Visual SLAM in Human Populated Environments: Exploring the Trade-off between Accuracy and Speed of YOLO and Mask R-CNN

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

***Abstract*— Simultaneous Localization and Mapping (SLAM) 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 direct or feature-based methods. Deep learning techniques can improve the SLAM solution in environments with *a priori* dynamic objects, providing high-level information of the scene. This paper presents a new approach to SLAM in human populated environments using deep learning-based techniques. The system is built on ORB-SLAM2, a state-of-the-art SLAM system. The proposed methodology is evaluated using a benchmark dataset, outperforming other Visual SLAM methods in highly dynamic scenarios.**

I. INTRODUCTION

The Simultaneous Localization and Mapping (SLAM) problem is a fundamental step for several mobile robotics applications, such as navigation. It consists of creating a map of the environment using only sensor measurements without external aid, while 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. RGB-D cameras have an extra advantage of providing dense depth information. There are several Visual SLAM systems in the literature, with high precision and efficiency, for instance, ORB-SLAM2 [1], LSD-SLAM [2], Henry *et al.* [3], rgbdslam [4], and RTAB-Map [5]. However, their majority assume a static environment, which imposes a limitation of their applicability in real-world scenarios.

ORB-SLAM2 is a state-of-the-art graph-based Visual SLAM system that can work with RGB-D, Stereo, or Monocular cameras. However, it does not work properly in dynamic environments. Fig. 1 shows an example of the point cloud map generated by the ORB-SLAM2 system, using data from an RGB-D dataset containing people moving in the scene. The map was corrupted by their movement.

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. Most SLAM systems that work in dynamic environments rely on classic approaches, such as optical flow, 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. Deep Learning techniques, on the other hand, have high-level information that allows the recognition of such objects. The focus of this work is to propose a SLAM system that uses deep learning techniques to enable its applicability in scenarios with *a priori* dynamic objects.

Fig. 1: Corrupted point cloud map

This paper proposes a complete SLAM system for human populated environments based on ORB-SLAM2 using two different approaches, both relying on deep learning techniques to detect and filter people in the scene. It is discussed the trade-off between them in terms of speed and accuracy. The proposed methodologies are evaluated using the TUM RGB-D dataset [6] and compared with other SLAM systems. The main contributions of this paper can be summarized as follows:

- Two new approaches for performing SLAM in human populated environments are proposed
- The proposed methodologies outperform the accuracy of ORB-SLAM2 and other state-of-the-art systems in highly dynamic environments
- The achieved run-time performance is superior comparing with approaches with similar accuracy

The paper is organized as follows. Section II discusses related work, section III details the proposed methodology, section IV shows the results and section V presents conclusions and suggestions for future work.

II. RELATED WORK

A. SLAM in dynamic environments

Most state-of-the-art SLAM systems are not able to handle dynamic scenarios, as they were designed with a static

[1]J. Soares and M. Meggiolaro are with the Department of Mechanical Engineering, Pontifícia Universidade Católica do Rio de Janeiro, R. Marquês de São Vicente 225, Gávea, Rio de Janeiro, RJ - Brazil virgolinosoares@gmail.com / meggi@puc-rio.br

[2]M. Gattass is with the Department of Informatics, Pontifícia Universidade Católica do Rio de Janeiro, R. Marquês de São Vicente 225, Gávea, Rio de Janeiro, RJ - Brazil mgattass@tecgraf.puc-rio.br

environment assumption. The Visual SLAM systems that deal with dynamic content in the scene usually treat it as noise, and filter it using feature-based or direct methods, such as Wang and Huang [7] and Kim *et al.* [8].

Dib and Charpillet [9] proposed a dense visual odometry system in dynamic environments using RANSAC. Alcantarila *et al.* [10] proposed a dense scene flow representation to detect moving objects using stereo cameras. Cheng *et al.* [11] proposed a method based solely on optical flow to perform localization in dynamic environments, integrating it with ORB-SLAM.

The previous approaches, however, are unable to detect *a priori* dynamic objects in the scene, such as people or cars. The DS-SLAM [12] system deals with dynamic objects combing the optical flow method with a semantic segmentation network, which allows the detection of people.

DynaSLAM [13] uses the Mask R-CNN [14] instance segmentation to obtain the pixel-wise information of people in the scene, using it to filter the outlier keypoints. Despite being accurate and robust, DynaSLAM cannot perform real-time due to the high computational requirement of the Mask R-CNN algorithm.

Object detection, on the other hand, can achieve real-time performance using GPU and also gives high-level information. This work proposes the use of a deep-learning based object detection module to detect *a priori* dynamic objects in the environment, claiming that the trade-off between accuracy and speed against instance segmentation is advantageous.

B. Object Detection

Object detection is the task of determining the location 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 [15] and Faster R-CNN [16], are known as two-stage detectors. They use an algorithm to find potential regions to contain objects and then use a convolutional neural network (CNN) in those regions.

Despite being accurate, these algorithms are slow due to their complex pipelines, and do not work in real time. On the other hand, the YOLO (You Only Look Once) technique proposed by Redmon *et al.* [17] can run at 45 frames per second using a GPU. YOLO is fast because it uses a single convolutional network for the whole image at once. This work uses YOLO for the object detection task.

III. METHODOLOGY

Fig. 2 shows the flowchart of the proposed methodology, composed of four main modules: Object Detection, Instance Segmentation, Outlier Removal, and SLAM. The RGB-D frames are sent to the object detection and instance segmentation modules. The high-level information given is used in an outlier removal system to filter the information submitted to the SLAM module.

Two approaches are considered in this work. In the first approach (a), the object detection module is active when there are no people detected in the scene. Once a person appears, the object detection module switches to instance segmentation. The objective is to use the segmentation only when it is necessary, to increase computational speed. In the second approach (b), only object detection is used to filter the dynamic features in the image. The objective of the second approach is to evaluate if object detection alone is sufficient for keypoint filtering, without the need of instance segmentation. The following section details each module.

A. SLAM

This work uses ORB-SLAM2 as the global SLAM solution. ORB-SLAM2 has three main threads: tracking, loop closing, and local mapping. In this work, both loop closing and local mapping threads are the same as in ORB-SLAM2. However, the tracking thread was modified to include the outlier removal module. This work also adds a thread for the deep learning stages.

The raw RGB images are processed in the deep learning thread and are sent to the outlier removal filter inside the tracking thread, which extracts ORB features [18] from the images and remove the dynamic ones.

ORB-SLAM2 works using a keyframe-based methodology. The tracking thread decides whether every new frame is a new keyframe and saves the corresponding depth images. The loop closing system searches for closed loops of every new keyframe with past keyframes. The place recognition algorithm DBoW2 [19] is used in the loop closing process. Once a loop is detected, the graph is optimized with g2o [20] to assure a consistent trajectory. The output of ORB-SLAM2 are a sparse point cloud map, and the optimized trajectory of the camera.

B. Object Detection

YOLO provides the classes of the detected objects, 2D bounding boxes with their corresponding positions and a confidence number for each box.

This work uses the OpenCV implementation of YOLO, trained with the COCO dataset [21]. Fig. 3 shows an image with detected objects of different classes, with their respective bounding boxes and confidence numbers.

C. Instance Segmentation

Instance segmentation is a combination of object detection and semantic segmentation. The Mask R-CNN [14] is used to perform instance segmentation in the color frames. It gives the pixel-wise information of the detected classes and the bounding boxes with the respective location of the objects. Fig. 4 shows a color frame from a dataset with the pixel-wise information of two persons and a chair.

D. Outlier Removal

Once the images pass through the object detector or instance segmentation, the keypoints that belong to people are removed from the image. For the instance segmentation approach, only the keypoints in pixels that represent people are removed. In the object detection approach, every keypoint

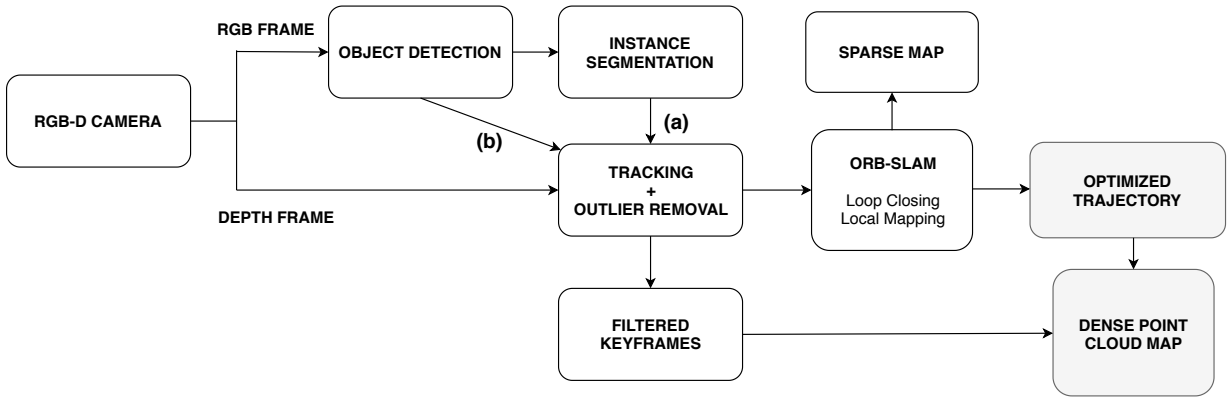

Fig. 2: Flowchart of the proposed approach

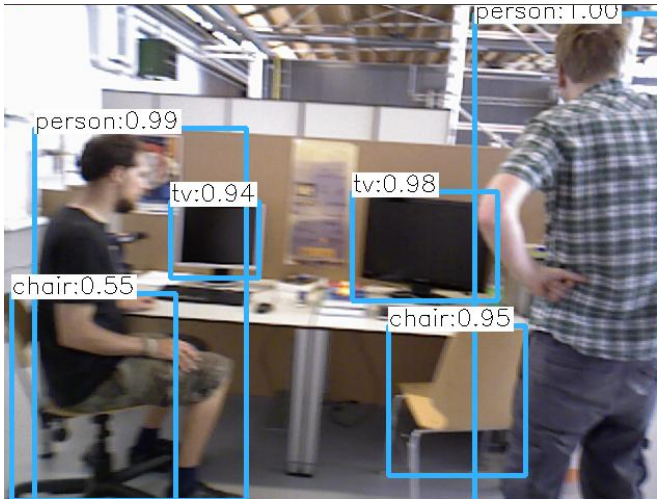

Fig. 3: YOLO Object Detection

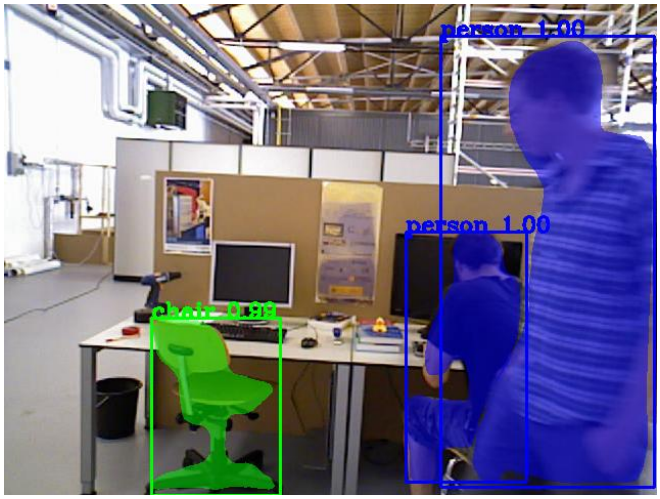

Fig. 4: Mask R-CNN Instance Segmentation

inside the people bounding boxes are removed. Figs. 5a, 5b and 5c show the keypoint detection of ORB-SLAM2 without filter, with the Mask R-CNN filter and with the object detector filter, respectively. The filters successfully erased all keypoints in the regions with people. However, using only object detection, the keypoints appearing in the left chair are also erased, for being merged with the person bounding box, resulting in an indirect filtering of potential dynamic objects.

In the approach using only object detection, a large number of people in the scene can reduce the information available drastically for the SLAM system, causing lost tracking. To overcome this issue, the number of features per image is variable, depending on the number of people in the detected image. If there are no people, the number of features is reduced to decrease the computational time without jeopardizing the accuracy.

E. Point cloud mapping

The ORB-SLAM2 system generates an optimized trajectory of the camera and a sparse map. To generate a dense point cloud map, a post-processing tool is used. During the tracking thread, for each keyframe added, the corresponding depth image is altered in the regions filtered after the object detection or instance segmentation steps and saved. The modified depth images are combined with the RGB images to create colored point clouds, using the camera intrinsic parameters. The point clouds are, then, transformed into a common coordinate frame using the optimized trajectory resulted from the SLAM process. To prevent any occasional false negative in people detection, a keyframe is used for the map only if the number of people remained constant in the past three frames.

IV. RESULTS

The system was numerically evaluated using the TUM RGB-D dataset [6]. 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.

The walking (fr3_w) sequences were chosen for the evaluation. In the sequences, two people are walking in the room, moving behind a desk, passing in front of the camera and sitting in chairs. This dataset is, therefore, highly dynamic.

There are four types of camera motion considered: xyz, rpy, half and static. For the motion xyz, the camera is moved

(a) ORB-SLAM (b) With Mask R-CNN filter (c) With object detection filter

Fig. 5: Feature detection comparison

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 in the same position and orientation. Tab. I shows the duration, length and average translational velocity of the camera for every sequence.

TABLE I: Details of each dataset sequence

Sequence	Duration [s]	Length [m]	Avg. Transl. Vel. [m/s]
fr3_w_static	24.83	0.282	0.012
fr3_w_xyz	28.83	5.791	0.208
fr3_w_rpy	30.61	2.698	0.091
fr3_w_half	35.81	7.686	0.221

A. Evaluation Metrics

The Absolute Trajectory Error (ATE) [6] 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.

All the tests were made 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.

B. Quantitative Results

Tab. II shows the comparison between ORB-SLAM2 [1] and the proposed methodology using the ATE metric [m], with both instance segmentation (IS) and object detection (OD) approaches, for the fr3_walking_xyz sequence. Both approaches achieved similar results and both considerably improved the results of ORB-SLAM2. Thus, the OD approach is more advantageous for having a better run-time performance, maintaining the accuracy obtained by the IS approach.

Figs. 6 through 9 show the ATE plots from ORB-SLAM2 and the proposed methodology with object detection for the fr3_w_xyz and fr3_w_rpy sequences. The proposed methodology significantly reduced the trajectory errors of ORB-SLAM2.

TABLE II: ATE [m] Comparison between the proposed system and ORB-SLAM2 for the fr3_walking_xyz sequence

Error	ORB-SLAM2	Prop. system (IS)	Prop. system (OD)
RMS	0.9083	**0.0165**	0.0169
Mean	0.7699	**0.0143**	0.0145
Median	0.7091	**0.0128**	**0.0128**
Min	0.0654	**0.0003**	**0.0003**
Max	1.9977	**0.0532**	0.0556

The proposed system was also compared with two state-of-the-art systems: DS-SLAM and DynaSLAM. Their error values were taken from their respective papers [12] and [13]. Tab. III shows the ATE comparison between ORB-SLAM2, DS-SLAM, DynaSLAM, and the proposed methodologies. The OD approach outperforms DS-SLAM in three of four sequences.

DynaSLAM, has better results than the OD approach in three sequences. However, the maximum difference between their results is approximately 3 mm for a sequence with 7.686 m of length.

Despite this work not explicitly handling movable objects such as chairs or notebooks, it filters their small displacements when they are inside people's bounding boxes. However, this work does not handle long-term object displacements, which would need an object tracking system, for example.

C. Qualitative Results

Fig. 10 shows the final point cloud map for the sequence fr3_walking_xyz using the object detection approach. Comparing to the final map obtained using ORB-SLAM2, shown in Fig. 1, it can be stated that the method was successful in erasing the people from the map.

D. Implementation and Run-time Analysis

All tests were performed on a laptop with an Intel Core i7 6700 HQ 2.60 GHz and 16 GB of RAM running Ubuntu Linux 16.04 LTS. The overall system is implemented using C++. Using the object detection approach, the total average run-time performance was 550 ms per frame for the

TABLE III: Comparison of the RMSE of ATE [m] of the proposed system against DS-SLAM, DynaSLAM and ORB-SLAM2

Sequence	ORB-SLAM2	DS-SLAM	DynaSLAM	Prop. system (OD)	Prop. system (IS)
fr3_w_static	0.3900	0.0081	**0.0060**	0.0086	0.0070
fr3_w_xyz	0.9083	0.0247	**0.0150**	0.0169	**0.0150**
fr3_w_rpy	0.8705	0.4442	0.0350	0.0332	**0.0303**
fr3_w_half	0.5071	0.0303	**0.0250**	0.0274	0.0260

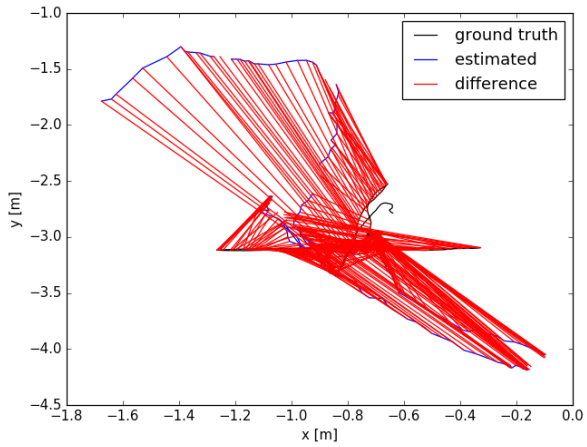

Fig. 6: Ground truth and trajectory estimated by ORB-SLAM2 in the sequence fr3_walking_xyz

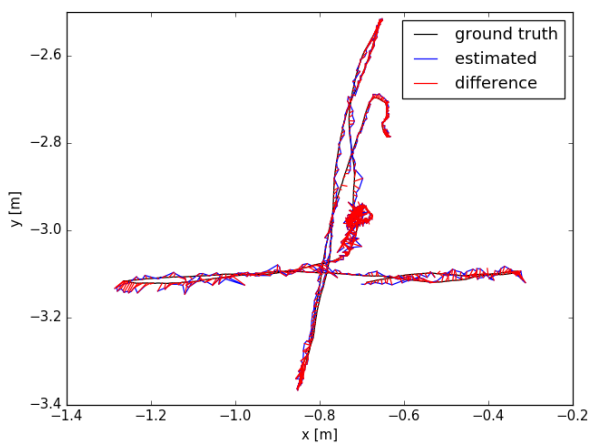

Fig. 7: Ground truth and trajectory estimated by the proposed methodology in the sequence fr3_walking_xyz

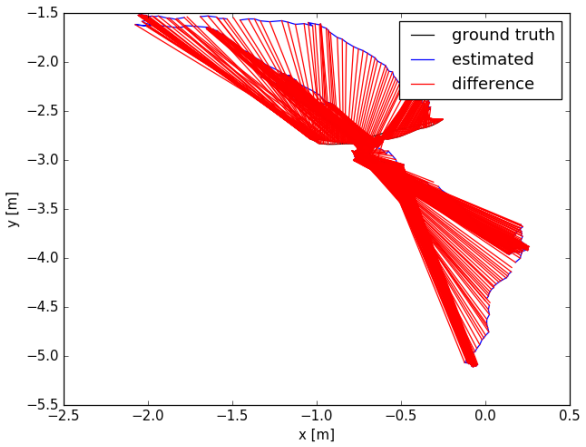

Fig. 8: Ground truth and trajectory estimated by ORB-SLAM2 in the sequence fr3_walking_rpy

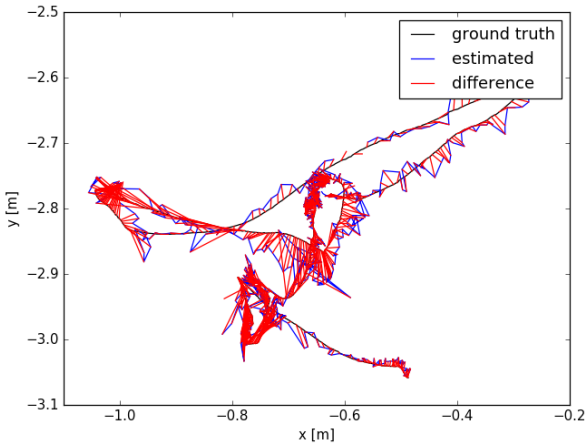

Fig. 9: Ground truth and trajectory estimated by the proposed methodology in the sequence fr3_walking_rpy

fr3_w_rpy sequence. Most of the computational effort comes from the object detector.

DynaSLAM, on the other hand, takes an average time of 335 ms per frame to process their Multi-view Geometry and tracking stages for the fr3_w_rpy sequence. However, the Mask R-CNN runs at more than 2 seconds per frame in a CPU. Even using a NVidia Tesla M40 GPU, the Mask R-CNN alone would run at 195 ms per frame [13].

According to Redmon *et al.* [17], the average computational time of the YOLO object detection is 45 fps running on a GPU. Therefore, the second proposed approach can achieve real-time performance.

V. CONCLUSIONS

This work presented two new approaches to perform SLAM in dynamic environments based on the ORB-SLAM2

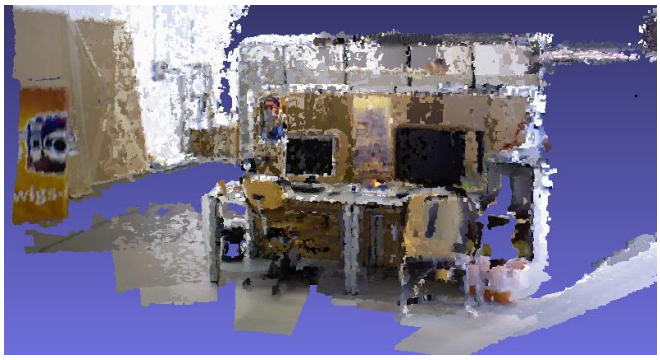

Fig. 10: Filtered point cloud map of fr3_w_xyz sequence

system. The object detection approach proved to be more advantageous than the instance segmentation one, due to its better run-time performance, maintaining the accuracy. Dataset tests indicate that the proposed methodology is successful, with a lower computational time and a better accuracy compared to other methods in the literature. The major drawback of this work is to not explicitly consider other moving objects besides people. Future works include the addition of an object tracking system to filter those moving objects.

REFERENCES

[1] R. Mur-Artal, and J. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Transactions on Robotics, vol. 33, 2017, pp. 1255-1262.

[2] J. Engel, T. Schöps, and D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, In European conference on computer vision, 2014, pp. 834-849.

[3] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, RGB-D mapping: using kinect-style depth cameras for dense 3D modeling of indoor environments, The International Journal of Robotics Research, vol.31, 2012, pp. 647-663.

[4] F. Endres, J. Hess, J. Sturm, D. Cremers and W. Burgard, 3D mapping with an RGB-D camera, IEEE transactions on robotics, v. 30, n. 1, 2013, pp. 177-187.

[5] M. Labbé and F. Michaud, Rtab-map as an open-source lidar and visual slam library for large-scale and long-term online operation, Journal of Field Robotics, v. 36, n. 2, 2019, pp. 416-446.

[6] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, A benchmark for the evaluation of RGB-D SLAM systems, in Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 573-580.

[7] Y. Wang, and S. Huang, 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, 2014, pp. 1841-1846.

[8] D. Kim, and J. Kim, Effective background model-based RGB-D dense visual odometry in a dynamic environment, IEEE Transactions on Robotics, v. 32, n. 6, pp. 1565-1573, 2016.

[9] A. Dib, and F. Charpillet, Robust dense visual odometry for RGB-D cameras in a dynamic environment, in Proc. of the International Conference on Advanced Robotics (ICAR), Istanbul, 2015, pp. 1-7.

[10] P. F. Alcantarilla, J. J. Yebes, J. Almazan, and L. M. Bergasa. 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.

[11] J. Cheng, Y. Sun, W. Chi, C. Wang, H. Cheng, and M. Q. H. Meng, 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), 2018, pp. 723 - 728.

[12] C. Yu, Z. Liu, X. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, Ds-slam: A semantic visual slam towards dynamic environments, in Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

[13] B. Bescós, J. Fácil, J. Civeira, and J. Neira, DynaSLAM: Tracking, mapping and inpainting in dynamic environments, IEEE Robotics and Automation Letters, vol. 3, n. 4, 2018, pp. 4076-4083.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, Mask r-cnn, in Proc. of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969.

[15] R. Girshick, Fast r-cnn, in Proc. of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448.

[16] S. Ren, K. He, R. Girshick, and J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems, 2015, pp. 91-99.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You only look once: Unified, real-time object detection, in Proc. of the IEEE conference on computer vision and pattern recognition, 2016.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, Orb: An efficient alternative to sift or surf, In proc. of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 2564 2571.

[19] D. Gálvez-López, and J. D. Tardos, Bags of binary words for fast place recognition in image sequences, IEEE Transactions on Robotics, v. 28, n. 5, 2012, pp. 1188 1197.

[20] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, g2o: A general framework for graph optimization, in IEEE Int. Conf. on Robot. and Autom. (ICRA), 2011, pp. 36073613.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, Microsoft coco: Common objects in context, in Proc. of the 13th European Conference on Computer Vision, 2014.