Journal of Intelligent & Robotic Systems (2021) 102:50
https://doi.org/10.1007/s10846-021-01414-1

REGULAR PAPER

Crowd-SLAM: Visual SLAM Towards Crowded Environments using Object Detection

João Carlos Virgolino Soares[1] · Marcelo Gattass[2] · Marco Antonio Meggiolaro[1]

Received: 1 July 2020 / Accepted: 4 May 2021

Abstract
Simultaneous Localization and Mapping is a fundamental problem in mobile robotics. However, the majority of Visual SLAM algorithms assume a static scenario, limiting their applicability in real-world environments. Dealing with dynamic content in Visual SLAM is still an open problem, with solutions usually relying on purely geometric approaches. Deep learning techniques can improve the SLAM solution in environments with *a priori* dynamic objects, providing high-level information of the scene. However, most solutions are not prepared to deal with crowded scenarios. This paper presents Crowd-SLAM, a new approach to SLAM for crowded environments using object detection. The main objective is to achieve high accuracy while increasing the performance, in comparison with other methods. The system is built on ORB-SLAM2, a state-of-the-art SLAM system. The proposed methodology is evaluated using benchmark datasets, outperforming other Visual SLAM methods.

Keywords Visual SLAM · Crowded environments · Object detection · Deep learning

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in mobile robotics, especially in unknown and unstructured environments, being a prerequisite for several tasks, such as navigation. It consists of creating a map of the environment and, simultaneously, estimating the pose of the robot in the created map.

A camera is a common choice as the main sensor in a SLAM system due to its low cost and richness of information, allowing accurate visual odometry systems, robust loop closure algorithms, and the use of semantic information, among other benefits. RGB-D cameras have an extra advantage of providing dense depth information. Visual SLAM is the problem of solving localization and mapping using only a camera as sensor.

There are several Visual SLAM systems in the literature, with high precision and efficiency, for instance, ORB-SLAM2 [1], LSD-SLAM [2], RGBDSLAM [3], and RTAB-Map [4]. However, their majority assume a static environment, which imposes a limitation of their applicability in real-world scenarios.

The main challenges of performing SLAM in dynamic environments are: to detect dynamic objects in the scene, to prevent those objects from being tracked, and to exclude them from the map. Some SLAM systems that work in dynamic environments rely on purely geometric approaches to detect moving objects. However, they usually fail to detect the presence of *a priori* dynamic objects, e.g., people, when they are initially static, which can lead to odometry drifts or long-term wrong loop closures. Computer vision tasks, such as object detection and instance segmentation, provide semantic information of the scene that allows the recognition of such objects.

Our previous work [5] presented the weakness and strengths of both tasks for Visual SLAM in human-populated environments. The YOLO object detection proved to be

✉ João Carlos Virgolino Soares
virgolinosoares@gmail.com

Marcelo Gattass
mgattass@tecgraf.puc-rio.br

Marco Antonio Meggiolaro
meggi@puc-rio.br

[1] Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil

[2] Department of Informatics / Tecgraf Institute, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil

Published online: 28 May 2021

more advantageous than the Mask R-CNN instance segmentation for SLAM, in terms of speed and accuracy. However, despite the higher speed of YOLO, it was not fast enough to be considered real-time.

There is an increasing number of Visual SLAM systems relying on deep learning object detectors to filter the dynamic content of images. However, they are neither efficient nor suitable for working with crowds. The main goal of this work is to create a SLAM system capable of working in crowded environments in real-time, by using a custom YOLO Tiny network specialized in people detection in crowds, and an algorithm for keypoint filtering.

Thus, this paper proposes Crowd-SLAM, a new open-source[1] Visual SLAM system for crowded environments based on ORB-SLAM2. The YOLO object detection is used to filter people in the scene. The YOLO network is trained using a dataset for crowded environments, achieving a real-time performance with high precision. The proposed methodology is evaluated using multiple datasets and compared with state-of-the-art systems. As ORB-SLAM2, Crowd-SLAM works with monocular, stereo, and RGB-D cameras. The main contributions of this paper can be summarized as follows:

- A complete open-source real-time Visual SLAM system for crowded environments with higher accuracy than other state-of-the-art approaches.
- A new network especially trained to operate in crowded environments, called CYTi, running in an independent thread.
- An algorithm to efficiently filter dynamic feature points, with a system to prevent lost tracks.

The paper is organized as follows. Section 2 presents the related work, Section 3 details the proposed methodology, Section 4 shows the results, and finally the conclusions and suggestions for future work are presented in Section 5.

2 Related Work

2.1 Visual SLAM

MonoSLAM [6], developed by Davison et al. was the first Visual SLAM system for monocular cameras. PTAM [7], developed in 2007 by Klein et al. was the first Visual SLAM system based on keyframe bundle adjustment, using two independent threads of mapping and tracking. Mur-Artal et al. [8] proposed a system with three threads called ORB-SLAM, with a second version called ORB-SLAM2 [1] that works with monocular, stereo, and RGB-D cameras.

[1] https://github.com/virgolinosoares/Crowd-SLAM

Visual SLAM systems are classified in two main categories: feature-based and direct methods. Feature-based methods use image feature extraction and matching to perform pose estimation, loop closing, and bundle adjustment. Examples of feature-based Visual SLAM methods include RGBDSLAM [3], the method from Henry et al. [9], and ORB-SLAM [1, 8]. Direct methods, on the other hand, perform optimization over image pixel intensities. KinectFusion [10], for instance, uses only depth data to create a dense volumetric model, and ICP to track the camera pose. DSO [11] combines photometric error minimization with a joint geometric and camera motion optimization to obtain visual odometry. There are also semi-direct systems such as LSD-SLAM [2], that uses a direct method to create semi-dense maps, but also uses features to perform loop closing.

2.2 Visual SLAM in Dynamic Environments

Most state-of-the-art Visual SLAM systems were designed with a static environment assumption. Therefore, they are not able to handle dynamic scenarios. The ones that deal with dynamic content in the scene usually treat it as noise, and filter it using direct or feature-based methods.

StaticFusion [12] and ReFusion [13], for instance, are two direct methods for RGB-D cameras. ReFusion combines the TSFD model representation of KinectFusion with a purely geometric approach to filter the dynamic content. StaticFusion also uses a geometric approach, but with a surfel representation.

Dib and Charpillet [14] proposed a dense visual odometry system for RGB-D cameras in dynamic environments using RANSAC. Alcantarila et al. [15] proposed a dense scene flow representation to detect moving objects using stereo cameras. Sun et al. [16] combined image differencing and a Maximum-a-posterior estimator to perform motion removal. In another work [17], Sun et al. proposed another method for motion removal using dense optical flow. Other direct approaches include the works of Wang and Huang [18], and Kim et al. [19]. Feature-based methods include the work of Cheng et al. [20], who proposed a system based on ORB-SLAM that uses optical flow to distinguish dynamic feature points.

The previously cited approaches, however, are unable to detect *a priori* dynamic objects in the scene, such as people or cars, when they remain static. The DS-SLAM [21] system deals with dynamic objects combining the optical flow method with a semantic segmentation network, which allows the detection of people. SOF-SLAM [22] is a feature-based method, built on ORB-SLAM2, that combines semantic segmentation and epipolar geometry to filter dynamic features. Sun et al. [23] proposed

a weakly-supervised semantic segmentation network to provide a binary mask to ORB-SLAM2 indicating movable objects, without the need for expensive annotations in the training process. However, their semantic segmentation step required 1.27 seconds to process a single image.

DynaSLAM [24] uses the Mask R-CNN [25] instance segmentation 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, DynaSLAM cannot perform real-time due to the high computational requirement of the Mask R-CNN technique.

On the other hand, Object Detection is a task that can be performed in real time, depending on the technique used. In our previous work [5], we proposed a comparison between Instance Segmentation and Object Detection for filtering *a priori* dynamic features from the scene, evaluating which one is most advantageous in terms of speed and accuracy. The Mask R-CNN and YOLO frameworks were used for instance segmentation and object detection, respectively. The object detection proved to be the most advantageous one.

2.3 Object Detection

Object detection is the task of determining the location, bounding box, and class of objects in an image. There are different types of deep learning-based object detection algorithms. The R-CNN detectors, for example, such as Fast R-CNN [26] and Faster R-CNN [27], use an algorithm to find potential regions to contain objects and then use a convolutional neural network (CNN) in those regions. Therefore, they are known as two-stage detectors.

Despite being accurate, these algorithms do not work in real time due to their complex pipelines. On the other hand, single-stage detectors, such as YOLO (You Only Look Once) [28] and SSD (Single Shot Multibox Detector) [29], are much more efficient as they use only one convolutional network to obtain both bounding box and object class without the need of generating candidate regions, as opposed to the R-CNN detectors. The YOLOv3 [30], proposed by Redmon et al. can run at 20 FPS in a computer with GPU, with 57.9 mAP, trained with the MS COCO dataset [31].

2.4 Visual SLAM in Dynamic Environments with Object Detection

Deep learning-based object detection has been widely applied in SLAM systems to filter dynamic features. In Detect-SLAM [32], Zhong et al. used SSD object detection only on keyframes to overcome the slow inference time of 0.31 s. Despite only happening when a new keyframe is inserted, this still can disturb the process.

Liu et al. [33] used YOLOv3 combined with optical flow for dynamic feature point removal. However, their method does not remove *a priori* dynamic objects, potentially causing wrong loop closures and odometry drifts in a scene with initially static people. Xiao et al. [34] proposed the Dynamic SLAM, which also uses SSD 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 the dynamic objects. However, the mask creation can be demanding in images with high resolution. Furthermore, they need the additional step for dynamic object filtering that highly depends on the number of objects in the scene.

None of the mentioned works consider the crowded environment scenario, which can lead to a drop in performance or accuracy.

3 Methodology

Figure 1 shows an overview of the proposed methodology, composed of four threads: Object Detection, Tracking, Local Mapping, and Loop Closing. The four threads run in parallel. The RGB images are processed in the object detection and tracking threads simultaneously. The tracking thread extracts ORB features [35] and waits for the bounding boxes provided by the object detection thread. The feature points and bounding boxes are sent to the dynamic keypoint filter and feature number update system inside the tracking thread. The dynamic keypoint filter removes the keypoints inside the bounding boxes, and the feature update system changes the feature number proportionally to the total filtered area, in order to prevent lost tracks.

3.1 SLAM

This work uses ORB-SLAM2 as the global SLAM solution. ORB-SLAM2 has three main threads: tracking, loop closing, and local mapping. Crowd-SLAM uses the same loop closing and local mapping threads of ORB-SLAM2. However, the tracking thread was modified to include the outlier removal algorithm and the feature number update system. Also, a new thread was added for Object Detection.

ORB-SLAM2 works using a keyframe-based methodology. The tracking thread decides whether every new frame is a new keyframe. The loop closing system compares the information of every new keyframe with past keyframes, searching for new closed loops using a bag-of-words place recognition module based on DBoW2 [36]. Once a loop is detected, the graph is optimized with the g2o framework [37] to assure a consistent trajectory. ORB-SLAM2 outputs

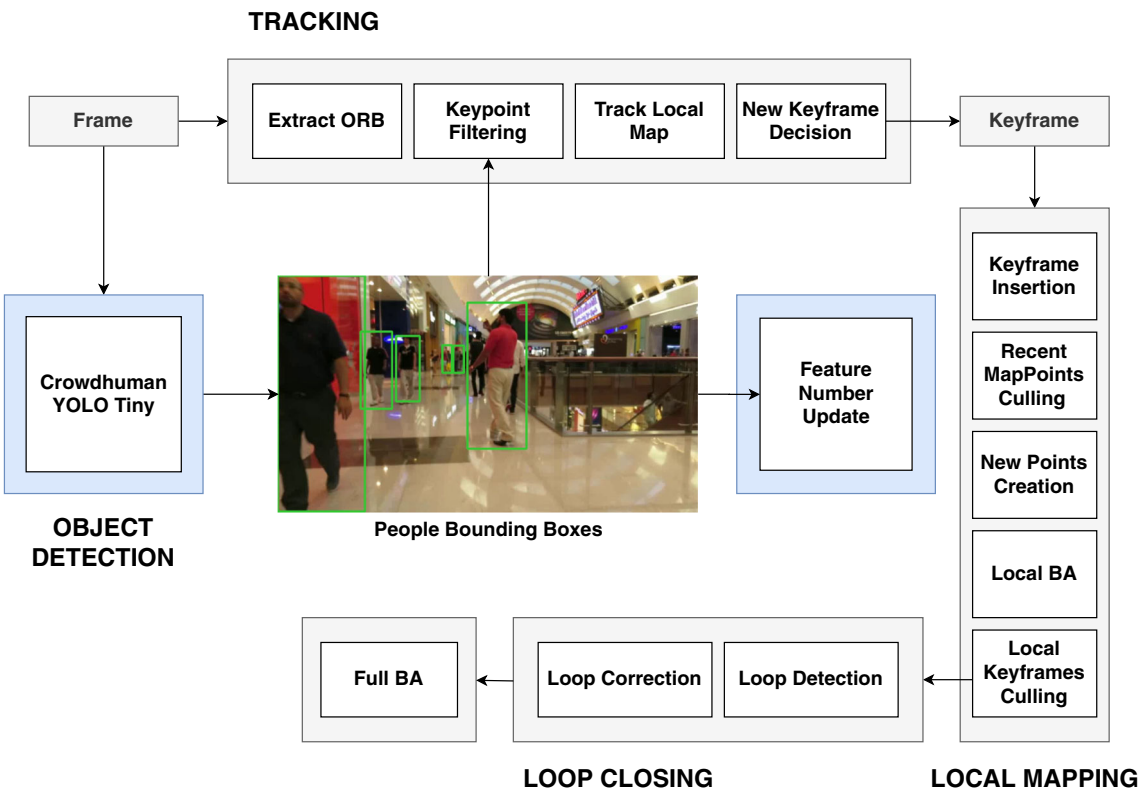


Fig. 1 Framework of Crowd-SLAM

a sparse point cloud map, and the optimized trajectory of the camera.

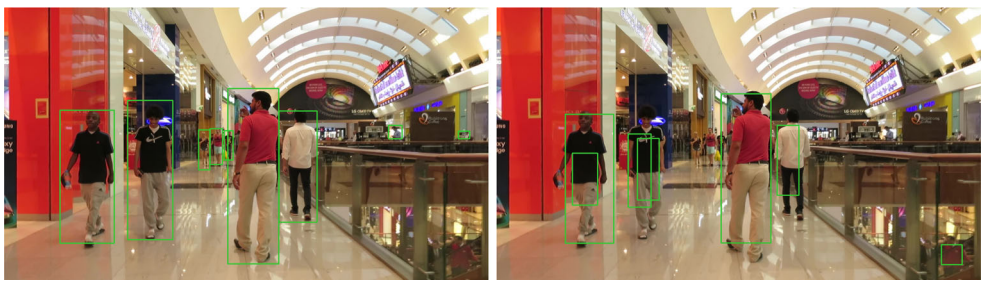
3.2 People Detection

YOLOv3 provides the classes of the detected objects, 2D bounding boxes with their corresponding positions, and a confidence number for each box. Figure 2a shows a detection in an image from the MOT Challenge 2020 [38], using a YOLOv3 framework trained with the COCO dataset [31]. YOLOv3 Tiny is a version of YOLO with fewer layers and filters, that has 10 times higher inference speed. However, it has a lower accuracy. Figure 2b shows an example of a YOLOv3 Tiny detection, also trained with the COCO dataset.

The MS COCO [31], used in YOLOv3, has 80 different object classes. However, only the people class is needed in this work. We propose the Crowdhuman YOLO Tiny (CYTi), a specialization of YOLO-Tiny that is more accurate in crowded environments with people.

CYTi is trained with the Crowdhuman dataset [39], composed of more than 20000 pictures of people in crowded environments, with 470000 humans and an average density of 23 people per image, much more than other people datasets. Figure 3 shows one image of the used training data. 15000 images were used for training and 4370 for validation. The training and validation were performed on an NVIDIA Quadro P2000 GPU using the Darknet framework [40]. The batch size was set to 24 with a learning rate of 0.001.

Fig. 2 Object detection output



(a) YOLOv3 (b) YOLOv3 Tiny

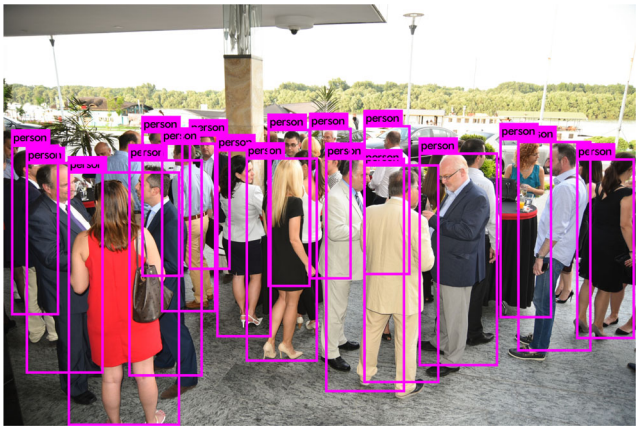


Fig. 3 YOLO detection in an image from the Crowdhuman Dataset

The Multiple Object Tracking (MOT) Challenge dataset [38, 41] is used to evaluate the improvements of the newly trained network in crowded environments. Several metrics are used to measure the detection capabilities. The *precision*, stated in Eq. 1, is the rate between true positives (TP) and the total number of detections, which is the sum of true positives and false positives (FP).

$$Precision = \frac{TP}{(TP + FP)} \tag{1}$$

The *recall*, stated in Eq. 2, is the rate between true positives and the sum of true positives and false negatives (FN).

$$Recall = \frac{TP}{(TP + FN)} \tag{2}$$

The Multiple Object Detection Precision (MODP) is stated by Eq. 3.

$$MODP = \frac{\sum_{t=1}^{N_{frames}} \frac{OverlapRatio}{N_{mapped}(t)}}{N_{frames}} \tag{3}$$

where N_{frames} is the total number of frames, $N_{mapped}(t)$ is the number of mapped objects in the frame t, and the overlap ratio is the sum of the intersection over union of every object for every frame.

According to Stiefelhagen et al. [42], the Multiple Object Detection Accuracy (MODA) is defined by Eq. 4.

$$MODA = 1 - \frac{\sum_{i=1}^{N_{frames}} (m_i + fp_i)}{\sum_{i=1}^{N_{frames}} N_G^i} \tag{4}$$

where m_i and fp_i are, respectively, the missed detections and false positives in the frame i, and N_G^i is the number of objects in the frame i.

Two sequences of the MOT17 and MOT20 challenges were selected: MOT20-01, MOT20-02, MOT17-09, and MOT17-11. The first two are crowded scenes in an indoor train station with a static camera. The MOT17-11 is a sequence in a crowded shopping mall with a forward-moving camera. The MOT17-09 is an outdoor crowded scene, with a static camera close to the people.

Table 1 shows the detection results using the YOLO Tiny, YOLOv3, and CYTi for the MOT challenge sequences. Besides the previous metrics, they also show the total number of true positives, false negatives, and false positives. The down and up arrow symbols next to the metric names means that the lower or higher the number, the better is for the overall detection performance, respectively.

CYTi outperforms the YOLO Tiny network in every metric of every sequence. YOLO Tiny has a poor performance in these sequences, especially in MOT20-01, not finding a single true positive. The results of YOLOv3 are also compared as reference. CYTi maintained the inference speed of YOLO Tiny, being more than ten times faster than YOLOv3, while increasing the detection accuracy.

Figures 4a and 4b show, respectively, the object detection output in a crowded scene from the MOT Challenge 2020 [38] using the YOLO Tiny network and CYTi. The improvement in both precision and accuracy is noticeable.

3.3 Outlier Removal

Once the images pass through the people detector, the keypoints that belong to people are removed from the image. The Dynamic keypoint filtering algorithm is as follows. The point (x, y) of D^{F_k} corresponds to the coordinates of the top left corner of the bounding box. w and h are the width and height of the box, respectively.

Unlike other systems, this algorithm does not need a mask of the frame with information about static and dynamic regions. It uses directly the bounding boxes to perform the filtering, therefore it does not depend on the image size.

Figures 5a and 5b show the keypoint detection of ORB-SLAM2 and with the proposed object detection filter, respectively. The filters successfully erased all keypoints in the regions with people. The keypoints appearing in the left chair are also erased, due to becoming merged with the person bounding box, resulting in an indirect filtering of potential dynamic objects.

3.4 Feature Number Update

If a large number of keypoints are filtered from a single frame, the information available for the SLAM system may not be enough to perform tracking. Two main problems can occur simultaneously or independently. First, there can be too many people in the scene. Secondly, one person can be too close to the camera, occupying most of the image. In both situations, the problem is not the number of people, but the total filtered area of the image.

Table 1 Detection results for YOLO Tiny, YOLOv3 and CYTi

Sequences	Method	MODA ↑	MODP ↑	TP ↑	FN ↓	FP ↓	Prec. ↑	Recall ↑
MOT20-01	Tiny	–69.4	0.0	0	8924	6192	0.0	0.0
	YOLOv3	4.6	70.0	**6851**	**2144**	6437	51.6	**76.2**
	CYTi	**11.4**	**73.2**	6609	2441	**5575**	**54.2**	73.0
MOT20-02	Tiny	–67.4	64.2	78	20001	13612	0.4	0.6
	YOLOv3	8.1	69.7	**15971**	**4279**	14324	**78.9**	52.7
	CYTi	**14.0**	**73.4**	14872	5356	**12037**	73.5	**55.3**
MOT17-09	Tiny	–71.8	75.4	856	2184	3040	22.0	28.2
	YOLOv3	1.8	**77.4**	**2722**	**398**	2665	50.5	**87.2**
	CYTi	**60.6**	77.2	2528	559	**657**	**79.4**	81.9
MOT17-11	Tiny	–35.6	74.9	2064	3666	4104	33.5	36.0
	YOLOv3	29.5	**80.3**	**4917**	**1062**	3156	60.9	**82.2**
	CYTi	**52.4**	78.4	4786	1169	**1665**	**74.2**	80.4

The bold entries are the best results for the respective sequences

Algorithm 1: Dynamic keypoint filtering algorithm.

```
Data: Frame F_k, bounding box list D^{F_k}(x,y,w,h),
      keypoints p^{F_k}
new_keypoints
for p_i in Frame F_k do
    bool_key = false;
    for people in D^{F_k} size do
        box = D[people];
        if p_i inside box then
            bool_key is true;
            break;
        end
    end
    if bool_key is false then
        new_keypoints append p_i;
    end
end
p^{F_k} = new_keypoints;
```

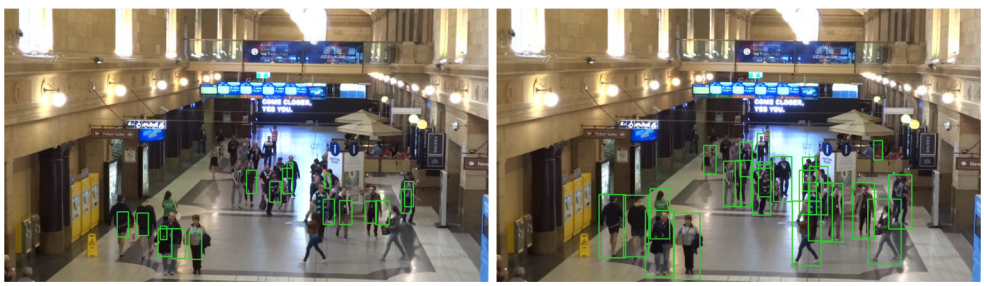
Even with the robust relocalization system of ORB-SLAM2, specifically designed to recover from a lost track, a crowded scene can prohibit the SLAM process. To overcome this issue, we propose a module to check the filtered area and update the number of detected ORB features, instead of setting a static high number. The number of feature points starts with a given initial value, and increases 300 for 30% of filtered area, 500 for 60%, 700 for 90%, and 1200 for more than 95%. This method benefits the performance, because more features extracted implies more computational effort, and simply defining a high static value would slow down the tracking without need, in the case of no people in the scene.

Figure 6a-c shows three scenes with (a) no people, (b) one person, and (c) two persons. The number of detected keypoints is increased in the third image due to the increased filtered area.

4 Results

4.1 TUM Dataset

Crowd-SLAM was numerically evaluated using the TUM RGB-D dataset [43]. It contains sequences of RGB and depth images obtained from a Microsoft Kinect camera, with their corresponding ground truth trajectories. The data was recorded at 30Hz with a 640 x 480 resolution.



(a) YOLO Tiny (b) CYTi

Fig. 4 People detection comparison between YOLO tiny and CYTi in a crowded scene

(a) ORB-SLAM (b) Proposed approach

Fig. 5 Feature detection comparison

Two types of sequences were used in this evaluation. In the fr3_w sequences, two people are walking in the room, moving behind a desk, passing in front of the camera, and sitting on chairs. These sequences are, therefore, highly dynamic. The fr3_s sequences can be considered low-dynamic, as people are sitting, making movements mainly with their hands. Both types are used in this evaluation.

There are four types of camera motion considered: xyz, rpy, half, and static. For the motion xyz, the camera is moved along the three axes, keeping the same orientation. In the rpy sequence, the camera is rotated over roll, pitch, and yaw axes. In the half sequence, the camera follows the trajectory of a half-sphere. In the static sequence, the camera is manually kept at the same position and orientation. Table 2 shows the duration, trajectory length, average translational velocity, and average rotational velocity of the camera for every sequence.

The Absolute Trajectory Error (ATE) [43] 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 5 shows the computation of the ATE at a time step i:

$$ATE_i = E_i^{-1} T G_i \tag{5}$$

where E is the estimated trajectory, G represents the ground truth, and T is the transformation that aligns the two trajectories. For a sequence of N poses, the RMSE of ATE is given by Eq. 6.

$$RMSE(ATE_{1:N}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|trans(ATE_i)\|^2} \tag{6}$$

The Relative Pose Error (RPE) is used to evaluate the translational and rotational drifts of the trajectory over a fixed interval Δ. The RPE at a time step i is shown in Eq. 7. The RMSE of RPE is given by Eq. 8.

$$RPE_i = (G_i^{-1} G_{i+\Delta})^{-1} (E_i^{-1} E_{i+\Delta}) \tag{7}$$

$$RMSE(RPE_{1:N}, \Delta) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \|trans(RPE_i)\|^2} \tag{8}$$

where $m = N - \Delta$.

All tests were performed five times and the median results were used for the evaluation, as proposed by Mur-Artal and Tardós [1], to consider the non-deterministic nature of the system.

Figures 7, 8, 9, 10 and 11 show the ATE plots from ORB-SLAM2, Soares et al. [5] (OD approach), and

(a) With no people in the scene (b) With 1 people in the scene (c) With 2 people in the scene

Fig. 6 ORB feature detection

Table 2 Details of each TUM dataset sequence

Sequence	Duration [s]	Length [m]	Vel. [m/s]	Vel. [deg/s]
fr3_s_static	23.63	0.259	0.011	1.699
fr3_s_xyz	42.50	5.496	0.132	3.562
fr3_s_rpy	27.48	1.110	0.042	23.841
fr3_s_half	37.15	6.503	0.180	19.094
fr3_w_static	24.83	0.282	0.012	1.388
fr3_w_xyz	28.83	5.791	0.208	5.490
fr3_w_rpy	30.61	2.698	0.091	20.903
fr3_w_half	35.81	7.686	0.221	18.267

Crowd-SLAM for the fr3_w_static, xyz, rpy, halfsphere, and fr3_s_xyz sequences.

In the fr3_sitting_xyz sequence, all three trajectories are close to the ground truth. The low-dynamic nature of this sequence allows ORB-SLAM2 to eliminate the few dynamic features through its outlier detection methods, such as RANSAC. In the walking sequences, on the other hand, ORB-SLAM2 is not able to detect the highly dynamic features and the estimated trajectories deviate from the ground truth.

Table 3 shows the Root Mean Square (RMSE) and Mean of the ATE comparison between Crowd-SLAM and four direct methods for dynamic environments: ReFusion [13], StaticFusion [12], and the works of Sun et al. [16, 17]. ReFusion and StaticFusion results were obtained in the work of Palazzolo et al. [13]. Our system outperformed the direct methods in all evaluated sequences.

Crowd-SLAM was also compared with ORB-SLAM2 and three feature-based methods for dynamic environments based on ORB-SLAM2: DS-SLAM [21], DynaSLAM [24], and SOF-SLAM [22]. The results are shown in Table 4. DynaSLAM has the best RMSE results in four sequences. However, the difference between their results and Crowd-SLAM is between 1mm and 9mm, depending on the sequence. As DynaSLAM is an offline method, our results are satisfactory.

Crowd-SLAM was also compared with three Visual SLAM systems that use object detection to filter dynamic content: Liu et al. [33], Detect SLAM [32], and Dynamic SLAM [34]. The results are shown in Table 5. Our system achieved better results in two sequences, Dynamic SLAM achieved better results in four sequences, and Liu et al. in two sequences. Overall the results of the four methods are similar, except in the fr3_w_rpy result of Detect-SLAM which had a higher error.

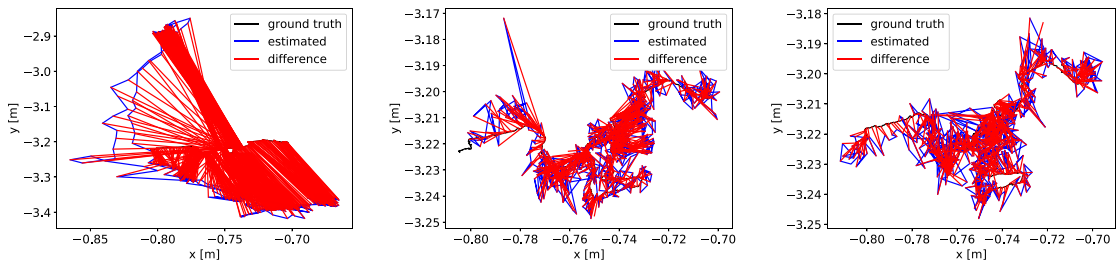
Tables 6 and 7 show the RMSE and Mean of translational and rotational drifts (RPE), respectively, of Crowd-SLAM against ORB-SLAM2, DS-SLAM, and two works of Sun et al. [16, 17], in m/s and deg/s. In the translational drift analysis, Crowd-SLAM outperformed the other works in five sequences, achieving values similar to the best results on the other three sequences. In the rotational drift analysis, Crowd-SLAM achieved the best results in six sequences. For instance, in the fr3_w_rpy sequence, Crowd-SLAM achieved nearly half the error of DS-SLAM.

4.2 Bonn RGB-D Dynamic Dataset

Another evaluation was made using the Bonn RGB-D Dynamic Dataset [13]. It is a dataset with highly dynamic sequences, with people walking and performing different tasks. It was recorded with an Asus Xtion Pro Live Sensor and an Optitrack Prime 13 motion capture system for the ground truth. It also has the same evaluation metrics of the TUM dataset.

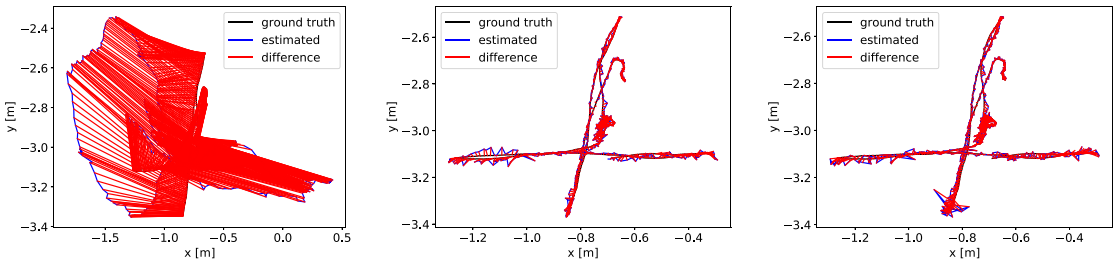
Five sequences of the dataset were chosen for the evaluation: crowd1, crowd2, crowd3, synchonous1, and synchronous2. Despite having less abrupt camera movements in comparison with the TUM sequences, for example roll spin, the sequences of the Bonn dataset have more challenging scenarios. Figure 12 shows a frame where most of the keypoints are filtered by the presence of 3 people close to the camera. Even so, the system is able to perform tracking.

As done in the TUM sequences, all tests were performed five times and the median results were used for the evaluation. Figures 13, 14, 15 and 16 show the ATE plots from ORB-SLAM2 and Crowd-SLAM for the crowd and synchronous sequences. All ORB-SLAM2 trajectories



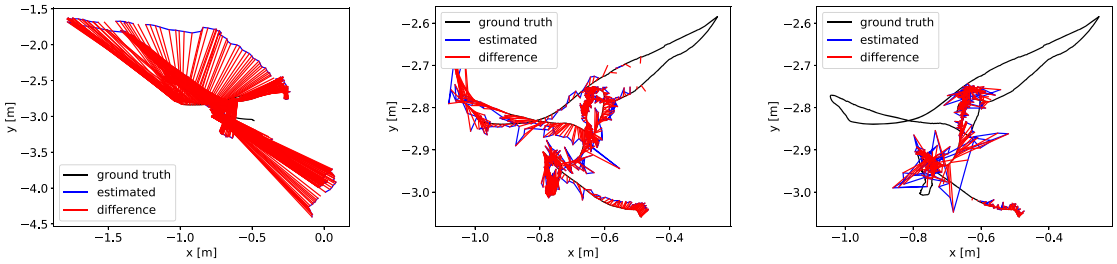
(a) ORB-SLAM2 (b) Soares et al. (OD) (c) Crowd-SLAM

Fig. 7 Ground truth and estimated trajectory in the sequence fr3_walking_static



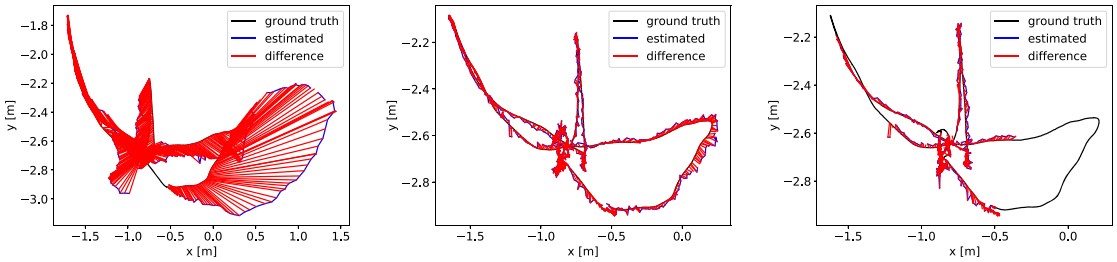
(a) ORB-SLAM2 (b) Soares et al. (OD) (c) Crowd-SLAM

Fig. 8 Ground truth and estimated trajectory in the sequence fr3_walking_xyz



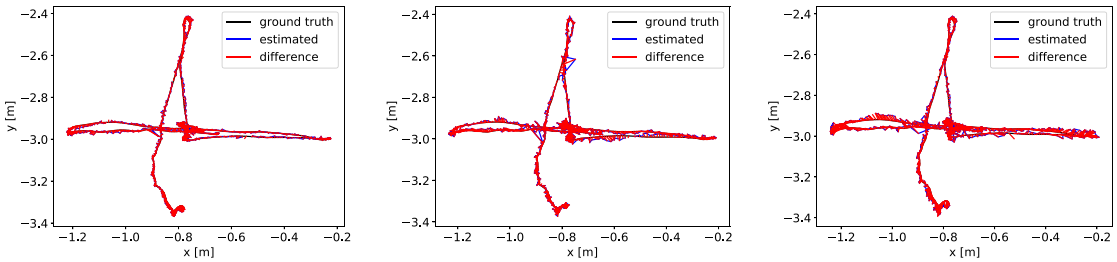
(a) ORB-SLAM2 (b) Soares et al. (OD) (c) Crowd-SLAM

Fig. 9 Ground truth and estimated trajectory in the sequence fr3_walking_rpy



(a) ORB-SLAM2 (b) Soares et al. (OD) (c) Crowd-SLAM

Fig. 10 Ground truth and est. trajectory in the sequence fr3_walking_halfsphere



(a) ORB-SLAM2 (b) Soares et al. (OD) (c) Crowd-SLAM

Fig. 11 Ground truth and estimated trajectory in the sequence fr3_sitting_xyz

Table 3 Comparison of the RMSE and Mean of ATE [m] of Crowd-SLAM against Sun et al. StaticFusion and ReFusion using the TUM dataset

	Sun et al. [16]	Sun et al. [17]	StaticFusion	ReFusion	Crowd-SLAM
Sequence	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean
fr3_s_static	– / –	– / –	0.014 / –	0.009 / –	**0.008 / 0.007**
fr3_s_xyz	0.048 / 0.039	0.051 / 0.043	0.039 / –	0.040 / –	**0.018 / 0.017**
fr3_s_half	0.047 / 0.040	0.066 / 0.054	0.041 / –	0.110 / –	**0.020 / 0.017**
fr3_w_static	0.065 / 0.038	0.033 / 0.026	0.015 / –	0.017 / –	**0.007 / 0.007**
fr3_w_xyz	0.093 / 0.076	0.066 / 0.055	0.093 / –	0.099 / –	**0.020 / 0.017**
fr3_w_rpy	0.133 / 0.103	0.073 / 0.065	– / –	– / –	**0.044 / 0.031**
fr3_w_half	0.125 / 0.087	0.067 / 0.061	0.681 / –	0.104 / –	**0.026 / 0.022**

The bold entries are the best results for the respective sequences

Table 4 Comparison of the RMSE and Mean of ATE [m] of Crowd-SLAM against ORB-SLAM2, DS-SLAM, DynaSLAM, and SOF-SLAM using the TUM dataset

	ORB-SLAM2	DS-SLAM	DynaSLAM	SOF-SLAM	Crowd-SLAM
Sequence	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean
fr3_s_static	0.008 / 0.008	**0.006 / 0.006**	– / –	0.010 / –	0.008 / 0.007
fr3_s_xyz	**0.009 / 0.008**	– / –	0.015 / –	– / –	0.018 / 0.017
fr3_s_rpy	0.019 / 0.016	– / –	– / –	– / –	**0.015 / 0.013**
fr3_s_half	0.021 / **0.017**	– / –	**0.017** / –	– / –	0.020 / **0.017**
fr3_w_static	0.409 / 0.367	0.008 / **0.007**	**0.006** / –	0.007 / –	0.007 / **0.007**
fr3_w_xyz	0.724 / 0.621	0.024 / 0.019	**0.015** / –	0.018 / –	0.020 / **0.017**
fr3_w_rpy	0.781 / 0.676	0.444 / 0.377	0.035 / –	**0.027** / –	0.044 / **0.031**
fr3_w_half	0.374 / 0.303	0.030 / 0.026	**0.025** / –	0.029 / –	0.026 / **0.022**

The bold entries are the best results for the respective sequences

Table 5 Comparison of the RMSE and Mean of ATE [m] of Crowd-SLAM against Liu et al. Detect SLAM, and Dynamic SLAM using the TUM dataset

	Liu et al.	Detect-SLAM	Dynamic SLAM	Crowd-SLAM
Sequence	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean
fr3_s_static	**0.006 / 0.005**	– / –	– / –	0.008 / 0.007
fr3_s_xyz	– / –	0.020 / –	**0.006 / 0.006**	0.018 / 0.017
fr3_s_rpy	– / –	– / –	0.034 / 0.032	**0.015 / 0.013**
fr3_s_half	– / –	0.023 / –	**0.015 / 0.013**	0.020 / 0.017
fr3_w_static	0.010 / **0.007**	– / –	– / –	**0.007 / 0.007**
fr3_w_xyz	0.016 / 0.014	0.024 / –	**0.013 / 0.011**	0.020 / 0.017
fr3_w_rpy	**0.042 / 0.030**	0.296 / –	0.060 / 0.054	0.044 / 0.031
fr3_w_half	0.031 / 0.026	0.051 / –	**0.021 / 0.018**	0.026 / 0.022

The bold entries are the best results for the respective sequences

Table 6 RMSE and Mean values of the Translational Drift (RPE) in m/s of Crowd-SLAM against ORB-SLAM2, DS-SLAM, and Sun et al. using the TUM dataset

	ORB-SLAM2	DS-SLAM	Sun et al. [16]	Sun et al. [17]	Crowd-SLAM
Sequence	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean
fr3_s_static	0.009 / 0.008	**0.008 / 0.007**	– / –	– / –	0.009 / 0.008
fr3_s_xyz	**0.011 / 0.010**	– / –	0.033 / 0.024	0.036 / 0.028	0.020 / 0.018
fr3_s_rpy	0.025 / 0.020	– / –	– / –	– / –	**0.021 / 0.018**
fr3_s_half	0.024 / **0.017**	– / –	0.046 / 0.037	0.055 / 0.044	**0.022** / 0.019
fr3_w_static	0.234 / 0.099	**0.010 / 0.009**	0.084 / 0.045	0.031 / 0.023	**0.010 / 0.009**
fr3_w_xyz	0.384 / 0.297	0.033 / 0.024	0.121 / 0.089	0.067 / 0.056	**0.025 / 0.022**
fr3_w_rpy	0.373 / 0.262	0.150 / 0.094	0.175 / 0.136	0.097 / 0.082	**0.065 / 0.047**
fr3_w_half	0.323 / 0.188	**0.030 / 0.026**	0.167 / 0.108	0.061 / 0.055	0.037 / 0.031

The bold entries are the best results for the respective sequences

Table 7 RMSE and Mean values of the Rotational Drift (RPE) in deg/s of Crowd-SLAM against ORB-SLAM2, DS-SLAM, and Sun et al. using the TUM dataset

	ORB-SLAM2	DS-SLAM	Sun et al. [16]	Sun et al. [17]	Crowd-SLAM
Sequence	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean	RMSE / Mean
fr3_s_static	0.289 / 0.261	0.273 / 0.245	– / –	– / –	**0.261** / **0.235**
fr3_s_xyz	0.483 / **0.406**	– / –	0.983 / 0.806	1.036 / 0.890	**0.478** / 0.413
fr3_s_rpy	0.784 / 0.667	– / –	– / –	– / –	**0.508** / **0.446**
fr3_s_half	**0.598** / **0.538**	– / –	2.375 / 1.893	2.268 / 1.795	0.653 / 0.565
fr3_w_static	4.207 / 1.830	0.269 / 0.242	2.049 / 1.055	0.900 / 0.625	**0.265** / **0.237**
fr3_w_xyz	7.302 / 5.652	0.826 / 0.584	3.235 / 2.256	1.595 / 1.366	**0.658** / **0.524**
fr3_w_rpy	7.229 / 5.169	3.004 / 1.919	4.375 / 3.360	2.593 / 2.232	**1.519** / **1.130**
fr3_w_half	5.960 / 3.615	**0.814** / **0.703**	5.010 / 3.288	1.900 / 1.740	0.820 / 0.725

The bold entries are the best results for the respective sequences

deviate from the ground truth. Our system, on the other hand, was able to achieve low errors.

Table 8 shows the ATE comparison between Crowd-SLAM and DynaSLAM, ReFusion [13], and StaticFusion [12]. Their results were obtained in the work of Palazzolo et al. [13]. Our system outperformed all methods in three sequences, including DynaSLAM. In all three *crowd* sequences, Crowd-SLAM outperformed ORB-SLAM2, StaticFusion, and ReFusion by a high margin.

4.3 ETH Stereo Dataset

Despite being broadly used as a benchmark for Visual SLAM systems, the TUM Dataset is not ideal for a good evaluation of crowded environments, as their dynamic sequences only contain at maximum two people in the scene. Also, the Bonn RGB-D Dynamic Dataset sequences contain a maximum of three people. The ETH Loewenplatz sequence [44] was used to test the system in more crowded scenarios. It consists of an autonomous robot, SmartTer [45], traveling through a road with people along the sidewalks and crossing the streets, with a stereo camera pair recording images at 13 FPS. It provides the cameras calibration and odometry. Figure 17a shows an image of the sequence, together with the people detection in Fig. 17b, and the corresponding Dynamic Feature Removal in Fig. 17c.

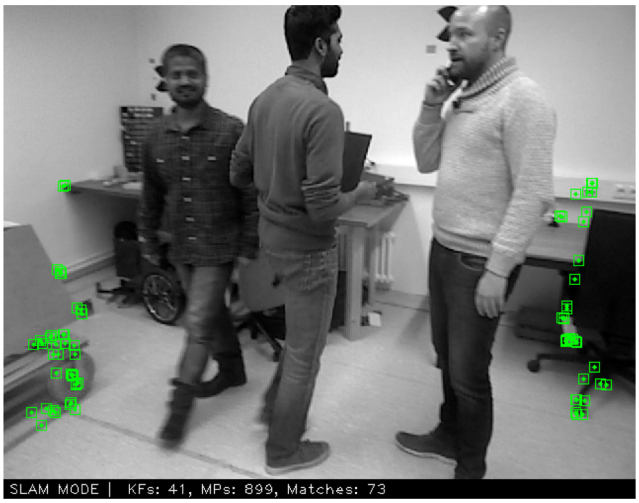


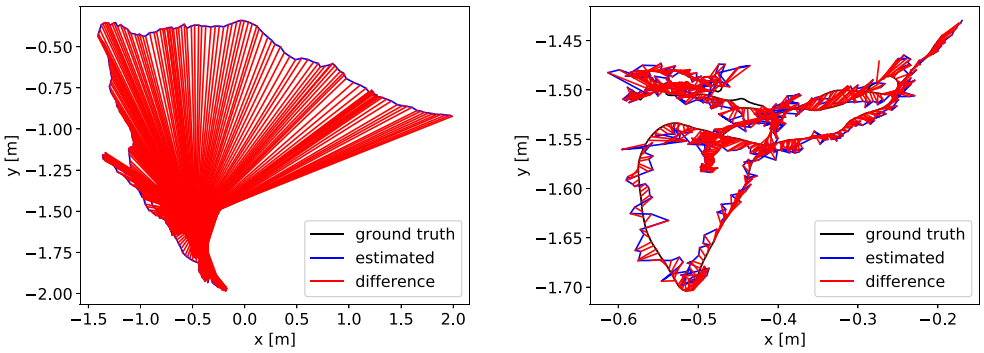
Fig. 12 Keypoint filtering in Bonn Crowd scene

The provided odometry was made with visual feature tracking using the static background and offline bundle adjustment. An ATE comparison was made between Crowd-SLAM and the provided odometry, and between ORB-SLAM2 and the provided odometry. Crowd-SLAM achieved a RMSE of 33.90 m, and ORB-SLAM2 achieved a RMSE of 41.35 m. As there were no major loop closures in the trajectory, this improvement is due to the tracking system of Crowd-SLAM. Figure 18 shows the trajectories estimated by Crowd-SLAM and ORB-SLAM2 compared to the provided odometry. With an absence of loop closures, an eventual gradual deviation was expected. However, the trajectory of ORB-SLAM2 immediately deviates from the odometry due to the presence of people.

4.4 Feature Number Update Improvements

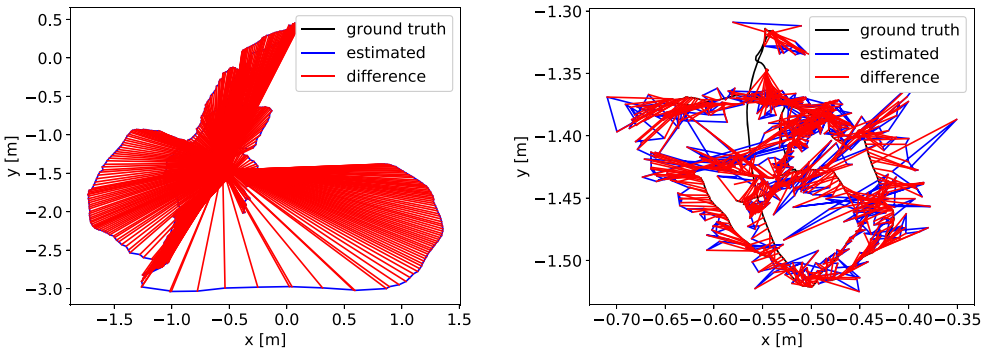
The objective of the feature number update (FNU) step is to prevent lost track due to keypoint filtering. Table 9 shows the percentage of successfully tracked frames in six sequences, with and without the update, using the standard ORB-SLAM2 feature number.

Using the standard number of features without update, the system is not able to recover from lost track in the fr3_s_half sequence. Using the proposed approach, the system is able to track more frames. In other sequences, the FNU also provided a considerable improvement. The low percentage of tracked frames in the $fr3\_s\_half$ sequence is caused by two main reasons: (i) this sequence has



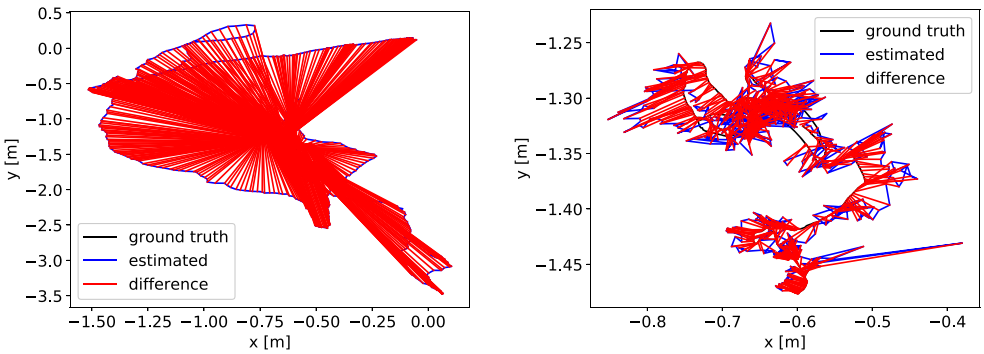
(a) ORB-SLAM2 (b) Crowd-SLAM

Fig. 13 Ground truth and estimated trajectory in the sequence crowd1



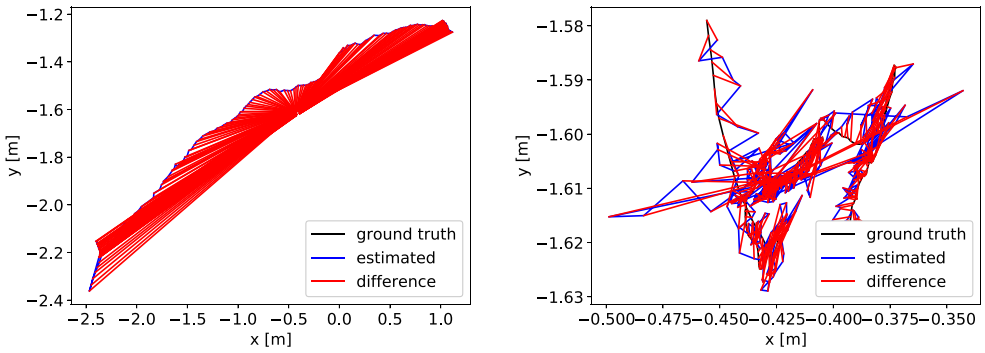
(a) ORB-SLAM2 (b) Crowd-SLAM

Fig. 14 Ground truth and estimated trajectory in the sequence crowd2



(a) ORB-SLAM2 (b) Crowd-SLAM

Fig. 15 Ground truth and estimated trajectory in the sequence crowd3



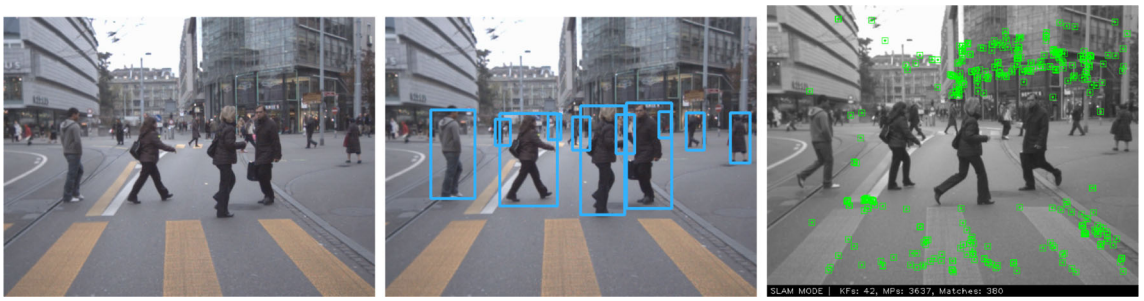
(a) ORB-SLAM2 (b) Crowd-SLAM

Fig. 16 Ground truth and estimated trajectory in the sequence synchronous2

Table 8 Comparison of the RMSE of ATE [m] of Crowd-SLAM against ORB-SLAM2, StaticFusion, ReFusion, and DynaSLAM using the Bonn Dataset

Sequence	ORB-SLAM2	StaticFusion	ReFusion	DynaSLAM	Crowd-SLAM
crowd1	0.963	3.586	0.204	**0.016**	0.018
crowd2	1.372	0.215	0.155	0.031	**0.030**
crowd3	1.262	0.168	0.137	0.038	**0.034**
synchronous1	1.121	0.446	0.441	0.015	**0.009**
synchronous2	1.507	0.027	0.022	**0.009**	0.012

The bold entries are the best results for the respective sequences



(a) RGB Left Image (b) Object Detection (c) Outlier removal

Fig. 17 Crowd-SLAM steps in ETH Loewenplatz sequence

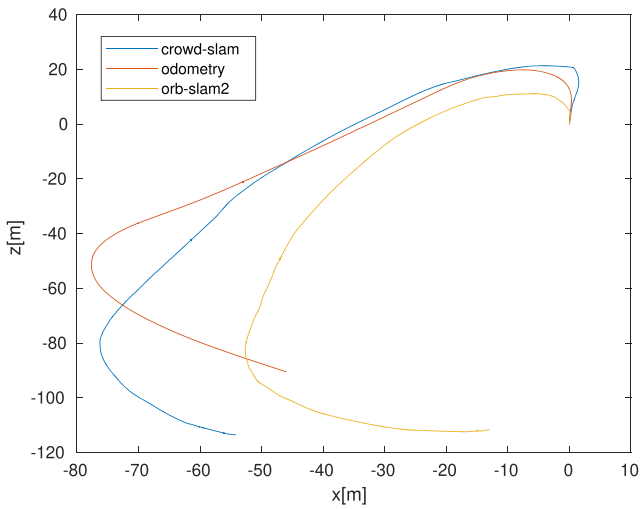


Fig. 18 Trajectory results from Crowd-SLAM and ORB-SLAM2 compared with the provided odometry for the Loewenplatz sequence

Table 9 Percentage of successfully tracked frames

Sequence	Without FNU	With FNU
fr3_s_static	73.09	99.41
fr3_s_xyz	73.83	90.97
fr3_w_static	71.27	95.67
fr3_w_xyz	83.31	99.88
crowd2	82.46	85.47
fr3_s_half	0.0	23.09

several frames with the camera close to one person, causing depletion of features; and (ii) it has several large roll camera movements. However, the system was able to correctly relocalize in all situations.

4.5 Implementation and Run-time Analysis

All tests were performed on a notebook with an Intel Core i7 6700 HQ 2.60 GHz 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, using only CPU. Table 10 shows the mean frame rate in FPS of ORB-SLAM2 and Crowd-SLAM for every sequence of Bonn and TUM datasets used for evaluation. Crowd-SLAM achieved an average frame rate of 26.22 FPS, while ORB-SLAM2 achieved 21.50 FPS. For comparison, Detect-SLAM, which also uses object detection, spends 0.34 seconds per frame just for moving-object removal. Dynamic-SLAM achieved a mean performance of 22.2 FPS on $fr3\_w\_xyz$ with GPU. Also, the achieved frame rate of Crowd-SLAM is higher than those of other systems that use GPU, for instance,

Table 10 Mean tracking time [FPS] of ORB-SLAM2 and Crowd-SLAM in the TUM and Bonn sequences

Sequence	ORB-SLAM2	Crowd-SLAM
fr3_s_static	24.260	28.765
fr3_s_xyz	27.894	25.953
fr3_s_rpy	24.343	30.049
fr3_s_half	21.886	30.206
fr3_w_static	17.361	25.893
fr3_w_xyz	18.416	23.960
fr3_w_rpy	21.218	28.432
fr3_w_half	18.487	28.211
crowd1	18.702	28.216
crowd2	18.318	26.492
crowd3	17.721	28.293
synchronous1	25.189	35.534
synchronous2	25.773	29.330
Average	21.505	**26.223**

The bold entries are the best results for the respective sequences

Fig. 19 Bounding box occupying a large portion of the image

DynaSLAM (1.35 FPS on $fr3\_w\_halfsphere$), and DS-SLAM (13.08 FPS).

4.6 Limitations

Our approach suffers from two main drawbacks. Although the FNU was able to prevent lost track in many scenarios, there is a limitation. Figure 19 shows a frame of the $fr3\_s\_half$ sequence where the bounding box of a person occupies a large portion of the scene, due to the pose of the person and the proximity of the camera, which may cause the depletion of features, even with the FNU system. To overcome this issue, it would be necessary to use the static features detected inside the bounding box. A possible approach to this problem is to use epipolar geometry, matching the current frame with past frames, and selecting features inside the bounding box using a geometric constraint.

Secondly, our approach does not filter other moving objects besides people. However, in the considered crowded environment scenarios, the main source of movement indeed came from people.

5 Conclusion

This work presented Crowd-SLAM, a new open-source Visual SLAM system designed to perform in crowded human environments. The system is based on ORB-SLAM2, with four main threads: tracking, object detection, local mapping, and loop closing. An efficient dynamic keypoint filtering algorithm was proposed, together with a newly trained network for object detection, and a feature number update system.

The effectiveness of Crowd-SLAM was evaluated on challenging dynamic sequences of the TUM, Bonn, and

Loewenplatz datasets. The results indicate that the proposed methodology was successful, with a lower computational time and a better accuracy compared to state-of-the-art methods. To our knowledge, the proposed system has the best results in the crowd2, crowd3, and synchronous1 sequences of the Bonn dataset.

There are several open problems to explore for future works. For instance, to allow filtering other moving objects without jeopardizing the performance, and to use the static features inside the bounding boxes in order to prevent more lost track cases. Other promising works include the integration of tracking and loop closing modules, and the extension of the methodology to work with long-term dynamic changes. Moreover, we aim to test our system in a real robot in a crowded environment.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10846-021-01414-1.

Author Contributions All authors contributed to the methodology conception and design. João Carlos Virgolino Soares collected the data, performed the analysis, and wrote the manuscript. Marcelo Gattass and Marco Antonio Meggiolaro reviewed and approved the manuscript.

Funding Partial financial support was received from the Brazilian National Council for Scientific and Technological Development (CNPq).

Availability of Data and Materials The code and datasets used are publicly available in:

- Crowd-SLAM: https://github.com/virgolinosoares/Crowd-SLAM
- MOT Challenge: https://motchallenge.net/
- TUM RGB-D Dataset: https://vision.in.tum.de/data/datasets/rgbd-dataset
- LOEWENPLATZ: https://data.vision.ee.ethz.ch/cvl/aess/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. Mur-Artal, R., Tardós, J.: ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**, 1255–1262 (2017)
2. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale direct monocular SLAM. In: European Conference On Computer Vision, pp. 834–849 (2014)
3. Endres, F., Hess, J., Sturm, J., Cremers, D.: Burgard, w.: 3D mapping with an RGB-D camera. IEEE Trans. Robot. **30**(1), 177–187 (2013)
4. Labbé, M., Michaud, F.: RTAB-Map as an open-source lidar and visual SLAM library for large-scale and long-term online operation. J. Field Robot. **36**(2), 416–446 (2019)
5. Soares, J.C.V., Gattass, M., Meggiolaro, M.: Visual SLAM in human populated environments: exploring the trade-off between accuracy and speed of YOLO and Mask R-CNN. In: Proc of the IEEE International Conference on Advanced Robotics (2019)
6. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Trans. Pattern Anal. Machine Intell. **29**(6), 1052–1067 (2007)
7. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (2007)
8. Mur-Artal, R., Montiel, J., Tardós, J.: ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transact. Robot. **31**(5), 1147–1163 (2015)
9. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. Int. J. Robot. Res. **31**, 647–663 (2012)
10. Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 10th IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136 (2011)
11. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Trans. Pattern Anal. Machine Intell. **40**(3), 611–625 (2017)
12. Scona, R., Jaimez, M., Petillot, Y.R., Fallon, M., Cremers, D.: StaticFusion: background reconstruction for dense RGB-D SLAM in dynamic environments. In: IEEE International Conference on Robotics and Automation (2018)
13. Palazzolo, E., Behley, J., Lottes, P., Giguère, P., Stachniss, C.: ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2019)
14. Dib, A., Charpillet, F.: Robust dense visual odometry for RGB-D cameras in a dynamic environment. In: Proc. of the International Conference on Advanced Robotics, Istanbul, pp. 1–7 (2015)
15. Alcantarilla, P.F., Yebes, J.J., Almazan, J., Bergasa, L.M.: On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In: Proc. of the International Conference on Robotics and Automation (2012)
16. Sun, Y., Liu, M., Meng, M.Q.H.: Improving RGB-D SLAM in dynamic environments: a motion removal approach. Robot. Autonom. Syst. **89**, 110–122 (2017)
17. Sun, Y., Liu, M., Meng, M.Q.H.: Motion removal for reliable RGB-D SLAM in dynamic environments. Robot. Auton. Syst. **108**, 115–128 (2018)
18. Wang, Y., Huang, S.: Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios. In: Proc. of the 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, pp. 1841–1846 (2014)
19. Kim, D., Kim, J.: Effective background model-based RGB-D dense visual odometry in a dynamic environment. IEEE Trans. Robot. **32**(6), 1565–1573 (2016)
20. Cheng, J., Sun, Y., Chi, W., Wang, C., Cheng, H., Meng, M.Q.H.: An accurate localization scheme for mobile robots using optical flow in dynamic environments. In: Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), 723–728 (2018)
21. Yu, C., Liu, Z., Liu, X., Xie, F., Yang, Y., Wei, Q., Fei, Q.: DS-SLAM: a semantic visual SLAM towards dynamic environments. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2018)

22. Cui, L., Ma, C.: SOF-SLAM: A semantic visual SLAM for dynamic environments. IEEE Access **7**, 166528–166539 (2019)
23. Sun, T., Sun, Y., Liu, M., Yeung, D.Y.: Movable-object-aware visual SLAM via weakly supervised semantic segmentation. arXiv:1906.03629 (2019)
24. Bescós, B., Fácil, J., Civeira, J., Neira, J.: DynaSLAM: Tracking, mapping and inpainting in dynamic environments. IEEE Robot. Autom. Lett. **3**(4), 4076–4083 (2018)
25. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 2961–2969 (2017)
26. Girshick, R.: Fast R-CNN. In: Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. Advances in neural information processing systems 91–99 (2015)
28. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
29. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: Proc. of the European Conference on Computer Vision, pp. 21–37 (2016)
30. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement. In: Arxiv:1804.02767 (2018)
31. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proc. of the 13th European Conference on Computer Vision (2014)
32. Zhong, F., Wang, S., Zhang, Z., Wang, Y.: Detect-SLAM: making object detection and SLAM mutually beneficial. In: IEEE Winter Conference on Applications of Computer Vision (WACV)., pp. 1001–1010 (2018)
33. Liu, H., Liu, G., Tian, G., Xin, S., Ji, Z.: Visual SLAM based on dynamic object removal. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 596–601 (2019)
34. 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)
35. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571 (2011)
36. Gálvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Robot. **28**(5), 1188–1197 (2012)
37. Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: A general framework for graph optimization. In: IEEE Int. Conf. on Robot. and Autom. (ICRA), pp. 3607–3613 (2011)
38. Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., Leal-Taixé, L.: MOT20: A benchmark for multi object tracking in crowded scenes. In: arXiv:2003.09003 (2020)
39. Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., Sun, J.: Crowdhuman: A benchmark for detecting human in a crowd. In: arXiv:1805.00123 (2018)
40. Redmon, J.: Darknet: Open source neural networks in C. http://pjreddie.com/darknet (2016)
41. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: MOT16: A benchmark for multi-object tracking. In: arXiv:1603.00831 (2016)
42. Stiefelhagen, R., Bernardin, K., Bowers, R., Garofolo, J., Mostefa, D., Soundararajan, P.: The CLEAR 2006 evaluation. In: International evaluation workshop on classification of events, activities and relationships, pp. 1–44 (2006)
43. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573–580 (2012)
44. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: A mobile vision system for robust multi-person tracking. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
45. Scaramuzza, D., Spinello, L., Triebel, R., Siegwart, R.: Key technologies for intelligent and safer cars-from motion estimation to predictive collision avoidance. In: IEEE International Symposium on Industrial Electronics, pp. 2803–2808 (2010)

João Carlos Virgolino Soares is a Ph.D. Candidate at the Mechanical Engineering Department of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). He graduated in Mechanical Engineering at PUC-Rio in 2015, having obtained his M.Sc. in the same institution in 2018. His research interests include Mobile Robotics, Visual SLAM, Long-term Mapping, and Pose-graph Optimization.

Marcelo Gattass is the director of the Tecgraf Institute, and a Full Professor at the Department of Informatics of PUC-Rio. He graduated in Civil Engineer at PUC-Rio in 1975, obtained his M.Sc. in the same institution in 1977, and his Ph.D. in Civil Engineer from the Computer Graphics Program of Cornell University in 1982. Gattass is an expert in the field of Computational Science, with emphasis on Graphics Processing, focusing on the following themes: Visualization, Numerical Simulation, Augmented Reality, Geometric Modeling, and Computer Vision. His research currently focuses on Modeling of Natural Objects based on Computer Vision. He has published over 70 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). 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 380 published scientific works.