Journal of Intelligent & Robotic Systems (2023) 109: (2023) 109:95
https://doi.org/10.1007/s10846-023-02019-6

REGULAR PAPER

Visual Localization and Mapping in Dynamic and Changing Environments

João Carlos Virgolino Soares[1,3] · Vivian Suzano Medeiros[2] · Gabriel Fischer Abati[3] · Marcelo Becker[2] · Glauco Caurin[4] · Marcelo Gattass[5] · Marco Antonio Meggiolaro[3]

Received: 18 March 2023 / Accepted: 11 November 2023 / Published online: 15 December 2023

Abstract
The real-world deployment of fully autonomous mobile robots depends on a robust simultaneous localization and mapping (SLAM) system, capable of handling dynamic environments, where objects are moving in front of the robot, and changing environments, where objects are moved or replaced after the robot has already mapped the scene. This paper proposes Changing-SLAM, a method for robust Visual SLAM in both dynamic and changing environments. This is achieved by using a Bayesian filter combined with a long-term data association algorithm. Also, it employs an efficient algorithm for dynamic keypoints filtering based on object detection that correctly identifies features inside the bounding box that are not dynamic, preventing a depletion of features that could cause lost tracks. Furthermore, a new dataset was developed with RGB-D data specially designed for the evaluation of changing environments on an object level, called PUC-USP dataset. Six sequences were created using a mobile robot, an RGB-D camera and a motion capture system. The sequences were designed to capture different scenarios that could lead to a tracking failure or map corruption. Changing-SLAM does not assume a given camera pose or a known map, being also able to operate in real time. The proposed method was evaluated using benchmark datasets and compared with other state-of-the-art methods, proving to be highly accurate.

Keywords SLAM · Object detection · Segmentation and categorization · Localization · RGB-D perception

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics, especially considering the challenges of real-world scenarios, such as the presence of dynamic objects and environments where objects can be constantly removed, added or moved to different locations.

There are several visual-SLAM systems in the literature with high efficiency and accuracy, such as PTAM [1] and ORB-SLAM [2], although these systems were not designed to operate in dynamic environments. Recent works have presented visual SLAM systems able to overcome this limitation, such as DynaSLAM [3], DS-SLAM [4], SaD-SLAM [5], and DOTMask [6]. However, these methods do not have specific mechanisms to handle other types of dynamic factors that can happen in a real environment, such as changing scenes, where objects are moved outside the field of view of the robot, after the scene had already been mapped, and are not guaranteed to work in such scenarios.

Dynamic changes in the scene usually cause an immediate drift in the pose estimation, but the changing environment problem can cause a wrong loop closure detection. Several works [7–9] proposed robust graph optimization algorithms to deal with wrong loop closures, but Lee and Myung [10] showed that traditional robust graph optimization approaches cannot deal with the problem of changing environments. They have shown that seeing an object that was previously mapped and changed its position caused the graph optimization process to fail, even using robust techniques do prevent wrong loop closures such as Max-mixture [7] or Dynamic covariance scaling [8]. They proposed a solution dealing with the graph structure in the back-end. However, they only dealt with poses in the 2D plane. This work has a different approach from Lee and Myung [10], preventing the graph to be created with wrong loop closures, instead of removing the loop closures after they were already created.

There are methods that deal with changing environments that consider the pose of the camera known, i.e., do not

✉ João Carlos Virgolino Soares
virgolinosoares@gmail.com

Extended author information available on the last page of the article

perform SLAM, but mapping with known poses [11]. Other methods [12] perform localization in changing environments with a known map. DXSLAM [13] has an increased robustness to changing environments, but fails in regular dynamic environments. In general, the SLAM methods found in the literature that are robust to both dynamic and changing environments only perform 2D SLAM using LiDAR fused with other sensors such as IMU and odometry [14].

In this context, this paper proposes a real-time visual SLAM system able to work with highly dynamic environments and changing environments. A combination of a robust object tracker and a filtering algorithm enables our visual SLAM system to perform well in highly dynamic environments containing moving objects. Also, the system maintains a semantic map of the environment, updating the belief about the poses of objects over time, making it also robust to changing environments.

1.1 Contributions

This work proposes Changing-SLAM, a real-time visual SLAM system based on ORB-SLAM3 [15] that is robust to both dynamic and changing environments, with the following contributions:

1. *Robustness to highly dynamic environments*: The proposed method uses a robust keypoint classification algorithm that filters a priori dynamic objects and uses an Extended Kalman Filter to track movable objects in the scene. The problem of feature depletion caused by filtering features from the background in the bounding boxes is solved with a fast and reliable method, using statistical data of the depth in each bounding box, called feature repopulation.
2. *Robustness to changing environments*: The system uses 3D mapped points derived from feature detection, combined with the output of an object detector to determine the 3D centroid of the objects in the scene, and create an object-level semantic map that maintains a belief about the pose of each mapped object. This results in a real-time 3D object detection using a semantic point clustering approach, without the need for instance or panoptic segmentation or an off-the-shelf 3D object detector. A robust long-term data association is also proposed, using the object's centroid. The state of the objects in the map is updated using a Bayesian filter. Different from other approaches found in the literature, the proposed method does not assume a known camera pose, nor a known map *a priori*.
3. *The PUC-USP dataset for changing environments*: Public datasets are fundamental elements for the evolution of SLAM systems. In contrast to other datasets publicly available, this work presents a dataset especially designed for the evaluation of the robustness of visual SLAM methods in changing environments. The data is collected using an RGB-D camera attached to a mobile robot, while a motion capture system is used to generate the ground-truth. It consists of sequences recorded in an indoor environment showing simple and challenging situations: vanishing objects, objects that are moved to different positions and replaced objects. Evaluation metrics for the estimated trajectory show that the proposed sequences could lead to failure in pose estimation for SLAM systems not robust to changing environments.

Changing-SLAM is evaluated using benchmark datasets such as the TUM dataset and the PUC-USP dataset for changing environments, and compared with several state-of-the-art methods, including ORB-SLAM3, DynaSLAM, SaD-SLAM, DOTMask and DXSLAM, achieving better camera localization accuracy in both dynamic and changing environments.

2 Preliminaries

This section briefly presents the main concepts of the ORB-SLAM system, used as a basis for this work, that are used in the following sections.

2.1 ORB-SLAM Overview

ORB-SLAM [2] is a state-of-the-art visual-SLAM system based on ORB [16] feature detection that has high accuracy and efficiency. It has three versions. The first one was designed only for monocular cameras, the second works with monocular, stereo or RGB-D cameras, and the third version [15] has a multi-mapping framework and can be integrated with IMU sensors.

It has three threads running in parallel: tracking, local mapping, and loop closing. In the tracking thread, the system tracks the camera's pose by matching ORB features between the current and the past frame. The mapping thread creates a sparse map of 3D points generated using the tracked ORB features, and the loop closing thread identifies when the camera revisits a previously visited location.

ORB-SLAM3 includes a multi-mapping representation called Atlas, and a map-merging system that runs in the loop closing thread. Atlas can store a set of disconnected maps, and merge them when a loop is detected. In this work, we use the RGB-D version of ORB-SLAM3.

2.2 Keyframes

Keyframe is a concept used in ORB-SLAM to maintain a consistent and efficient map. It consists of a specific frame captured by the camera, strategically chosen to represent significant parts of the trajectory. The selection of keyframes is based on several criteria such as changes in motion, time intervals and scene complexity. Each keyframe stores the camera's pose and the ORB features extracted in that frame.

After a new keyframe is chosen, the system searches for a loop closure using a place recognition algorithm.

2.3 MapPoints

MapPoints are one of the fundamental elements in ORB-SLAM. They are created from the ORB features detected in each frame and represent 3D points in the environment that have been observed by the camera. They are also part of the map built by the system. They are used for all tasks, including camera tracking, local optimization, and loop closure.

MapObjects are structures proposed in this work to give semantic and a common geometric meaning to a group of MapPoints that belong to the same structure.

2.4 Graph Optimization

In ORB-SLAM, the SLAM problem is represented as a graph consisting of nodes and edges. The nodes are keyframe poses, and the edges are geometric constraints between two keyframes, which are observations of the same map points.

After a loop closure, the camera poses and MapPoint positions are jointly optimized. Thus, both map and camera trajectory are refined, which maintain the accuracy of the system, reducing the drift caused by visual odometry. This consistency, however, is only achievable in static environments.

3 Related Work

This section presents a review of the latest research in visual localization and mapping for autonomous mobile robotics applications, including dynamic and changing environments.

3.1 SLAM in Dynamic Environments

Recent approaches to Visual SLAM focus on the highly dynamic environment problem. Some of them [17–19] rely on object detection combined with a dynamic feature point removal algorithm. Xiao et al. proposed the Dynamic SLAM [19] system, which uses SSD [20] object detection to filter dynamic features. They proposed a semantic correction module to create a mask with the same size of the image to map static and dynamic points, and a selective tracking algorithm to eliminate dynamic objects. The main problem with this approach is that it could lead to feature depletion and lost tracks under certain conditions, such as when a person is too close to the camera, or in a scene with many people. This happens because when the keypoints inside the bounding box are filtered, some keypoints that belong to the background are also filtered. SGC-VSLAM [18] handled this problem by using optical flow and computing the fundamental matrix between two frames to decide which keypoints belong to objects, which is computationally demanding. Detect-SLAM [17] employs SSD object detection to remove dynamic features in the scene, but only on keyframes, to overcome the slow inference time of the network. It uses GrabCut [21] for generating segmentation masks that separate the object from the background inside the bounding box. However, background removal requires a few iterations. In this work, a solution to this problem is proposed in Section 4.1, with a robust and fast feature repopulation algorithm for object detectors.

Another approach to solving the problem of feature depletion is to use an instance segmentation framework to differentiate objects from the background, but the inference time of instance segmentation networks is very high. DynaSLAM [3] uses the Mask R-CNN [22] instance segmentation framework to obtain the pixel-wise information of people in the scene, using it to filter a priori dynamic features. Despite its high accuracy and robustness, it cannot perform real-time due to the high computational requirement of the Mask R-CNN framework. Similarly, DP-SLAM [23] combines the semantic information of Mask R-CNN with a geometric approach based on epipolar geometry and probability propagation to classify dynamic keypoints. SaD-SLAM [5] combines depth information and Mask R-CNN instance segmentation to find dynamic features in the image. Each feature point is individually classified as static, dynamic, or static and movable. SaD-SLAM has a high accuracy, higher than DynaSLAM [3] in some scenarios. Its main drawback is that the semantic segmentation is processed offline.

DOTMask [6] uses instance segmentation to obtain the pixel-wise information of the objects in the image, and an Extended Kalman Filter to track these objects. Their aim was to provide a faster SLAM system in exchange of a lower accuracy, in comparison with DynaSLAM, for example. The main problem with this approach is that the use of instance segmentation makes it still too slow, and the accuracy is considerably lower than SaD-SLAM [5] or DynaSLAM [3].

Ji et al. [24] proposed a faster Semantic RGB-D SLAM method for dynamic environments extracting semantic information only from keyframes. Also, they combined K-Means with depth reprojection to identify unknown moving objects in the other frames. Despite achieving an accuracy comparable with DynaSLAM with less computational effort, their tracking thread runs at approximately 13 FPS.

Some works consider people as *a priori* dynamic objects, such as DynaSLAM and Crowd-SLAM [25]. Crowd-SLAM is an open-source visual SLAM system for crowded environments based on ORB-SLAM2 [26] that uses a custom YOLO [27] Tiny network specialized in people detection. However, it only removes dynamic features from people in the environment, not other objects. The assumption of considering people as dynamic *a priori* may seem strong, but

in reality people are dynamic by nature and they eventually move. Mapping a scenario where most people are static for a long period is unrealistic. Furthermore, even if people in a scene are static, mapping them would lead to a future wrong loop closure when revisiting that scene after they have moved. One way to overcome this issue is to use features from static people only for tracking purposes, as done in SaD-SLAM [5].

3.2 Dataset for Visual SLAM in Changing Environments

Datasets and benchmarks are very important for the advances of SLAM research, as they provide an accessible way for comparing multiple methodologies and evaluate them with clear criteria. There are several datasets for visual SLAM in the literature, each one focused on a different problem, with different types of raw data and ground-truth.

The KITTI dataset [28] is used for the evaluation of several outdoor problems, including visual odometry, visual SLAM, multi-object tracking, segmentation, among others. It contains monocular, stereo and RGB-D data.

The TUM RGB-D dataset [29] is one of the most used for evaluating visual SLAM systems. It has 39 sequences of static scenarios, scenes with dynamic objects, with low texture, among others. It uses two evaluation metrics: the absolute trajectory error, which is suited to evaluate SLAM systems, and the relative pose error, which is suited to evaluate visual odometry drift. The ground-truth was made using a motion capture system. Similar to the TUM dataset is the Bonn RGB-D [30]. It uses the same evaluation metrics from the TUM dataset, but with the focus on highly dynamic scenarios.

The previously mentioned datasets are not designed to deal with the changing environment problem, but rather focused on dynamic objects appearing in front of the camera. On the other hand, the OpenLORIS-Scene Dataset [31] was developed for real environments with several challenges that were not embraced by past datasets, such as changing environments, changing view point, and illumination.

Zhao et al. [14] proposed a framework for lifelong localization and 2D mapping, and released a dataset with several sequences of indoor and outdoor changing environments, such as markets, parking lots and offices. The dataset contains 2D and 3D LIDAR, IMU and wheel encoder data. However, their dataset does not include RGB-D data. Furthermore, their ground-truth is not made using external measurement sensors, such as a motion capture system.

The majority of SLAM systems that deal with changing environments do not rely on publicly available datasets for performance evaluation. Most of them carry out their own experiments to record data sequences more suited to the changes they expect to handle [10–12, 32–35]. These experiments require the use of several sensors and a platform for data collection, which can be expensive and time-consuming. A publicly available dataset focused on changing environments would greatly contribute for the advance of this field.

3.3 Visual SLAM in Changing Environments

One of the situations usually not considered in methods for dynamic environments, such as [3–5], is when a change happens after the robot has already mapped the scene. When it revisits the scene, some objects are in different locations, some are missing, and new objects may have appeared. This is often referred in the literature as SLAM in low dynamic environments [11, 36], semi-static environments [32], changing environments [14], or simply long-term mapping [37].

The term "changing environments" was chosen as the more appropriate for the task, as "low dynamic" or "semi-static" can be used in the context of a scene with objects moving slowly in front of the camera, and "long-term mapping" emphasizes the scalability issue.

An early solution to this problem was proposed by Walcott-Bryant et al. [36] in 2012. They proposed a method for planar indoor environments with robots using laser scanners in which the dynamic pose-graph could be edited, removing poses according to scan matching results.

Lee and Myung [10] showed through experiments that the wrong loop closures caused by a moved object could not be solved by pose-graph optimization techniques robust to outliers, such as Switchable Constraints [9], Max Mixtures [7] or Dynamic Covariance Scaling [8].

Rosen et al. [32] proposed a method to model environmental change of features over time, called feature persistence, using a recursive Bayesian estimator, but their method was not actually implemented in a visual SLAM system. Hashemifar and Dantu [33] extended Rosen's formulation, incorporating the persistence filter into ORB-SLAM and testing in a real environment. However, their method has a high accuracy error in dynamic environments.

Gomez et al. [11] developed a method for dealing with changing environments on an object level. To create a 3D bounding box of an object, they use 2D object detection and point cloud to estimate the centroid position and object dimensions. They use a floodfill algorithm and the median of the 2% smallest depths within the 2D bounding box to extrapolate the maximum and minimum depths of the object. Also, they create an object-based pose graph, connecting the robot poses and objects. The graph is updated computing the probability of finding the object in that location based on new measurements. The main drawback of their formulation is that the robot always revisits the same locations to update the object-graph. Thus, they do not perform SLAM, but mapping with known poses.

Zhao et al. [14] proposed a framework for lifelong localization and 2D mapping, tracking the changes in the scene

and maintaining an updated map accordingly through a technique called pose-graph refinement. Their method uses IMU, wheel encoders and LiDAR measurements. Lazaro et al. [34] also proposed a method for changing environments using laser scans, but focusing only on 2D mapping and localization.

Derner et al. [12] proposed a method for visual localization in changing environments. Their method uses a previously built visual database, used to perform matching against query images to determine the pose of the robot.

Schmid et al. [35] proposed a method for mapping in changing environments using panoptic segmentation to build and maintain volumetric maps during operation, receiving robot poses from an external estimator.

DXSLAM [13] is a visual SLAM method that uses features from a deep convolutional neural network. Despite considerably improving robustness in changing environments, deep features alone did not improve robustness in dynamic environments. Xie et al. [38] proposed a learning-based method to perform only localization in changing environments. Adkins et al. [39] proposed a localization system for long-term operations by modeling the likelihood of the poses of movable objects. Their focus, however, was in LiDAR-based approaches.

In general, most of the previous works that deal with changing environments only perform localization in a known map [11, 12, 38, 39], or perform mapping with known poses [35], i.e., do not perform SLAM. The systems presented in [14, 34] focus specifically on 2D mapping and/or localization using LiDAR. DXSLAM [13] is a 3D visual SLAM system that handles changing environments but is not robust to highly dynamic environments. The persistence filter developed in [32] was designed to handle changing environments and was implemented for ORB-SLAM in [33]; however, it also presents a high accuracy error in highly dynamic environments. In contrast to the previous work, Changing-SLAM performs 3D visual SLAM and it is robust to both changing and dynamic environments.

4 Methodology

Changing-SLAM works by giving semantic and temporal meaning to mapped points in the environment, either acting as a pre-filter of dynamic points to prevent odometry drift, or as a map management system to prevent wrong loop closures. In the proposed method, an object belief map is created and updated, making the SLAM process occur in two levels. Within the high level (objects), a Bayesian filter is used to create a belief map about the poses of objects in the map. This belief map decides which features are used for the low-level step. Within the low level (points), the SLAM problem is solved using the feature-based methods proposed by ORB-SLAM3, adapted for dynamic environments. This approach results in a reliable tracking system, robust to changes in the map, and a semantic map in an object-level that can be used for other problems such as autonomous navigation.

Figure 1 shows the framework of the proposed methodology. The system is built on ORB-SLAM3, and it is composed of four threads running in parallel: Object Detection, Tracking, Local Mapping, and Loop Closing. Changing-SLAM modifies all three threads of ORB-SLAM3 and adds a new thread for object detection. ORB features are extracted from the RGB image in the tracking thread, and each associated keypoint is initially classified as dynamic, movable or static, according to the semantic information provided by the object detection thread. A feature repopulation algorithm is proposed to differentiate object features from background ones, and features from people are filtered *a priori*.

The local mapping thread adds new keyframes and points to the map. The MapPoints, created from the detected and classified features, generate MapObjects with a semantic class and an unique ID. An Extended Kalman Filter is used to track the objects and predict their state, based on their ID. If an object has a velocity above a threshold, its keypoints are filtered from the image.

The graph-optimization remains the same of ORB-SLAM3. The output is the pose of the camera frame by frame, processed with filtered sensor information, and the sparse map clear of outliers.

Changing-SLAM explores one of the main novelties of ORB-SLAM3, the Atlas framework [40], explained in Section 2. This considerably improves the SLAM solution in scenarios with lost tracks. The proposed approach modifies Atlas to include storage and operations with MapObjects.

A long-term data association is also proposed to decide whether an object detected in the current frame is already in the map, or if it is a new object. The long-term data association is also responsible for updating the belief about the persistence of the objects in the environment. Finally, MapPoints associated to MapObjects with a low belief are filtered from the loop closing and graph-optimization steps.

4.1 Keypoint Classification

The keypoints detected in the tracking thread are initially classified into three categories: dynamic, movable or static. All keypoints belonging to people are classified as *a priori* dynamic, keypoints that belong to objects are classified as movable, and the keypoints from the background are static.

Different from SaD-SLAM [5], two keypoints that belong to the same object cannot have different classifications. This improves the speed of the process, because evaluating the dynamics of each individual keypoint can be unfeasible in real time. Figure 2 shows an example of the initial keypoint

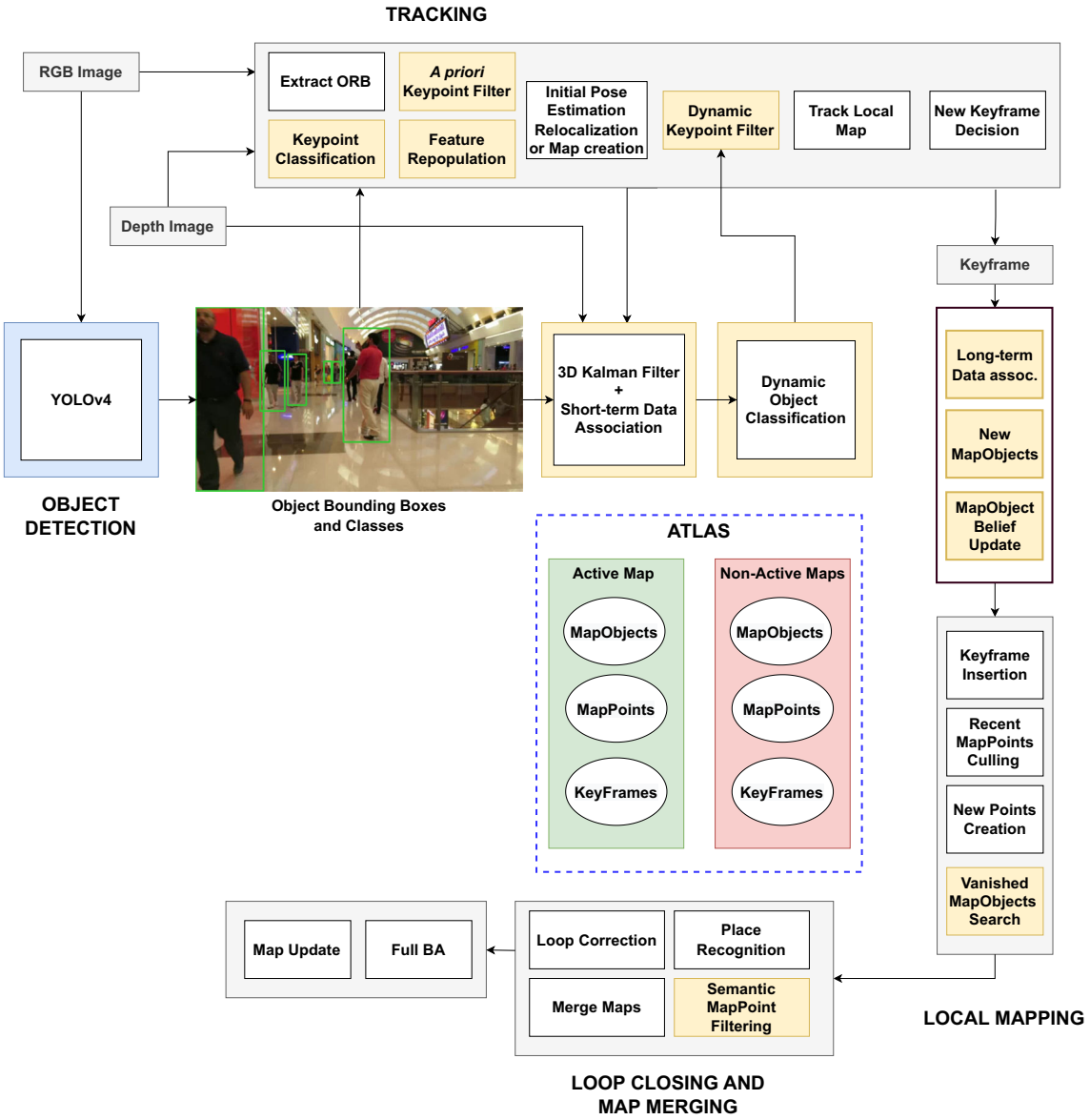


Fig. 1 Main components of Changing-SLAM. The system is built on ORB-SLAM3, with modifications in the threads for Tracking, Local Mapping, Loop Closing, and an additional thread for Object Detection. The new modules specific for the proposed methodology are highlighted in yellow

classification being performed in a frame with two people and one chair.

Using object detection for feature removal can lead to a depletion of features, especially when there are many dynamic objects in the scene, or when a dynamic object occupies a large portion of the image. Instead of using computationally expensive methods based on epipolar geometry and RANSAC, this work presents an efficient method to correctly classify the features that belong to the bounding box but are not dynamic, called feature repopulation.

In each bounding box the median, mean, maximum and minimum pixel depth values are extracted, as well as the standard deviation. With this information, together with the intersection over union (IoU) matrix, it is possible to evaluate whether the detected object is being occluded, and to differentiate the object from the background.

To correctly classify the keypoints, it is necessary to consider a few possible conditions. If the object is not being occluded, then the classification is straightforward. Keypoints with a depth greater than the minimum depth plus a threshold value are considered belonging to the background. The threshold depends on the class of the object. If the object is being partially occluded by another known detected object, the depths inside the overlapping area are not considered. This occlusion is evaluated by calculating the IoU between the bounding boxes of the frame. The main problem arises

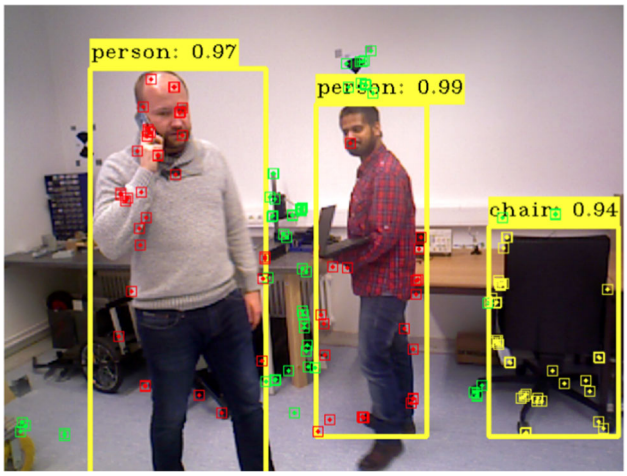


Fig. 2 Initial Keypoint Classification. Dynamic keypoints are red, movable keypoints are yellow, and static keypoints are green. Note that all keypoints inside the bounding box have the same classification, even if they are part of the background

when a non-labeled or non-detectable object is occluding the target object. This problem is identified when the standard deviation of the depths is too high, or if the median depth is higher than the depth of the center of the bounding box. If this happens, it means that the center of the bounding box is being occluded. In this last scenario, the keypoints of this object are not considered. Figures 3a and b show the result of the feature repopulation technique. In Fig. 3a, all keypoints inside the bounding box are filtered out. However, in Fig. 3b, just the keypoints that belong to people are filtered out, while the keypoints from the background are kept.

The number of the class associated with each keypoint is the same number of that class in the COCO dataset [41]. For example, "person" is 0, "tvmonitor" is 62, and "chair" is 56. When a keypoint has its class set as -1, it means that the keypoint belongs to the background.

4.2 Map Objects

A MapObject is an element created to represent a group of MapPoints that belong to the same entity, with a common body and class. In the proposed methodology, besides all other pieces of information needed for the ORB-SLAM3 framework, MapPoints store a 3D position $X_{w,i}$ in the world coordinates, the class, and the ID of the MapObject associated with the MapPoint. When the MapPoint is created, it receives the class and ID of the MapObject associated with the bounding box where the keypoint was located. Each MapObject stores the first bounding box, the current bounding box, class, list of associated MapPoints, the 3D position in world coordinates, the unique global ID, the belief of persistence, and the 3D bounding box dimensions.

The belief of persistence is the numerical value that will determine whether the MapObject, and its associated MapPoints, will remain active in the map. Details of its initialization and update are given in Sections 4.6 and 4.7.

The 3D position is obtained by computing the centroid of the associated MapPoints. The maximum object dimensions are obtained by computing the median of the 5% maximum and minimum x, y and z coordinates of the associated MapPoints.

4.3 Short-term Data Association

The short-term data association evaluates if the new bounding boxes detected in the current frame correspond to the

(a) Without feature repopulation (b) With feature repopulation

Fig. 3 *A priori* people keypoint filtering proposed in this methodology. The figure shows the effect of the feature repopulation technique that keeps keypoints that belong to the background but are located inside the bounding box

bounding boxes of the MapObjects present in the last frame. This is done using the IoU.

In each new frame, for every bounding box the IoU is computed with all detections of the same class in the previous frame. If no matches are found, a new MapObject instance is created. When a new Keyframe is created, the system checks whether a tracked MapObject has new associated MapPoints. If it has, then its pose is updated.

4.4 Object Tracking

The position of a MapObject is given by the 3D position of its centroid. The position of the centroid is obtained through the position of each associated MapPoint. With the short-term data association algorithm, presented in Section 4.3, each new detection is matched with all MapObjects from the last frame, which allows the possibility to track objects over time. In order to filter dynamic keypoints, it is necessary to infer if the tracked objects are moving or not.

The Extended Kalman Filter (EKF) is a non-linear state estimator that considers motion and observations corrupted by a zero mean Gaussian noise. The objective of the EKF is to obtain the best estimate of $\mathbf{x}$ given the measurements $\mathbf{z}$. An EKF is used to track each MapObject, in order to predict which one is moving, and it is initialized for each new MapObject. The state of the object is defined as its 3D position and velocity in world frame, as stated by

$$\mathbf{x} = [x \;\; y \;\; z \;\; \dot{x} \;\; \dot{y} \;\; \dot{z}]^T \tag{1}$$

The prediction step at frame k is given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} \tag{2}$$

where k is the current frame, $k-1$ is the last frame, and $\mathbf{F}$ is given by

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{3}$$

where Δt is the time between predictions, $\mathbf{I_3}$ is a 3x3 identity matrix, and $\mathbf{0_3}$ is a 3x3 matrix with zeros. The state covariance is estimated as stated by

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q} \tag{4}$$

where $\mathbf{P}_0 = \mathbf{I}$. The process covariance matrix $\mathbf{Q}$ is defined based on the random-acceleration model [42]:

$$\mathbf{G} = \begin{bmatrix} \frac{\Delta t^2}{2}\mathbf{I}_3 & \Delta t \mathbf{I}_3 \end{bmatrix}^T \tag{5}$$

$$\mathbf{Q} = \mathbf{G} \begin{bmatrix} \sigma_{a_x}^2 & 0 & 0 \\ 0 & \sigma_{a_y}^2 & 0 \\ 0 & 0 & \sigma_{a_z}^2 \end{bmatrix} \mathbf{G}^T \tag{6}$$

where $\sigma_{a_x}^2$, $\sigma_{a_y}^2$ and $\sigma_{a_z}^2$ are the covariance values for each acceleration component, which are empirically assigned based on the class of the object. For highly dynamic objects, such as "person", the assigned covariance values were $\sigma_{a_x} = \sigma_{a_y} = \sigma_{a_z} = 0.5$; for movable objects, such as "chair", they were defined as $\sigma_{a_x} = \sigma_{a_y} = \sigma_{a_z} = 0.02$, and $\sigma_{a_x} = \sigma_{a_y} = \sigma_{a_z} = 0.01$ for other objects.

The measurement update is given by

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \hat{\mathbf{h}}(\hat{\mathbf{x}}_{k|k-1}) \tag{7}$$

where $\mathbf{z}_k$ corresponds to the measured coordinates of the object's centroid

$$\mathbf{z}_k = \begin{bmatrix} \frac{(\mu_x - c_x)z}{f_x} \\ \frac{(\mu_y - c_y)z}{f_y} \\ z \end{bmatrix} \tag{8}$$

in which c_x, c_y, f_x and f_y are the camera intrinsic parameters, μ_x and μ_y are the image coordinates of the centroid of the MapObject, and z is the measured depth. The $\hat{\mathbf{h}}$ function transforms the predicted state from the world frame to the camera frame

$$\hat{\mathbf{h}}(\hat{\mathbf{x}}_k) = \begin{bmatrix} {}_{\mathrm{C}}\mathcal{R}_{\mathrm{W}} & \mathbf{0}_3 \end{bmatrix} \hat{\mathbf{x}}_k + \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{9}$$

where C_x, C_y and C_z are the coordinates of the camera in the world frame and ${}_{\mathrm{C}}\mathcal{R}_{\mathrm{W}}$ is the rotation matrix from the world frame (W) to camera frame (C).

The innovation covariance $\mathbf{S}_k$ is given by

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R} \tag{10}$$

where $\mathbf{H}_k$ is the Jacobian of $\hat{\mathbf{h}}$. The observation covariance matrix $\mathbf{R}$ is a 3×3 diagonal matrix whose terms represent the noise covariance of camera measurements.

Finally, the Kalman gain is stated by

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{11}$$

The updated state estimate and covariance are given by Eqs. 12 and 13.

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{12}$$

$$\mathbf{P}_{k|k} = (\mathbf{I}_6 - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{13}$$

4.5 Dynamic Object Classification

The output of the object tracking is the state of each tracked MapObject. The dynamic object classification module outputs a Boolean result, establishing whether the object is moving or not in that particular frame based on a threshold, called "DOC threshold".

Keypoints belonging to new objects are filtered *a priori*. If the object tracker establishes that the object is static, the keypoints are used for feature tracking.

The DOC threshold is defined as a different value for each object and empirically adjusted depending on how potentially dynamic the object is. For example, a chair is more likely to achieve higher velocities than a couch or a desk.

4.6 MapObject Persistence

This work proposes a recursive Bayes' filter to estimate the belief about the MapObjects' persistence in the map. When a MapObject is created, the belief is set to 0.5. M_O is a discrete random variable that represents the persistence of a given MapObject initialized at a certain 3D position. It can assume the values 0 or 1, i.e., either the object is not there or it is there. The belief about the persistence of a given MapObject is stated by

$$bel(M_O) = \eta\, p(Y|M_O)p(M_O) \tag{14}$$

where η is a normalization factor given by

$$\eta = \frac{1}{bel(M_O = 1) + bel(M_O = 0)} \tag{15}$$

$$bel(M_O = 1) = \eta\, p(Y|M_O = 1)p(M_O = 1) \tag{16}$$

$$bel(M_O = 0) = \eta\, p(Y|M_O = 0)p(M_O = 0) \tag{17}$$

For each iteration, the prior $p(M_O)$ is the last belief. The likelihood $p(Y|M_O)$ is sampled from a Gaussian distribution with a mean equal to the current estimated position of the centroid of the object, and a covariance inversely proportional to the number of registered MapPoints. If an object is not detected at the place it was previously seen, its belief is lowered. The belief update of each MapObject is performed during the long-term data association, which is explained in the next section.

4.7 Long-term Data Association

The long-term data association evaluates if the MapObjects created in a new keyframe correspond to existing objects in the map. This process is detailed in Algorithm 1. First, the centroid of a MapObject candidate is computed using its associated MapPoints. Then, the Euclidean norm between the candidate's pose and close MapObjects with the same class is computed. If they are close enough, they are merged and their MapPoints are combined. Also, the belief of the object in the Map is updated accordingly.

If the belief of a MapObject is below a threshold (BeliefThreshold), then all its associated MapPoints are marked as inactive and cannot be used for tracking, mapping or loop closure. As all MapObjects start with a 0.5 belief, initially all MapPoints that belong to MapObjects are inactive. With this approach, objects that are moved or vanish from the map cannot interfere in the mapping process. As an object continues to be seen, its belief grows and, eventually, it is added to the map and its MapPoints become active.

If a MapObject has sequentially been seen in the past frames, its belief is not updated. This is to prioritize long-term updates in the belief estimation. The system also searches for MapObjects in the vicinity of the current pose that are not there anymore.

Algorithm 1 Long-term Data Association

Require: Current frame, last frame, list of MapObject candidates, current Map, current keyframe, last keyframe, list of Map Objects in the current Map

for every new MapObject candidate **do**
  Compute centroid and pose of the candidate
  for every MapObject in the Map **do**
    if class of candidate == class of object in the map **then**
      Compute the euclidean distance between the centroids
      if euclidean distance < ltdaThreshold **then**
        Merge candidate and object in map
        if object in map was not saw in the past N Keyframes **then**
          Update Belief of the object in map
          Update the Keyframe number
          if Belief < BeliefThreshold **then**
            MapPoints of the object are inactive
          else
            MapPoints of the object are active
          end if
        else
          Update Keyframe where it was last saw
        end if
      else
        Candidate was not in the map
        Create new MapObject
      end if
    end if
  end for
end for

4.8 Semantic Map

The final output of Changing-SLAM is the complete camera trajectory and a metric-semantic map. The metric map is composed of active MapPoints. The semantic map is

composed of MapObjects with their respective centroids and 3D bounding boxes, as shown in Fig. 4. Only objects with a high belief are active in the map.

5 Experiments and Results

This section presents experiments made with a mobile robot, and a dataset created from the experiments. Also, the present work is compared with several methods from the literature using this data and other public datasets.

5.1 PUC-USP Dataset

Experiments were performed to test the robustness of the methodology in changing environments and recorded as a dataset. It is composed of 6 sequences recorded in an indoor environment. Each sequence contains color and depth images captured by an Intel RealSense camera, and a ground-truth trajectory obtained using a motion capture system. The sequences were designed to capture different scenarios that could lead to a tracking failure or a map corruption. All data is publicly available.

The robot used in the experiments is shown in Fig. 5. The TerraSentia robot, developed by EarthSense [43], is a mobile robot with a size of 0.32x0.54x0.4 meters and equipped with four active wheels for skid-steer locomotion. All data acquisition, processing, and locomotion control are performed using an Intel i7 NUC in combination with a RaspberryPi 3B board. The ground-truth was generated using a motion capture system, consisting of seven OptiTrack Flex13 cameras. The robot has an Intel RealSense D435i camera with reflective markers used by the OptiTrack system for computing the ground-truth trajectory.

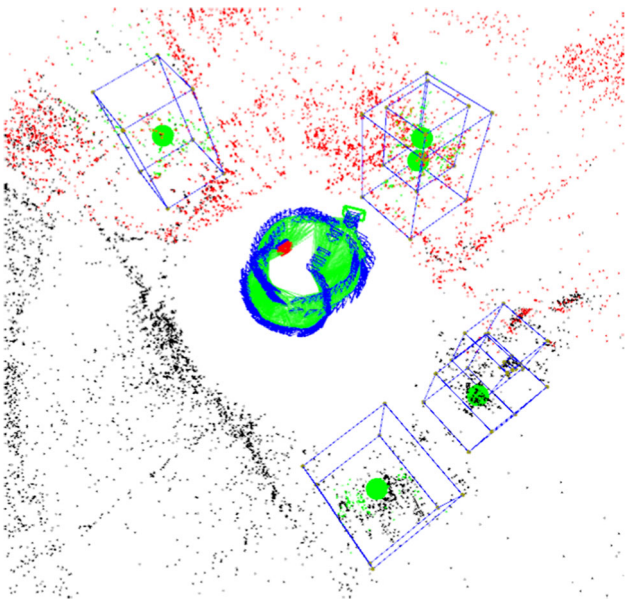


Fig. 4 Semantic Map generated by Changing-SLAM. It combines the output of the object detector with a long-term data association algorithm that updates the belief about the poses of objects over time and keeps only the ones with high belief

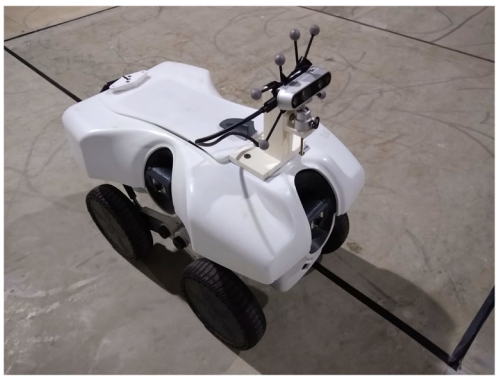


Fig. 5 TerraSentia, the mobile robot used for the experiments and for the recording of the dataset

All data, including the RGB and depth images, is provided in the same format used in the TUM Dataset [29].

The motion capture system was calibrated using the software provided by OptiTrack. The procedure consisted of waving a calibration stick while the calibration software registers its movement and receives information from each camera. A global precision of 4 mm was achieved after measuring the length of a metal rod with two reflective markers, following a process similar to that used in the TUM Dataset.

The proposed dataset uses the Absolute Trajectory Error (ATE) for ground-truth evaluation, which is the same evaluation metrics from the TUM Dataset [29]. It is used to evaluate the global consistency of the estimated trajectory, comparing the absolute distances between the translational components of the estimated and ground-truth trajectories. Equation 18 shows the computation of the ATE at a time step i.

$$ATE_i = E_i^{-1} T G_i \tag{18}$$

where E represents the estimated trajectory, G is the ground truth, and T describes the transformation that aligns the two trajectories. For a sequence of N poses, the RMSE of ATE is given by

$$RMSE(ATE_{1:N}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||trans(ATE_i)||^2} \tag{19}$$

where $trans(ATE_i)$ correspond to the translational components of ATE_i.

Several objects from the COCO Dataset [41] were used in the sequences, such as a teddy bear, umbrellas, chairs, and monitors. These objects were chosen because several

Table 1 Information about each sequence in the PUC-USP dataset

Sequence	Duration [s]	Length [m]	Avg. T. Vel. [m/s]	Avg. R. Vel. [deg/s]
Static	109.73	9.09	0.0828	27.08
OneChair	294.83	18.78	0.0637	14.07
Vanishing	277.53	23.17	0.0833	24.44
Changing	195.46	13.89	0.071	25.36
Shift	366.36	25.19	0.0687	16.26
Replacing	174.53	15.49	0.0887	46.55

semantic detectors use COCO for training, such as YOLOv3 [44], YOLOv4 [27], and Mask R-CNN [22].

Six sequences were recorded in this dataset: *Static*, *OneChair*, *Vanishing*, *Changing*, *Shift*, and *Replacing*. All sequences were recorded with the camera fixed on the robot, with the robot moving in the workspace with flat terrain. Despite being fixed and located on a wheeled mobile robot, the motion of the Intel RealSense camera is not entirely restricted to a plane due to irregularities on the floor as well as robot vibration.

Table 1 shows the statistics for the six recorded sequences, containing the duration, total length, average translational velocity and average rotational velocity. The actual velocity of the robot was higher, because the robot eventually needed to stand still for some moments so that the objects could be moved in the scene out of its field of view.

Table 2 shows an overview of the main characteristics of each dataset and how they are used to evaluate the performance of the algorithm in changing environments.

The *Static* sequence is a baseline for the evaluations. No objects were moved in this sequence. The robot just wanders within a static scene.

The *Vanishing* sequence is suitable to evaluate the ability of SLAM systems to eliminate vanished objects on the map. It starts with the robot facing a chair with a teddy bear. As the robot wanders, some objects are moved and others are removed until there are no more objects in the scene, as shown in Fig. 6. Even though the missing objects would not cause a track failure or wrong loop closure in a SLAM system not robust to changing environments, they would be present on the final map. This would interfere with a path-planning algorithm that uses this map, for example.

The *OneChair* sequence is suitable to evaluate the ability of SLAM systems to avoid wrong loop closures caused by moved objects. Figure 7 shows the ground-truth trajectory of the *OneChair* sequence. It starts with the robot facing a chair, as shown in Fig. 8a. As the robot moves, the chair is also moved outside of the robot's field of view. When the robot sees the chair again, as shown in Fig. 8c, the chair is in a different position, which can cause a wrong loop detection.

The *Changing* sequence initially contains two umbrellas and two chairs. Also, there is one teddy bear on one of the chairs. It starts with the robot facing the chair with the teddy bear. As the robot wanders within the scene, the teddy bear is moved to the other chair. This can potentially trigger a wrong loop closure. Selected scenes from this sequence are shown in Fig. 9.

The *Shift* sequence initially contains an umbrella on the floor, a teddy bear on a chair, and a suitcase with a monitor and a mug. The robot starts facing the chair with the teddy

Table 2 Description of the objects and changes in each sequence of the PUC-USP dataset

Sequence	Number of objects	Objects	Number of changes	Type of changes
Static	8	microwave, suitcase, monitor, mug, bottle, chair, umbrella, teddy bear	0	no changes
OneChair	2	chair, teddy bear	4	objects moved
Vanishing	8	suitcase, monitor, mug, chair, teddy bear, book	6	objects moved and removed
Changing	6	two chairs, teddy bear, umbrella, mug, monitor	4	objects removed, replaced and added
Shift	9	two chairs, books, teddy bear, monitor, suitcase, mug, umbrella, backpack	4	objects moved
Replacing	5	two chairs, teddy bear, two umbrellas	6	objects moved and replaced

(a) Scene 1 (b) Scene 2 (c) Scene 3

Fig. 6 Selected scenes from the *Vanishing* sequence, in which the objects on the scene are removed one by one outside the robot's field of view

bear. As the robot wanders within the scene, the umbrella and chair are moved to other positions. After several turns, the suitcase together with the monitor and mug are also moved. This sequence is suitable to evaluate the ability of SLAM systems to detect changes of multiple objects.

The *Replacing* sequence initially contains a chair with books, another chair with a teddy bear on it, and a suitcase with a monitor and a mug. The robot starts facing the chair with the teddy bear. After a while, the chair with the teddy bear is replaced by an umbrella. Finally, the other chair is removed.

5.2 Results

The performance of Changing-SLAM was evaluated in several different conditions. The robustness of the method against highly dynamic environments was evaluated using the TUM RGB-D dataset [29], in particular the fr3_w sequences, in which two people are walking around an office scenario and moving in front of the camera. For testing the robustness in changing environments, the PUC-USP dataset was used, which contains sequences with objects that are moved, removed, and/or replaced outside the camera's field of view.

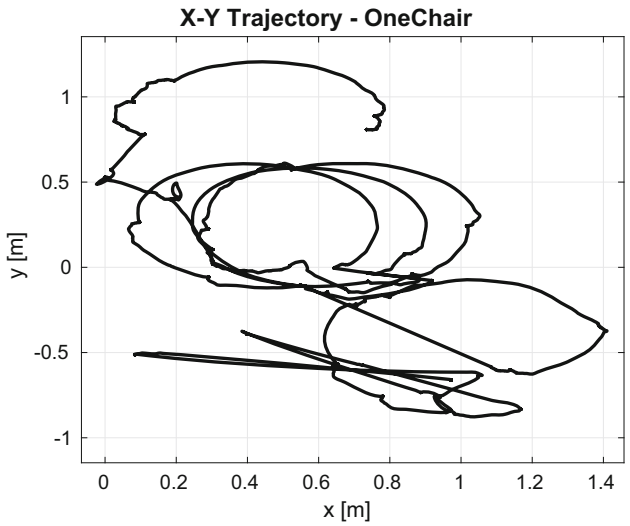


Fig. 7 Ground-truth trajectory of the *OneChair* sequence

5.2.1 TUM Dataset Results

Five dynamic sequences from the TUM Dataset were used to evaluate the robustness of the proposed method: the four walking sequences (fr3_w) and the fr3_sitting_xyz sequence, which was used as a baseline. The system was compared to 15 systems from the literature. The comparison includes feature-based methods designed for static environments, such as ORB-SLAM3 [15], direct methods designed for dynamic environments, such as StaticFusion [45] and ReFusion [30], feature-based methods designed for dynamic environments, for instance DynaSLAM [3], DS-SLAM [4], SaD-SLAM [5], and DOTMask [6], and DXSLAM [13], a method for changing environments.

The results from Liu et al. [46], DynaSLAM, DS-SLAM, SOF-SLAM [47], DetectSLAM [17], SaD-SLAM, DOTMask, Ji et al. [24] and Crowd-SLAM were obtained in their respective publications. The results from ORB-SLAM3 were obtained running the code using 1500 keypoints in each frame. Finally, the results from StaticFusion and ReFusion were obtained in [30]. For the results from the proposed work, the tests were performed 5 times and the median results were used for the evaluation, as proposed by Mur-Artal and Tardós [2].

Figures 10a and b show, respectively, the comparison between the ground-truth and estimated trajectories of ORB-SLAM3 and Changing-SLAM for the fr3_w_xyz sequence. The trajectory estimated by ORB-SLAM3 completely deviates from the ground-truth. Since ORB-SLAM3 does not have an algorithm to deal with dynamic environments, it fails to the presence of moving people in the scene. Figure 11 shows the difference between Changing-SLAM and ORB-SLAM3 during the feature detection process in this situation.

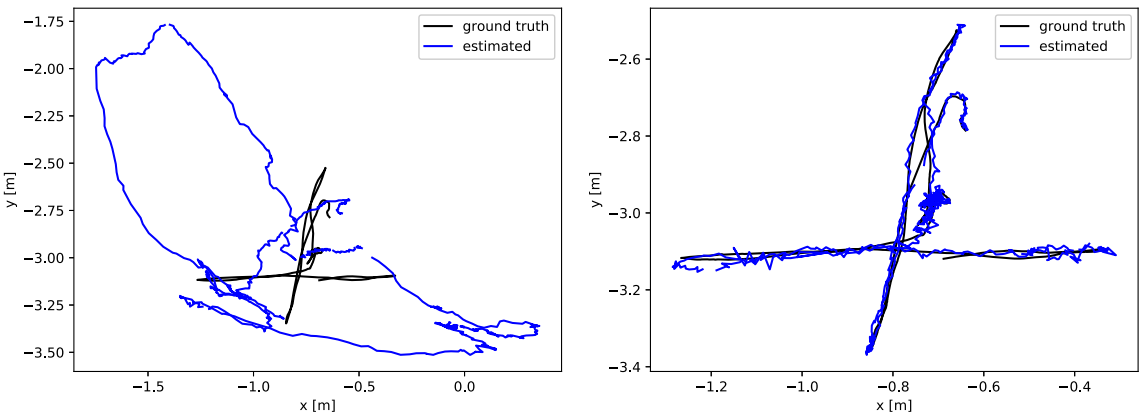
Table 3 shows the RMSE of the ATE comparison between the proposed method and the 15 methods. The bold values represent the best results in each sequence, and the underlined values represent the second-best ones. The results show that ORB-SLAM3 cannot cope with dynamic environments. Their results are satisfactory only in the sitting sequence,

(a) Scene 1 (b) Scene 2 (c) Scene 3

Fig. 8 Selected scenes from the *OneChair* sequence. The chair with the teddy bear has its position changed outside the field of view of the robot

(a) Scene 1 (b) Scene 2 (c) Scene 3

Fig. 9 Selected scenes from the *Changing* sequence. Similar objects have their positions switched to each other outside the field of view of the robot



(a) ORB-SLAM3 (b) Changing-SLAM

Fig. 10 Ground truth and estimated trajectory in the sequence fr3_w_xyz

Fig. 11 Comparison of feature detection between Changing-SLAM and ORB-SLAM3. This is a failure case for ORB-SLAM3 because the features associated to people cause a drift in the camera's pose. This does not happen for Changing-SLAM because the features from people are correctly filtered

(a) Changing-SLAM (b) ORB-SLAM3

Table 3 Comparison of the RMSE of ATE [m] of the proposed method against Crowd-SLAM, ORB-SLAM3, StaticFusion, ReFusion, DynaSLAM, DS-SLAM, SOF-SLAM, Detect-SLAM, Liu et al., Sun et al., Sun et al., SaD-SLAM, DOTMask, Ji et al., and DXSLAM using the TUM dataset

Sequence	*fr3_s_xyz*	*fr3_w_static*	*fr3_w_xyz*	*fr3_w_rpy*	*fr3_w_half*
Changing-SLAM	0.018	0.008	0.016	0.067	0.039
Crowd-SLAM [25]	0.018	0.007	0.020	0.044	0.026
ORB-SLAM3 [15]	**0.009**	0.038	0.819	0.957	0.315
StaticFusion [45]	0.039	0.015	0.093	—	0.681
ReFusion [30]	0.040	0.017	0.099	—	0.104
DynaSLAM [3]	0.015	**0.006**	**0.015**	0.035	**0.025**
DS-SLAM [4]	—	0.008	0.024	0.444	0.030
SOF-SLAM [47]	—	0.007	0.018	**0.027**	0.029
Detect-SLAM [17]	0.020	—	0.024	0.296	0.051
Liu et al. [46]	—	0.011	0.016	0.042	0.031
Sun et al. [48]	0.048	0.065	0.093	0.133	0.125
Sun et al. [49]	0.051	0.033	0.066	0.073	0.067
SaD-SLAM [5]	0.012	0.017	0.017	0.032	0.026
DOTMask [6]	0.018	0.008	0.021	0.053	0.040
Ji et al. [24]	0.012	0.011	0.020	0.037	0.029
DXSLAM [13]	—	0.017	0.309	—	—

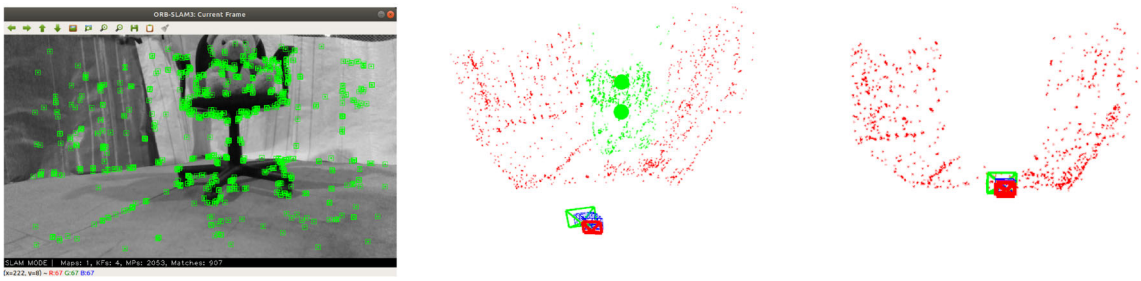
where the people in the scene are sitting, just moving their hands.

Overall, Changing-SLAM achieved the second-best result in the challenging fr3_w_xyz sequence, and achieved similar results in the other sequences in comparison to the other systems, showing that Changing-SLAM is robust to dynamic environments. DynaSLAM and SaD-SLAM had, overall, the best results. However, both methods are not real-time. Our method, on the other hand, achieved similar results working in real time. The results show that Changing-SLAM has the best trade-off between accuracy and speed for dynamic environments in the literature. The methods that achieved the best results have an inferior speed in comparison with Changing-SLAM [3, 6, 24]. For instance, SaD-SLAM runs offline, and Dyna-SLAM runs at more than 300ms per frame, without considering the time for instance segmentation, which is the bottleneck of the system. Other systems are either slower than Changing-SLAM or do not have information about the processing time in their papers. Changing-SLAM also outperformed all direct methods and two real-time feature-based ones: DOTMask and Detect-SLAM. The results also show that DXSLAM is not robust to dynamic environments.

5.2.2 PUC-USP Dataset Results

The PUC-USP Dataset, presented in Section 5.1, was used to evaluate the robustness of the proposed methodology in changing environments.

Figures 12a-c show an example of the proposed method in the *OneChair* sequence. There are two objects in the scene: a chair and a teddy bear. First, the features are detected in the RGB image, as shown in Fig. 12a. The keypoints are classified as belonging to the objects or to the background. Figure 12b shows the MapPoints in green classified as belonging to the objects, and the MapPoints in red as from the background. The larger green dots represent the



(a) Feature detection (b) Initial classification (c) Filtered map

Fig. 12 MapPoint filtering process performed by Changing-SLAM in the *OneChair* sequence. The two larger green dots represent two MapObjects in the map. The small green dots in the map represent the MapPoints associated to the MapObjects

centroids of the MapObjects. The green MapPoints are initially removed from the map, as shown in Fig. 12c, as they belong to MapObjects which beliefs are 0.5. However, both objects are detected, estimated and stored in memory. If the objects are observed again later, their belief would increase and they would be inserted again in the map together with their MapPoints.

Table 4 shows the ATE comparison between ORB-SLAM3, DXSLAM and Changing-SLAM. The bold values represent the best results in each sequence. As expected, all systems achieved good results in the static scenario. Despite the fact that Changing-SLAM improved every result of ORB-SLAM3, it is noticeable that the effect of changing environments is not always critical for the localization accuracy, as it happens in dynamic environments. The *Vanishing* sequence was not expected to cause a major error, because there are no new objects added in the scene, which could have caused a wrong loop closure. The *Changing* sequence caused an increase in the localization error of ORB-SLAM3, as shown in Fig. 13a and b. ORB-SLAM3 had lost tracks and drifts that were corrected by Changing-SLAM. The sequence *OneChair* triggered a wrong loop closure in ORB-SLAM3, which caused a considerable increase in the ATE (Fig. 14). DXSLAM achieved good results, although with lower accuracy than Changing-SLAM. However, DXSLAM is not robust to dynamic environments, according to the results presented in Table 3.

Figures 15a and b show the wrong loop closure made by ORB-SLAM3 in the *OneChair* sequence. The shapes of two chairs are noticeable in Fig. 15a, merged into one chair in Fig. 15b. The wrong loop occurred due to the change in the chair position and the inability of ORB-SLAM3 to detect this change. After this wrong loop closure, the system is no longer able to recover from the error, even with the robust multi-mapping and re-localization systems of ORB-SLAM3, as shown in Fig. 14a.

Table 4 Comparison of the RMSE of ATE [m] of Changing-SLAM against ORB-SLAM3 and DXSLAM using the PUC-USP dataset

Sequence	ORB-SLAM3 [15]	DXSLAM [13]	Changing-SLAM
Static	**0.033**	0.036	**0.033**
OneChair	0.407	0.097	**0.089**
Vanishing	0.052	0.062	**0.049**
Changing	0.071	0.044	**0.029**
Shift	**0.075**	0.077	**0.075**
Replacing	0.049	0.055	**0.047**

5.3 Influence of the number of Keypoints

It is important to verify the influence of the number of keypoints to the overall results, to evaluate if its change alters the accuracy of the system. Fig. 16 shows the ATE comparison between Changing-SLAM, ORB-SLAM2 and ORB-SLAM3 with different number of keypoints in the OneChair sequence of the PUC-USP dataset. The two systems were chosen for the comparison for being the base of many visual SLAM systems in dynamic environments, including ours. Five runs were made for each case and the median result was chosen for evaluation. The results show that our method maintains a low ATE for different numbers of keypoints. The other methods, on the other hand, have significantly increased errors. ORB-SLAM3 fails in fewer cases in comparison to ORB-SLAM2 due to its improved localization system. Their increased ATE are mainly caused by lost tracks, wrong re-localization, and wrong loop closures, which happen due to the different position of objects that were previously mapped.

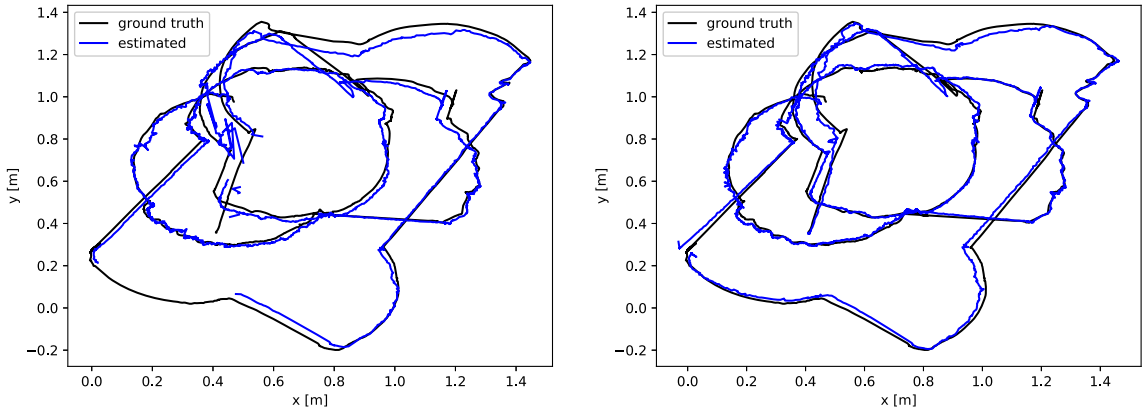
5.4 Run-time Analysis

All tests were performed on a notebook with an Intel Core i7-10750H and 16 GB of RAM running Ubuntu Linux 18.04 LTS. The system is implemented in C++, and the object detection is performed with OpenCV 4.5, using a Nvidia GeForce RTX 2060 GPU. Changing-SLAM achieved an average tracking speed of 23.8 FPS, considering all steps including object detection, which can be categorized as real time.

5.5 Limitations

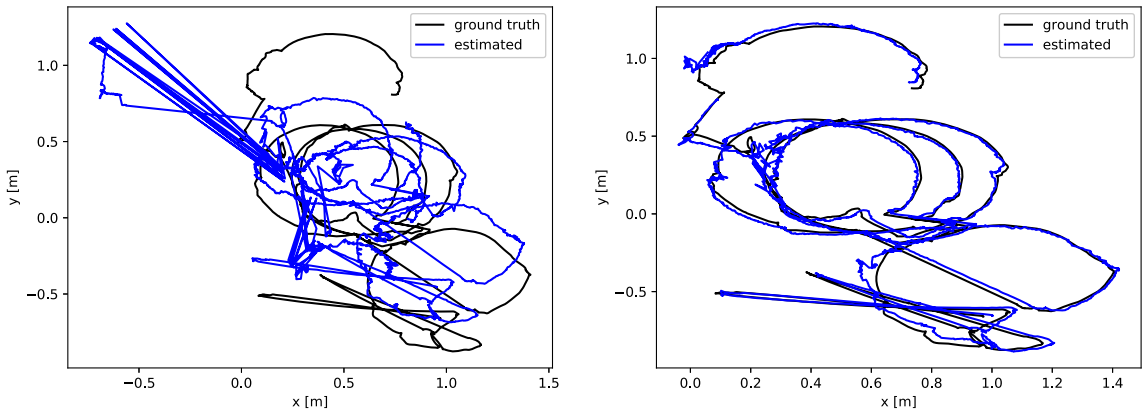
Changing-SLAM does not deal with deformable objects such as blankets, rigid objects that can change shape such as cabinets with closed and open doors, and objects not labeled in the object detector training process. The latter problem can be solved re-training the network to include more objects that are common to the environment.

The COCO Dataset [41] has several classes that are not suitable for indoor environments, which is the scope of this work, such as cars, and even other classes that are not common for regular outdoor situations, such as giraffes and other wild animals. Therefore, it would be beneficial to train the network with more common indoor objects. However, this approach still cannot deal with unexpected new objects, as performed in the approach proposed by Ji et al. [24].



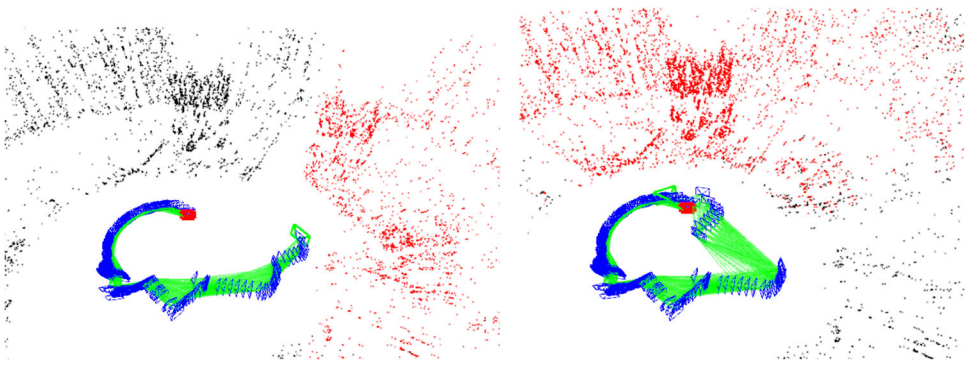
(a) ORB-SLAM3 (b) Changing-SLAM

Fig. 13 Comparison of ground truth and estimated trajectory in the sequence *Changing* between ORB-SLAM3 and Changing-SLAM



(a) ORB-SLAM3 (b) Changing-SLAM

Fig. 14 Comparison of ground truth and estimated trajectory in the sequence *OneChair* showing the wrong loop closure in the ORB-SLAM3 trajectory



(a) Before loop closure (b) After loop closure

Fig. 15 Wrong loop detection by ORB-SLAM3 in the OneChair sequence

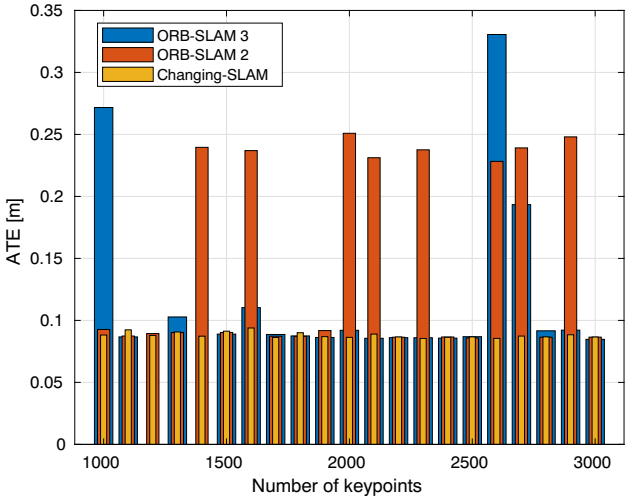


Fig. 16 ATE Comparison between ORB-SLAM2, ORB-SLAM3 and Changing-SLAM with different number of keypoints in the OneChair sequence

6 Conclusion

This work presented Changing-SLAM, our proposed method to perform visual SLAM in both dynamic and changing scenarios in real time. The proposed method was tested with a dataset especially designed for the evaluation of visual SLAM systems in changing scenarios, achieving a high accuracy in comparison with ORB-SLAM3 and DXSLAM. Changing-SLAM is very robust in such scenarios, preventing the detection of a wrong loop closure that would ruin the SLAM process.

Besides correcting localization, the semantic map generated by Changing-SLAM can be useful for a wide variety of applications. One example would be within the work from Chen and Liu [50], which generates navigable paths from the maps generated by ORB-SLAM2 and ORB-SLAM3. These maps would be corrupted if objects were moved in the scene.

Finally, the use of object detection and feature repopulation to differentiate object features from the background ones is a method to decrease the computational effort of the system. However, the semantic mapping, dynamic object filtering and belief update methods are not restricted to that. The proposed method can be performed using other types of semantic detection such as instance or panoptic segmentation, when they become computationally feasible. Therefore, for future work, panoptic segmentation could be used in the methodology to evaluate if there is an improvement in accuracy.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10846-023-02019-6.

Author Contributions Conceptualization and methodology, J.S, V.M and G.A; software, J.S, V.M and G.A; validation, J.S, V.M and G.A.; experiment J.S, V.M, G.A, G.C and M.B; writing – original draft preparation, J.S, V.M and G.A; writing – review and editing, J.S, V.M, G.A, M.B, M.G and M.M; supervision, M.B, M.G and M.M; All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Istituto Italiano di Tecnologia within the CRUI-CARE Agreement. This research was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and in part by FAPESP, process number 2021/05336-3.

Availability of data and materials The datasets used are publicly available in:
- PUC-USP Dataset
- TUM RGB-D Dataset: https://vision.in.tum.de/data/datasets/rgbd-dataset
- Bonn RGB-D Dynamic Dataset: http://www.ipb.uni-bonn.de/data/rgbd-dynamic-dataset/

Declarations

Competing Interests The authors declare that they have no competing interest.

References

1. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of the 2007 6th IEEE and ACM international symposium on mixed and augmented reality, pp. 1–10 (2007). IEEE Computer Society
2. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Trans Robot **31**(5), 1147–1163 (2015)
3. Bescos, B., Fácil, J.M., Civera, J., Neira, J.: DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. IEEE Robot Autom Lett **3**(4), 4076–4083 (2018)
4. Yu, C., Liu, Z., Liu, X.-J., Xie, F., Yang, Y., Wei, Q., Fei, Q.: DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 1168–1174 (2018). https://doi.org/10.1109/IROS.2018.8593691
5. Yuan, X., Chen, S.: SaD-SLAM: A Visual SLAM Based on Semantic and Depth Information. In: 2020 IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS), pp. 4930–4935 (2020). https://doi.org/10.1109/IROS45743.2020.9341180
6. Vincent, J., Labbé, M., Lauzon, J., Grondin, F., Comtois-Rivet, P., Michaud, F.: Dynamic Object Tracking and Masking for Visual SLAM. 2020 IEEE/RSJ Int Conf Intell Robot Syst (IROS), 4974–4979 (2020)
7. Olson, E., Agarwal, P.: Inference on networks of mixtures for robust robot mapping. Int J Robot Res **32**(7), 826–840 (2013). https://doi.org/10.1177/0278364913479413
8. Agarwal, P., Tipaldi, G.D., Spinello, L., Stachniss, C., Burgard, W.: Robust map optimization using dynamic covariance scaling. In: 2013 IEEE international conference on robotics and automation, pp. 62–69 (2013)
9. Sünderhauf, N., Protzel, P.: Switchable constraints for robust pose graph SLAM. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp. 1879–1884 (2012). IEEE
10. Lee, D., Myung, H.: Solution to the SLAM problem in low dynamic environments using a pose graph and an RGB-D sensor. Sensors **14**(7), 12467–12496 (2014)
11. Gomez, C., Hernandez, A., Derner, E.: Object-Based Pose Graph for Dynamic Indoor Environments. IEEE Robot Autom Lett **5**(4), 5401–5408 (2020)
12. Derner, E., Gomez, C., Hernandez, A.C., Barber, R., Babuska, R.: Change detection using weighted features for image-based localization. Robot Auton Syst **135**, 103676 (2021)
13. Li, D., Shi, X., Long, Q., Liu, S., Yang, W., Wang, F., Wei, Q., Qiao, F.: DXSLAM: A Robust and Efficient Visual SLAM System with Deep Features. 2020 IEEE/RSJ Int Conf Intell Robot Syst (IROS), 4958–4965 (2020)
14. Zhao, M., Guo, X., Song, L., Qin, B., Shi, X., Lee, G., Sun, G.: A General Framework for Lifelong Localization and Mapping in Changing Environment. In: 2021 IEEE/RSJ International conference on intelligent robots and systems (2021). IEEE
15. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M.M., Tardøs, J.D.: ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. IEEE Trans Robot **37**(6), 1874–1890 (2021). https://doi.org/10.1109/TRO.2021.3075644
16. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: 2011 International conference on computer vision, pp. 2564–2571 (2011)
17. Zhong, F., Wang, S., Zhang, Z., Chen, C., Wang, Y.: Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial. In: 2018 IEEE winter conference on applications of computer vision (WACV), pp. 1001–1010 (2018). https://doi.org/10.1109/WACV.2018.00115
18. Yang, S., Fan, G., Bai, L., Zhao, C., Li, D.: SGC-VSLAM: A Semantic and Geometric Constraints VSLAM for Dynamic Indoor Environments. Sensors **20**(8), (2020)
19. Xiao, L., Wang, J., Qiu, X., Rong, Z., Zou, X.: Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. Robot Auton Syst **117**, 1–16 (2019). https://doi.org/10.1016/j.robot.2019.03.012
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., Berg, A.: SSD: single shot multibox detector. In: Proceedings of the european conference on computer vision, pp. 21–37 (2016)
21. Rother, C., Kolmogorov, V., Blake, A.: GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. ACM Trans. Graph. **23**(3), 309–314 (2004). https://doi.org/10.1145/1015706.1015720
22. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp. 2961–2969 (2017)
23. Li, A., Wang, J., Xu, M., Chen, Z.: DP-SLAM: A visual SLAM with moving probability towards dynamic environments. Inf Sci **556**, 128–142 (2021). https://doi.org/10.1016/j.ins.2020.12.019
24. Ji, T., Wang, C., Xie, L.: Towards Real-time Semantic RGB-D SLAM in Dynamic Environments. In: 2021 IEEE international conference on robotics and automation (ICRA), pp. 11175–11181 (2021). https://doi.org/10.1109/ICRA48506.2021.9561743
25. Soares, J.C.V., Gattass, M., Meggiolaro, M.A.: Crowd-SLAM: Visual SLAM Towards Crowded Environments using Object Detection. J Intell Robot Syst **102**(2), 50 (2021). https://doi.org/10.1007/s10846-021-01414-1
26. Mur-Artal, R., Tardśs, J.D.: ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Trans Robot **33**(5), 1255–1262 (2017)
27. Bochkovskiy, A., Wang, C., Liao, H.M.: YOLOv4: Optimal Speed and Accuracy of Object Detection. CoRR (2020) arXiv:2004.10934
28. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Conference on computer vision and pattern recognition (CVPR) (2012)
29. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International conference on intelligent robots and systems, pp. 573–580 (2012)
30. Palazzolo, E., Behley, J., Lottes, P., Giguère, P., Stachniss, C.: ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 7855–7862 (2019). https://doi.org/10.1109/IROS40897.2019.8967590
31. Shi, X., Li, D., Zhao, P., Tian, Q., Tian, Y., Long, Q., Zhu, C., Song, J., Qiao, F., Song, L., Guo, Y., Wang, Z., Zhang, Y., Qin, B., Yang, W., Wang, F., Chan, R.H.M., She, Q.: Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM. In: 2020 International conference on robotics and automation (ICRA), pp. 3139–3145 (2020)
32. Rosen, D.M., Mason, J., Leonard, J.J.: Towards lifelong feature-based mapping in semi-static environments. 2016 IEEE Int Conf Robo Autom (ICRA), 1063–1070 (2016)
33. Hashemifar, Z., Dantu, K.: Practical Persistence Reasoning in Visual SLAM. In: 2020 IEEE international conference on robotics and automation (ICRA), pp. 7307–7313 (2020). https://doi.org/10.1109/ICRA40945.2020.9196913
34. Lázaro, M.T., Capobianco, R., Grisetti, G.: Efficient long-term mapping in dynamic environments. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 153–160 (2018). https://doi.org/10.1109/IROS.2018.8594310
35. Schmid, L.M., Delmerico, J.A., Schönberger, J.L., Nieto, J.I., Pollefeys, M., Siegwart, R., Cadena, C.: Panoptic Multi-TSDFs: a Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency. CoRR (2021) arXiv:2109.10165
36. Walcott-Bryant, A., Kaess, M., Johannsson, H., Leonard, J.J.: Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp. 1871–1878 (2012). IEEE
37. Bore, N., Ekekrantz, J., Jensfelt, P., Folkesson, J.: Detection and tracking of general movable objects in large three-dimensional maps. IEEE Trans Robot **35**(1), 231–247 (2019). https://doi.org/10.1109/TRO.2018.2876111
38. Xie, H., Deng, T., Wang, J., Chen, W.: Robust incremental long-term visual topological localization in changing environments. IEEE Trans Instrum Meas **72**, 1–14 (2023). https://doi.org/10.1109/TIM.2022.3220270
39. Adkins, A., Chen, T., Biswas, J.: Probabilistic object maps for longterm robot localization. In: 2022 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 931–938 (2022). https://doi.org/10.1109/IROS47612.2022.9981316
40. Elvira, R., Tardós, J.D., Montiel, J.M.M.: ORBSLAM-Atlas: a robust and accurate multi-map system. 2019 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 6253–6259 (2019)

41. Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, pp. 740–755 (2014). Springer
42. Saho, K.: Kalman filter for moving object tracking: Performance analysis and filter design. In: de Oliveira Serra, G.L. (ed.) Kalman Filters. IntechOpen, Rijeka (2017). Chap. 12. https://doi.org/10.5772/intechopen.71731
43. EarthSense. https://www.earthsense.co/home. Accessed: 2022-04-22
44. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement (2018). arXiv:1804.02767
45. Scona, R., Jaimez, M., Petillot, Y.R., Fallon, M., Cremers, D.: StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. In: 2018 IEEE international conference on robotics and automation (ICRA), pp. 3849–3856 (2018). https://doi.org/10.1109/ICRA.2018.8460681
46. Liu, H., Liu, G., Tian, G., Xin, S., Ji, Z.: Visual SLAM Based on Dynamic Object Removal. In: 2019 IEEE international conference on robotics and biomimetics (ROBIO), pp. 596–601 (2019). https://doi.org/10.1109/ROBIO49542.2019.8961397
47. Cui, L., Ma, C.: SOF-SLAM: A Semantic Visual SLAM for Dynamic Environments. IEEE Access **7**, 166528–166539 (2019)
48. Sun, Y., Liu, M., Meng, M.: Improving RGB-D SLAM in dynamic environments: A motion removal approach. Robot Auton Syst **89** (2016). https://doi.org/10.1016/j.robot.2016.11.012
49. Sun, Y., Liu, M., Meng, M.Q.-H.: Motion removal for reliable RGB-D SLAM in dynamic environments. Rob Auton Syst **108**, 115–128 (2018). https://doi.org/10.1016/j.robot.2018.07.002
50. Chen, Z., Liu, L.: Navigable space construction from sparse noisy pointclouds. IEEE Robot. Autom. Lett. **6**(3), 4720–4727 (2021). https://doi.org/10.1109/LRA.2021.3068939

João Carlos Virgolino Soares is a Postdoctoral Researcher at the Dynamic Legged Systems Lab of the Istituto Italiano di Tecnologia, Italy. He received his D.Sc. degree from the Mechanical Engineering Department of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 2022. He graduated in Mechanical Engineering at PUC-Rio in 2015, having obtained his M.Sc. degree in the same institution in 2018. He was a Postdoctoral Researcher at the RoboDesign Lab of the University of Illinois at Urbana-Champaign, USA, from 2022 to 2023. His research interests include Simultaneous Localization and Mapping, State Estimation of Wheeled and Legged Robots, Deep Learning for Computer Vision, and Autonomous Navigation.

Vivian Suzano Medeiros is a Postdoctoral Researcher at the Mobile Robotics Lab of the University of São Paulo, Brazil. She graduated in Control and Automation Engineering at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 2012. She received her D.Sc. and M.Sc. degrees in Mechanical Engineering from the same institution in 2020 and 2015, respectively. During her doctorate, she spent one year as a Visiting Ph.D. Student at the Autonomous Systems Lab, at ETH Zurich, Zurich, Switzerland. At that time, she was involved in research on trajectory optimization and control for hybrid wheeled-legged robots. Her research interests include Autonomous Mobile Robots, Motion Planning for Legged and Wheeled-legged Robots, and Control Systems.

Gabriel Fischer Abati is a Research Fellow at the Dynamic Legged Systems Lab of the Istituto Italiano di Tecnologia in Genova, Italy. He received his M.Sc. degree at the Robotics Laboratory (LabRob) of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2023. He graduated in Mechanical Engineering at PUC-Rio in 2020. His research interests include Mobile Robotics, Visual SLAM, Artificial Intelligence, Dynamics and Control.

Marcelo Becker received the M.Sc. and D.Sc. degrees in mechanical engineering from the State University of Campinas (UNICAMP), Brazil, in 1997 and 2000, respectively. During his D.Sc. studies, he spent eight months as a Guest Student with the Institute of Robotics, Swiss Federal Institute of Technology, Zurich, Switzerland. At that time, he was involved in research on obstacle avoidance and map-building procedures for indoor mobile robots. From August 2005 to July 2006, he was on sabbatical leave with the Autonomous System Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland, where he continued working on obstacle avoidance for indoor and outdoor mobile robots. Since 2008, he has been a Professor at the University of Sao Paulo, Sao Carlos, Brazil. His research interests focus on mobile robots, inspection robots, vehicular dynamics, design methodologies and tools, and mechanical design applied to robots and mechatronics.

Glauco Caurin holds a degree in Mechanical Engineering from EESC – USP Brazil (1988). He completed a specialization in Mechatronics from the Swiss Federal Institute of Technology - ETH - Zurich, Switzerland, in 1990. In 1994, he earned a Doctorate in Engineering from ETH Zurich. Between 2010 and 2011, Glauco Caurin took a sabbatical year at the Newman Laboratory for Biomechanics and Human Rehabilitation within the Department of Mechanical Engineering at the Massachusetts Institute of Technology - MIT - USA. Currently, Glauco Caurin serves as the Head of the Department of Aeronautical Engineering at EESC - USP. His research interests encompass a broad spectrum, including Interaction Control (Robot-aided Aircraft Assembly and Robot-Aided Surgery), Network Control Systems (NCS), Autonomous Systems (UAVs and AGVs), and expertise in Embedded and Real-Time Operating Systems (RTOS).

Marcelo Gattass is the Vice President of Development and Innovation and a Full Professor at the Department of Informatics of PUC-Rio. He graduated in Civil Engineering at PUC-Rio in 1975, obtained his M.Sc. in the same institution in 1977, and his Ph.D. in Civil Engineering from the Computer Graphics Program of Cornell University in 1982. Gattass is an expert in Computational Science, emphasizing Graphics Processing and focusing on the following themes: Visualization, Numerical Simulation, Augmented Reality, Geometric Modeling, and Computer Vision. His research currently focuses on Modeling Natural Objects based on Computer Vision. He has published 92 articles in international journals and over 120 papers in prominent conferences.

Marco Antonio Meggiolaro Ph.D., is an Associate Professor at the Mechanical Engineering Department of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) since 2000. He graduated in Mechanical Engineering at PUC-Rio in 1994, having obtained his M.Sc. in this same institution in 1996. He got his Ph.D. in Mechanical Engineering at the Massachusetts Institute of Technology (MIT) in 2000. Prof. Meggiolaro's main research areas are Robotics and Structural Integrity, being the author or co-author of more than 400 published scientific works.

Authors and Affiliations

João Carlos Virgolino Soares[1,3] · Vivian Suzano Medeiros[2] · Gabriel Fischer Abati[3] · Marcelo Becker[2] · Glauco Caurin[4] · Marcelo Gattass[5] · Marco Antonio Meggiolaro[3]

Vivian Suzano Medeiros
viviansuzano@usp.br

Gabriel Fischer Abati
fischerabati@gmail.com

Marcelo Becker
becker@sc.usp.br

Glauco Caurin
gcaurin@sc.usp.br

Marcelo Gattass
mgattass@tecgraf.puc-rio.br

Marco Antonio Meggiolaro
meggi@puc-rio.br

[1] Dynamic Legged Systems lab, Istituto Italiano di Tecnologia (IIT), Via S. Quirico, 19d, Genoa 16163, GE, Italy

[2] Department of Mechanical Engineering, University of São Paulo, Avenida Trabalhador São-Carlense, São Carlos 13566-590, SP, Brazil

[3] Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro, Marquês de São Vicente, Rio de Janeiro, Rio de Janeiro 22451-040, RJ, Brazil

[4] Department of Aeronautics, University of São Paulo, Avenida João Dagnone, São Carlos 13563-120, SP, Brazil

[5] Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Marquês de São Vicente, Rio deJaneiro, Rio de Janeiro 22451-040, RJ, Brazil