Received January 10, 2022, accepted January 27, 2022, date of publication February 7, 2022, date of current version February 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149885

The STDyn-SLAM: A Stereo Vision and Semantic Segmentation Approach for VSLAM in Dynamic Outdoor Environments

DANIELA ESPARZA AND GERARDO FLORES, (Member, IEEE)
Laboratorio de Percepción y Robótica [LAPyR], Centro de Investigaciones en Óptica (CIO), León, Guanajuato 37150, Mexico

Corresponding author: Gerardo Flores (gflores@cio.mx)

This work was supported in part by the Consejo Nacional de Ciencia y Tecnología (CONACYT), Fondo Institucional de Fomento Regional para el Desarrollo Científico, Tecnológico y de Innovación (FORDECYT) under Grant 292399.

ABSTRACT The Visual Simultaneous Localization and Mapping (VSLAM) is a system based on the scene's features to estimate a map and the system pose. Commonly, VSLAM algorithms are focused on a static environment; however, some dynamic objects are present in the vast majority of real-world applications. This work presents a feature-based SLAM system focused on dynamic environments using convolutional neural networks, optical flow, and depth maps to detect objects in the scene. The proposed system employs a stereo camera as the primary sensor to capture the scene. The neural network is responsible for object detection and segmentation to avoid erroneous maps and wrong system locations. Moreover, the proposed system's processing time is fast and can run in real-time, running in outdoor and indoor environments. The proposed approach has been compared with state-of-the-art; besides, we present several experimental results outdoors that corroborate the approach's effectiveness. Our code is available online.

INDEX TERMS VSLAM, dynamic environment, stereo vision, neural network.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) systems are strategic for developing the following navigation techniques. This is mainly due to its fundamental utility in solving the problem of autonomous exploration tasks in unknown environments such as mines, highways, farmlands, underwater/aerial environments, and in broad terms, indoor and outdoor scenes. The problem of SLAM for indoor environments has been investigated for years, where usually RGB-D cameras or Lidars are the primary sensors to capture scenes [1]–[3]. Indoors, dynamic objects are usually more controllable, unlike outdoors, where dynamic objects are inherent to the scene.

On the other hand, the vast majority of SLAM systems are focused on the assumption of static environments, such as HECTOR-SLAM [4], Kintinuous [5], MonoSLAM [6], PTAM [7], SVO [8], LSD-SLAM [9], among others. Since this assumption is strong, the system is restricted to work in static environments. However, in dynamic environments, the moving objects can generate an erroneous map and wrong poses because dynamic features cause a bad pose estimation and incorrect data. For this reason, new approaches have arisen for solving the dynamic environment problem, such as NeuroSLAM [10], hierarchical Outdoor SLAM [11], and Large-Scale Outdoor SLAM [12].

In this work, we propose a method called *STDyn-SLAM* for solving VSLAM's problem in dynamic outdoor environments using stereo vision [19]. Fig. 1 depicts a sketch of our proposal in real experiments. The first row shows the input images, where a potentially dynamic object is present on the scene and is detected by a semantic segmentation neural network. Fig. 1d depicts the 3D reconstruction excluding dynamic objects. To evaluate our system, we carried out experiments in different outdoor scenes, and we qualitatively compared the 3D reconstructions taking into account the excluding of dynamic objects. We conducted experiments using sequences from KITTI Dataset, and they are compared with state-of-the-art systems. Furthermore, our approach is implemented in ROS, in which we use the depth image from a stereo camera for making the 3D reconstruction using the octomap. Also, we analyzed the processing time using

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

TABLE 1. This table shows the state-of-the-art SLAM problem considering dynamic environments.

System	Sensor	Environment	Dynamic Objects	Real Time	Method
[13]	Mono	Indoor Outdoor	YOLO and MS-CNN	GPU	3-D Box Proposal Generation, standard 3-D map point reprojection error, constant motion model with uniform velocity
[14]	RGB-D/Stereo and monocular	Indoor Outdoor	MS COCO	–	Segmentation algorithm and inpainting background
[15]	RGB-D	Indoor	PASCAL VOC	–	Semantic segmentation, and optical flow
[16]	Mono	Indoor Outdoor	COCO	–	Semantic segmentation network, depth prediction network and geometry properties
[17]	RGB-D	Indoor	COCO	–	Mask R-CNN, edge refinement, and optical flow
[18]	RGB-D/Stereo and proprioceptive	Indoor Outdoor	COCO	–	Factor graph and instance-level object segmentation algorithm
Ours	RGB-D/Stereo	Indoor Outdoor	PASCAL VOC	GPU	Semantic segmentation, optical flow and epipolar geometry

different datasets. Further, we publish our code been available on GitHub.[1] Also, a video is available on YouTube. The main contributions are itemized as follows:

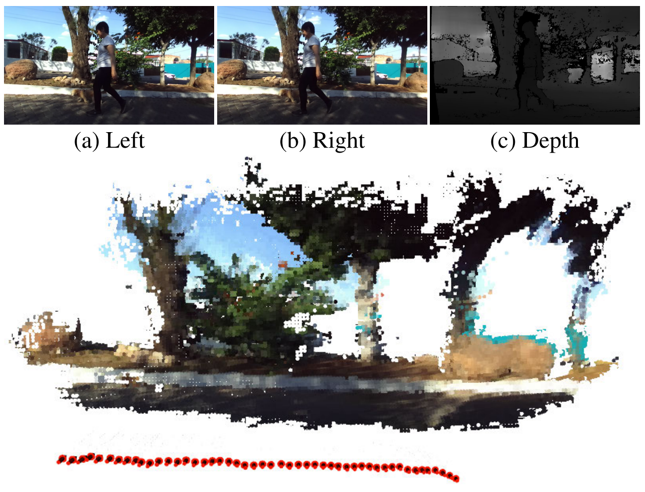
- We proposed a Stereo SLAM for dynamic environments using semantic segmentation neural network and geometrical constraints to eliminate the dynamic objects.
- We use the depth image from a stereo camera for making the 3D reconstruction using the octomap. The depth image is not necessary for the SLAM process.
- This work was tested using the KITTI and EurocMav datasets, and we compared our system with the stereo configuration systems from state-of-the-art. In addition, we obtained results from outdoor and indoor environments of our sequences.
- Some results are shown in a YouTube video, and the STDyn-SLAM is available as a GitHub repo.

The rest of the paper is structured as follows. Section II mentions the related work of SLAM in dynamic environments. Then, in Section III, we show the main results and the algorithm STDyn-SLAM algorithm. Section IV presents the real-time experiments of STDyn-SLAM in outdoor environments with moving objects; we compare our approach with state-of-art methods using the KITTI dataset. Finally, the conclusions and the future work are given in Section V.

II. RELATED WORK

A. CLASSIC APPROACHES

The classical methods do not consider artificial intelligence. Some of these approaches are based on optical flow, epipolar geometry, or a combination of the two. For example, in [20], Yang *et al.* propose a SLAM system using an RGB-D camera and two encoders for estimating the pose and building an OctoMap. The dynamic pixels are removed using an object detector and a K-means to segment the point cloud. On the other hand, in [21], Gimenez *et al.* present a CP-SLAM based on continuous probabilistic mapping and a Markov random field; they use the iterated conditional modes. Wang *et al.* [22] propose a SLAM system for indoor environments based on an RGB-D camera. They use the number of features on the static scene and assume that the parallax between consecutive images is a movement constraint. In [23], Cheng, Sun, and Meng implement an optical-flow and the five-point algorithm approach to obtain dynamic features. In [24], Ma and Jia proposed a visual SLAM for dynamic



(d) 3D reconstruction

FIGURE 1. The STDyn-SLAM results in scenes with moving objects. First raw: Input images with two dynamic objects. Second raw: 3D reconstruction performed by the STDyn-SLAM discarding moving objects.

[1]https://github.com/DanielaEsparza/STDyn-SLAM

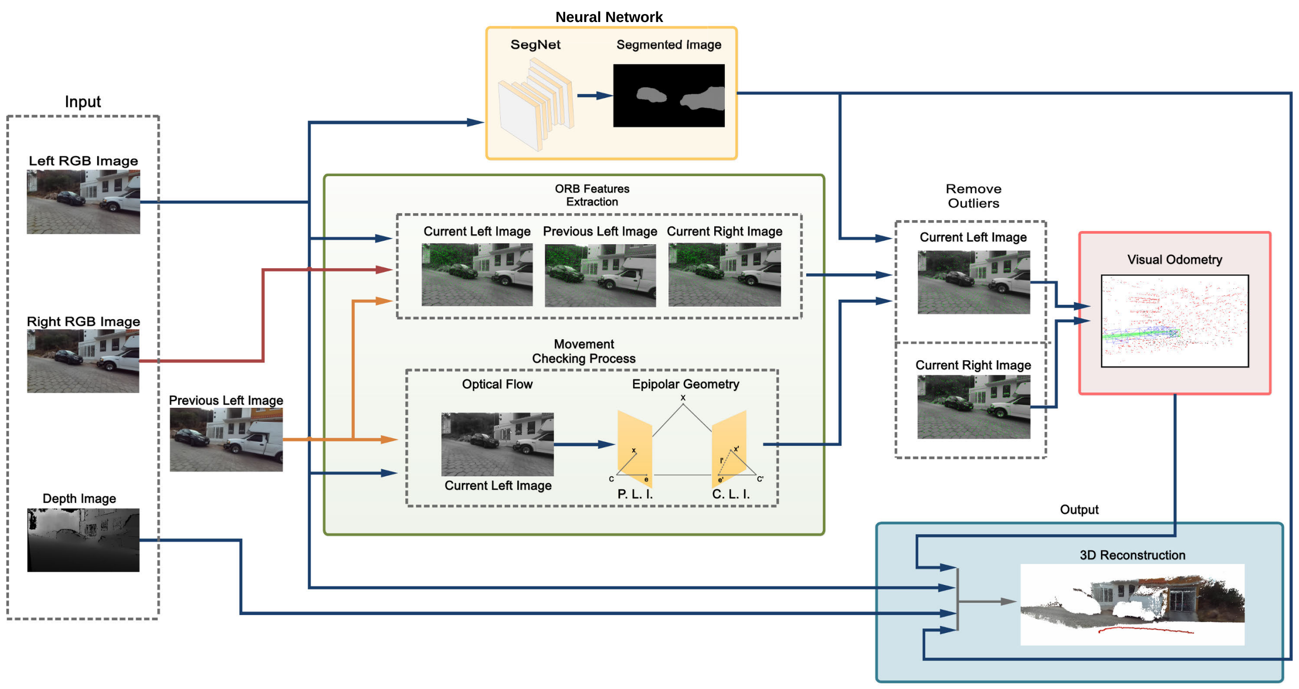


FIGURE 2. **A block diagram showing the algorithm steps of the STDyn-SLAM.**

environments, detecting the moving objects in the scene using optical flow. Furthermore, they use the RANSAC algorithm to improve the computation of the homography matrix. In [25], Sun *et al.* proposed an RGB-D system for detecting moving objects based on ego-motion to compensate for the camera movement, then obtaining the frame difference. The result of frame difference helps for detecting the moving object. After that, Sun *et al.* proposed in [26] an RGB-D system for motion removal based on a foreground model. This system does not require prior information.

B. ARTIFICIAL-INTELLIGENCE-BASED APPROACHES

Thanks to the growing use of deep learning, the researchers have proposed some SLAM systems using artificial-intelligence-based approaches. Table 1 resumes the state-of-art in this regard. Some works, such as Dosovitskiy *et al.* [27], Ilg *et al.* [28] and Mayer *et al.* [29], used optical flow and supervised learning for detecting and segmenting moving objects.

In [30], Xu *et al.* proposed an instance segmentation of the objects in the scene based on the COCO dataset [31]. The geometric and motion properties are detected and used to improve the mask boundaries. Also, they tracked the visible objects and moving objects and estimated the system's pose. Several works are based on RGB-D cameras, such as [15], [17], and [18]. Cui and Ma [15] proposed the SOF-SLAM, an RGB-D system based on ORB-SLAM2, which combines a neural network for semantic segmentation, and optical flow for removing dynamic features. Zhao *et al.* [17] proposed an RGB-D framework to dynamic scenes, where they combined the Mask R-CNN, edge refinement, and optical flow to detect the probably dynamic objects. Henein *et al.* [18] proposed a system based on an RGBD camera and proprioceptive sensors for tackling the SLAM problem. They employ a model of factor graph and an instance-level object segmentation algorithm to the classification of objects and the tracking of features. The proprioceptive sensors are used to estimate the camera pose. Also, some works use a monocular camera, for instance, the DSOD-SLAM presented in [16]. Ma *et al.* employ a semantic segmentation network, a depth prediction network, and geometry properties to improve the results in dynamic environments. Our work is built on the well-known ORB-SLAM2 [32], taking some ideas from DS-SLAM system [33]. In the DS-SLAM, the authors used stored images from an RGB-D camera for solving the SLAM problem in indoor dynamic environments. Nevertheless, the depth map obtained from an RGB-D camera is hard for external environments. In [34], Cheng *et al.* proposed a SLAM system for building a semantic map in dynamic environments using CRF-RNN for segmenting objects. Bescos *et al.* in [14] proposed a system for object detecting using the Mask R-CNN, and their method proposed for inpainting the background using the information from previous images. An update of [14] is [35], where Bescos *et al.* proposed a visual SLAM based on the trajectories of the objects and a bundle adjustment.

III. METHODS

In this section, we present and describe the framework of the STDyn-SLAM with all the parts that compose it. A block diagram describing the framework's pipeline is depicted in Fig. 2, where the inputs at the instant time t are the stereo pair, depth image, and the left image captured at $t-1$ (aka previous left image). The process starts with extracting ORB features in the stereo pair and the past left image. Then, it follows the optical flow and epipolar geometry image processing. Next, the neural network segments potentially moving objects parallelly in the current left image. To remove outliers (features inside dynamic objects) and estimate the visual odometry, it is necessary to computation the semantic information and the movement checking process. Finally, the 3D reconstruction is computed from the segmented image, visual odometry, the current left frame, and the depth image. These processes are explained in detail in the following subsections.

A. STEREO PROCESS

Motivated by the vast applications of robotics outdoors, where dynamic objects are presented, we proposed that our STDyn-SLAM system be focused on stereo vision. A considerable advantage of this is that the depth estimation from a stereo camera is directly given as a distance measure. The process described in this part is depicted in Fig. 2, where three main tasks are developed: feature extraction, optical flow, and epipolar geometry. Let's begin with the former.

The first step of the stereo process is acquiring the left, right, and depth frames from a stereo camera. Then, a local feature detector is applied in the stereo pair and the previous left image. As a feature detector, we use the Oriented fast and Rotated Brief (ORB) feature detector, which throws the well-known ORB features [36]. Once the ORB features are found, optical flow and a process using epipolar geometry are conducted.

To avoid dynamic objects not classified by the neural network (explained in the following subsection), the STDyn-SLAM computes optical flow using the previous and current left frames. This step employs a Harris detector to compute the optical flow. Remember, these features are different from the ORB ones. The Harris points pair is discarded if at least one of the points is on the edge corner or close to it.

From the fundamental matrix, ORB features, and optical flow, we compute the epipolar lines. Thus, we can map the matched features from the current left frame into the previous left frame. The distance from the corresponding epipolar line to the mapped feature into the past left image determines an inlier or outlier. Please refer to the remove outliers section in Fig. 2. Notice that the orb features of the car in the left image were removed, but the points on the right frame remain unchanged. This is because removing the points in the right images adds computational cost and is unnecessary.

B. ARTIFICIAL NEURAL NETWORK's ARCHITECTURE

The approach we use is eliminating the ORB features on dynamic objects. To address this, we need to discern the natural dynamic objects among all the objects in the scene. It is here where the NN depicted in Fig. 2 is introduced. In the NN block of that figure, a semantic segmentation neural network is shown, with the left image as input and a segmented image with the object of interest as output. This NN is a pixel-wise classification and segmentation framework. The STDyn-SLAM implements a particular NN of this kind called SegNet [37], which is an encoder-decoder network based on the VGG-16 model [38]. The encoder of this NN architecture counts with thirteen convolutional layers with batch normalization, a ReLU non-linearity divided into five encoders, and five non-overlapping max-pooling and sub-sampling layers located at the end of each encoder. Since each encoder is connected to a corresponding decoder, the decoder architecture has the same number of layers as encoder architecture, and every decoder has an upsampling layer at first. The last layer is a softmax classifier. SegNet classifies the pixel-wise using a model based on the PASCAL VOC dataset [39], which consists of twenty classes. The pixel-wise can be classified into one of the following classes: airplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train and TV/monitor.

Notwithstanding those above, not all feature points in the left frame are matched in the right frame. For that reason and to save computing resources, the SegNet classifies the objects of interest only on the left input image.

1) OUTLIERS REMOVAL

Once all the previous steps have been accomplished, a threshold is selected to determine the features as inlier or outlier. Fig. 3 depicts the three cases of a mapped feature. Let x_1, x_2, and x_3 denote the ORB features from the previous left image; x_1', x_2', and x_3' are the corresponding features from the current left image; X and X' represent the homogeneous coordinates of x and x', respectively; F is the fundamental matrix; and $l_1' = FX_1$, $l_2' = FX_2$, and $l_3' = FX_3$ are the epipolar lines. The first and second cases correspond to inliers, x_1' is over l_1', and the distance from x_2' to l_2' is less than the threshold. The third case is an outlier because the distance from x_3' to l_3' is greater than the threshold. To compute the distance between the point x' and the epipolar line, l', we proceed as follows,

$$d(X', l') = \frac{X'^T FX}{\sqrt{(FX)_1^2 + (FX)_2^2}} \tag{1}$$

where the subindex from $(FX)_1$ and $(FX)_2$ denotes the element of the epipolar line. If the distance is larger than the threshold, the feature point is considered an outlier, i.e., a dynamic feature.

Remember that the SegNet, described before, semantically segments the left image in object classes. The semantic segmentation enhances the rejection of ORB features on the possible dynamic objects. The ORB features inside

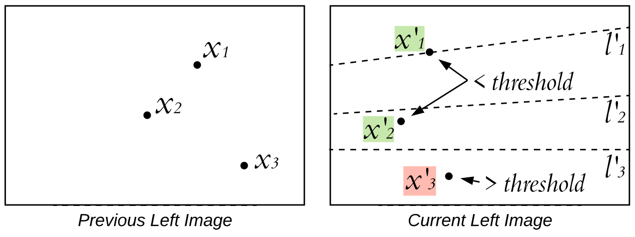


FIGURE 3. The cases of inliers and outliers. Green: the x'_1 and x'_2 are inliers; the distance from the point to their corresponding epipolar line l' is less than a threshold. Red: x'_3 is an outlier, since the distance is greater than the threshold.

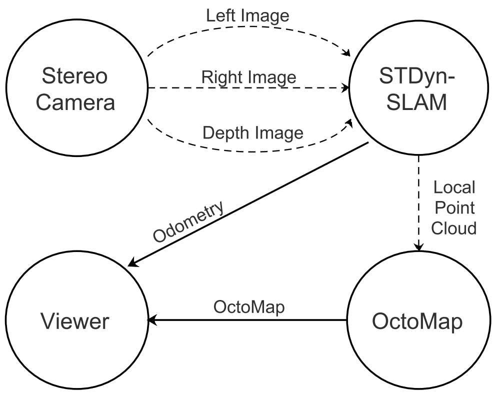


FIGURE 4. Diagram of the ROS nodes of the STDyn-SLAM required to generate the trajectory and 3D reconstruction. The circles represent each process's ROS node, and the arrows are the ROS topics published by the ROS nodes. The continued arrows depict the final ROS topics.

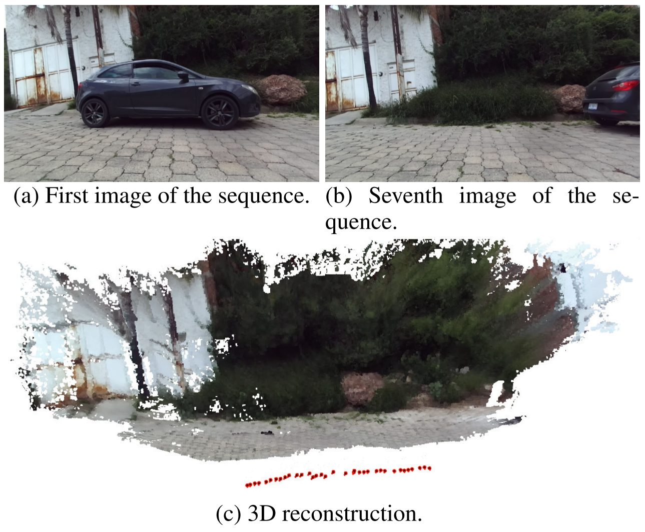
segmented objects, and thus possible moving objects, are rejected. The remained points are matched with the ORB features from the right image.

C. VISUAL ODOMETRY

Because the system is based on ORB-SLAM2, the VSLAM visually computes the odometry. Therefore, the next step needs the ORB features to estimate the depth for each feature pair. The features are classified in mono and stereo and will be necessary to track the camera's pose. Again, this step is merely a process from ORB-SLAM2.

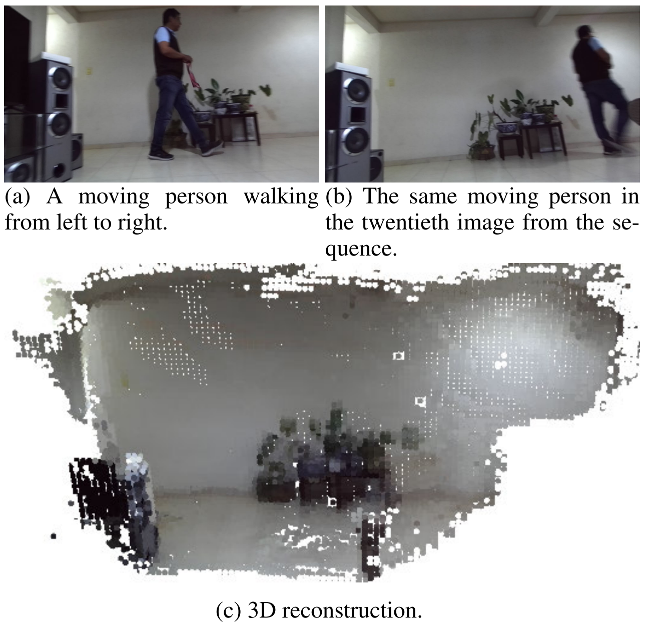
D. 3D RECONSTRUCTION

Finally, the STDyn-SLAM builds a 3D reconstruction from left, segmented, and depth images using visual odometry. First, the 3D reconstruction process checks each pixel of the segmented image to reject the point corresponding to the classes of the objects selected as dynamic in section III-B. Then, if the pixel is not considered a dynamic object, the equivalent pixel from the depth image is added to the point cloud, and the assigned color of the point is obtained from the left frame. This section builds a local point cloud only in the current pose of the system, and then the octomap [40] joins and updates the local point clouds in a full point cloud.



(a) First image of the sequence.
(b) Seventh image of the sequence.
(c) 3D reconstruction.

FIGURE 5. The STDyn-SLAM when a static object becomes dynamic. Images a) and b) corresponds to the left images from a sequence. Image c) is the 3D reconstruction of the environment; in red dots is the trajectory. The OctoMap node fills empty areas along the sequence of images.



(a) A moving person walking from left to right.
(b) The same moving person in the twentieth image from the sequence.
(c) 3D reconstruction.

FIGURE 6. The 3D reconstruction from STDyn-SLAM in an indoor environment. In the scene appears a moving person, which is crossing from left to right. The VSLAM system considers the person as a dynamic object.

Remark 1: It is essential to mention that we merely applied the semantic segmentation, optical flow, and geometry constraints to the left image to avoid increasing the time executing. Moreover, the right-hand-side frame segmentation is unnecessary because feature selection rejects the ORB features inside dynamic objects from the left image, so the corresponding points from the right frame will not be matched.

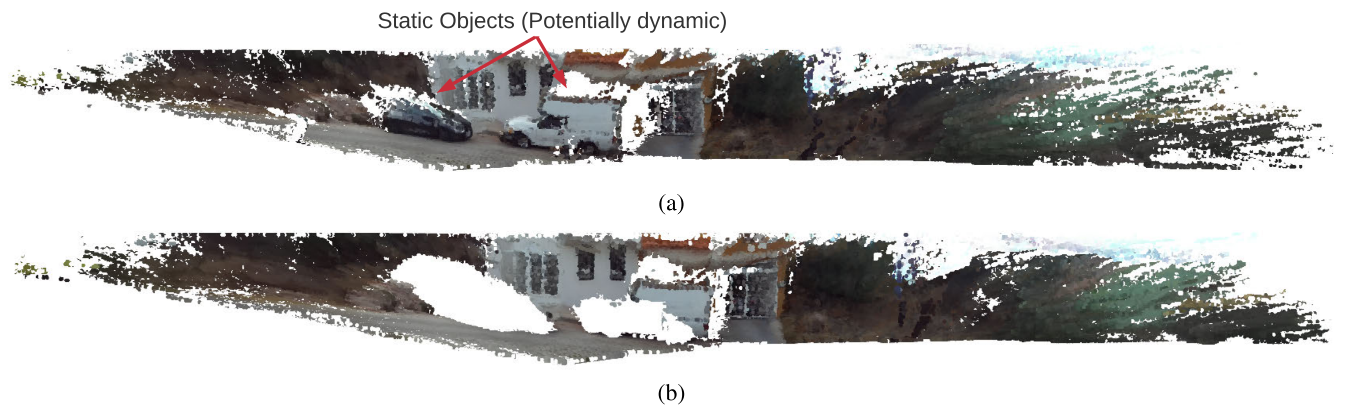


FIGURE 7. The 3D reconstruction, with the presence of static (two parked cars) and dynamic objects (a person and two dogs). Notice that the person and dogs are not visualized in the scene for the effect of the STDyn-SLAM. Fig. a) depicts the static objects. Nevertheless, the vehicles are potentially dynamic objects, thus in Fig. b), the STDyn-SLAM excludes the bodies considering its possible movement.

IV. EXPERIMENTS

This section tests our algorithm STDyn-SLAM in real-time scenes under the KITTI datasets. Our system's experiments were compared to other state-of-art systems to evaluate the 3D reconstruction and the odometry. The results of the 3D map were qualitatively measured because of the nature of the experiment. We employ the Absolute Pose Error (APE) metric for the odometry.

A. HARDWARE AND SOFTWARE SETUP

We tested our system on an Intel Core i7-7820HK laptop computer with 32 Gb RAM and a GPU GeForceGTX 1070. Moreover, we used as input a ZED camera, which is a stereo camera developed by Sterolabs. We selected an HD720 resolution. The ZED camera resolutions are WVGA (672×376), HD720 (1280×720), HD1080 (1920×1080), and 2.2K (2208×1242).

The STDyn-SLAM is developed naturally on ROS. Our system's main inputs are the left and right images, but the depth map is necessary to build the point cloud. However, if this is not available, it is possible to execute the STDyn-SLAM only with the stereo images and then obtain the trajectory. On the other hand, the *STDyn* node in ROS generates two main topics; the *Odom* and the *ORB_SLAM2_PointMap_SegNetM/Point_Clouds* topics. The point cloud topic is the input of the *octomap_server* node; this node publishes the joined point cloud of the scene.

Fig. 4 depicts the required ROS nodes by the STDyn-SLAM to generate the trajectory and the 3D reconstruction. The camera node publishes the stereo images and computes the depth map from the left and right frames. Then, the STDyn-SLAM calculates the odometry and the local point cloud. The OctoMap combines and updates the current local point cloud with the previous global map to visualize the global point cloud. It is worth mentioning that the user can choose the maximum depth of the local point cloud. All the ROS topics can be shown through the viewer.

B. REAL-TIME EXPERIMENTS

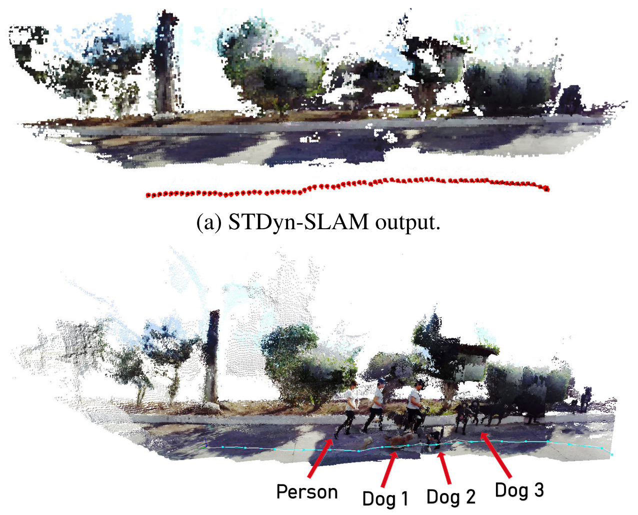
We present real-time experiments under three different scenarios explained next.

First, we test the STDyn-SLAM in an outdoor environment where a car is parked and then moves forward. In this case, a static object (a car) becomes dynamic, see Fig. 5. This figure shows the 3D reconstruction, where the car appears static in the first images from the sequence, Fig. 5 a). Then, the car becomes a dynamic object when it moves forward (Fig. 5 b), so the STDyn-SLAM is capable of filling the empty zone if the scene is covered again, as is the case in Fig. 5 c).

The second experiment tests our system in an indoor environment. The scene consists of a moving person crossing from left to right. Subfigures *a* and *b* depicts the left and right images from Fig. 6. And *c* shows the 3D reconstruction. The area occupied by the moving person is filled after the zone is visible.

The third experiment consists of a scene sequence with two parked cars, a walking person, and a dog. Even though the vehicles are static, the rest of the objects move. Fig. 7a shows the scene taking into account the potentially dynamic entities. However, a car can change its position; the STDyn-SLAM excludes the probable moving bodies (parked cars) to avoid multiple plotting throughout the reconstruction. This is depicted in Fig. 7b.

We compared the point clouds from the RTABMAP and the STDyn-SLAM systems as a fourth experiment. The sequence was carried out outdoors with a walking person and two dogs. Since RTABMAP generates a point cloud of the scene, we decided to compare it with our system. To build the 3D reconstructions from RTABMAP, we provided left and depth images, camera info, and odometry as inputs for the RTABMAP. We used stereo and depth images; the intrinsic parameters are saved in a text file in the ORB-SLAM2 package. Fig 8 shows the 3D reconstructions. In Fig. 8a our system excludes the dynamic objects. On the other hand, Fig 8b RTABMAP plotted the dynamic objects on different sides of the scene, resulting in an incorrect map of the environment.



(a) STDyn-SLAM output.

(b) RTAB-Map output.

FIGURE 8. Experiment comparison between the STDyn-SLAM and the RTABMAP [41]. Image a) shows the 3D reconstruction given by STDyn-SLAM; it eliminates dynamic objects' effect on the mapping. Image b) shows the point cloud created by RTABMAP; notice how dynamic objects are mapped along the trajectory. This is undesirable behavior.

TABLE 2. Comparison of Absolute Pose Error (APE) on KITTI dataset.

Sequence	RMSE			Mean		
	Dyna1	ORB2	Ours	Dyna1	ORB2	Ours
00	0.905	**0.786**	0.819	0.836	**0.717**	0.739
01	5.230	3.810	**2.168**	4.794	3.469	**2.040**
02	5.728	**2.974**	3.096	4.687	**2.692**	2.835
03	0.297	**0.256**	0.268	0.283	**0.237**	0.241
04	**0.137**	0.177	0.169	**0.119**	0.166	0.154
05	**0.370**	0.558	0.836	**0.323**	0.525	0.766
06	**0.547**	0.906	1.735	**0.511**	0.810	1.596
07	0.594	**0.406**	0.448	0.540	**0.406**	0.419
08	3.227	3.318	**3.224**	2.793	2.796	**2.781**
09	2.572	3.432	**1.953**	2.062	2.884	**1.826**
10	1.062	**1.020**	1.227	0.931	**0.901**	1.132

C. COMPARISON OF STATE-OF-ART AND OUR SLAM USING KITTI AND EurocMav DATASETS

We compare our VSLAM with DynaSLAM1 [14] and ORB-SLAM2 approaches. We selected sequences with dynamic objects, loop, and no-loop closure to evaluate the SLAM systems. Therefore, we chose the 00−10 sequences from the odometry KITTI datasets [42], furthermore all sequences from the EurocMav dataset excepting the V1_03 and V2_03. Moreover, we employed *EVO* [43] tools to evaluate the Absolute Pose Error (APE) and the Relative Pose Error (RPE), and RGB-D tools [44] to calculate the Absolute Trajectory Error (ATE).

We present the results of APE, RPE, and ATE in different tables. We divided the tables depending on the dataset evaluated. Tables 3 and 4 show the APE experiments on KITTI and EurocMav datasets, respectively. Tables 4 and 5 correspond to RPE, and tables 6 and 7 present the ATE results. We did not evaluate the EurocMav with the DynaSLAM1 due to excessive processing time to compute the trajectories.

TABLE 3. Comparison of Absolute Pose Error (APE) on Euroc-Mav dataset.

Sequence	RMSE		Mean	
	ORB2	Ours	ORB2	Ours
MH_01	**0.364**	0.370	0.275	**0.024**
MH_02	0.067	**0.037**	0.043	**0.024**
MH_03	0.105	**0.038**	0.091	**0.060**
V1_01	0.145	**0.086**	0.132	**0.022**
V1_02	0.331	**0.060**	0.259	**0.022**
V2_01	0.090	**0.065**	0.077	**0.022**
V2_02	0.132	**0.054**	0.111	**0.022**

TABLE 4. Comparison of Relative Pose Error (RPE) on KITTI dataset.

Sequence	RMSE			Mean		
	Dyna1	ORB2	Ours	Dyna1	ORB2	Ours
00	0.027	0.027	**0.026**	0.018	0.018	**0.017**
01	0.049	0.048	**0.035**	0.043	0.044	**0.032**
02	0.028	0.027	**0.025**	0.022	0.022	**0.020**
03	0.016	0.017	**0.015**	0.014	0.015	**0.013**
04	**0.018**	0.019	0.020	**0.016**	**0.016**	0.018
05	**0.016**	0.018	0.017	**0.012**	0.013	0.014
06	**0.017**	0.023	0.022	**0.013**	0.015	0.016
07	0.017	**0.016**	0.018	0.014	**0.013**	0.014
08	0.039	0.039	**0.038**	0.242	0.025	**0.024**
09	0.021	**0.019**	0.023	0.018	**0.017**	0.020
10	0.020	0.020	**0.018**	0.015	0.014	**0.013**

TABLE 5. Comparison of Relative Pose Error (RPE) on Euroc-Mav dataset.

Sequence	RMSE		Mean	
	ORB2	Ours	ORB2	Ours
MH_01	0.057	**0.027**	0.038	**0.024**
MH_02	0.052	**0.027**	0.029	**0.024**
MH_03	0.085	**0.071**	0.062	**0.060**
V1_01	0.035	**0.025**	0.027	**0.022**
V1_02	0.255	**0.051**	0.082	**0.022**
V2_01	0.022	**0.019**	**0.018**	0.022
V2_02	0.074	**0.042**	0.047	**0.022**

To evaluate the significative difference of the ATE evaluation, we implemented the *Score* S_ρ [45] over the sequences of EurocMav and KITTI datasets of tables 6 and 7. The results in table 8 show an improvement of our system against ORBSLAM2 in the trajectories of the EurocMav dataset. In the KITTI dataset, STDyn-SLAM and ORBSLAM2 are not significative different. In evaluating our system and DynaSLAM1, the Dyna is slightly better.

D. PROCESSING TIME

In this section, we analyzed the processing time of this work. For the study, we evaluate some datasets with different types of images. The analysis consists of obtaining the processing time of each sequence with the same characteristics and calculating the average of the sequence's mean. Table 9 shows the times getting with the datasets. We use the KITTI and EurocMav datasets for the RGB and Gray columns. Since the sequences do not provide a depth image, we did not map a 3D reconstruction. For the last column, we utilized our trajectories. In addition, our dataset contains depth images,

TABLE 6. Comparison of Absolute Trajectory Error (ATE) on KITTI dataset.

Sequence	RMSE (ATE)		
	Dyna1	ORB2	Ours
00	**165.229**	170.069	169.672
01	546.034	563.836	**536.394**
02	176.395	**171.177**	172.683
03	144.891	156.837	**143.910**
04	**0.225**	0.255	0.665
05	133.867	**128.425**	141.054
06	**1.713**	2.467	2.743
07	37.734	**35.975**	40.435
08	281.275	280.827	**229.148**
09	115.893	**111.776**	123.498
10	191.473	**169.045**	185.667

TABLE 7. Comparison of Absolute Trajectory Error (ATE) on Euroc-Mav dataset.

Sequence	RMSE (ATE)	
	ORB2	Ours
MH_01	**2.336**	2.344
MH_02	**2.112**	2.188
MH_03	2.921	**2.823**
V1_01	1.831	**1.811**
V1_02	**1.717**	1.729
V2_01	2.294	**2.193**
V2_02	2.092	**2.063**

TABLE 8. Comparison of Score $S_\rho(a, b)$ on the datasets.

Sequence	ρ	a	b	$S_\rho(a,b)$
Euroc-Mav	0.01	ORBSLAM2	STDyn	-0.3571
	0.05	ORBSLAM2	STDyn	-0.2857
	0.1	ORBSLAM2	STDyn	-0.2142
	0.25	ORBSLAM2	STDyn	-0.0714
KITTI	0.01	ORBSLAM2	STDyn	0.1818
	0.05	ORBSLAM2	STDyn	-0.0909
	0.1	ORBSLAM2	STDyn	0
	0.25	ORBSLAM2	STDyn	-0.0909
KITTI	0.01	DynaSLAM1	STDyn	0.1363
	0.05	DynaSLAM1	STDyn	0.0454
	0.1	DynaSLAM1	STDyn	0.0454
	0.25	DynaSLAM1	STDyn	0

TABLE 9. Processing time.

STDyn-SLAM	RGB images (1241 × 376)	Gray images (752 × 480)	ROS (1280 × 720)
Time (sec)	0.153662	0.0993046333	0.2683495

so we plotted a 3D reconstruction. For this reason, the processing time is longer.

V. CONCLUSION

This work presents the STDyn-SLAM system for outdoor and indoor environments where dynamic objects are present. The STDyn-SLAM is based on images captured by a stereo pair for 3D reconstruction of scenes, where the possible dynamic objects are discarded from the map; this allows a trustworthy point cloud. The system capability for computing a reconstruction and localization in real-time depends on the computer's processing power, since a GPU is necessary to support the processing. However, with a medium-range computer, the algorithms work correctly.

In the future, we plan to implement an optical flow approach based on the last generation of neural networks to improve dynamic object detection. The implementation of neural networks allows replacing classic methods such as geometric constraints. Furthermore, we plan to increase the size of the 3D map to reconstruct larger areas and obtain longer reconstructions of the scenes. The next step is implementing the algorithm in an aerial manipulator constructed in the lab.

SUPPLEMENTARY MATERIAL

The implementation of our system is released on GitHub and is available under the following link: https://github.com/DanielaEsparza/STDyn-SLAM

Besides, this letter has supplementary video material available at https://youtu.be/3tnkwvRnUss, provided by the authors.

REFERENCES

[1] J. Castellanos, J. Montiel, J. Neira, and J. Tardos, "The SPmap: A probabilistic framework for simultaneous localization and map building," *IEEE Trans. Robot. Autom.*, vol. 15, no. 5, pp. 948–952, 1999.

[2] G. Dissanayake, H. Durrant-Whyte, and T. Bailey, "A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem," in *Proc. IEEE Int. Conf. Robot. Automation. Symposia (ICRA)*, 2000, pp. 1009–1014.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI Nat. Conf. Artif. Intell.*, 2002, pp. 593–598.

[4] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, Nov. 2011, pp. 155–160.

[5] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially extended KinectFusion," in *Proc. RSS Workshop RGB-D, Adv. Reasoning with Depth Cameras*, Jul. 2012, pp. 1–10.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[7] Y. Kameda, "Parallel tracking and mapping for small AR workspaces (PTAM) augmented reality," *J. Inst. Image Inf. Telev. Engineers*, vol. 66, no. 1, pp. 45–51, 2012.

[8] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[9] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 834–849.

[10] F. Yu, J. Shang, Y. Hu, and M. Milford, "NeuroSLAM: A brain-inspired SLAM system for 3D environments," *Biol. Cybern.*, vol. 113, nos. 5–6, pp. 515–545, Dec. 2019.

[11] D. Schleicher, L. M. Bergasa, M. Ocana, R. Barea, and M. E. Lopez, "Real-time hierarchical outdoor SLAM based on stereovision and GPS fusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 440–452, Sep. 2009.

[12] R. Ren, H. Fu, and M. Wu, "Large-scale outdoor SLAM based on 2D LiDAR," *Electronics*, vol. 8, no. 6, p. 613, May 2019.

[13] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[14] B. Bescos, J. M. Fácil, J. Civera, and J. L. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[15] L. Cui and C. Ma, "SOF-SLAM: A semantic visual SLAM for dynamic environments," *IEEE Access*, vol. 7, pp. 166528–166539, 2019.

[16] P. Ma, Y. Bai, J. Zhu, C. Wang, and C. Peng, "DSOD: DSO in dynamic environments," *IEEE Access*, vol. 7, pp. 178300–178309, 2019.

[17] L. Zhao, Z. Liu, J. Chen, W. Cai, W. Wang, and L. Zeng, "A compatible framework for RGB-D SLAM in dynamic scenes," *IEEE Access*, vol. 7, pp. 75604–75614, 2019.

[18] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The need for speed," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 2123–2129.

[19] S. Trejo, K. Martinez, and G. Flores, "Depth map estimation methodology for detecting free-obstacle navigation areas," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2019, pp. 916–922.

[20] D. Yang, S. Bi, W. Wang, C. Yuan, W. Wang, X. Qi, and Y. Cai, "DRE-SLAM: Dynamic RGB-D encoder SLAM for a differential-drive robot," *Remote Sens.*, vol. 11, no. 4, p. 380, Feb. 2019.

[21] J. Gimenez, A. Amicarelli, J. M. Toibero, F. di Sciascio, and R. Carelli, "Continuous probabilistic SLAM solved via iterated conditional modes," *Int. J. Autom. Comput.*, vol. 16, no. 6, pp. 838–850, Aug. 2019.

[22] R. Wang, W. Wan, Y. Wang, and K. Di, "A new RGB-D SLAM method with moving object detection for dynamic indoor scenes," *Remote Sens.*, vol. 11, no. 10, p. 1143, May 2019.

[23] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Improving monocular visual SLAM in dynamic environments: An optical-flow-based approach," *Adv. Robot.*, vol. 33, no. 12, pp. 576–589, Jun. 2019.

[24] Y. Ma and Y. Jia, "Robust SLAM algorithm in dynamic environment using optical flow," in *Proc. Chin. Intell. Syst. Conf.*, Y. Jia, J. Du, and W. Zhang, Eds. Singapore: Springer 2020, pp. 681–689.

[25] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auton. Syst.*, vol. 89, pp. 110–122, Mar. 2017.

[26] Y. Sun, M. Liu, and M. Q.-H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robot. Auton. Syst.*, vol. 108, pp. 115–128, Oct. 2018.

[27] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.

[28] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2462–2470.

[29] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *2016 IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4040–4048.

[30] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-fusion: Octree-based object-level multi-instance dynamic SLAM," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 5231–5237.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[32] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[33] 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. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1168–1174.

[34] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Robust semantic mapping in challenging environments," *Robotica*, vol. 38, no. 2, pp. 256–270, Feb. 2020.

[35] B. Bescos, C. Campos, J. D. Tardos, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.

[36] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[37] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, Jul. 2015, pp. 1–14.

[39] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and W. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2010.

[40] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013. [Online]. Available: https://octomap.github.io

[41] M. Labbé and F. Michaud, "Long-term online multi-session graph-based SPLAM with memory management," *Auto. Robots*, vol. 42, no. 6, pp. 1133–1150, 2018.

[42] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. Int. Conf. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[43] (2017). U. Technologies. *EVO: Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[45] R. Muñoz-Salinas and R. Medina-Carnicer, "UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," *Pattern Recognit.*, vol. 101, May 2020, Art. no. 107193.

DANIELA ESPARZA received the B.S. degree in robotic engineering from the Universidad Politécnica del Bicentenario, México, in 2017, and the master's degree in optomechatronics from the Center for Research in Optics, in 2019, where she is currently pursuing the Ph.D. degree in mechatronics and mechanical design.

Her research interests include artificial vision, such as 3D reconstruction and deep learning applied to SLAM developed over platforms as mobile robots.

GERARDO FLORES (Member, IEEE) received the B.S. degree (Hons.) in electronic engineering from the Instituto Tecnológico de Saltillo, Mexico, in 2007, the M.S. degree in automatic control from CINVESTAV-IPN, Mexico City, in 2010, and the Ph.D. degree in systems and information technology from the Heudiasyc Laboratory, Université de Technologie de Compiègne–Sorbonne Universités, France, in October 2014.

Since August 2016, he has been a full-time Researcher and the Head of the Perception and Robotics Laboratory, Center for Research in Optics, León, Guanajuato, Mexico. His current research interests include the theoretical and practical problems arising from the development of autonomous robotic and vision systems. He has been an Associate Editor of *Mathematical Problems in Engineering*, since 2020.