in [12], several representations are maintained simultaneously from multiple timescales, allowing the robot to detect patterns. Patterns between different experiences are also sought in [13], [14] through spectral representations that model the frequency of appearance of different features.

The main drawback of the aforementioned works is the lack of semantic meaning of the information stored in the map. For this reason, some authors started to focus on objects as the elements to map the environment. Relevant works in static environments are [1], [2], where robust pose graphs of indoor environments and object reconstruction are proposed. Recently, an adaptation of this work for coping with high-dynamic environments was proposed in [15]. Semantic, geometric and motion information is used for object tracking and pose estimation. Other solutions for high-dynamic environments have been proposed [16], [17], [18], [19]. In [17], a static weighing method estimates whether an object is static or not based on the Euclidean distance between object edges in two situations.

Regarding low-dynamic environments, changes at an object level can be detected inferring if they are static or movable. In [20], an approach is introduced where a map of the static environment and a database of the discovered objects is maintained over time. Both the map and the objects are 3D reconstructions that are refined as the robot discovers the environment. In [21], several representations of the environment are maintained and overlaps between objects and representations are checked for changes. An object-based pose graph is developed in [22], where the most up to date representation of the environment is maintained by merging new objects and removing old ones. In [23] tracking of objects is performed in 3D maps. When an object has disappeared from its mapped position, the system looks for it until its new location is found. An updated 3D representation of the environment is maintained over time.

In contrast to the related work, we propose to map the environment as an object-based pose graph that captures the movability of the objects. To the best of our knowledge, such works have been proposed for features but not for objects. In addition, a new definition for describing the probability of an object to be in an already-mapped position is presented. Our resulting map is a probabilistic object-based pose graph where static and movable objects are included and improved object classification is obtained.

III. OBJECT DETECTION AND CUBOID REPRESENTATION

The most common approaches for object representation in pose graphs are object reconstruction [2] and cuboid generation [19]. Both methods require an object detector that provides information on the object position and class. With this information, reconstruction approaches group the pixels or scans that belong to an object and extract the object geometry; on the contrary, cuboid generation approaches identify the 3D bounding box of the object representing the space that it occupies. Regarding limitations, object reconstruction is highly demanding in terms of memory and computing power, but it gives an individual and detailed representation of objects. Cuboid generation overlooks the specific characteristics of each object instance within the class with the advantage of reducing computational costs. In this work, a cuboid generation method is used as we are interested in classifying objects according to their class.

Object detection is performed in RGB images through ResNet-101 [24] trained with COCO Dataset [25]. Detections contain all the objects, o_i, present in the image frame I_k at time step k. Each object is defined by the detection confidence, $p(o_i|I_k)$, its 2D bounding box, $b_{i,2D}$, and the object class. 2D bounding boxes and point cloud information are used to calculate the centroid $c_i = [x_i, y_i, z_i]$ and the size of the 3D bounding box $b_{i,3D} = [w_i, h_i, d_i]$, where the object, centroid and bounding box are linked through their indices. Only objects with high detection confidence, $p(o_i|I_k) > 0.7$, are included in the map. We consider such detections reliable, without further parameter tuning, as the details of the object detection algorithm are out of the scope of this paper. Cuboid generation is shown in Figure 2.

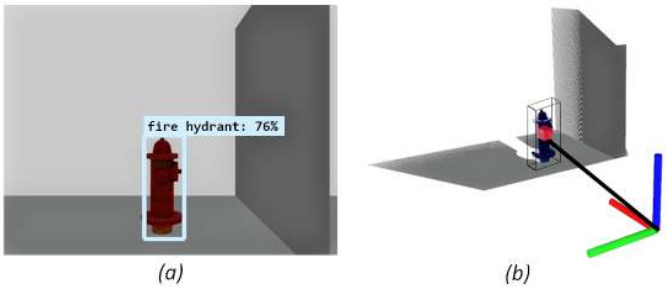

Fig. 2. Object detection and generation of object cuboid. In (a), the 2D bounding box, class and confidence of a detected object is shown. In (b), the cuboid and centroid for the object are calculated using the 2D bounding box of the object and the current point cloud.

Point cloud information is used to calculate transformations between image pixels and coordinates in the map, as presented in [26]. Such transformations are firstly used to obtain the object depth. Object depth is characterized using minimum, mean and maximum values. Minimum and maximum values are calculated for defining vertex depth and mean value for centroid depth. Minimum object depth is calculated as the median of the 2% smallest depths within the 2D bounding box of the object. Maximum object depth is calculated through an adaptation of the flood fill algorithm, starting from the minimum-object-depth pixel and recursively visiting its neighbors. A threshold, θ, is defined for the difference in depth of two neighboring pixels. If the difference in depth between the minimum-object-depth pixel and its neighbor is smaller than the threshold, the two pixels are connected and they belong to the object. Otherwise, the neighboring pixel is considered part of the background and it is discarded. Mean object depth is calculated as the mean value between minimum and maximum object depths.

3D coordinates of the object centroid are obtained using the mean object depth and pixel-to-coordinate transformation. Similarly, 3D coordinates of the bounding box vertices are estimated using the minimum and maximum object depths and the width and height of the 2D bounding box.

This method provides a representation of every object that the robot sees as a 3D cuboid characterized by its 3D bounding box and its centroid from RGB images and point cloud information.

IV. INITIAL POSE GRAPH

The initial pose graph captures the objects and trajectory as the robot explores the environment for the first time. This process implies the generation of robot poses, mapping the detected objects and connecting poses with objects. In addition, identifying whether the detected objects have been already mapped is required for pose graph consistency.

A. Building the initial pose graph

Our representation of the world is an object-based pose graph that consists of robot poses, objects, and connections between them. For pose generation, reliable information about the robot pose is assumed. In addition, this pose is used to determine when the robot has returned to an already-mapped pose.

Neighboring poses are connected and they are also connected with the objects detected from them. In addition, every time an object is detected, it is annotated with the connections to the poses from where it was seen, the object class, the probability of finding the object in that location, $p(o_i)_0$, and whether the object is active (present in the mapping session), inactive (missed in the mapping session) or unknown to be active (not visisted in the mapping session). Newly detected objects are always identified as active and their probability is set as an initial probability that depends on the detection confidence $p(o_i|I_k)$. This probability refers to how likely it is to find that object i again in the same location. For this reason, and as it is the first time that the object is detected, an initialization factor of 0.5 is used, as we do not know yet how dynamic the object is:

$$p(o_i)_0 = 0.5p(o_i|I_k) \tag{1}$$

Object-based pose graphs require to control the addition of objects as redundant mapping can become a major problem. This is solved through online object merging.

B. Object merging

Object merging involves identifying that an object detected in image frame I_l at time step l was previously seen from the same or another position in image frame I_k, where $l > k$. In this work, every object to be added to the map is analyzed according to its class and position. Firstly, two objects are only going to be merged if they belong to the same object class. Secondly, two conditions are introduced to evaluate the relation between object positions. If the centroid of the new object lies inside the cuboid of the already-mapped object, $c_j(x, y, z) \in b_{i,3D}$, they are automatically merged without any modification. If the centroid of the new object is within the area of influence, $a_{inf}(o_i)$ of the already-mapped object, the two objects are merged and the cuboid of the object is reshaped. The area of influence of an object is defined as a sphere centered in the object centroid where given the size of the object no other centroid of the same object class can be found. This sphere is defined by the diagonal of the object $\sqrt{b_{i,3D}^2}$ multiplied by a tolerance factor α, as objects from the same object class may have approximately similar sizes but not exactly the same:

$$a_{inf}(o_i) = \alpha\sqrt{b_{i,3D}^2} \tag{2}$$

If the centroid lies in the area of influence, $c_j(x, y, z) \in a_{inf}(o_i)$, the cuboid of the already-mapped object is reshaped to include the centroid of the new detection. This situation is illustrated in Figure 3.

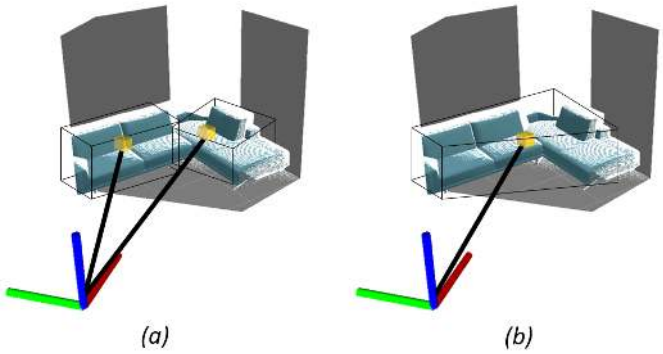

Fig. 3. Object merging for a sofa. In (a), the two instances of the object as shown. As the centroid of the second detection (right) is within the area of influence of the first detection, in (b) both instances are merged.

V. POSE GRAPH ADAPTATION OVER TIME

Every time that a new mapping session, m, starts the pose graph is updated. The robot is assumed to travel through the first mapping session path (although it can drive it partially or in different directions), and we assume that the robot can always be localized in one of the mapped poses. Updating the pose graph implies adding new objects and detecting whether the already-mapped objects are still present or have been moved/removed. While objects detected within one mapping session are updated and merged online, matching objects between different mapping sessions is performed offline when the mapping session has finished. Therefore, object probability is just evaluated and modified once and efficiency is improved as unnecessary evaluations are avoided while the robot is moving.

A. Updating already-mapped objects

As the robot moves, the expected already-mapped objects are registered. An object becomes expected when it was already mapped from the current robot pose and it is supposed to enter the frustum of the camera according to its mapped position, $c_i \in F$, where F denotes the frustum defined by the vertical and horizontal angles along with the minimum and maximum detection distances of the camera. When the mapping session has finished, the register of expected objects is compared to the objects detected during the session.

If an expected object has been seen again, its probability increases according to the following equation and it is considered to be active:

$$p(o_i)_m = \frac{s(c_i^m, c_i^{m-1})p(o_i|I_k) + \xi + p(o_i)_{m-1}}{2} \tag{3}$$

Here, $s(c_i^m, c_i^{m-1})$ refers to the similarity between the centroid position of both detections according to their Euclidean distance and ξ relates to the false-negative rate of the detector maintaining a low value for the new measurement even if the object was not detected. The similarity takes values between 0 and 1, where the higher the measure is, the closer the two detections are; and it is computed as follows:

$$s(c_i^m, c_i^{m-1}) = \frac{1}{1 + \|c_i^m - c_i^{m-1}\|_2} \tag{4}$$

If an expected object is not detected again, its probability decreases and it is considered inactive. In such a case, $p(o_i|I_k)$ is zero and 3 simplifies as follows:

$$p(o_i)_m = \frac{\xi + p(o_i)_{m-1}}{2} \tag{5}$$

If an already-mapped object was not expected (the robot has not visisted it) and it was not detected its probability is not affected by the map update, so it is labelled as unknown to be active.

B. Adding new objects

The objects to add are those ones detected by the robot that do not correspond to any of the expected objects. Before adding a new object a check is performed to confirm that the object was not mapped to another pose. If it was already mapped, a new connection is created. Otherwise, the new object is added and annotated with its probability, class and cuboid. The probability for newly added objects is calculated similarly to the initial probability (first mapping session) as described in (1).

C. Inferring object class movability

After every mapping session, m, the individual object probabilities are updated. Grouping them according to object class reveals valuable information about object movability and allows to classify static and movable objects. Movability, $M_{m,a} \in [0, 1]$, is the measure that captures whether an object class, a, is static or movable. Movability is the complement of the mean object probability for all the objects that belong to an object class, n_a, defined by:

$$M_{m,a} = 1 - \frac{\sum_{i \in a} p(o_i)_m}{n_a} \tag{6}$$

Movability for different object classes is inferred by the robot based on its own experience in the environment.

VI. EXPERIMENTAL RESULTS

Experiments were conducted in an indoor environment (15 x 6) m^2 using a Turtlebot 2 robot equipped with an Asus Xtion depth camera. The robot gathered information in 20 different mapping sessions during a month where 22 object were present in the environment: 12 chairs, 3 sofas, 2 cups, 3 bottles, 1 plant and 1 laptop.

All the experiments presented in this paper were performed with the same values for the threshold for depth, θ, tolerance factor for merging, α and false-negative rate, ξ (including the experiments shown in Figures 1, 2 and 3). The value of θ was set to 15 cm, α to 0.9 and ξ is assumed to be 0. We have empirically evaluated that this choice is suitable for different objects present in the environment and for different environments, see Table I. The first three columns in Table I refer to θ: $\overline{d}$ represents the average difference between depth of neighboring pixels for each object class (cm); $\%d^*$ refers to the percentage of differences that are greater than 15 cm; and $\%\varepsilon$ refers to the final error in size comparing the real and calculated depth for the object. Although several objects have differences between individual pixels higher than the defined threshold, this only affects the chairs with a 5.91% error (a 50 cm-wide object will be detected as 47.05 cm wide). Next three columns refer to α: min(s) column shows the minimum separation (m) between objects of the same class present in any of the environments; max(D) column shows the maximum diagonal value (m) for each of the objects classes; and finally, 0.9max(D) refers to the area of influence of the object. Although the maximum diagonal values for some objects are larger than the minimum distance (what would lead to an error), using the α value of 0.9 solves these possible errors.

TABLE I
EVALUATION OF PARAMETERS θ AND α

Object	$\overline{d}$	$\%d^*$	$\%\varepsilon$	min(s)	max(D)	0.9max(D)
chair	1.62	1.86	5.91	0.67	0.69	0.62
sofa	1.23	1.06	0	1.85	1.89	1.70
plant	1.57	1.41	0	-	0.63	0.57
cup	0.41	0	0	0.44	0.15	0.13
bottle	0.42	0	0	1.54	0.09	0.08
laptop	0.35	0	0	-	0.46	0.41

Images gathered by the robot during some of the mapping sessions are shown in Figure 4. Red boxes highlight the changes in the sample images. All the processing took place on a PC with IntelCore i7-6500U CPU@2.50GHz 12GB RAM.

Fig. 4. Sample of images captured by the robot in three different mapping sessions. Changes as movement of chairs, presence of new objects such as cups or bottles are introduced between mapping sessions (red boxes).

Experiments presented in sections VI-A and VI-B show the building and the adaptation process for the object-based pose graph presented in this work. In addition, section VI-C shows the improvement in object classification regarding movability.

A. Real-world experiments in a dynamic environment

This first experiment evaluates the performance of the mapping system and its adaptation to the changes in the map. For this purpose, two mapping sessions ($m = 0$ and $m = 1$) are evaluated in detail. The initial map, generated in $m = 0$, is shown in Figure 5 (a), the active elements for the next mapping

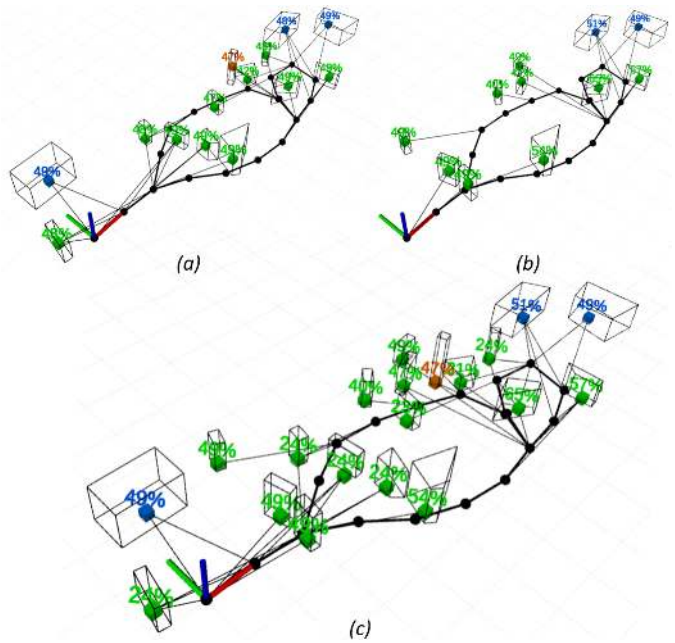

Fig. 5. Result for two mapping sessions. In (a) the objects of the first mapping session are included (10 chairs, a plant and 2 sofas). In (b), the new objects detected are shown (9 chairs, some of them were in a different position than in the previous mapping session and 2 sofas). Finally, in (c), the adaptation of the map to the new situation is shown.

session are shown in Figure 5 (b), and, finally, Figure 5 (c) shows the resulting map after the two mapping sessions. Every object is represented using its cuboid, object class (color-coded) and the connections to the poses where they were detected.

Table II shows the detail of object probabilities. Objects not entering the frustum of the camera (unknown) are listed with - and their probabilities remain constant, undetected objects (inactive) decrease their probabilities whereas detected objects (active) increase them after the two mapping sessions (forth column).

TABLE II
OBJECT PROBABILITY FOR THE TWO FIRST MAPPING SESSIONS

Object	m = 0	m = 1	m = 0 & m = 1
0	0.4919	0.00	0.2460
1	0.4954	0.00	0.2477
2	0.4901	0.5956	0.5428
3	0.4961	0.6477	0.5719
4	0.4981	0.8087	0.6534
5	0.4861	0.5414	0.5138
6	0.4961	-	0.4961
7	0.4896	0.00	0.2448
8	0.4279	0.00	0.2139
9	0.4733	0.00	0.2366
10	0.4785	-	0.4785
11	0.4856	0.00	0.2428
12	0.4864	0.00	0.2432
13	0.4904	-	0.4904
14	-	0.4975	0.4975
15	-	0.4978	0.4978
16	-	0.4787	0.4787
17	-	0.4983	0.4983
18	-	0.4030	0.4030
19	-	0.4980	0.4980

B. Long-term inference of object movability

After validating the performance of the proposed method for two mapping sessions, the map resulting from the complete set of mapping sessions is evaluated. The initial map (first mapping session) and the resulting map (all the objects mapped along their probabilities) are shown in Figure 6 (a) and (b), respectively. From the resulting map, the active map (objects present in the last mapping session) and static map (object probability $> 50\%$) can be obtained as shown in Figure 6 (c) and (d). Figure 6 (c) shows the objects that were active in the last mapping session. Figure 6 (d) shows the objects that the robot considers static after including the complete set of mapping sessions. They are the three sofas, the plant and the three chairs, which corresponds to the elements that were not moved during the experiments.

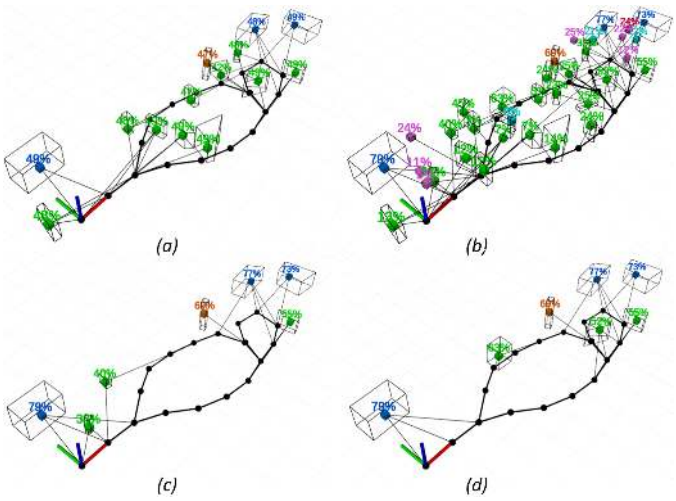

Fig. 6. Results after 20 mapping sessions and the adaptation in the environment. (a) and (b) show the resulting map for the first and last mapping sessions respectively, (c) shows the active elements for the last mapping session and (d) shows the objects that are learned as static after the 20 mapping sessions.

Updating object probabilities during 20 mapping sessions results in a polarization between the objects that have not being detected in most of the sessions (movable objects) and those that remain for almost all the sessions (static objects). As shown in Table III, object movability is scaled in a realistic fashion and the robot has effectively learned which objects are more movable (higher values of movability). The evolution of movability within all the mapping sessions is shown in Figure 7.

TABLE III
MOVABILITY ACCORDING TO OBJECT CLASS

Object class	Movability
Bottle	0.8765
Cup	0.8302
Laptop	0.7516
Chair	0.7343
Plant	0.3007
Sofa	0.2319

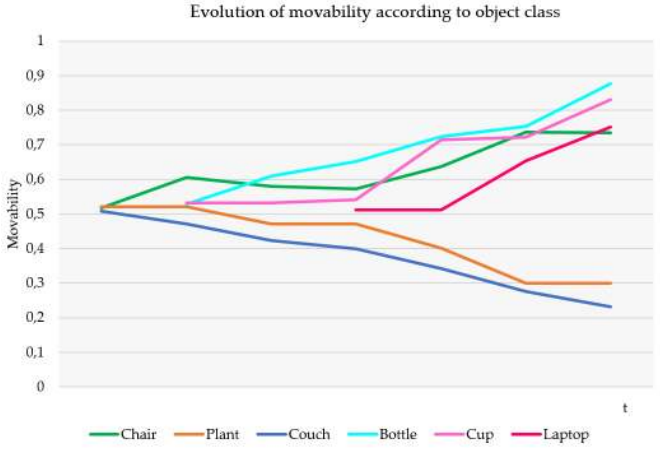

Fig. 7. Evolution of the movability for each object class during the mapping sessions.

C. Comparison to binary object classification

Comparison to binary object classification is included to further evaluate the performance of the method. Binary object classification identifies objects as movable or static. Most of the approaches that use binary classification are meant to map the static objects and discard the movable ones [21], [22], [23]. In order to replicate this behavior, only the objects that have been labeled as active or unknown for all of the mapping sessions are included in the resulting map as they are supposed

to be static. Figure 8 (a) and (b) show the resulting static map for binary classification and the static map of the proposed method after evaluating the complete set of mapping sessions, respectively.

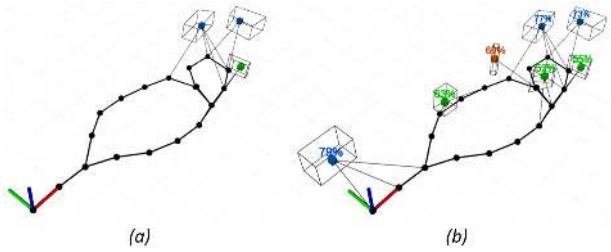

Fig. 8. Comparison of object classification according to the movability of objects between the binary classification method (a) and using the proposed method (b) after evaluating all the mapping sessions.

As shown in Figure 8, the method proposed outperforms binary object classification as object probability increases the robustness of object classification. Binary classification overlooks static objects just because they were not detected in one mapping session, obtaining a 42.85% of successful static objects mapped in contrast to the 100% of the method proposed. Therefore, robustness is increased in our method thanks to employing the proposed object movability calculation in object classification. In addition, binary classification can just infer about static and movable objects, but it does not give any insight in the degree of movability.

D. Discussion

Quantitative results of the proposed method and comparison to binary classification have been presented in this section. Comparison to other methods that define the movability of the environment elements is not appropriate as those works map the environment using features instead of objects. Although both methods pursue the same objective, as they do not use equivalent information, methods based on feature descriptors cannot be applied to objects. The only possible way to compare these approaches is through the improvement in localization, which is beyond the scope of this paper. Therefore, we give an extensive discussion on how our method presents a contribution in the light of these prior works.

Feature-based approaches [3], [4], [5] gather information from the salient regions of images, scans or point clouds. Features can be merged, removed or assigned a probability that could be registered for dynamic environments, but the meaning of these features is not easy to transfer to the real world. Features represent an abstraction level that allows to know which regions of the environment are prone to changes, but they need a second step to determine which elements are located in that region to gain environment understanding. In contrast, object-based approaches implicitly provide environment understanding, as changes are directly associated to objects. They also share the advantages of feature-based approaches, as they determine the regions of the environment that change more.

Although it was not possible to compare to any other specific method, the movability of objects can be calculated with other well-known methods, such as Bayesian filtering. Here we briefly discuss the comparison of our method with a standard Kalman filter [27]. Kalman filters can estimate the belief of a specific object remaining static or being movable. As proposed for our method, object class movability can be calculated by grouping the objects probabilities (or beliefs) for each object class. Kalman filter and our method can be compared through the results obtained regarding object class movability as shown in Figure 9. Our method (red) and three instances of Kalman filter (blue) are compared for a static object and a highly movable object. The process model for the Kalman filter is initialized with $\mu_0 = 0.5$ and $\Sigma_0 = 0.2$, and it is assumed to be static if measurements are not received ($\mu_t = \mu_{t-1}$ and $\Sigma_t = \Sigma_{t-1}$). The measurement model is defined by each new observation and the measurement noise covariance, R_t. The results show that noisier measurements (higher R_t) lead to a slow evolution of object movability, being more difficult to distinguish between static and movable objects. On the contrary, more precise measurements (lower R_t) lead to a more polarized estimation of object movability, especially for movable objects. Our method performs similarly to a Kalman filter of $R_t = 0.2$ for increasing movability (bottle). However, our performance for static objects is increased (sofa), as the system infers faster that the object is static. For these reasons, we conclude that our method performs better than Bayesian filtering for the task of object movability estimation.

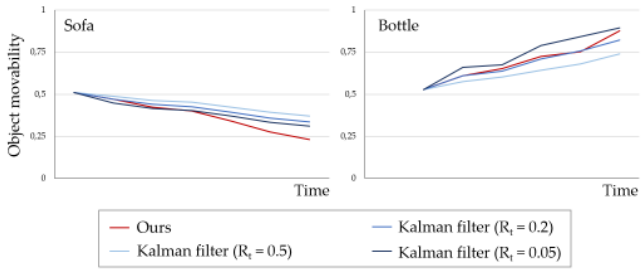

Fig. 9. Comparison of object movability between the Kalman filter and our method.

Some advantages can also be found regarding the resulting map. In our method, object probability and active objects are maintained through the different mapping sessions resulting in an improvement compared to other works. Active elements for each mapping session are included, as for [9], [10]. Also the static and dynamic maps of the environment, as for [6], [7]. Comparing the different representations, we can say that the resulting map for the proposed method gives more complete and representative information of the environment than other state-of-the-art methods.

VII. CONCLUSIONS AND FUTURE WORK

We have proposed a method for mapping and map adaptation through object-based pose graphs for low dynamic indoor environments. This new method calculates and maintains object probability depending on whether an object is seen again or not and capturing whether it is static or movable. As shown in the experimental results, including object probability improves the resulting map. In addition, it provides the

robot with a realistic estimation of the movability of objects according to its class gathered from its own experience that outperforms object classification using a binary method.

Overall, this paper strengthens the idea that real-world environments have to be treated as dynamic environments consisting of objects that may be more or less movable and important information can be obtained from capturing their movability. The method proposed in this paper allows for a straightforward extension to improve other tasks such as localization or object search. Therefore, future research includes developing a localization algorithm that builds upon the method proposed in this work and improves the localization of the robot thanks to the individual object probabilities and object class movability. In addition, improvements to this work include the recalculation of the path (not only the objects), as the original path could become impassable and new paths could be created to overcome this situation.

ACKNOWLEDGMENT

This research has 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). 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.

REFERENCES

[1] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1352–1359.

[2] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 international conference on 3D vision (3DV)*. IEEE, 2018, pp. 32–41.

[3] B. Bacca, J. Salvi, and X. Cufí, "Appearance-based mapping and localization for mobile robots using a feature stability histogram," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 840–857, 2011.

[4] M. Bürki, M. Dymczyk, I. Gilitschenski, C. Cadena, R. Siegwart, and J. Nieto, "Map management for efficient long-term visual localization in outdoor environments," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 682–688.

[5] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[6] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1871–1878.

[7] J. Ginés, F. Martín, V. Matellán, F. J. Lera, and J. Balsa, "Dynamics maps for long-term autonomy," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2017, pp. 85–90.

[8] F. Dayoub and 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.

[9] M. T. Lázaro, R. Capobianco, and G. Grisetti, "Efficient long-term mapping in dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 153–160.

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

[11] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.

[12] P. Biber, T. Duckett *et al.*, "Dynamic maps for long-term operation of mobile service robots," in *Robotics: science and systems*, 2005, pp. 17–24.

[13] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localisation for service robots in dynamic environments using spectral maps," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4537–4542.

[14] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.

[15] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic SLAM," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5231–5237.

[16] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4471–4478.

[17] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2263–2270, 2017.

[18] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[19] "Cubeslam: Monocular 3D] object slam, author=Yang, Shichao and Scherer, Sebastian, journal=IEEE Transactions on Robotics, volume=35, number=4, pages=925–938, year=2019, publisher=IEEE."

[20] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, and 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.

[21] J. Mason and B. Marthi, "An object-based semantic world model for long-term change detection and semantic querying," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3851–3858.

[22] D. Wilbers, L. Rumberg, and C. Stachniss, "Approximating marginalization with sparse global priors for sliding window SLAM-graphs," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 25–31.

[23] N. Bore, J. Ekekrantz, P. Jensfelt, and J. Folkesson, "Detection and tracking of general movable objects in large three-dimensional maps," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 231–247, 2018.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[26] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[27] C. K. Chui, G. Chen *et al.*, *Kalman filtering*. Springer, 2017.