CENTRO DE INVESTIGACIONES
EN ÓPTICA, A.C.

STEREO VISION BASED SLAM IN DYNAMIC OUTDOOR ENVIRONMENTS USING DEEP LEARNING

MASTER IN OPTOMECHATRONICS

Advisor:

PhD. Gerardo Ramón Flores Colunga

Student:

Karla Daniela Martínez Esparza

October 2019
León, Guanajuato, México

Contents

Acknowledgements **8**

Publications **9**

Abstract **10**

Introduction **11**

1 State of the Art **13**

1.1 The SLAM problem 16

1.2 SLAM in dynamic environments 16

1.2.1 Robust Visual SLAM 17

1.2.2 Dynamic Object Segmentation and 3D Tracking 18

2 Theoretical Knowledge **20**

2.1 ORB-SLAM 20

2.1.1 Tracking 21

2.1.2 Local Mapping 23

2.1.3 Loop closing 25

2.2 SegNet . . . 26
2.2.1 Architecture . . . 26
2.3 DS-SLAM . . . 28
2.3.1 Outliers Removal . . . 30
2.3.2 Mapping: Octo-Tree Map . . . 32
2.4 ROS: Robot Operating System . . . 33

3 Methodology **35**
3.1 Camera configurations . . . 35
3.1.1 RGB-D camera . . . 36
3.1.2 Monocular camera . . . 37
3.1.3 Stereo camera . . . 37
3.2 SLAM Parameters . . . 40
3.3 Real-time configuration . . . 42
3.4 Stereo configuration . . . 44

4 Experiments **46**
4.1 ROS . . . 46
4.2 Depth Comparison . . . 47
4.2.1 Outdoor Environment . . . 50
4.2.2 Indoor Environment . . . 50
4.3 3D reconstruction . . . 53
4.3.1 Outdoor Environment . . . 53
4.3.2 Indoor Environment . . . 55
4.4 Process Rate . . . 55

5 Conclusions **62**

Bibliography **63**

List of Figures

1.1 Types of sensors. 14

1.2 Maps, a) shows a 2D map generated by HECTOR-SLAM [1], b) were built by LSD-SLAM [2], c) ORB-SLAM [3] generates a sparse map and d) corresponds to dense map created by KinectFusion [4]. 15

2.1 ORB-SLAM system consists of three threads, the first one is the tracking (the upper gray block), where the images are processing, the ORB features are extracted too, the next thread is local mapping, the keyframes are using to obtained the map and the last is loop closing, where if the keyframe is detected previously [5]. 21

2.2 Graphs, a) a covisibility graph shows all links from a keyframe with other keyframes, b) a essential graph depicts the strongest links from a keyframe with others. [3] 24

2.3 SegNet is an encoder-decoder architecture, the encoder consists of thirteen convolutional layers, batch normalization, ReLU as activation function and pooling, the decoder contains thirteen convolutional layers, upsampling and softmax functions [6]. 26

2.4 Up-sampling: 2×2 window corresponds to input feature map, a, b, c and d are feature values, 4×4 window is an up-sampled feature map using the corresponding max-pooling indices to locate each value from input feature map [6]. 27
2.5 The system receives two images sequences (RGB and depth images), DS-SLAM uses the RGB image to detect ORB features and semantic segmentation frame, then the outliers are removed and then the system computes the visual odometry. Semantic octo-tree map uses as input the depth image, semantic segmentation data and visual odometry. [7]. 29
2.6 Epipolar geometry representation. (a) represents the ideal mapping of a 3-space point (X) on both keyframes, the ray is traced from the center of the camera C to X passing through x and the same for the other frame. (b) The x point has an inexact position and hence the position in the second frame is considered on the epipolar line instead of particular point [8]. 31
2.7 ROS: A set of nodes, topics, tools, several applications, compatibility of programming languages [9]. 34

3.1 Kinect sensor, an example of a RGB-D camera [10]. 36
3.2 IR pattern used to calculate the depth of the real world. [10] 37
3.3 Monocular camera 38
3.4 Depth is obtained from two images, (a) the left image corresponds to the left camera, so, the right image corresponds to the right camera, (b) depth is obtained from disparity and disparity is calculated from the difference in pixels between left and right images. 39
3.5 Zed camera. 40
3.6 Diagram of main code. 43

4.1 ROS graphs, a) and b) depict the connected nodes and topics for stereo and RGB-D cameras respectively. 47

4.2 Depth images, a) and b) were captured with a kinect camera, the size of the image is VGA (640×480) and c) is the corresponding scene to depth images. 48
4.3 a), b) and c) images were captured from a ZED camera, compared to Kinect, the stereo camera is able to compute the depth map in open-air environments. 49
4.4 Images were captured inside a building, the first column corresponds to a similar perspective, in the second column the pictures capture a corner inside the room, a) and b) images were obtained from a RGB-D camera, c) and d) from a stereo camera. 51
4.5 Images captured from ZED camera, the first row corresponds to HD720 resolution, in the second one, the image quality is HD1080. 52
4.6 3D reconstructions in outdoor environment with dynamic objects. 54
4.7 The top image is a 3D reconstruction built using a stereo camera, the bottom image shows a person representing the dynamic object. 57
4.8 3D reconstructions. The top image is built using a stereo camera, the middle image is generated using a Kinect sensor, the bottom image shows the dynamic objects which were in the scene whereas it was built. 58
4.9 Indoor environment with high illumination from sun. 59
4.10 Hallway reconstruction. 60
4.11 3D reconstruction, indoor environment. a) and b) are 3D reconstructions using stereo and RGB-D camera respectively. c) is the scene where the maps were generate. 61

List of Tables

3.1 Camera parameters. 41
3.2 Viewer (RVIZ) parameters. 41
3.3 ORB parameters. 42

4.1 Average rate. 56

Acknowledgements

I am very thankful to my parents and brother, who have always been supported me. Also, I would like to thank Carlos, for supporting and being always for me, for his love and giving me encouragement. Another person, who helped me during the master's degree is Sergio, I am very thankful to him. Moreover, I would like to show my gratitude to my friends of the mastery generation and to "los brothers" of Perception and Robotics Lab for helping me and for sharing many experiences.

I am very thankful to my advisor Gerardo Flores, for guiding me along my thesis work. Also, I would like to thank my thesis reviewers Dr. Abundio Dvila and Dr. Francisco Cuevas, researchers and engineers from Centro de Investigaciones en ptica A.C. for sharing their knowledge.

Finally, I would like to thank CIO and CONACyT, since this thesis was supported through project 292399 "Generation of scientific-technological strategies with a multidisciplinary and interinstitutional approach to face the threat posed by ambrosial complexes in the agricultural and forestry sectors of Mexico" by Fondo Institucional de Fomento Regional para el Desarrollo Cientfico, Tecnolgico y de Innovacin (FORDECyT) of the Consejo Nacional de Ciencia y Tecnologa (CONACyT).

Publications

Conference Papers

- S. Trejo, **K. Martnez** and G. Flores. "Depth map estimation methodology for detecting free-obstacle navigation areas" International Conference on Unmanned Aircraft Systems 2019 (PUBLISHED).

Abstract

This thesis presents a Simultaneous Localization and Mapping (SLAM) system focused on dynamic environments using convolutional neural networks. The proposed system employs a stereo camera as the input of the SLAM for the acquisition of left and right images and depth map. The neural network is used for object detection and segmentation to avoid erroneous maps and wrong system location. The main job of the neural network is to find out objects within the scene, and to use its features for dynamic detection. Moreover, the processing time of the proposed system is fast and can run in real-time being able to run in outdoor and indoor environments.

Introduction

In the last years, researchers around the world have directed particular attention to problem solution using computers, for instance, the idea for offering to a system the ability to sense the world as we do. Despite we are complex systems, there are a lot of sensors which can do the same or similar function like our sensors, which obtain information about the environment with different capabilities. For instance, some sensors were made to detect temperatures variations, and others to detect distance. In particular, an essential sense we have is the vision, and the equivalent sensor is the camera, either monocular or stereo these sensors act as our eyes. Current research in computer vision, is mostly oriented to find out solutions for objects recognition. Therefore, it is desirable to have computer systems to sense objects color, shape, size, or 3D information. There are a lot of approaches, one of them is the 3D reconstruction using stereo cameras or RGB-D sensor, among others.

Simultaneous Localization and Mapping (SLAM) is a combination of the environment reconstruction in 2D or 3D, and the robot position over unknown scenes while the robot is moving. Due to this movement, the reconstruction and the pose are updated after each movement, and then a 3D map is created assembling each reconstruction. Moreover, SLAM has several applications, some examples are for recognizing areas where the human lacks access or its maneuver is hard, for instance, in environments like underwater, inside of mines,

navigation in indoor and outdoor scenes, for mounting on UAVs (Unmanned Aerial Vehicles), mobile robots, among others. Therefore, SLAM systems are strategin for the development of the next navigation techniques.

This work is addressed as follow. The first chapter presents the state of art about SLAM, its main problem, and some approaches in dynamic environments. Chapter 2 introduces to the fundamental knowledge, which this thesis is based. Chapter 3 mentions the methodology and contributions of this work. Chapter 4 reports the results of the tested system. Finally, in Chapter 5, the conclusion and future work are presented.

CHAPTER 1

State of the Art

The SLAM problem dates from 1986 in IEEE Robotics and Automation Conference, where researchers had a conversation about probabilistic methods in robotics area [11]. Since that year, SLAM has become a relevant topic in this area.

In general, SLAM is a system focused on recognizing the environment and generating a map and finding its location at the same time [12]. A map is a visual representation of the environment and it can be very useful in several applications, as the representation of places where the human do not have open access, or for carrying out some tasks as path planning [13], mobile robots [14–16], among others [17]. As it is mentioned before, there are 2D and 3D maps; in the case of the first one, it is commonly generated by sensors like LiDARs, sonars and scanners, among others (Figure 1.1), since its detection range covers only two dimensions, x (width) and z (depth). On the other side, the 3D maps can be generated by stereo cameras, RGB-D sensors or 3D LiDARs, and others. These sensors provide information about 3D scenes, therefore, the maps can be dense, semi-dense or sparse. The Figure 1.2 depicts each type of map.

(a) 2D LiDAR [18].

(b) 3D LiDAR [19].

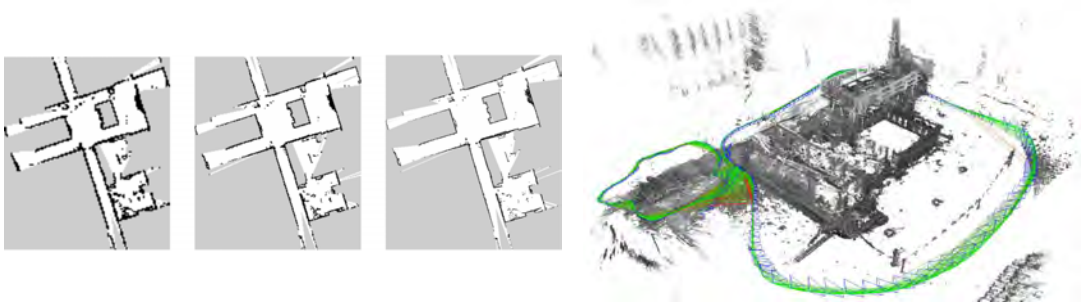
(c) Depth camera [20].

(d) Stereo camera [21].

(e) RGB-D camera [22].

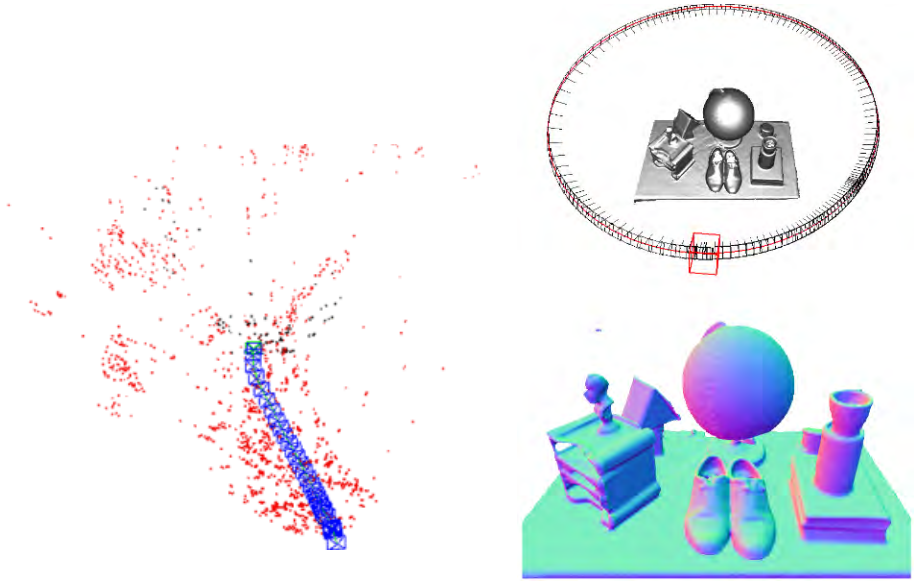
(f) Event-based camera [23].

Figure 1.1: Types of sensors.



(a) 2D map.

(b) 3D semi-dense map.



(c) 3D sparse map.

(d) 3D dense map.

Figure 1.2: Maps, a) shows a 2D map generated by HECTOR-SLAM [1], b) were built by LSD-SLAM [2], c) ORB-SLAM [3] generates a sparse map and d) corresponds to dense map created by KinectFusion [4].

1.1 The SLAM problem

As it is mentioned before, the SLAM problem implies the generation of a map using the system localization, and the estimation of current location using the map, to wit, both need each other, it is usually compared with the famous chicken-or-egg problem. However, there are various approaches to solve it, which imply the implementation of some algorithms using different sensors. Some researchers have proposed various probabilistic approaches for solving the SLAM problem, and these are based in Bayesian networks as Kalman Filter, Particle Filter [24], and Kalman Filter variations, among others [25]. However, the large environments [26] are a main constrain of this type of filters due to the increase in the size of the system variables. On the other hand, another problem is the measurement error from sensors. There are sensors which are more accurate than others, it is always present, as a result of this, the sensors add error to the system and then the system location and mapping have a cumulative error. For solving this, some techniques are used to reduce the error like loop closing [27]. In addition, not only the sensors cause error, the dynamic objects are within several scenes cause error in some processes such as the calculation of the perspective transform, among others. A consequence of this is the erroneous map and wrong system location. The next section mentions some techniques focused on dynamic environments.

1.2 SLAM in dynamic environments

There are many SLAM approaches focus on static environments, it is a strong assumption, since it restricts the system a lot. However, there are several dynamic environments, where the moving objects can generate an error in outdoor or indoor environments, it is because dynamic features cause a bad pose estimation and erroneous data. For this reason, new approaches have arisen for solving the dynamic environment problem. For instance, in [28], Wei Tan *et al*. focused their work on appearance and structure changes due to occlusions and false corresponding. The occlusions are non-data pixels caused by the separation of the sensors and the difference of perspective, and the false corresponding are wrong matched

features from the first frame to second frame.

In [29], Saputra *et al.* divide the techniques used in dynamic environments in three classes, the first one is *Robust Visual SLAM*, *Dynamic Object Segmentation and 3D Tracking* is the second one and *Joint Motion Segmentation and Reconstruction* as the third class. The next subsections are based in the work of Saputra.

1.2.1 Robust Visual SLAM

One of the robust visual SLAM techniques is based on prior knowledge, which implies that the system knows information about the scene when it starts mapping. In this approach, the static information corresponds to background and the dynamic features belong to foreground. Some SLAM systems are based on this kind of approaches, for instance, two works about foreground initialization are mentioned in [30] and [31], Wangsiripitak *et al.* and Chhaya *et al.* respectively use prior information for detecting the dynamic features. The first approach uses a 3D object to remove the dynamic features, on the other hand, in the second one, the system knows dynamic SURF feature descriptors [32] and then they are stored. On the other side, there are works using a background initialization, where the system initializes with a prior knowledge about static information of the scene. These approaches subtract the background information for avoiding the dynamic objects, in [33], Piccardi mentions some works related to background subtraction. Another technique on the Robust Visual SLAM branch is the geometric constraint [8], this method is based on epipolar geometry [34], triangulation [35] or the reprojection error [36] for determining of dynamic objects, since these objects disobey the geometric constraints.

Another approach of this class is the usage of optical flow [37], it determines the motion direction from two consecutive images. For instance, in [38], Alcantarilla *et al.* use an image obtained from dense optical flow to detect the dynamic objects using two images from a stereo camera. And then, they use a metric and a Jacobian matrix for eliminating the measurement error. On the other hand, Cheng *et al.* [39] compute the optical flow in current image from monocular camera, and then, they use a metric for dynamic object distinction.

Continuing with the same class, the ”ego-motion constraint” is another technique. In [40], Scaramuzza estimates the motion from a monocular camera, the system is restricted with nonholonomic constraints, which allows the usage of only 1-point correspondence. He uses the 1-point RANSAC algorithm [41] and histogram voting. Finally, the last approach uses neural networks (NN), for instance, Dosovitskiy *et al.* [42], Ilg *et al.* [43] and Mayer *et al.* [44] use optical flow with supervised learning for detecting and segmenting the moving objects. On the other side, Lin and Wang [45] obtain the images of the scene from a stereo camera, after that, the neural network learns high-level features for detecting the moving areas of the image.

1.2.2 Dynamic Object Segmentation and 3D Tracking

The second class also has several approaches, one of them is the statistical model selection, which the data sampling is done from image features and then the system attempts to fit the motion models using the data sampling, after, the process selects the best fits. The static features are described by one motion model, the dynamic features uses several models. This process is used to find the inlier and outlier features, there are several works using this type of approach like [46], [47], [48] and [49]. The next technique is the subspace clustering, which uses subspaces to define each moving object fitting the data to each one. The basis vectors represent from high-dimensional data to low-dimensional spaces, these vectors can be used for representing the subspaces if these are linear [50, 51].

On the other hand, “geometry” is another technique of the second class, its main idea is based on multiple fundamental matrices, to wit, if a fundamental matrix describes the motion of camera in the static scenes, n fundamental matrices characterize the n objects motion in dynamic environments. Vidal *etal*. have focused on this type of approach in [52]. On the other hand, another technique used for the segmentation of dynamic objects is based on deep learning, for instance, the work of Vijayanarasimhan *et al.* [53] is relied on two convolutional/deconvolutional subnetwork, the first one generates a depth map (an image with a corresponding depth value to each pixel) from a single frame, once the depth map is predicted,

the process creates a point cloud from depth image using the intrinsic camera parameters. A point cloud is a set of mapped data points in the space. The second subnetwork uses a pair of images for predicting a set of K segmentation masks, which is the same quantity of predefined dynamic objects, also, the subnetwork computes the motion of each mask and the camera motion. After that, the system computes a scene flow from a previously transformed point cloud, finally, the process generates the scene flow. On the other hand, the last class will not be explain, since it is based on Structure From Motion (SFM).

CHAPTER 2

Theoretical Knowledge

This chapter is divided in four sections, each one explains the main knowledge which the SLAM is based, since the system is a several processes union to achieve the mapping and localization. The following topics that will be addressed and the system uses are: ORB-SLAM [3, 5], SegNet [6] and DS-SLAM [7]. Finally, the last section talks about Robot Operating System (ROS), a framework, where the SLAM runs.

2.1 ORB-SLAM

The main algorithm is based on a well-known SLAM system that is one the best implementations of the mapping and localization problem, since is fast due to the way to obtain the interest features about environment, this system is named ORB-SLAM due to used features to recognize the scene, these features have the same name (ORB, Oriented FAST [54] and Rotated BRIEF [55, 56]), and ORB features consist of finding potentially invariant points to rotations and escalations. This kind of features allows to obtain the same point from different perspectives or views and to match with previous points.

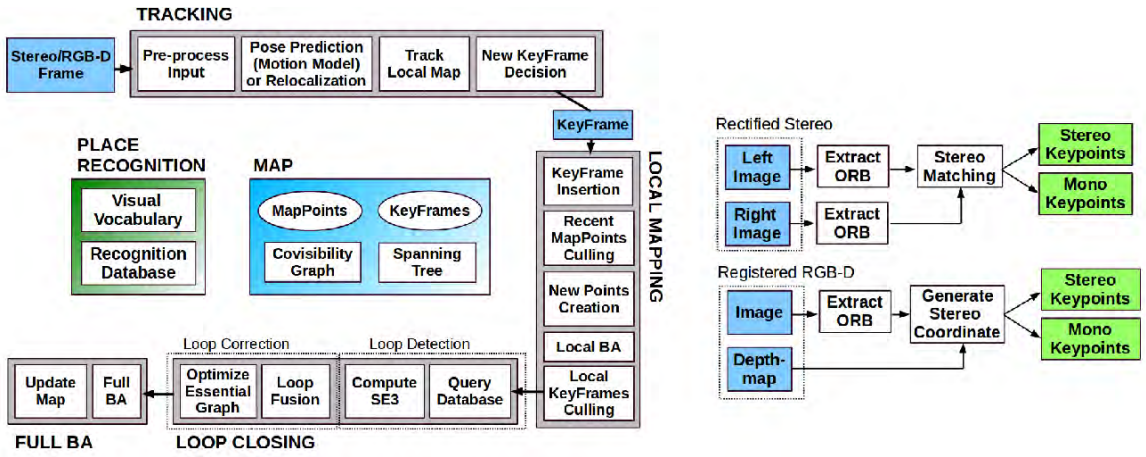


Figure 2.1: ORB-SLAM system consists of three threads, the first one is the tracking (the upper gray block), where the images are processing, the ORB features are extracted too, the next thread is local mapping, the keyframes are using to obtained the map and the last is loop closing, where if the keyframe is detected previously [5].

ORB-SLAM approach is separated in three threads, tracking, local mapping and loop closure as we can appreciate in the Figure 2.1. Below are mentioned the main threads of the ORB-SLAM.

2.1.1 Tracking

The tracking thread has four main processes, which are *Pre-process input*, *Pose prediction or Relocalization*, *Track local map* and *New Keyframe Decision*. At first, the input data is obtained from a monocular, stereo or RGB-D camera and then the ORB features are extracted from each image. In the right side of the Figure 2.1, the pre-process is shown, it is applied to the input images, either RGB-D or stereo images, both have two types of keypoints, monocular and stereo, and the points are classified in close or far depending on intrinsic parameters of the camera.

The stereo keypoints have three defined coordinates and are determined as $x_s = (u_L, v_L, u_R)$, where u_L and v_L are the coordinates on the left image, and u_R is the horizontal coordinate on the right image, the reason for ORB-SLAM only needs one coordinate is because the right camera is fixed respect to the left camera, so, the ORB features detected in the left image

tend to be in the same vertical position but different horizontal position. On the other hand, the RGB-D input only capture an image RGB, therefore, the process generates a virtual right coordinate using the depth image (d), and some intrinsic parameters as focal distance (f_x) and baseline (b). The way for obtaining it is using the disparity formula (2.1), this concept is the difference between horizontal coordinates corresponding to an ORB match.

$$disparity = u_L - u_R = \frac{f_x b}{d} \tag{2.1}$$

Since, f_x, b and d are known, $disparity$ is unknown and the aim is find u_R, the formula to obtain it is:

$$u_R = u_L - \frac{f_x b}{d} \tag{2.2}$$

Once the coordinates are obtained, the system classifies them as close or far. This classification allows to safely triangulate the keypoints using close keypoints, since the captured objects in the image have better resolution. The keypoints are considered close if the corresponding depth estimation for the keypoint is less than 40 times the baseline. The far keypoints are used for giving rotation information, since they can appear in many frames, therefore, they are used in multiple views.

The monocular keypoints are defined as $x_m = (u_L, v_L)$, these points correspond to not matched features or RGB input, in this case, the depth is invalid. The monocular keypoints are triangulated in multiple views. On the other hand, the system obtains RGB input frames whereas the process is running, however, if the SLAM system stores all these frames, the memory would be saturated. A solution of this, some frames are selected to be stored depending on a criteria, these frames are named as keyframes.

After obtaining the stereo and monocular keypoints, the system predicts the system pose, the first time that the system is executed, the thread selects a keyframe and its pose as the SLAM origin, then the SLAM generates an initial map using the stereo keypoints. Since the threads run all the time while the ORB-SLAM is running, in some situations, the system loses its position, in this case, a bag of words based in DBoW2 [57] is used to find candidate

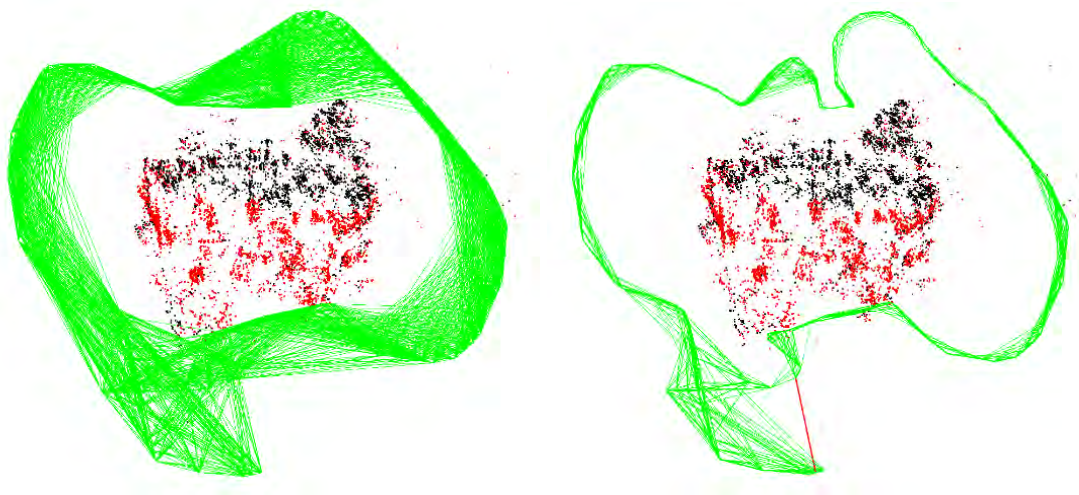
keyframes. Then, the system corresponds ORB features with map points in the selected keyframes. The camera pose is found using RANSAC iterations [8] for each keyframe and using PnP algorithm [58]. This process is named "global relocalization" and it is based on [59]. After that, the SLAM optimizes the pose camera using bundle adjustment (BA) [60,61], which is a technique for functions optimizing. The system uses Levenberg-Marquard algorithm, a method implemented in g2o [62], to carry out the optimizations.

Once the camera pose is estimated, the 3D map is created. While ORB-SLAM is running, the map is increasing, it means that the processing time is getting bigger, to solution it, the SLAM uses a local map composed of a set of keyframes K_1, a set K_2 and a reference keyframe $K_{ref} \in K_1$, where the set K_1 has matching map points with the current frame, the set K_2 shares map points with the set K_1, and K_{ref} is the keyframe with more map points in common with the current frame.

Finally, the last process in the tracking thread is the new keyframe decision, here, the input frames are converted in keyframes or not depending on different conditions, the first one is related with relocalization mode, since at least 20 frames must have passed between the last global relocalization frame and the current frame, the second condition is similar to the last condition, the difference is that this condition depends on keyframe insertion, at least 20 frames must have passed after the last keyframe insertion. If a frame has at least 50 points (condition 3) and less that 90% points than K_{ref}, the frame is considered as keyframe. In general, all these process are within tracking thread.

2.1.2 Local Mapping

The next thread is the local mapping, where it has as input the keyframe obtained from the tracking thread, only if the frame was accepted as keyframe. If this is the situation, the covisibility graph (a graph where the keyframes are linked among others depending on common points, see a) from Figure 2.2) is updated and the process adds a node for K_i and updates the edges between keyframes depending on shared map points. After that, the bag of words is computed for new keyframe.



(a) Covisibility graph (b) Essential graph

Figure 2.2: Graphs, a) a covisibility graph shows all links from a keyframe with other keyframes, b) a essential graph depicts the strongest links from a keyframe with others. [3]

Once the keyframe insertion is done, local mapping thread culls the map points if these do not satisfy the next conditions. The first one says that a point must be in more than 25% of the frames where it must appear. And the second condition refers to that map point must be observed in at least three keyframes after its creation. The next step of local mapping thread consists in the new map points creation. In the covisibility graph, there are K_i keyframes connected by edges, also there are unmatched points, so, the SLAM system attempts to match these points with another unmatched points and then triangulate these points to obtain new points only if these have the scale consistency, parallax, positive depth in both cameras, and the reprojection error (difference between the the real 2D point and the 2D projected point from the 3D point) is small. After that, a local BA optimizes the current keyframe with a connected set of keyframes and all points observed in that set. Also, there are keyframes with the some points seen in the set of keyframes but their keyframe are not in the set, however, for this situations, the points are also included in the local BA and remain fixed.

The last step of this thread is the local keyframe culling, this step consists in discarding redundant keyframes. For this, the system checks if the 90% of points of a keyframe is in at least three keyframes in the same or finer scale [63]. If this is the situation, the process

deletes the keyframe.

2.1.3 Loop closing

The loop closure thread tries to find a loop closure using the last keyframe processed by local mapping thread. This thread has two main processes; loop detection and loop correction. Below each step will be mentioned.

The loop detection has two main steps, the first step is to detect a candidate loop using a similarity between the bag of words vector of K_i and its neighbors. An angle is necessary to discard non-relevant information, the proposed angle is $\theta_{min} = 30$, it is to avoid using much information, since the process will be slow, the step retains the lowest score s_{min}. The keyframes with a lower score than s_{min} and directly connected to the last keyframe processed are discarded and it only considered a loop candidate, if there are at least three loop candidates with keyframes connected. After that, a similarity transform is computed, this is because the current frame and loop candidate keyframe do not have the same perspective, therefore, it will help to validate the loop and know the accumulated error. This step computes the similarity transform using the map points of the current keyframe and loop candidate keyframes. The way to do it is using ORB correspondences between both and then, the similarity transform is computed with method proposed by Horn [64] with iterations of RANSAC algorithm with each candidate keyframe. Once the last step is finished and a similarity transform is successful (enough inliers) and optimizes, the system optimizes the similarity matrix again using more correspondences. If the similarity transform is still successful, the result is a loop closure.

Like the loop detection, the loop correction has two steps: loop fusion and optimize essential graph (see b) from Figure 2.2). If the last process detects a loop, the loop correction fuses the duplicated map points, the recent keyframes generates new edges, this edges are inserted in the covisibility graph. The current pose is corrected according to the loop keyframe, and the correction is propagated to the last poses and all points inside of the keyframes set are fused. Finally, the edges are inserted to covisibility graph. The last step of loop detection is

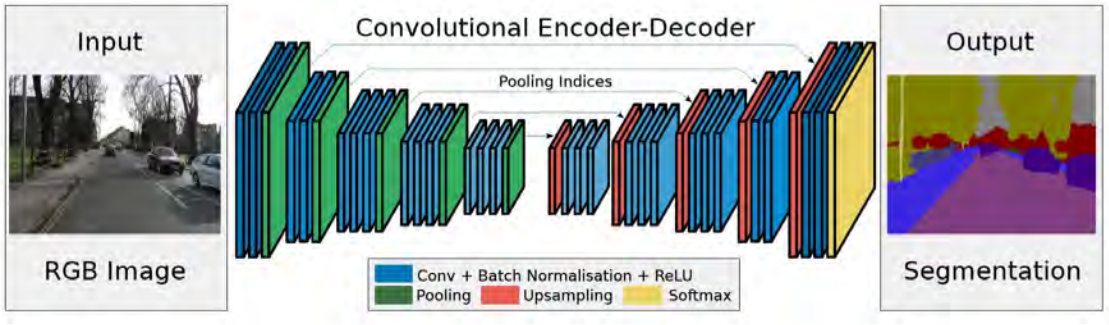


Figure 2.3: SegNet is an encoder-decoder architecture, the encoder consists of thirteen convolutional layers, batch normalization, ReLU as activation function and pooling, the decoder contains thirteen convolutional layers, upsampling and softmax functions [6].

the optimization of the essential graph which contains the nodes of keyframes as covisibility graph, but the difference is that the essential graph has less edges, it allows a better computational cost. This step applies an optimization using the similarity transform which corrects the scale drift. Finally, a full BA is used, BA adjusts all points and keyframes except the origin keyframe.

2.2 SegNet

SegNet is a semantic segmentation neural network, this type of NN searches certain objects previously trained and then it detects the object classification, after that, the NN determines the location of detected objects and enclose it with boundary boxes. Finally, the network labels each pixel of the image depending on the corresponding class. In resume, SegNet is a neural network developed to detect objects and enclose in many classes with boundary boxes and finally the pixels are segmented.

2.2.1 Architecture

This system has a convolutional encoder-decoder architecture and a pixelwise classification layer, each encoder layer has a corresponding decoder as is shows in the Figure 2.3. The encoder contains thirteen convolutional layers, this step is designed for object classification

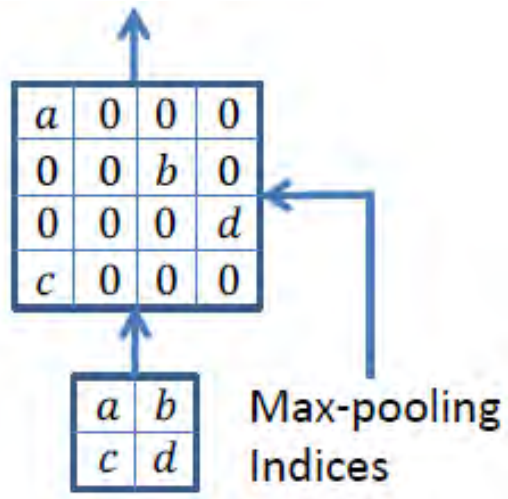


Figure 2.4: Up-sampling: 2×2 window corresponds to input feature map, a, b, c and d are feature values, 4×4 window is an up-sampled feature map using the corresponding max-pooling indices to locate each value from input feature map [6].

and is based on VGG16 network [65] but the fully connected is discarded. This architecture allows the use of trained weights for the training process.

Encoder

The encoder network contains different processes, one of them is a filter bank in charge of obtaining feature maps, a filter bank is a set of convolution kernels applied to the input image. After that, the network performs a batch normalization, it speeds up the training since the input data is more stable, and then the system applies an element-wise operation for data rectification, the activation function is named rectified-linear non-linearity $(ReLU)max(0,x)$. Following that, the maximum value selection is implemented, this type of process is named max-pooling, in this work, the neural network implement a 2×2 window with a stride 2 in the max-pooling, it can achieve a translation invariance over is important to mention, each encoder stores the max-pooling indices, since the decoder network needs the information for up-sampling.

Decoder

As it is mentioned before, each encoder has a corresponding decoder and this is in charge of up-sampling the input data from encoder network, for this, the decoder uses feature maps and max-pooling indices, since each encoder is linked to its own decoder, the features maps and indices belong to corresponding encoder layer. When a feature map is up-sampled, the process produces a sparse feature map, it is because the up-sampled step does not have enough information to fill in the blanks. The Figure 2.4 shows a 2×2 feature map and then a 4×4 window with feature values using max-pooling indices. Once the system has a sparse feature map, the next process consists in a trainable decoder filter bank convolution, the output results in a dense feature map. Finally, the last decoder has a trainable soft-max classifier which is used to obtain a K-class probability of each pixel, where K is the number of classes, in other words, soft-max classifies pixel-by-pixel with a certain probability depending on the K-classes, therefore, the output is a K channel image. The maximum value of probability at each pixel corresponds to the predict segmentation.

2.3 DS-SLAM

DS-SLAM [7] is a system which its main aim is to avoid the dynamic objects and reduce the error of the mapping and localization and therefore obtain a 3D representation about the real world.

In general, the system consists in a semantic segmentation network implemented in the ORB-SLAM2 [3], this network is adopted to achieve the elimination of dynamic objects which produce an error when the system attempts to find the sensor position and mapping the environment using the detected features of the environment. On the other hand, DS-SLAM uses this information to create an octo-tree map representing the environment where the system is running and is able to better the mapped space. The Figure 2.5 shows the overview of DS-SLAM.

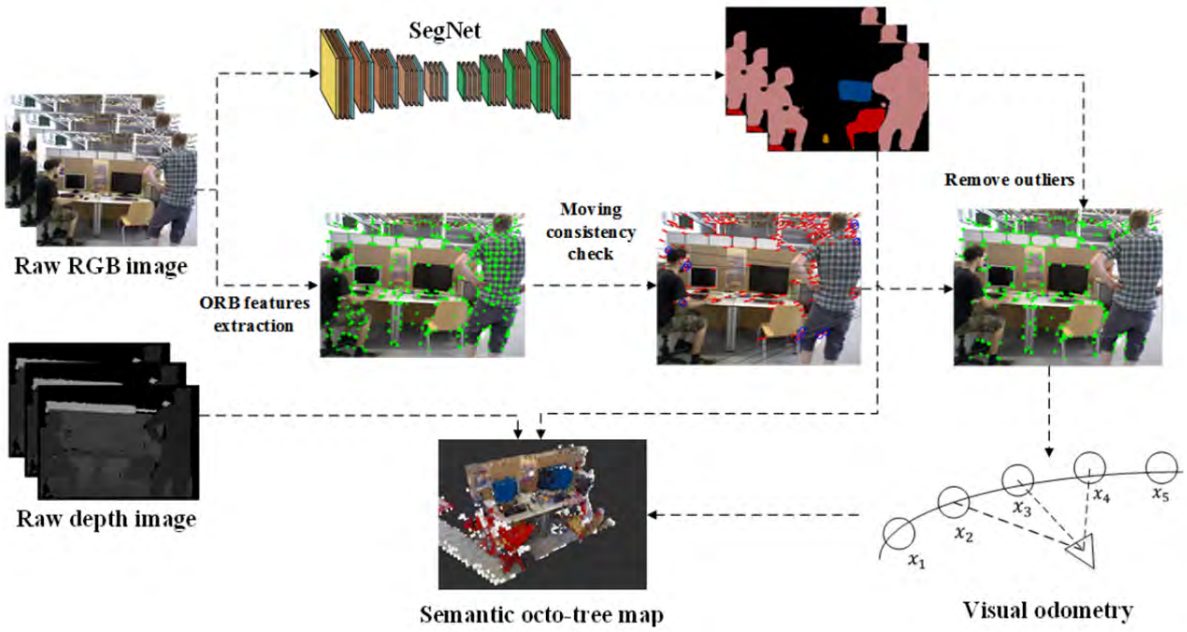


Figure 2.5: The system receives two images sequences (RGB and depth images), DS-SLAM uses the RGB image to detect ORB features and semantic segmentation frame, then the outliers are removed and then the system computes the visual odometry. Semantic octo-tree map uses as input the depth image, semantic segmentation data and visual odometry. [7].

2.3.1 Outliers Removal

The main error obtained in the measures is the presence of dynamic objects, since a total features fraction are localized on the moving objects. Therefore, it is important to avoid it during the system process. In contrast with ORB-SLAM, DS-SLAM uses the SegNet to detect objects in motion. The SegNet is the first used function for selecting the possible dynamic objects, which the NN has previously trained. However, some dynamic objects has not trained by the NN, as result of this, the dynamic objects are not segmented.

A semantic segmentation is a good method to find objects, however, many of them are sometimes static, if the system is not able to determinate whether the object is dynamic or not, then some features points will be removed, it means that the scene has fewer feature points. As a consequence, another function is implemented to detect dynamic objects, it is named "*Moving Consistency Check*". In other words, the function is in charge of detecting the possible outliers, the idea to is calculate an optical flow pyramid and then the good matches selection is done. A matched pair of features is considered "good" if the features are not near to the edge of the image, another parameter to discard the feature points is the difference of pixels between the 3×3 image block and the center of matched pair.

Since the main idea is to remove features belonging to dynamic objects, a fundamental matrix is found, this in order to obtain their corresponding epipolar lines. In other words, in geometric approaches, an epipolar line is obtained from a feature and a fundamental matrix using the equation (2.3), where $x_1 = [u_1, v_1, 1]^T$ and $x_2 = [u_2, v_2, 1]^T$ are homogeneous coordinates, u and v are the coordinates in pixel, $l_1 = [X_1, Y_1, Z_1]^T$ and $l_2 = [X_2, Y_2, Z_2]^T$ are the epipolar lines and F is the fundamental matrix. In particular, a fundamental matrix maps a feature point from first frame to the second one, the result is a line where the point lies on the another frame. Therefore, F maps from a point to a line as is showed in the Figure 2.6.

$$l_2 = F x_1 \tag{2.3}$$

Once the fundamental matrix is obtained, dynamic points are selected from the distance

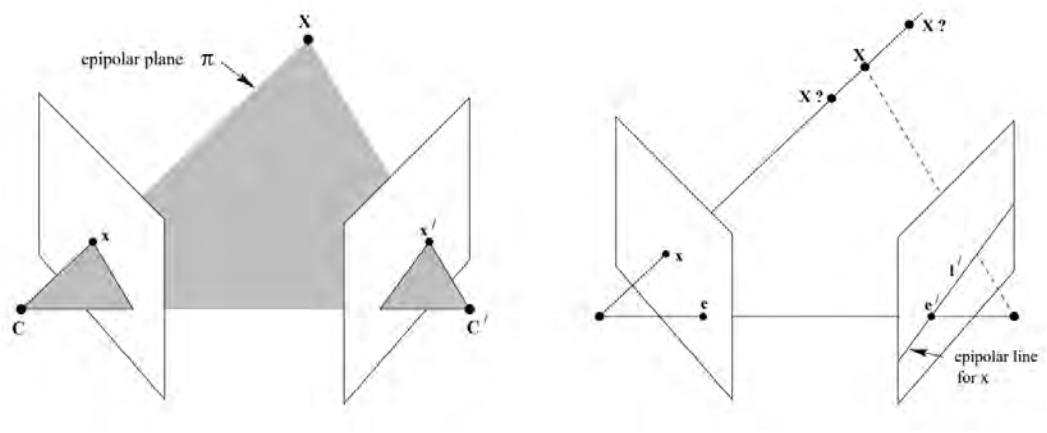


Figure 2.6: Epipolar geometry representation. (a) represents the ideal mapping of a 3-space point (X) on both keyframes, the ray is traced from the center of the camera C to X passing through x and the same for the other frame. (b) The x point has an inexact position and hence the position in the second frame is considered on the epipolar line instead of particular point [8].

calculation between the matched point and the epipolar line. The equation (2.4) is used to determinate if the matched points are dynamic or not depending on a threshold, in other words, when the fundamental matrix is found, the points from the first frame are projected to the second one as a line, if the distance between this line and the matched point in the second frame is bigger than a threshold, the point is rejected.

$$D = \frac{|{x_2}^T F x_1|}{\sqrt{\|X_2\|^2 + \|Y_2\|^2}} \tag{2.4}$$

Thus, the last filter to remove the moving points is the selection of dynamic objects using the semantic segmentation information and moving consistency check results. To achieve the aim, a quantity of dynamic points obtained by the "moving consistency check" function must lie on the borders of semantic information, if there are enough points fulfilling it, then the object is considered dynamic and the points are removed.

2.3.2 Mapping: Octo-Tree Map

An octo-tree map [66] is the environment representation, this map depends on the semantic data, the depth image, and new keyframes to generate a local point cloud, which will later be incorporated into a global octo-tree map. During point cloud creation, the semantic information represents the detected objects by the neural network, in other words, when a sofa is found, the voxels are drawn in red color, in general, each object detected by NN has a corresponding color. A voxel is considered as a pixel but the difference lies in the extra coordinate, instead of "x" and "y" coordinates, the voxel has "z" coordinate.

DS-SLAM uses a NN to find possible dynamic objects, however, this neural network is not perfect. It exist situations where the objects are overlapped among themselves. It causes inconsistencies in the octo-tree map. So, DS-SLAM implements a log-odd score method, it is used to avoid bad mapping applying a probability to each voxel being occupied. The equation (2.5) represents log odds score and is denoted by $l \in \mathbb{R}$ and $p \in [0,1]$ is the probability being occupied.

$$l = log\left(\frac{p}{1-p}\right) = logit(p) \tag{2.5}$$

The inverse function of *logit* (2.6) shows how it is possible to obtain the probability from log-odd score.

$$p = log\left(\frac{\exp(l)}{\exp l + 1}\right) = logit^{-1}(p) \tag{2.6}$$

For representing the increase or decrease in log-odds score of a voxel from the beginning to time t, the used term is $L(n|Z_{1:t})$, where n is each voxel and Z_t denotes the observed voxel at time t (sensor measurements). Then, for calculating the log-odd score of voxel n at time $t+1$, it uses the equation (2.7)

$$L(n|z_{1:t+1}) = L(n|z_{1:t-1}) + L(n|z_t) \tag{2.7}$$

In this equation, $L(n|Z_t)$ term is τ depending on if the observed voxel can be occupied at time t, otherwise the term is 0. Therefore, if a voxel is seen many times, the probability of a voxel to be mapped is greater. A pre-defined threshold is responsible for allowing a voxel to be added to the octo-map, that is, if the occupying probability is above the threshold, the voxel is mapped. Finally, this method avoids mapping dynamic objects in the octo-tree map.

2.4 ROS: Robot Operating System

Before considering the methodology of this work, it is important to know how ROS (Robot Operating System) works, which will be explained below, since this system is the base of the SLAM execution, and it allows exchanging information from different algorithms and then use the data in other programs.

As we mentioned in the past, ROS is a system which it allows communicating various algorithms simultaneously. This characteristic allows sharing information (messages) from an algorithm when it is executing. Each executed program establishes a *node* representing a process, which can provide information computes during of the process execution. The way for distributing the information is the usage of *topics*. A topic is the bus where the process (node) sends and receives the messages, in other words, a topic is the plumbing of the system where the data navigate from a node to another. It exists different type of topics which depends of the kind of information, for instance, the way to share an image from a process to another is using "sensor_msgs/Image" [67], each topic has different parameters, continuing with the same example, this topic has the size of the image (width and height), also it has the type of encoding, this can be "8UC1", "16UC3", "16SC1", among others.

In the Figure 2.7 shows an equation representing the general meaning of ROS, in this picture we can appreciate four characteristics, *plumbing* is the connections among nodes through topics, *tools* are those packages as RViz, Gazebo and others that allow visualize the programs, among other applications, *capabilities* are different applications of algorithms to achieve general or specific implementations as image processing, pose estimation, filters, and more. Finally, *ecosystems* mean the capability of the program has to join ROS package from

Figure 2.7: ROS: A set of nodes, topics, tools, several applications, compatibility of programming languages [9].

different authors with distinct applications and functions.

In general, ROS is a platform where several programs are able to share messages among themselves. The algorithms are executed using two different ways, the first one is using *rosrun*, but it needs to execute the "roscore", which it is a nodes collection and necessary services to run the algorithms, the second way to achieve running a program is executing it with *roslaunch*, it needs a ".launch" file, which contains information about name of the nodes we want to execute, the name of the ROS package and optional parameters about the algorithm, in this case, it is not necessary to run the roscore since roslaunch is in charge of doing it. A difference among rosrun and roslaunch is the number of nodes executed when the command is used. In the case of rosrun, it allows to run only one node. On the other hand, roslaunch allows the executing of only one node or more nodes. However, the best way to execute the nodes depends on the user.

CHAPTER 3

Methodology

This chapter presents a description of the necessary configurations to develop the SLAM system, the first configuration is about different camera models and which is the chosen camera to use as input, after that, the next section is about camera, SLAM and RVIZ parameters. The next section of this is about the sequence of the main code to achieve the real-time approach. Finally, the last section talks about the code to run the program using a stereo camera in a dynamic environment.

3.1 Camera configurations

Once we have the previous knowledge about ROS, SLAM and CNN, we can address the implementation of the program. As we mentioned before, the key idea of SLAM is obtaining information about the environment. Therefore, it is necessary as input a camera which could be a monocular, RGB-D or a stereo. The selection of the sensor depends on the approach. Therefore, the characteristics of each kind of cameras are mentioned below.

Figure 3.1: Kinect sensor, an example of a RGB-D camera [10].

3.1.1 RGB-D camera

As is well known, the RGB-D sensor works very well in places where the rays of the sun are not direct, in other words, indoor scenes and it is not trustworthy in outdoor environments, it is due to its operating mode, which consists in four channels of data, the RGB abbreviation means the three channels of typical image, in other ways, RGB are *Red*, *Green* and *Blue* layers of an image, the fourth channel is a matrix with depth values of the scene. Therefore, RGB-D camera has an RGB camera, an IR projector and a IR camera. The last one is used to detect the dot pattern projected by the IR sensor (IR projector), these dots provides information about depth of the environment, the IR camera measures the "time of flight" of the projected light by the infrared transmitter, in other words, the distance between sensors and objects is measuring the time that light takes to travel from the transmitter to object and after returning to receptor. Since the sensor works with infrared light, it is very hard to detect the dots under the sun, so, it is not trustworthy in outdoor.

One of the most known RGB-D cameras is the "Kinect" by Microsoft, in the Figure 3.1, the internal structure is observable. The Figure 3.2 shows the IR pattern from a Kinect, the dots are light and dark speckles and are generated from several diffraction gratings [10].

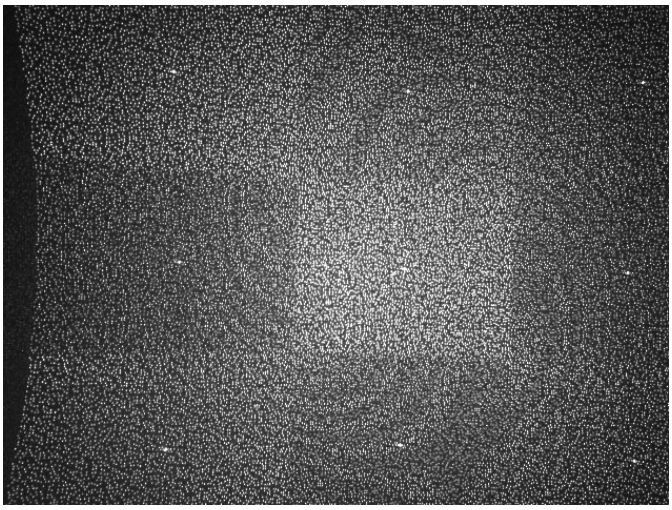


Figure 3.2: IR pattern used to calculate the depth of the real world. [10]

3.1.2 Monocular camera

Another sensor is the monocular camera, the main characteristic is the absence of a depth map obtained from only one position, in other words, the camera is not able to obtain information of depth about environment. In the case of RGB-D camera, as it is mentioned earlier, there are two sensors which obtain the depth, in the case of the stereo camera, the baseline is known, therefore, the calculate of depth has more constrains, it means that the depth is easier calculated than using a monocular camera. An example of a monocular camera is shown in the Figure 3.3.

3.1.3 Stereo camera

The stereo camera consists in two RGB cameras separated by a fixed distance, the depth map from stereo camera is obtained from the matching of features, which are found in both images (left and right), once we obtain the features, the difference (in pixels) between the features on left image and right image has to be calculated, when the features correspond to distant objects the distance between the left features and right features is small, on the other hand, the distance when the objects near the camera is larger, this separation is known as *disparity* and it is elementary to calculate the depth because the disparity is inversely

Figure 3.3: Monocular camera

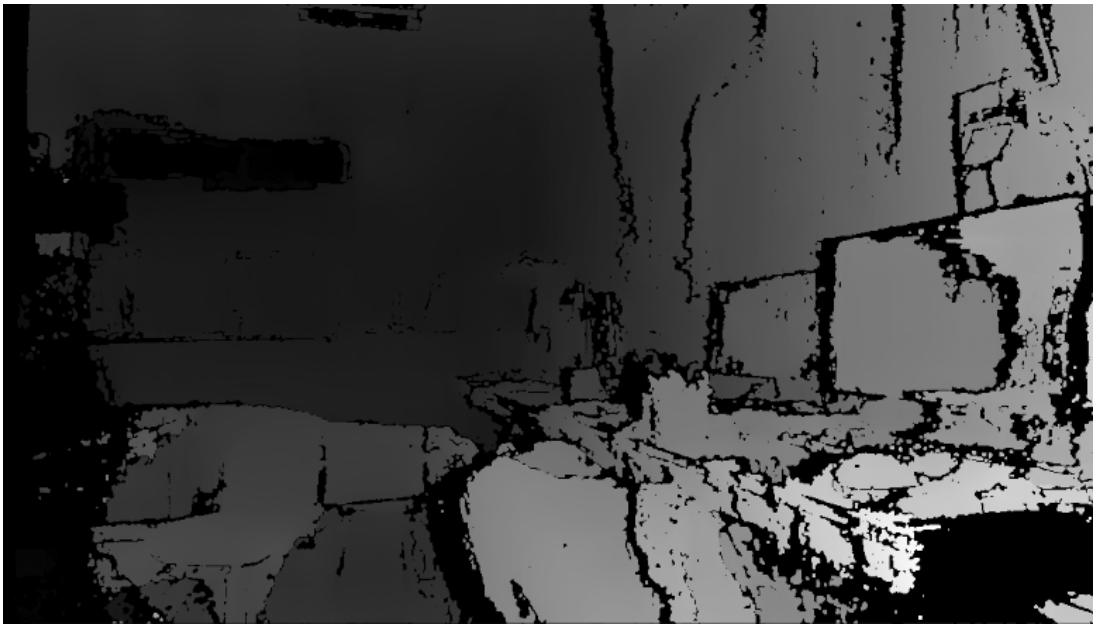
proportional to depth including some internal parameters of the cameras as baseline and focal distance, in other words, the depth is calculated from the equation (3.1), where D is the depth, b is the baseline and f is the focal distance. By this reason, the method to find the depth from a stereo camera is more robust to the rays of the sun, therefore, is better in outdoor environments.

$$D = \frac{b * f}{disparity} \tag{3.1}$$

In (a) of the Figure 3.4 the images correspond to the real-time captured pictures, which are separated by a fixed distance, so, we can appreciate the images are moving between them, it allows obtaining the disparity, in the picture (b) the depth is shown, the light gray represents the near objects, and the dark gray shows the distant objects, the dark shadows are caused by occlusions, therefore, the disparity is not defined and consequently the depth.

Considering the last properties, the stereo camera is chosen since it allows to work in outdoor environments, one of the main purposes of this work. There are a lot models of stereo cameras, in this case, ZED [68] is the chosen camera to implement, this camera is developed by StereoLabs, the advantage is its ROS package, since it allows to modify different parameters depending on the own necessities as quality of depth (Perfomance, Medium, Quality,

(a) Left and right images



(b) Depth image

Figure 3.4: Depth is obtained from two images, (a) the left image corresponds to the left camera, so, the right image corresponds to the right camera, (b) depth is obtained from disparity and disparity is calculated from the difference in pixels between left and right images.

Figure 3.5: Zed camera.

Ultra), sensing mode (Standard or Fill), the quality of image (VGA, HD720, HD1080 and 2K), name of topics, quantity of frames per second among others. Another advantage is the ROS package named "zed-ros-wrapper" [69] publishes the depth of the image as we can observe in the picture (b) of the Figure 3.4. Other important published topic is about information of the cameras, as intrinsic parameters, the size of the images, and more.

As we mentioned in one of the last sections, this work is based on DS-SLAM, this system allows to make a 3D map and works well in indoor environments, the reason is because DS-SLAM is only available for RGB-D cameras, the advantage is the accuracy of the depth, however, due to its architecture it is not advisable to use in outdoor environments. Therefore, the ZED stereo camera is used as input of the system. In the Figure 3.5, the ZED camera is show.

3.2 SLAM Parameters

There are many necessary parameters for the use of SLAM, these parameters refer to stereo camera configuration and SLAM and they are listed in Tables 3.1, 3.2 and 3.3. The first column shows the name of parameters and the second column presents a brief description of them.

These parameters can be modified depending on different factors. The camera parameters are unique values of each camera, for instance, the baseline depends on the structure of the

Table 3.1: Camera parameters.

Camera Parameter	**Description**
Camera.fx	Focal distance in pixels on the x axis: component in x of the distance from center camera to image plane.
Camera.fy	Focal distance in pixels on the y axis.
Camera.cx	Center of the camera on the x axis, it corresponds the half of width of the image in pixels approximately.
Camera.cy	Center of camera on the y axis.
Camera.k1	Radial distortion coefficient [70].
Camera.k2	Radial distortion coefficient..
Camera.p1	Tangential distortion coefficient
Camera.p2	Tangential distortion coefficient..
Camera.k3	Radial distortion coefficient.
Camera.width	Size of the image along x axis in pixels.
Camera.height	Size of the image along y axis in pixels.
Camera.fps	Quantity of frames per second.
Camera.bf	Result of multiplying the baseline and the horizontal focal distance, the baseline is on meters.
Camera.RGB	Configuration of layers of the image, it could be RGB or BGR, if it is monochromatic, the parameter is ignored.
ThDepth	Threshold to determinate if an object is near or far from stereo camera.
DepthMapFactor	Depth map value factor, it depends on the scale factor.

Table 3.2: Viewer (RVIZ) parameters.

RVIZ Parameter	**Description**
PointCloudMapping.Resolution	Size of the flat squares.

Table 3.3: ORB parameters.

ORB Parameter	Description
ORBextractor.nFeatures	Number of ORB features for image.
ORBextractor.scaleFactor	Scale factor to reduce the size of the image.
ORBextractor.nLevels	Number of times to reduce an image.
ORBextractor.iniThFAST	Initial number of features to detect in each cell of the image, the image is divided in cell along of width and height.
ORBextractor.minThFAST	If the number of features is lower than iniThFAST, it is the minimum quantity to detect features, it depends on the texture of the environment.

stereo camera, the intrinsic parameters (f_x, f_y, c_x, c_y, k_1, k_2, p_1, p_2 and k_3) are values that are different for each camera, since they bet on the physical structure and size of the image. On the other hand, ORB and viewer parameters build upon the user, that means of the user can choose the value of the parameters taking into consideration the environment in the case of ORB parameters or the capabilities of the computer or the environment texture and the viewer parameters depend on the taste of the person to visualize the point cloud. The user is able to change these parameters (viewer parameters) during the code is running.

3.3 Real-time configuration

It is important to note that there are programs which are considering “offline” or “online”, the difference between them could be bigger or not, it depends on the applications.

An “offline” program refers to an algorithm which process the data after to obtain all topics, for instance, if the messages from the topic are recorded in a rosbag when a drone is in flight and is publishing information from a GPS, the topic will be processed after to the drone takes off, that means that a program can process the data at any moment. So, it is an example of an “offline” processing. On the other hand, an “online” program is when the processing is in real-time, since the information is used while it is received in the main algorithm.

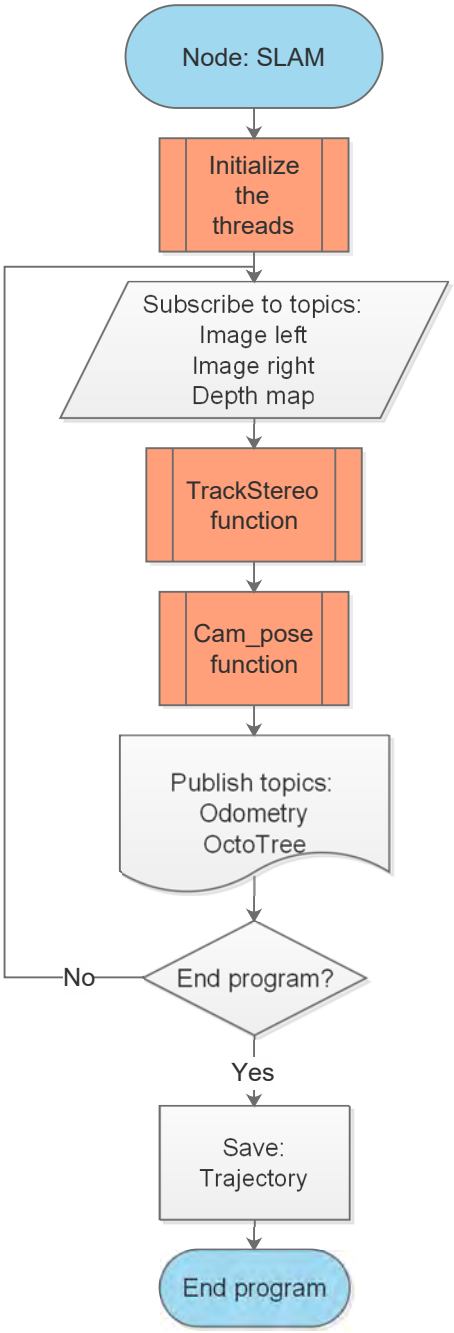


Figure 3.6: Diagram of main code.

In general, the formulation of the problem is how to run the SLAM in real-time on outdoor environments, so, DS-SLAM was modified to be executed "online", the Figure 3.6 shows the sequence of the main program to achieve the aim. The program begins with the block "Initialize the threads", this process as the name indicate, it starts the different threads to run in parallel, it has the capability to decrease the total time in which the program works. As it is mentioned, the initialized threads are: Tracking, Local Mapping, Loop Closing, Segment and Viewer.

In the case of stereo configuration, the program needs three topics to achieve the aim. Two of them correspond to the left and right images, and the last topic is about depth data. The images are necessary to find ORB features and the depth map is used to create the octo-tree map.

Once the program subscribes to topics, the "TrackStereo" function starts with the full process, which corresponds to the detector and extractor of ORB features, and more process mentioned before. After that, the last function of the code is "Cam_pose", which is in charge of the conversion of data from pose and odometry of the camera to topics with the information. So, these topics have the information about the path and orientation of the camera and the odometry of the system. During the main process, if an octo-tree map is generated then a topic is published with information about point cloud. Therefore, there are four published topics: "/Camera_Pose", "Camera_Odom", "Odom" and "/ORB_SLAM2_PointMap_SegNetM". Finally, if the program is closed, the trajectories are saved in a text file. On the other hand, if the program is not closed, the loop is repeated again, hence, the main process continues.

3.4 Stereo configuration

An aim of this work is the mapping and localization in outdoor environments, therefore, as it has been mentioned before, the best camera to use in this kind of environments is the stereo camera. So, the stereo approach is the contribution and hence, the capability of mapping in open-air scenes.

In past sections, it says about the base SLAM for dynamic scenes, however, it only works

in indoor environments given that the SLAM is made for the use of RGB-D sensors. It is for this reason that we decided to do the implementation of the stereo configuration in the code, and the program is able to use not only in indoor scenes, also in outdoor scenes, and taking into account that the detected objects by the neural network may or not may be present in the scene. For instance, in rural environments, there may be sheep, cows, among others, but cars, buses, bicycles or others may not be in this kind of environments. In other words, the neural network allows to find objects in different scenes.

In general, there are three modified functions, which are in charged of finding and extracting ORB features for different scales of size, it aims to maintain invariant the ORB features. Also, the dynamic points are detected with the "ProcessMovingObject" function and then check the moving of the points from one frame to another, if the points are in moving, they will be removed. After that, the distortion of the keypoints (the static features) is removed, in this case, if the input image is rectified, the points will not be rectified, otherwise, the distortion parameters will be used to eliminate the distort. Finally, the depth of the points is computed.

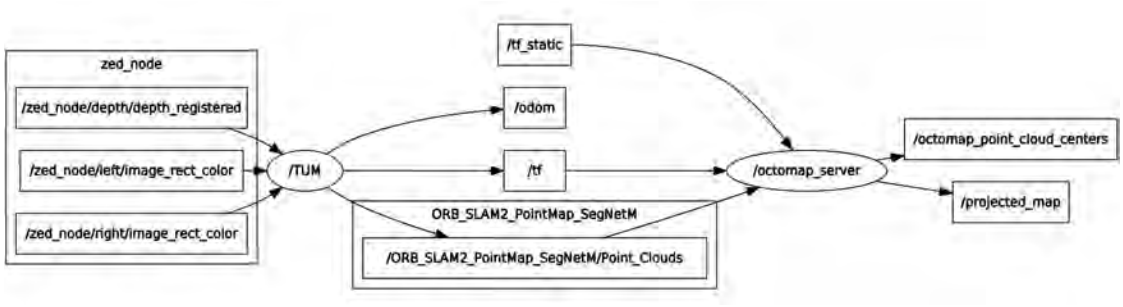
CHAPTER 4

Experiments

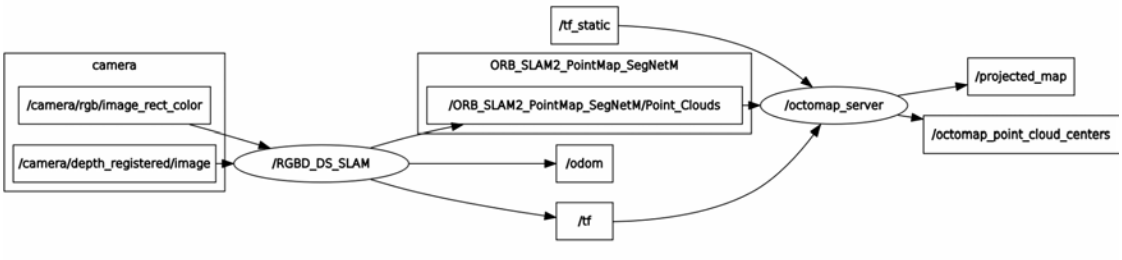
This chapter has four sections, the first section presents the ROS system because the SLAM program is executed in this framework. After that, the second section shows the depth images from RGB-D and stereo cameras in outdoor and indoor environments using different image resolution. In the third section, SLAM is tested and the 3D reconstruction results are shown using both cameras. Finally, the average rates for each configuration of camera and its resolution are shown in the fourth section. For carrying out the experiments, it is used a Kinect as RGB-D camera and the ZED camera as a stereo camera. Also, the system was tested in an Intel Core i7-7820HK laptop computer with 32 Gb RAM and a GPU GeForce GTX 1070.

4.1 ROS

The SLAM process was executed using ROS, as it is mentioned before, this framework allows share data from multiple processes using “topics”. The SLAM system has three main nodes: camera, SLAM and octomap. The Figure 4.1 depicts the necessary nodes and topics to



(a) ROS graph for stereo configuration.



(b) ROS graph for RGB-D configuration.

Figure 4.1: ROS graphs, a) and b) depict the connected nodes and topics for stereo and RGB-D cameras respectively.

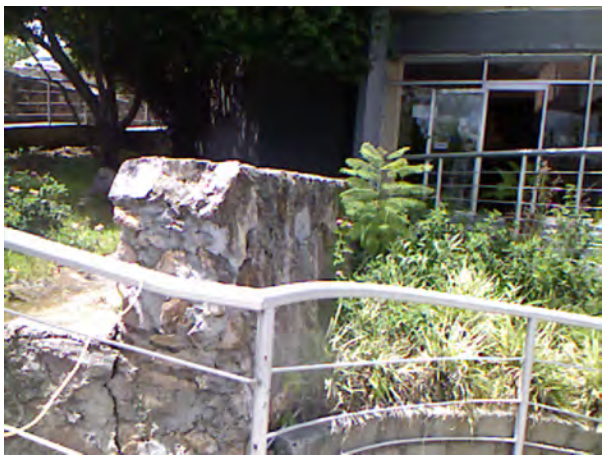
obtain a 3D reconstruction and system localization. For stereo configuration, the main nodes are: zed_node, /TUM and /octomap_server (see Figure a)), on the other hand, for RGB-D camera the nodes are named: camera, /RGBD_DS_SLAM and /octomap_server (see Figure b)). The RGB image/s and depth map are published from camera node, then the SLAM node subscribes to these topics depending on the type of camera configuration and then publishes the corresponding topics to the odometry (/odom) and the point cloud for $K-$keyframe (/ ORB_SLAM2_PointMap_SegNetM/Point_Clouds), after that, octomap node publishes the point cloud topic and its pose for current keyframe. The topics for visualizing of mapping and localization are: /odom, /projected_map and /octomap_point_cloud_centers.

4.2 Depth Comparison

The depth image is very important to obtain a 3D reconstruction of the environment, hence, therefore, this section shows many indoor and outdoor experiments capturing the depth image

(a) Depth image using Kinect camera (VGA resolution.)

(b) Depth image using Kinect (VGA resolution).

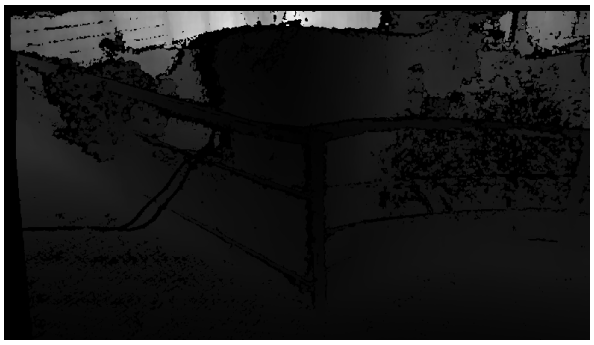


(c) RGB image from Kinect camera.

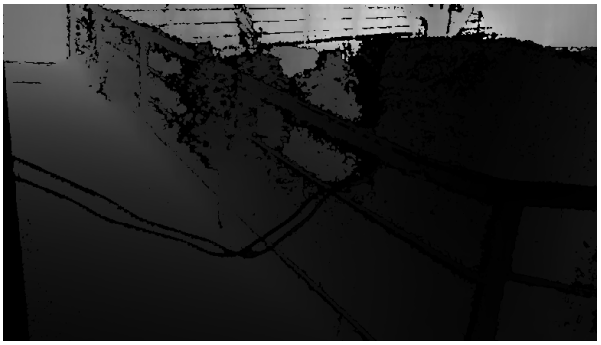
Figure 4.2: Depth images, a) and b) were captured with a kinect camera, the size of the image is VGA (640×480) and c) is the corresponding scene to depth images.

from RGB-D and stereo camera. As it is mentioned, this work obtains a map and a system localization in outdoor environments. Below, there are two subsections showing the depth images in different scenes and changing the image resolution (VGA, WVGA, 1280×720 and 1920×1080).

(a) Depth image from stereo camera (WVGA resolution)



(b) Depth image using ZED camera (HD720)



(c) Depth image from ZED camera (HD1080)

Figure 4.3: a), b) and c) images were captured from a ZED camera, compared to Kinect, the stereo camera is able to compute the depth map in open-air environments.

4.2.1 Outdoor Environment

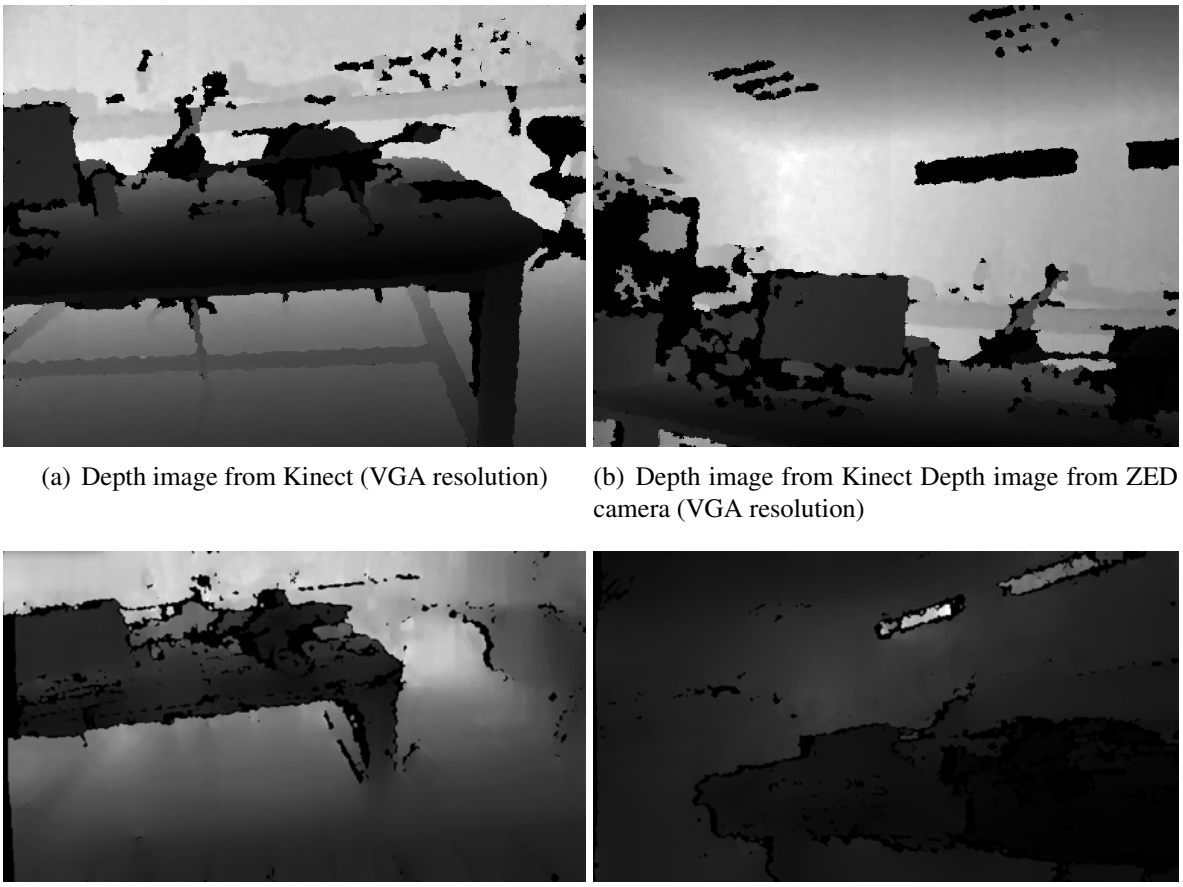
The cameras were tested outside a CIO building. a) and b) from Figure 4.2 show the depth image using a Kinect camera, as is appreciable, the image is almost entirely dark, to wit, the computed depth is zero, since the IR sensor is unable to obtain information from the IR projector. The sun emits infrared rays which alter the pattern projected by Kinect. Therefore, the camera cannot correctly compute a depth map.

The stereo cameras have an advantage respect to RGB-D cameras; these do not depend on an infrared pattern. This type of cameras is based on visual algorithms as Semi-Global Block Matching (SGBM) [71] and others to obtain the disparity and later, the depth. The Figure 4.3 shows three images with various types of resolutions, the a) image has a WVGA resolution, b) has a HD720 resolution and the size of the c) image is HD1080. The depth quality depends on the image resolution, whereas the image quality is bigger, the computed depth has more detail and the detected distance is greater.

A conclusion related to the Figure 4.2 and Figure 4.3 is notable, in outdoor environments the RGB-D camera cannot compute the depth unless there are no direct sun rays, on the other hand, the stereo camera is able to compute it and besides that it can do it using different image resolution.

4.2.2 Indoor Environment

The results obtained from Kinect and ZED camera are shown in the Figure 4.4, where a) and b) are obtained from Kinect camera, c) and b) by ZED camera, the difference of depth image quality between RGB-D and stereo camera is pretty clear. a) and b) have more defined detail than c) and d). However, the variety of image resolutions in stereo camera allows to improve the depth map, since the pixels size is smaller, as a result of this, the scenes have more detail and depth is better. It is shown in Figure 4.5, where a) and b) are captured from a stereo camera, the image resolution of these is HD720, c) and d) are also captured using the same camera but the camera resolution is HD1080. All images from Figures 4.4 and 4.5 were captured inside of “Perception and Robotics LAB”.



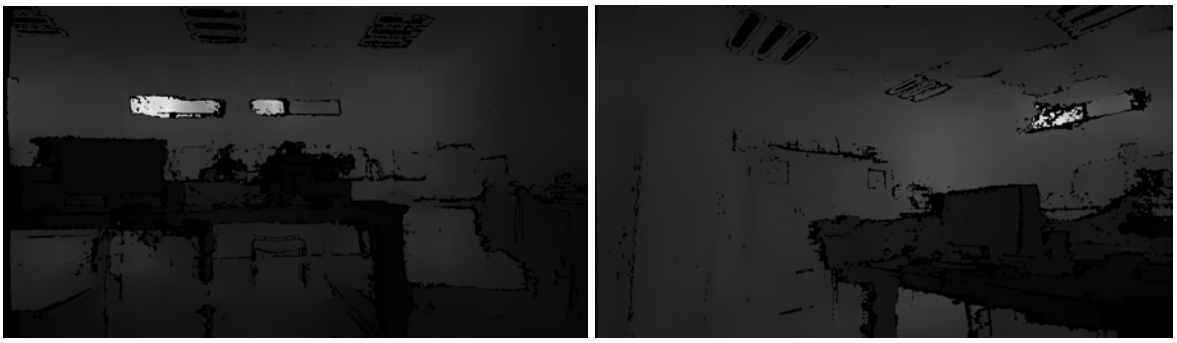
(a) Depth image from Kinect (VGA resolution)

(b) Depth image from Kinect Depth image from ZED camera (VGA resolution)

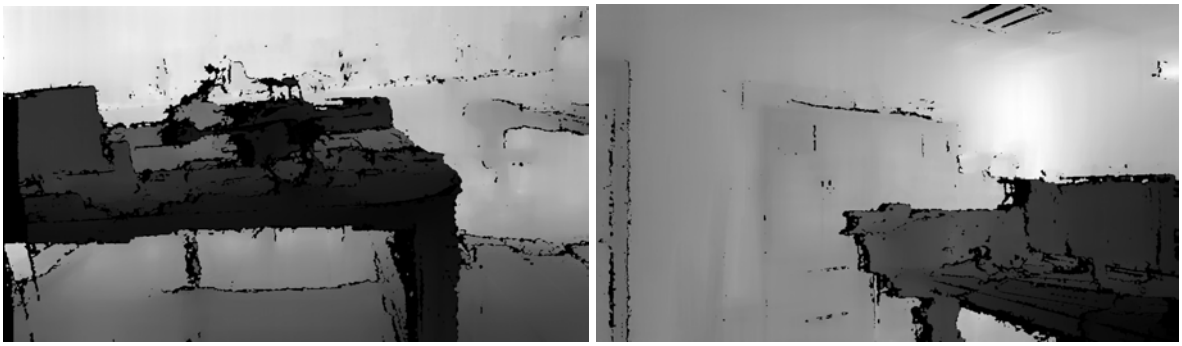
(c) Depth map from ZED camera (WVGA resolution)

(d) Depth map from ZED camera (WVGA resolution)

Figure 4.4: Images were captured inside a building, the first column corresponds to a similar perspective, in the second column the pictures capture a corner inside the room, a) and b) images were obtained from a RGB-D camera, c) and d) from a stereo camera.



(a) Depth map from ZED camera (HD720 resolution) (b) Depth map from ZED camera (HD720 resolution)



(c) Depth map from ZED camera (HD1080 resolution) (d) Depth map from ZED camera (HD1080 resolution)

Figure 4.5: Images captured from ZED camera, the first row corresponds to HD720 resolution, in the second one, the image quality is HD1080.

4.3 3D reconstruction

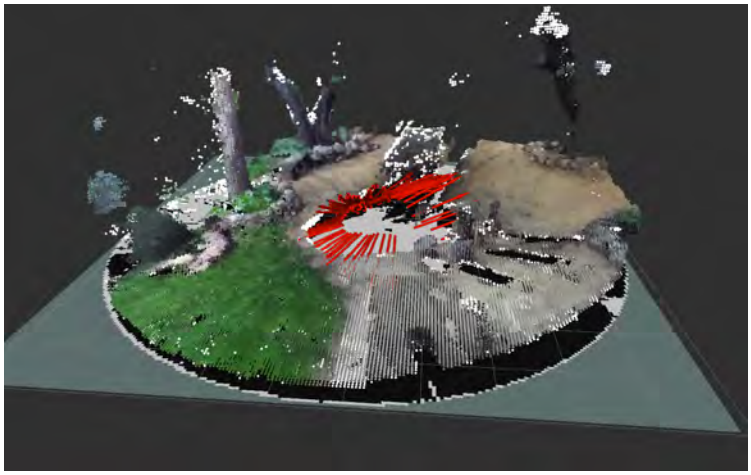
This section shows the corresponding results to SLAM, where a point cloud represents the mapping and red arrows mean the system location. The system generates a point cloud with a certain radius from system origin. Slam was tested in three diverse scenes. Each one has different type of illumination, either from sun, artificial light source or indirect light from sun.

The mapping and localization are visualized in RViz (3D visualization tool for ROS), this allows to modify the type of point cloud and the voxels size. Also, the representation of system trajectory can change between arrows and axes, the results are showed using arrows, on the other hand, some point clouds have different sizes and types of voxels. Below are shown the results in outdoor and indoor environments. Besides, the results were carried out in dynamic environments, since the semantic segment neural network detects: airplanes, bicycles, boats, bottles, buses, cars, cats, chairs, cows, dining tables, dogs, horses, motorbikes, persons, potted plants, sheep, trains and TVs.

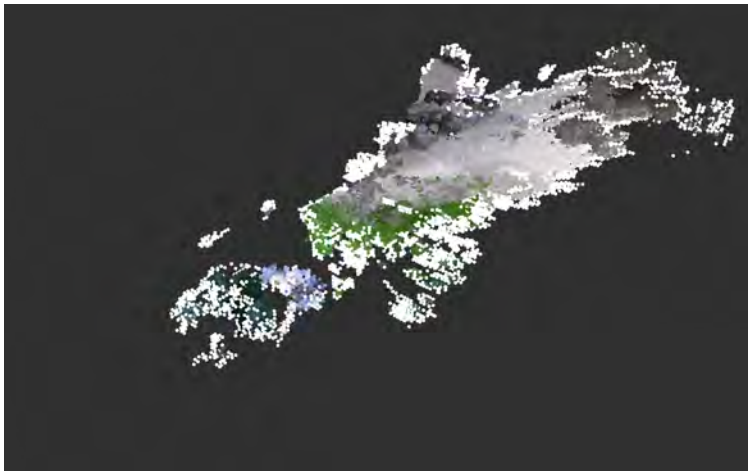
4.3.1 Outdoor Environment

The Figure 4.6 shows three images, where a) is a point cloud generated by SLAM using the stereo camera, b) point cloud was created using the Kinect and c) is the scene, in this image there is a person who was walking in the scene. As it is shown, in a) and b) the dynamic object is not visible, it is because the object is detected, therefore the SLAM does not map it. However, the map is constantly updated, as a result of this, if the system visualize the same position where the person was, the map is updated filling in the gaps.

On the other hand, the 3D reconstructions of the Figure 4.7 were also built in dynamic scenes, where a person was walking. In this case, there are only two images, since SLAM was not able to generate a correct map. So, a) is a point cloud generated by the system using a stereo camera, b) shows a person, and the system recognizes it as dynamic object, and therefore, it is not mapped, the person was walking in the right part of a).



(a) Point cloud using stereo camera



(b) Point cloud using the Kinect camera

(c) RGB image of the scene

Figure 4.6: 3D reconstructions in outdoor environment with dynamic objects.

The last reconstruction in Figure 4.8 in an outdoor environment was built in the same environment that the Figure 4.7, in this reconstruction, the dynamic objects were an airplane and a person. As it is appreciable, in the a) the objects were not mapped again. On the other hand, b) corresponds to a map by Kinect, however, there are not point cloud, it reaffirms the last section, where the depth maps computed by RGB-D sensor do not have information about the outdoor environment. Finally, c) shows the dynamic objects.

4.3.2 Indoor Environment

The last reconstructions were done in an indoor environment. The Figure 4.9 depicts a reconstruction outside of the *Perception and Robotics LAB*, a) is a 3D map generates using the ZED stereo, there are areas where the reconstruction cannot be computed due to stereo camera is not giving data in that specific zones. b) shows the scene and two dynamic objects. In this case, the Kinect sensor cannot compute the reconstruction, since there are areas with high intensity of light. However, the reconstruction of the Figure 4.10 is generated rotating the camera in the same environment. In this instance, the Kinect camera reconstructs the scene, b) shows the 3D map using the RGB-D camera. On the other side, a) depicts the map with the stereo approach. Both images present disperse points which are wrong plotted due to presence of glass door.

On the other hand, the Figure 4.11 depicts two reconstructions, the first one is a map built by the system using the ZED camera and the second one presents the 3D reconstruction using the Kinect sensor, finally the last image shows the dynamic objects which are not mapped.

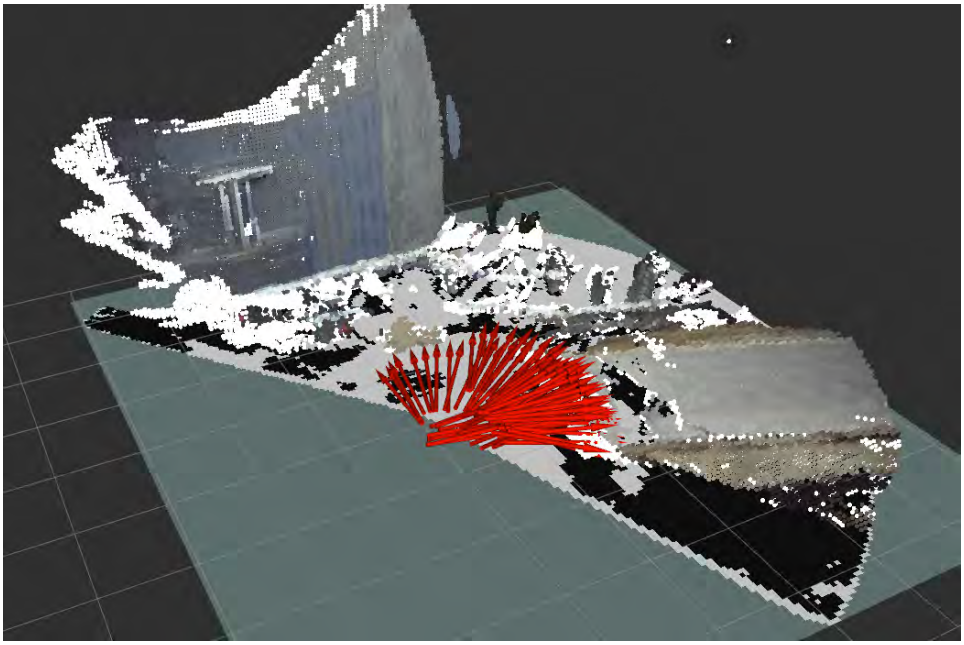
4.4 Process Rate

Finally, this section shows the rate of the process execution. The Table 4.1 depicts the average rate obtains during the execution of Kinect sensor and ZED camera, in the case of stereo camera, the rate was obtained with two camera resolutions. As it is observable, the Kinect has better average rate than stereo camera due to the internal processes of the SLAM. In the stereo configuration, the processes of the system are a little longer than the processes

Table 4.1: Average rate.

	Kinect camera (640×480)	**ZED camera** (672×376)	**ZED camera** (1280×720)
Average rate (hz)	12.58	8.15	5.66

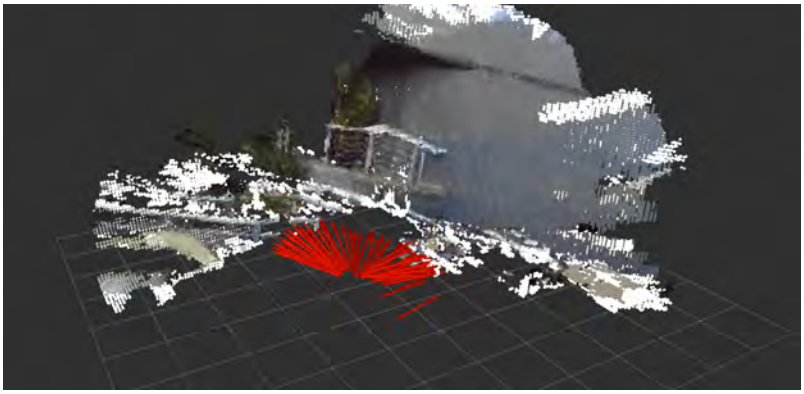
for RGB-D configuration, therefore, the average rate for VGA resolution is bigger for the Kinect camera than the ZED camera. Moreover, the HD720 resolution is slower than VGA resolution because the first resolution has more pixels and then the system has more data to process.



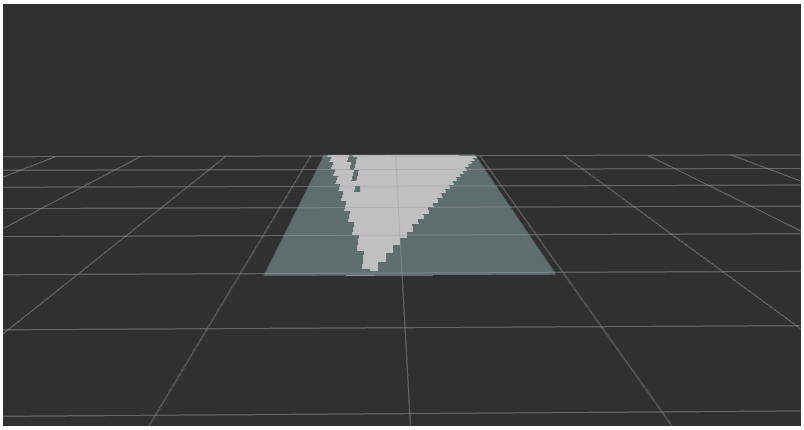
(a) High illumination from sun. 3D reconstruction using a ZED camera.

(b) RGB image with dynamic object

Figure 4.7: The top image is a 3D reconstruction built using a stereo camera, the bottom image shows a person representing the dynamic object.



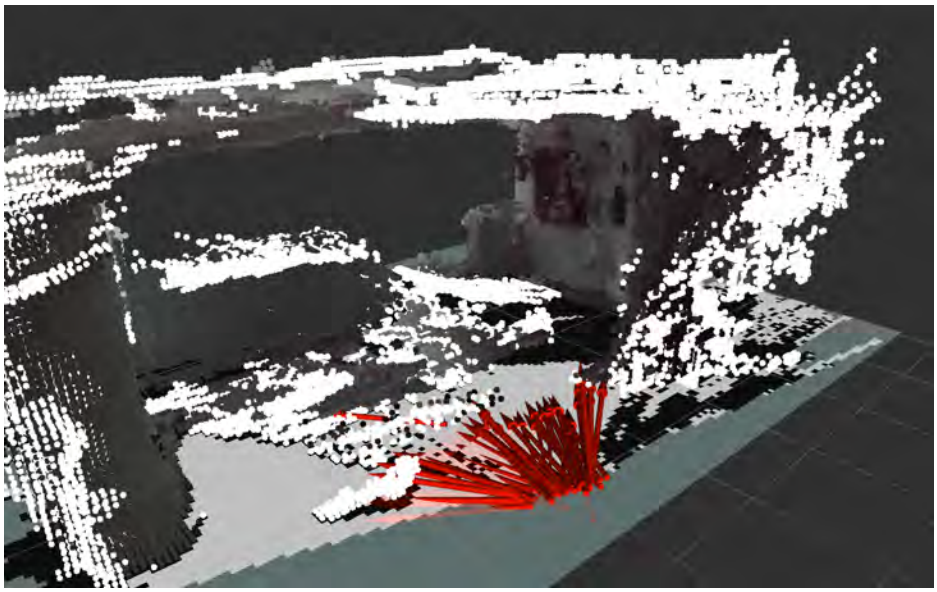
(a) 3D reconstruction using a ZED camera



(b) 3D reconstruction using Kinect camera

(c) Dynamic objects: Airplane and person

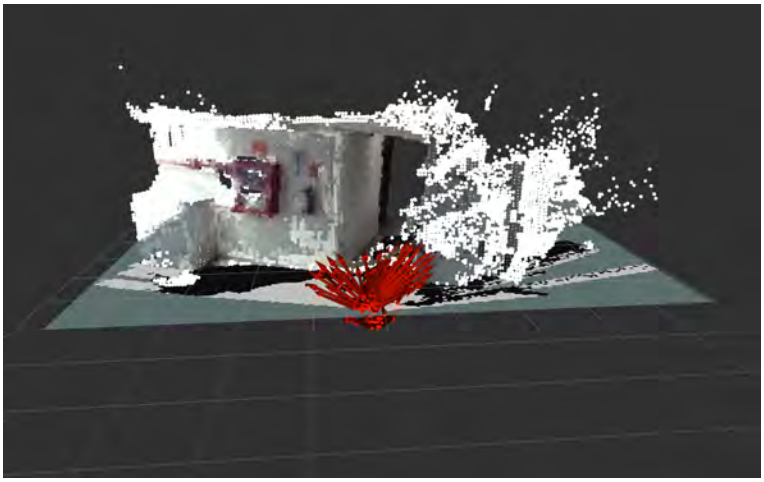
Figure 4.8: 3D reconstructions. The top image is built using a stereo camera, the middle image is generated using a Kinect sensor, the bottom image shows the dynamic objects which were in the scene whereas it was built.



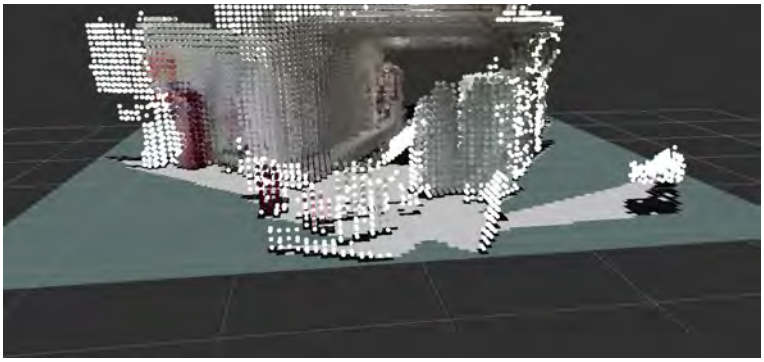
(a) Reconstruction from stereo camera.

(b) RGB image with dynamic objects.

Figure 4.9: Indoor environment with high illumination from sun.



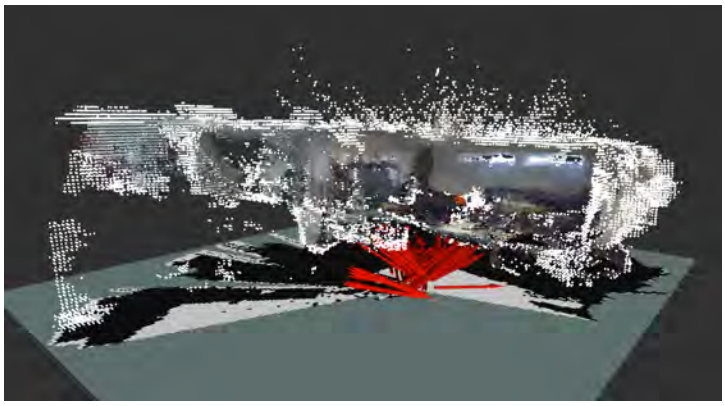
(a) 3D reconstruction using stereo camera.



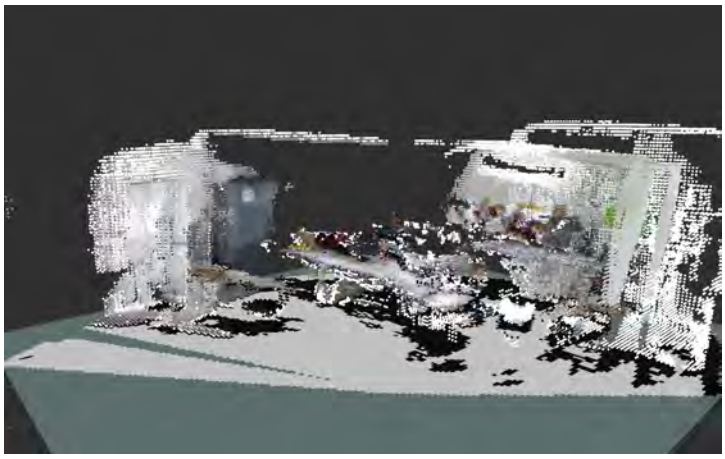
(b) 3D reconstruction using RGB-D camera.

(c) Scene with dynamic object.

Figure 4.10: Hallway reconstruction.



(a) Map using ZED camera.



(b) 3D Reconstruction from Kinect camera.

(c) Dynamic objects. Inside of the Perception and Robotics LAB

Figure 4.11: 3D reconstruction, indoor environment. a) and b) are 3D reconstructions using stereo and RGB-D camera respectively. c) is the scene where the maps were generate.

CHAPTER 5

Conclusions

This work presents a SLAM system for outdoor environments where the dynamic objects are present. The experiments related to depth map images demonstrate the RGB-D cameras compute a depth map better than stereo cameras for indoor environments. However it is possible to obtain better quality depth images from stereo cameras, it depends on the camera resolution. In other words, RGB-D cameras are better inside of the buildings or another type of scenes where there are no direct rays from the sun or the infrared rays do not cause interference with the infrared sensor and the stereo cameras are able to obtain a depth map in almost any scenes.

On the other side, the results of the SLAM in outdoor environments show the incapability for computing a 3D reconstruction of the scene using a RGB-D camera. Nevertheless, the reconstruction in an indoor environment is really good. On the other hand, the stereo approach for SLAM demonstrate good reconstruction in outdoor and indoor environments, it allows the usage of the system in a great variety of scenes. However, in environments where there are a lot of small objects, the 3D map is not able to reconstruct them in great detail. he system capability for computing a reconstruction and localization in real-time, however, it

also depends on the processing power of the computer. Finally, in every 3D reconstruction, the dynamic objects are not presents, it allows a trustworthy map.

In conclusion, the system is able to be executed in different environments using a stereo camera in real-time. Moreover, the dynamic objects are detected whereas the system is running and they are not plotted in the point cloud.

Bibliography

[1] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A Flexible and Scalable SLAM System with Full 3D Motion Estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[4] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct 2011, pp. 127–136.

[5] R. Mur-Artal and J. D. Tards, "ORB-SLAM2: An open-source SLAM System for Monocular, Stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct 2017.

[6] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017.

[7] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1168–1174.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[9] "About ros." [Online]. Available: https://www.ros.org/about-ros/

[10] "Technical description of kinect calibration." [Online]. Available: http://wiki.ros.org/kinect_calibration/technical

[11] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[12] T. Bailey and H. Durrant-Whyte, "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art."

[13] W. Nie, Q. Li, G. Zhong, and H. Deng, "Indoor 3D path planning using a Kinect V2 sensor," in *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, Oct 2017, pp. 527–531.

[14] Y. Cheng, J. Bai, and C. Xiu, "Improved rgb-d vision slam algorithm for mobile robot," in *2017 29th Chinese Control And Decision Conference (CCDC)*, May 2017, pp. 5419–5423.

[15] Yusuke Misono, Yoshitaka Goto, Yuki Tarutoko, Kazuyuki Kobayashi, and Kajiro Watanabe, "Development of laser rangefinder-based slam algorithm for mobile robot navigation," in *SICE Annual Conference 2007*, Sep. 2007, pp. 392–396.

[16] G. Xin, X. Zhang, Xi Wang, and Jin Song, "A rgbd slam algorithm combining orb with prosac for indoor mobile robot," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 01, Dec 2015, pp. 71–74.

[17] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec 2016.

[18] "Sick sensor intelligence." [Online]. Available: https://www.sick.com/es/es/

[19] "Velodyne lidar." [Online]. Available: https://www.ros.org/about-ros/

[20] "Intel realsense technology." [Online]. Available: https://www.intelrealsense.com/stereo-depth/

[21] "Tara - usb 3.0 stereo vision camera." [Online]. Available: https://www.e-consystems.com/3D-USB-stereo-camera.asp

[22] "Xtion pro live." [Online]. Available: http://xtionprolive.com/

[23] "inivation ag." [Online]. Available: https://inivation.com/

[24] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, pp. 1985–1991 vol.2.

[25] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi, "Fast and accurate slam with rao-blackwellized particle filters," *Robotics and Autonomous Systems*, vol. 55, pp. 30–38, 2007.

[26] C. D. C. Lerma and J. Neira, "Slam in o(log n) with the combined kalman - information filter," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2069–2076, 2009.

[27] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, 2009.

[28] Wei Tan, Haomin Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular slam in dynamic environments," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Oct 2013, pp. 209–218.

[29] M. R. U. Saputra, A. Markham, and A. Trigoni, "Visual slam and structure from motion in dynamic environments: A survey," *ACM Comput. Surv.*, vol. 51, pp. 37:1–37:36, 2018.

[30] S. Wangsiripitak and D. W. Murray, "Avoiding moving outliers in visual slam by tracking moving objects," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 375–380.

[31] F. Chhaya, D. Reddy, S. Upadhyay, V. Chari, M. Z. Zia, and K. M. Krishna, "Monocular reconstruction of vehicles: Combining slam with shape priors," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5758–5765.

[32] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[33] M. Piccardi, "Background subtraction techniques: a review," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, vol. 4, Oct 2004, pp. 3099–3104 vol.4.

[34] A. Kundu, K. M. Krishna, and J. Sivaswamy, "Moving object detection by multi-view geometric techniques from a single camera mounted robot," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 4306–4312.

[35] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti, "Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments," in *ICRA 2009*, 2009.

[36] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 354–366, 2013.

[37] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1980.

[38] P. F. Alcantarilla, J. J. Yebes, J. Almazn, and L. M. Bergasa, "On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 1290–1297.

[39] J. Cheng, Y. Sun, W. Chi, C. Wang, H. Cheng, and M. Q. . Meng, "An accurate localization scheme for mobile robots using optical flow in dynamic environments," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2018, pp. 723–728.

[40] D. Scaramuzza, "1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *International Journal of Computer Vision*, vol. 95, pp. 74–85, 2011.

[41] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 4293–4299.

[42] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with

convolutional networks," *CoRR*, vol. abs/1504.06852, 2015. [Online]. Available: http://arxiv.org/abs/1504.06852

[43] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2017/IMKDB17

[44] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2016, pp. 4040–4048.

[45] T. Lin and C. Wang, "Deep learning of spatio-temporal features with geometric-based moving point detection for motion segmentation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3058–3065.

[46] P. H. S. Torr, "Geometric motion segmentation and model selection," *Phil. Trans. Royal Society of London A*, vol. 356, pp. 1321–1340, 1998.

[47] K. Schindler and D. Suter, "Two-view multibody structure-and-motion with outliers through model selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 983–995, June 2006.

[48] N. Thakoor, J. Gao, and V. Devarajan, "Multibody structure-and-motion segmentation by branch-and-bound model selection," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1393–1402, June 2010.

[49] K. Schindler, D. Suter, and H. Wang, "A model-selection framework for multibody structure-and-motion of image sequences," *International Journal of Computer Vision*, vol. 79, no. 2, pp. 159–177, Aug 2008. [Online]. Available: https://doi.org/10.1007/s11263-007-0111-7

[50] J. P. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *International Journal of Computer Vision*, vol. 29, no. 3, pp. 159–179, Sep 1998. [Online]. Available: https://doi.org/10.1023/A:1008000628999

[51] C. Gear, "Multibody grouping from motion images," *International Journal of Computer Vision*, vol. 29, no. 2, pp. 133–150, Aug 1998. [Online]. Available: https://doi.org/10.1023/A:1008026310903

[52] R. Vidal and R. Hartley, "Three-view multibody structure from motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 214–227, Feb 2008.

[53] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "Sfm-net: Learning of structure and motion from video," *ArXiv*, vol. abs/1704.07804, 2017.

[54] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, pp. 105–19, 01 2010.

[55] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792.

[56] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[57] D. Galvez-Lpez and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, Oct 2012.

[58] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, pp. 155–166, 2008.

[59] R. Mur-Artal and J. D. Tards, "Fast relocalisation and loop closing in keyframe-based slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 846–853.

[60] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Workshop on Vision Algorithms*, 1999.

[61] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual slam: Why filter?" *Image Vision Comput.*, vol. 30, pp. 65–77, 2012.

[62] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3607–3613.

[63] Wei Tan, Haomin Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular slam in dynamic environments," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Oct 2013, pp. 209–218.

[64] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr 1987. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629

[65] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov 2015, pp. 730–734.

[66] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[67] "Raw message definition of sensor_msgs/image." [Online]. Available: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Image.html

[68] “Meet ZED.” [Online]. Available: https://www.stereolabs.com/zed/

[69] “Zed ros wrapper.” [Online]. Available: http://wiki.ros.org/zed-ros-wrapper

[70] “What is camera calibration?” [Online]. Available: https://la.mathworks.com/help/vision/ug/camera-calibration.html

[71] “Stereo correspondance algorithms.” [Online]. Available: https://docs.opencv.org/3.4.2/d1/d9f/classcv_1_1stereo_1_1StereoBinarySGBM.html