A HYBRID ALGORITHM FOR FEATURE-BASED APPROACH IN VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING

By

Wondimagegn Zewudu Wakjira

A Thesis submitted to the Department of Computer Science and Engineering,

School of Electrical Engineering and Computing

Office of Graduate Studies

Adama science and technology University

July 2020

Adama Ethiopia

A HYBRID ALGORITHM FOR FEATURE-BASED APPROACH IN VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING

By

Wondimagegn Zewudu Wakjira

Advisor: Prof.Yun Koo Chung

A Thesis submitted to the Department of Computer Science and Engineering,

School of Electrical Engineering and Computing

Office of Graduate Studies

Adama science and technology University

July 2020

Adama Ethiopia

Approval Page

We, the undersigned, members of the Board of Examiners of the final open defense by "Wondimagegn Zewudu" have read and evaluated his thesis entitled "a hybrid algorithm for feature-based approach in Visual Simulation Localization and Mapping" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the degree.

Name of student	Signature	Date
Supervisor /Advisor	Signature	Date
Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date
Head of Department	Signature	Date
School Dean	Signature	Date
Post graduate Dean	Signature	Date

Acknowledgment

I would like to thank almighty God whose everlasting and undying love kept me a life throughout my thesis work. Next I would like to thank my adviser Prof.Yun Koo Chung for his encouragement and support during the completion of this thesis. I am grateful for his willingness to invest in all the equipment I required. And above all, I am thankful for his belief in me and my research idea. He not only provided me with a robotic development kit but also assisted me with the process of computer vision for this thesis. He was always ready to help and encourage.

I would also like to thanks Dr. Mesfin Abebe for his kindly approach to help me in technical support and for his brilliant comments and suggestions. I am thankful also for his aspiring guidance, invaluably constructive criticism and friendly advice during the project work.

Table of Contents

Abstract xvii
CHAPTER 1 1
Introduction 1
1.1.Background of the study 1
1.2.Motivation 3
1.3 Statement of the Problem 4
1.4 Objective of the study 5
1.4.1.General Objective 5
1.4.2. Specific Objectives 5
1.5. Scope and Limitation 5
1.6. Contribution 6
1.7. Organization of the Thesis 6
CHAPTER 2 8
Literature Review and Related Work 8
2.1. Robot Navigation 8
2.1.1.SLAM (Simultaneous Localization and Mapping) 8
2.1.1.1. Landmark Extraction 9
2.1.2 VSLAM (Visual Simultaneous Localization and Mapping) 9
2.1.2.1 VSLAM Basic modules 10
2.1.2.2 Feature-based approach 12
2.1.2.3 Feature-based methods 12
2.1.2.4 MonoSLAM 12
2.1.2.5 PTAM 14
2.1.2.6 ORB-SLAM 15
2.2 Minutiae extraction 16
2.3.Related Work 18
Chapter Three 23
Methodology of the Study 23
3.1 Overview 23
3.2. Development Tools 23
3.2.1 Design Tools 23
3.2.2 Simulator Selection 23
3.2.2.1. ROS 26
3.2.2.2 ROS Architecture 27
3.2.2.3 ROS Tools 28
3.2.2.4.Data visualization Tool(RVIZ) 28

3.2.2.5. ROS GUI Development Tool i(rqt) 29
3.2.2.6.Gazebo 29
3.3. Data collection method 29
3.3.1. Dataset content and sampling techniques 29
3.3.2.Training and testing techniques 31
3.4. Proposed System Implementation Method 31
3.5 Evaluation Method 33
Chapter Four 35
Proposed Solution 35
4.1.Overview of the Proposed Solution 35
4.2 Computer Vision Pipeline 35
4.2.1.Detection Step 37
4.3. Controller Algorithm 41
4.4. Deployment model 41
Chapter Five 43
Implementation of the Prototype 43
5.1. Prototype Setup 44
5.2. Extraction Step 44
5.3. Map Building 47
5.4 Obstacle Avoidance 48
5.5. Mapping Graph 50
Chapter six 52
6.1 Key Point extraction and formation with core value 52
6.1.1.In the case of high Intensity 56
6.2. Computer vision and map formation 57
6.3. Computer vision with ROS and Test Bed 58
Chapter Seven Conclusion and Future work 61
7.1. Conclusion 61
7.2.Future work 61

Table of figures

Figure1. Component of the navigational system 2
Figure2. 1 Overview of the SLAM process 9
Figure2.2 Summary of feature-based methods 15
Figure2.3 various types of minutiae 17
Figure3. 1 multiple language supporter by meta operating system ROS 26
Figure3. 2 ROS architecture with their respective section and level concept 27
Figure3. 3 Over view of RVIZ simulator 28
Figure.4.1. ORB-SLAM 36
Figure4.2 simple steps of minutiae extraction 37
Figure4.3 general sachema move_bas path planner 40
Figure4.4 Message Communication Architecture 42
Figure5. 1 Turtlebot3 44
Figure5. 2 video changed to frame by frame 44
Figure5.3 Examples of a ridge ending and bifurcation pixel so all the basic steps are : 46
Figure5.4 Node graph of ros 51
Figure6.1 the snip shoot of implementation of training image frame 52
Figure6.2 the snip shoot of another implementation of training image frame 53
Figure6. 2Two original frame of two different video 53
Figure6. 3 The process of resize and Conversion to gray scale of test frame 54
Figure6. 4 extraction using minutiae points (Key points) from consecutive frames 54
Figure6. 5 extraction using ORB slam (Key points) 55
Figure6. 6 key points counter 56
Figure6. 7 In different light of intensity(on high sun light around my home) 56
Figure6. 8 from combined key point creating estimated map 57
Figure6.9 SIG test bed 58
Figure6.10 ROS sample test bed on gazebo 58
Figure6.11 sample test bed 59
Figure6.12 rough map output with combinig LADAR in ROS with gmaping node 59

List of tables

Table 3.1 Characteristics of Open-Source Robotic Simulation Software 24
Table 6.1 key point /matches 56
Table 6.2 intensity measurement 57

Acronyms

Abbreviation	Expanded form
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust invariant scalable key points
DOG	Difference of Gaussian
EKF	Extended Kalman Filter
FAST	Feature from accelerated segment test
FLANN	Fast Library for approximate nearest neighbors
IR	Infrared radiation
LDS	Laser distance sensor
LOG	Laplacian of Gaussian
LIDAR	Light Detection and ranging
OCR	Optical character recognition
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random sample consensus
RFID	Radio frequency identification
ROS	Robot operating System
R-CNN	Region Classification neural network
SIFT	Scale invariant transform
SURF	Speeded up Robust features
SLAM	Simultaneous Localization and Mapping
SVM	Support vector machine
VSLAM	visual Simultaneous Localization and Mapping

Abstract

Robot navigation means the robot's ability to determine its own position in its frame of reference (map) and then to plan a path towards some goal location by avoiding dangerous situations such as collisions and unsafe conditions. The use of obstacle avoidance for the navigation of the robot has been increasing due to its importance in many sectors such as for autonomous cars, for military surveillance robots, for indoor surveillance robots, and for assisting visual impaired people with robot. Therefore, the ability to observe and understand the meaning of the robot is an essential part of navigating the robot safe and reaching the desired end without any difficulty. Therefore, integrating fast vision algorithm that can enable the robot to understand its environment and avoid obstacle make the robot to navigate safely within an unknown indoor environment. This study enables the robot to detect and recognize a map with high accuracy and enable the robot to avoid any damage. But there are a small robot who only got one camera, those robots need an accurate map and avoid collision this thesis shows solving this problem only using a hybrid algorithm for feature-based approach.

For micro-robot which only got one camera to discover the environment so such a future need to be applied. Since navigating the robot through an unfamiliar environment requires an understanding of details and other useful visual information, the robot has to have a method for detecting recognizing the environment as accurately as possible .so our work focus on this behave.

1) It needs to model and understand its environment, with much keyframe points along with its attendant uncertainties, as well as find its own location within the environment, using only the noisy observations from its monocular camera so we propose hybrid algorism for VSLAM based on monocular camera and the algorithms are Minutiae extraction, commonly used in fingerprint recognition and ORB-VSLAM one of the common used VSLAM algorism

2) The proposed algorithm got tested bygone through some identified simulation tools with well-known data and local data then finally got the appreciable results this thesis paper explains this in detail.

Keywords: VSLAM algorism, ORB-VSLAM, Monocular camera, Minutiae extraction, Hybrid algorism.

CHAPTER 1

Introduction

1.1. Background of the tudy

Existing robotic systems are already able to perform various services such as delivery, education, providing tele-presence, cleaning, or entertainment. Likewise, there are prototypes of autonomous wheelchairs and intelligent assistance robots which are designed to assist people in their homes. So generally, with the development of mobile robots, it is expected that robots and humans can coexist in the same environment, wherein they can provide support and assistance for healthy and other services. We can bring the transport system as example.

The design and functioning of the road transport system is intended to ensure socioeconomically efficient and sustainable transportation. Normally the drivers need to be very vigilant and cautious to identify collusion at the right time and predict the situation. Therefore, predictive techniques are being developed to maximize transportation safety and efficiency by introducing fast obstacle avoidance system in which the traffic detection and recognition holds the most important position. National Highway Traffic Safety Administration (NHTSA), the main cause (94%) of vehicle crashes is human error [1] [2]. Among all possible types of driver fault, recognition errors, decision mistake, and performance errors are the most frequent driver-related critical reasons for accidents. Based on this investigation, we deduce that there should be a strong motivation to develop and implement technologies that alleviate and avoid accidents.

Currently in Ethiopia having a growth rate of 2.1% the population of Addis Ababa constitutes 32.27 % of the total urban population of the country [3]. In year 2012 the total population of Addis Ababa reaches 4 million and expected to be reaching 6million and 10million in the year 2025 and 2040 respectively. And the city has an area of 540 km2 [3]. Thus the increase in population size coupled with the economic growth and increase in the size of the city demands a transport service supply in line with the increase in the mobility need of the people. The poor road infrastructure of the city, coupled with inadequate and inefficient transport activities, the erratic behavior of drivers and poor usage and identification of traffic details complicate Addis Ababa traffic problems [3]. Yet among the various reasons of major traffic problems in the city the weak traffic environment identification and usage are identified as a major one by different scholar studies in the area [3]. This motivates us to study highly optimized computer vision Algorithms that can inform the computer to identify environment with high speed and which can take action for the user accordingly.

Object recognition in real scenes is one of the most challenging problems in computer vision, as it is necessary to deal with complications, such as viewpoint changes, occlusions, illumination variations, background clutter or sensor noise [7]. Because of this, there have been many methods for object recognition cultivated over the last few decades. Most of them concentrate on specific cases, like faces or pedestrians, with different techniques based on feature descriptors, shape descriptors, gradient-based, derivative-based, template matching, etc. Almost all object recognition systems can be split into two successive phases: the offline phase, including the generation of the model, and the online phase, in which the constructed model is used to find the object in the search image. Thus, only the computation time of the online phase is critical considering the real-time requirement

In order to deal with these issues, the solution uses the interaction among several components as a platform to capture and process the user and environment information, and to generate and deliver navigation messages to users or robot while they are moving in an indoor area or outdoor area. The system's main components are the following: camera with 8 mega pixel from that make some features for all tasks: tracking, mapping, relocalization and loop closing. This makes our system more efficient, simple and reliable. We use ORBSLAM2 which allow real-time performance without GPUs with addition of minute extraction, providing good invariance to changes in viewpoint and illumination

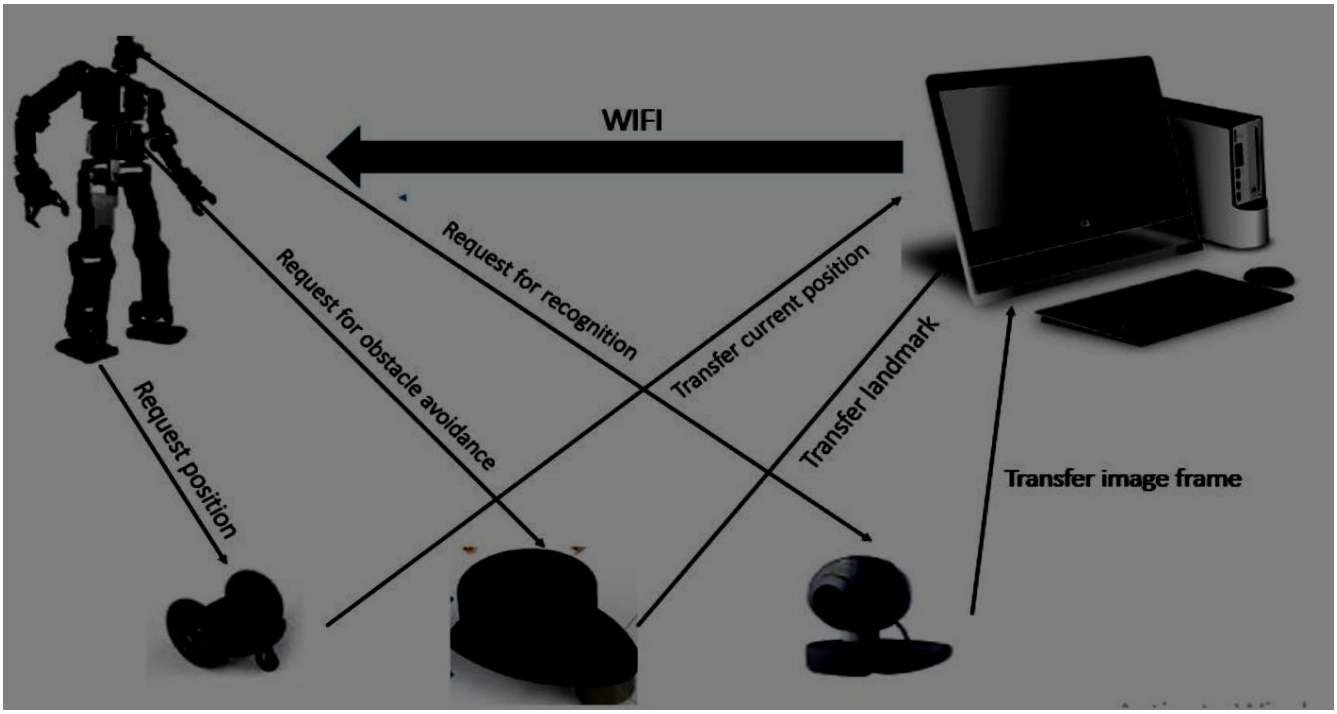


Figure1. Component of the navigational system [10]

When the robot request navigation information, it activates the camera to continually taking up landmark in order to build 3D reference called map. At the same time the camera mounted on the

robot start to continually to record video to make able to transfer the image frame by frame for recognition. The software application instantly tries to determine the user's position, the presence of obstacles in the surrounding area and whether obstacle is existing in front of the robot. The devices transmit that information via Wireless modem to a software application running on the computer. The application then uses such information and also the data that it has about the obstacles in the area, to generate single navigation messages that are delivered through WIFI. The system follows a common-sense approach for the message delivery.

In our case In general, the technical difficulty of VSLAM is higher than that of other sensor-based SLAMs because cameras can acquire less visual input from a limited field of views compared to 360° laser sensing which is typically used in robotics. From such input, camera poses need to be continuously estimated and the 3D structure of an unknown environment is simultaneously reconstructed. The early work of VSLAM using a monocular camera was based on tracking and mapping feature points in 2000s. This is called "feature-based approach."

1.2. Motivation

Autonomous robots which work without human operators are required in robotic fields. Nowadays, mobile service robots in indoor environments such as houses, offices, hospitals, are introduced widely. These mobile robots have to be equipped with a capability to navigate in dynamic environments to execute a given task while avoiding obstacles and recognizing environment. A number of sensors such as laser finders, ultrasonic sensors, stereo-camera-based range sensors, and etc – are used widely in order to detect obstacles in natural environments and various detection and recognition algorithm are used to detect and recognize map. However, most of these sensors are too expensive to apply for low-cost service robots like vacuum cleaning robots or guide robots. Hence, visual navigation took much attention after web cameras were introduced a few years ago since their cost is attractive as compared to the previous sensors they are also small to install. In order to achieve tasks, autonomous robots have to be intelligent and should decide their own depending on their tasks. More, it is necessary to recognize map and to plan a collision free path minimizing a cost such as time, energy and distance. When an autonomous robot moves from a point to a target point in its given environment, it is necessary to plan an optimal or feasible path avoiding obstacles in its way and necessary to detect and recognize environment.

1.3 Statement of the Problem

Environment detection and recognition in real scenes is one of the most challenging problems in computer vision, as it is necessary to deal with difficulties, such as viewpoint changes, occlusions, illumination variations and scale variation. The time required to recognize and the reliability or the quality of detection is still existing problem in computer vision [8], also most of the new robots (medium and large size) use LIDAR and other sensors because of this reason people decrease interest research to make accurate VSLAM.

To modified VSLAM and make it more accurate and additional key points the research has been done in the real time environment for indoor map recognition, there are three outmost aspects considered, to be achieved by the hybrid algorithm.

1. Efficiency: Overall System's Response Time.
2. Reliability: Accuracy in real environment map.
3. Speed: Processing Time in recognition of the environment.

So generally, the problem of existing system can be summarized as follow.

- Difficulty in detecting and recognition of environment accurately.
- Difficulty in detecting and recognition of environment with lower time
- Difficulty in detecting and recognition of environment within difficulties, such as viewpoint changes, occlusions, illumination variations and scale variation
- Difficulty in detecting obstacle from maximum distance and avoiding obstacle with less required time while navigating within indoor environment

Generally, we have the following research questions:

RQ1. How can the robot detect and recognize environment with accuracy?

RQ2. How can we improve the robustness or reliability for the detection and recognition environment through different image transformation?

RQ3. How can the robot detect and recognize environment with high speed?

1.4 Objective of the study

we propose hybrid algorism for VSLAM based on monocular camera and the algorithm are Minutiae extraction, commonly used in finger print recognition and ORB-VSLAM one of the commonly used VSLAM algorithm

1.4.1. General Objective

The general objective of the research is to develop a robot that can recognize environment with additional accuracy and less required time and that can detect and avoid by combining fast feature detection and recognition algorithm with obstacle avoidance algorithm.

1.4.2. Specific Objectives

To achieve the above aforementioned general objective, the thesis work will address the following specific objectives:

- Review previous related work in the area of image and minutiae extraction, detection and recognition for robot navigation and obstacle avoidance mechanism so as to understand the domain area and assess different algorithms for object recognition technique
- Finding algorithm that can detect and recognize object with less required time
- Designing algorithms by combining fast, selective and robust algorithm for recognition of environment - Collecting and training the image for the testing purpose
- Evaluating the recognition algorithm through different evaluation techniques
- Designing the mechanism to avoid obstacle
- Designing and building the map
- Integrating the recognition algorithm with navigation algorithm
- Designing and Building the system in which capable of detecting and recognize the map and that can detect and avoid obstacle at the same time

1.5. Scope and Limitation

Since to develop safe navigating robot requires four main points that are path planning, obstacle avoidance, localization and vision mechanism, this study will focus on image based obstacle avoidance, detection and recognition of its environment. It is focus on improving the accuracy in which the robot can recognize image and the mechanism to detect and avoid obstacle. The speed for recognition of the image within indoor environment has to be compatible with the speed of the robot this study is all about merging recognition algorithm with obstacle avoidance mechanism to make the navigation mission by the robot safer. Navigating the robot in outdoor environment can use GPS which is practically best mechanism for localization of the robot and no light variation at outdoor environment this study only focus on the indoor environment. Due to time restriction, text recognition from the given image will be researched in future.

There is some limitation which is found during the study due to different constraints like the time schedule and resource limitation such as tools. Since we are using turtlebot3 the robot requires

battery that enable to move long distance without any wire restriction but for our case turtlebot3 battery was difficult to get and the distance that the robot can cover was restricted.

1.6. Contribution

This thesis focuses on the autonomous navigation problem with special emphasis on indoor navigation in the semi-structured environment of the test area. The main contributions from this thesis are:

□□A vision-based perception of new available path through detecting and recognizing image. In this process we showed based on the type of environment the robot detect the robot make decision either to re plan a new path or to cease motion for moment.

Additionally:

- High-Speed Object Detector. In this work we suggest and implement one of the fastest object detectors available in current computer vision literature. Building on a solid foundation of integral channel features in a boosting frame work we suggest two approximation techniques for speeding up the detection process without a decrease in detection quality.
- Performance Study. Starting with our baseline detector we identify the bottleneck in the detection process and propose the implementation of two fast detection and recognition algorithm by merging together for speeding up our detector. We also show the contribution and effect of the approximation techniques to both detection quality, detection speed and detection reliability through different image transformation.

1.7. Organization of the Thesis

The organization of this thesis is as follows chapter one introduces about robot navigation and about image detection and recognition system. This chapter also contains the problems observed while navigating robot with in unfamiliar environment and the significance of the study.

Chapter two discuss the literature review including information for robot navigation techniques that previously done to navigate robots, observe some characteristics of robot navigation, over view for image detection and recognition system, state the merit of map detection and recognition, observing about obstacle detection and focusing on reviewing other related models.

Chapter three provides the methodology used to carry out the research. Chapter four deals with the proposed solution is provided in detail and the development life cycle used in the research are explained.

Chapter five gives the implementation step taken for the proposed solution in the above chapter.

Chapter six gives the result observed from the implementation step and some result is discussed.

Finally, chapter seven gives the conclusion and future works.

CHAPTER 2

Literature Review and Related Work

2.1. Robot Navigation

Peoples and researchers think robots will soon take part in everyone's daily life. In industrial production this has been the case for many years, but up to now the use of mobile robots has been limited to a few and isolated applications like lawn mowing, surveillance, agricultural production and military applications. Navigation of an autonomous robot is concerned with the ability of the robot to direct itself from the current position to a desired destination. So robot navigation is to make the robot able to reach the desired destination with ability to determine its own position in its frame of reference and then to plan a path towards some goal location or destination [9]. Therefore, in order to navigate the robot requires map information and the ability to intemperate the information. So generally navigation is the combination of three general concepts. The first one is self-localization or the localization which is the robot has to realize the current position of it and its own orientation with frame of reference. The second is the mission or path planning in which the robot from the localization stage has realizes how to reach the required destination either with same frame of reference or coordinates. The third is map building and interpretation or how do I get there in this building the map to isolate the required path that makes the robot to reach its intended destination

2.1.1. SLAM (Simultaneous Localization and Mapping)

Since the navigation of autonomous robot requires realization of the robot position along with the orientation, to build and interpret the map, navigation requires the SLAM techniques. SLAM is the method that the mobile robot builds its own map and differentiate its current position by using the built map [4]. In SLAM the trajectory of the platform and the location of all landmarks are estimated online without the need for any a prior knowledge of position. A simultaneous estimate of both robot and landmark locations is required in order to compute an estimate of robot location with respect to these landmarks [10]. However, localization problem is arises when computing the probability distribution when that much of the error between estimated and true landmark locations is common between landmarks which means errors in knowledge of where the robot is when landmark observations are made. So to overcome this problem SLAM is proposed.

So generally SLAM deal with building the map of unknown environment and at the same time navigating the robot using the built map [11]. SLAM consists of multiple parts; Landmark extraction, data association, state estimation, state update and landmark update

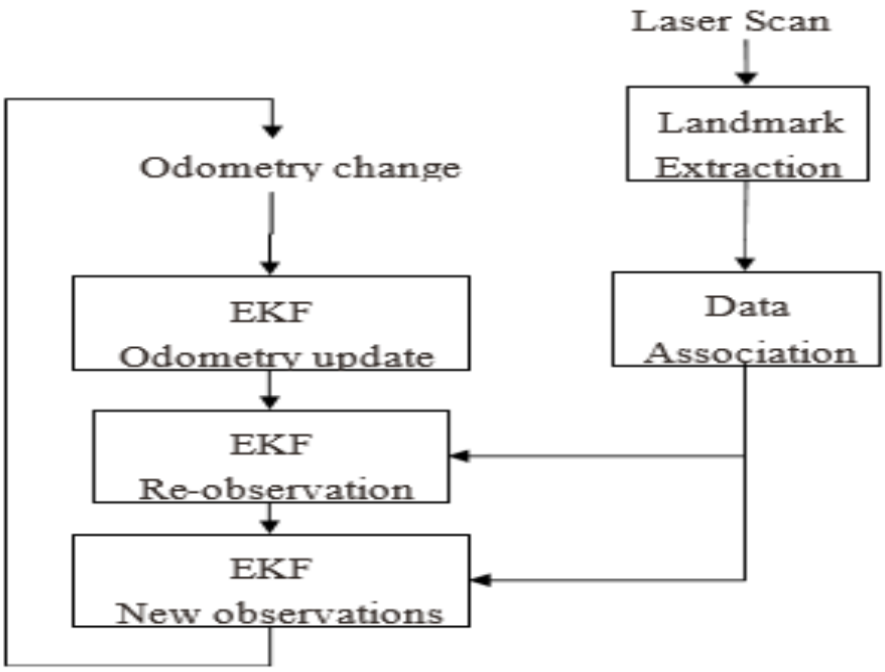


Figure2. 1 Overview of the SLAM process [20]

2.1.1.1. Landmark Extraction

Landmark is a feature which is used by the robot to find out where it is which is easily differentiated from the surroundings [2]. To say one object to be as landmark it has to fulfill the following criteria

- It has to be detected or observable from different point of view
- It has to be unique or it must have some criteria that makes it to be different from other landmark to say it as new landmark (i.e. it should be easily distinguished from other landmark)
- It should be stationary

After fulfilling those criteria, the next step is landmark extraction by using the sensory inputs. In this extraction process there are two kinds of algorithm that is used for extracting the land mark one is minute extraction and VSLAM (ORB_SLAM) [2],

2.1.2 VSLAM (Visual Simultaneous Localization and Mapping)

In recent years, SLAM using cameras only has been actively talked over the robotic world because the sensor configuration is simple and the technical difficulties are higher than others. Since the input of such SLAM is visual data only, the technique is specifically referred to as visual SLAM

(VSLAM). Simply Visual SLAM, a system that uses a camera as its data input sensor, is widely used in platforms moving in indoor environments.[11]VSLAM algorithms have widely proposed in the field of computer vision, robotics, and AR [8]. Especially, they are suitable for camera pose estimation in AR systems because the configuration of the systems can be simple such as camera-mounted tablets or smart- phones. One of the important requirements in AR systems is real-time response to seamlessly and interactively merge real and virtual objects. To achieve the response with a limited computational resource on a light-weighted hand-held device, various low computational-cost VSLAM algorithms have been proposed in different literature.[9]

The application of such VSLAM algorithms is not bordered to AR systems. For example, it is also useful for unmanned autonomous vehicles (UAV) in robotics [10]. Even though VSLAM algorithms have been advised for different purposes in different research communities, they basically share overall parts of technical core ideas and can be used to achieve different purposes each other. [9]

Vision based robot navigation uses computer vision algorithms to extract the visual information to make further processing it in order to detect the intended area from the image and to recognize it by the robot from the surrounding environment [12]. However, there are more techniques for vision based navigation which can be classified under indoor environment and outdoor environment.

2.1.2.1VSLAM Basic modules

The framework is mainly composed of three modules as follows.

1. Initialization
2. Tracking
3. Mapping
4. Relocalization
5. Global map optimization

To start VSLAM, it is essential to define a certain coordinate system for camera pose estimation and 3D reconstruction in an unidentified environment. Therefore, in the initialization, the global coordinate system have to be defined first, and a part of the environment is reconstructed as

an initial map in the global coordinate system. After the initialization, tracking and mapping are performed to continuously estimate camera poses. In the tracking, the reconstructed map is tracked in the image to estimate the camera pose of the image with respect to the map. In order to do this, 2D–3D correspondences between the image and the map are first obtained from feature matching or feature tracking in the image. Then, the camera pose is computed from the correspondences by solving the Perspective-n-Point (PnP) problem [12,13]. It should be noted that most of VSLAM algorithms assumes that intrinsic camera parameters are calibrated beforehand so that they are known. Therefore, a camera pose is normally equivalent to extrinsic camera parameters with translation and rotation of the camera in the global coordinate system. In the mapping, the map is expanded by computing the 3D structure of an environment when the camera observes unknown regions where the mapping is not performed before.[9]

The relocalization is required when the tracking is broke down due to fast camera motion or some interferences and disturbances. In this case, it is necessary to compute the camera pose with respect to the map again. Therefore, this process is called “relocalization.” If the relocalization is not incorporated into VSLAM systems, the systems do not work anymore after the tracking is lost and such systems are not practically useful. Therefore, a fast and efficient method for the relocalization has been discussed in the literature. Note that this is also referred to as kidnapped robot problems in robotics.

The other module is global map optimization. The map generally includes accumulative estimation error according to the distance of camera movement. In order to suppress the error, the global map optimization is normally performed. In this process, the map is refined by considering the consistency of whole map information. When a map is revisited such that a starting region is captured again after some camera movement, reference information that represents the accumulative error from the beginning to the present can be computed. Then, a loop constraint from the reference information is used as a constraint to suppress the error in the global optimization.

Loop closing is a technique to acquire the reference information. In the loop closing, a closed loop is first searched by matching a current image with previously acquired images. If the loop is detected, it means that the camera captures one of previously observed views. In this case, the accumulative error occurred during camera movement can be estimated. Note that the closed loop detection procedure can be done by using the same.[11]

2.1.2.2 Feature-based approach

Starting from 2003 there is algorithm by using monocular camera was based on tracking and mapping feature points and continuously estimated and the 3D structure of an unknown environment is simultaneously reconstructed.

2.1.2.3 Feature-based methods

Most of the time there exist two types of feature-based methods in most literature: filter-based and BA (Bundle adjustment)-based methods. We will explain the methods and algorism plus provide the comparison, because they can be considered as fundamental frameworks for other methods.

2.1.2.4 MonoSLAM

First monocular vSLAM was formulated in 2003 by Davison et al. [14 15]. They named it MonoSLAM. MonoSLAM actually considered as a representative method in filter-based vSLAM algorithms. In MonoSLAM, camera motion and 3D structure of an unknown environment are simultaneously estimated using an extended Kalman filter (EKF). Degree of freedom (DoF) camera motion and 3D positions of feature points are represented as a state vector in EKF. Uniform motion is accepted in a prediction model, and a result of feature point tracking is used as observation. Depending on camera movement, new feature points are added to the state vector. Note that the initial map is created by observing a known object where a global coordinate system is defined. In summary, MonoSLAM is composed of the following elements.

- Map initialization is done by using a known object.
- Camera motion and 3D positions of feature points are estimated using EKF.[13]

EKF (extended kalman filter)

EKF is used to estimate the current position of the robot from the odometry reading and from the landmark observation [22]. In this process since we can't rely on the odometry reading and land mark observation directly to predict the current position of the robot, EKF filter out the data which it gets from the odometry with land mark observation and predict the correct position of the robot**.** The

first step is prediction step in this step it updates the current position using the odometry data by using the following equation.

$$x + \Delta t\, cos\theta + q\Delta cos\theta$$
$$y + \Delta t\, sin\theta + q\Delta sin\theta$$
$$\theta + \Delta\theta + q\Delta\theta$$

This should be updated in the first three spaces in the state vector, X, which contains the position of the robot, x, y and theta. Furthermore, it contains the x and y position of each landmark. We also need to update the A matrix, the jacobian of the prediction model which defines how to compute an expected position of the robot given the old position and the control input, every iteration. Also Q(noise) since process is assumed to have a Gaussian Noise proportional to the controls, x, y and t, will be updated to reflect the control terms, x, y and t.

In the second step since the predicted current position by odometry is not exactly correct we will not exactly relay on it so we need extra process to compensate for these errors. This is done using landmarks. In this process try to predict where the landmark is using the current estimated robot position (x, y) and the saved landmark position (x, y).

With the following formula:

$$\text{Range bearing} = \begin{matrix} \sqrt{(\lambda_x - x)^2 + (\lambda_y - y)^2} + v_r \\ \tan^{-1}\left(\frac{\lambda_y - y}{\lambda_x - x}\right) - \theta + v_\theta \end{matrix}$$

After calculating error matrix R should also be updated to reflect the range and bearing in the current measurements it goes to compute the Kalman gain. It is calculated using the following formula.

$$K = P * HT * (H * P * HT + V * R * VT)\text{-}1$$

Finally, it computes a new state vector using the Kalman gain

$$X = X + K * (z - h)$$

In the final step which is in step three it updates the state vector X and the covariance matrix P with new landmarks in order to have more landmarks that can be matched, so the robot has more landmarks that can be matched.

2.1.2.5 PTAM

To solve the problem of a computational cost in MonoSLAM, PTAM [15] split the tracking and the mapping into different threads on CPU. These two threads are executed in parallel so that the computational cost of the mapping does not affect the tracking. As a result, BA that needs a computational cost in the optimization can be used in the mapping. This means that the tracking estimates camera motion in real-time, and the mapping estimates accurate 3D positions of feature points with a computational cost. PTAM is the first method which incorporates BA into the real-time vSLAM algorithms. After publishing PTAM, most vSLAM algorithms follow this type of multi-threading approaches.

In PTAM, the initial map is reconstructed using the five- point algorithm [16]. In the tracking, mapped points are projected onto an image to make 2D–3D correspondences using texture matching. From the correspondences, camera poses can be computed. In the mapping, 3D positions of new feature points are computed using triangulation at certain frames called key frames. f PTAM is to introduce this key frame based mapping in vSLAM. An input frame is selected as a keyframe when a large disparity between an input frame and one of the keyframes is measured. A large disparity is basically required for accurate triangulation. In contrast to MonoSLAM, 3D points of feature points are optimized using global BA with some keyframes and global BA with all keyframes with the map. Also, in the tracking process, the newer vision of PTAM employ a re-localization algorithm [17]. It uses a randomized tree-based feature classifier for searching the nearest keyframe of an input frame. In summary, PTAM is composed of the following four components.

• Map initialization is done by the five-point algorithm [16].

• Camera poses are estimated from matched feature points between map points and the input image.

• 3D positions of feature points are estimated by triangulation, and estimated 3D positions are optimized by BA.

• The tracking process is recovered by a randomized tree-based searching [17].

Compared to MonoSLAM, in PTAM, the system can handle thousands of feature points by splitting the tracking and the mapping into different threads on CPU. There have been proposed many extended PTAM algorithms. Castle et al. developed a multiple map version of PTAM [18]. Klein et al. developed a mobile phone version of PTAM [19]. In order to run PTAM on mobile phones, input image resolution, map points, and number of keyframes are reduced. In addition, they consider rolling shutter distortion in BA to get an accurate estimation result because a rolling shutter is normally installed in most mobile phone cameras due to its cheap cost. Since PTAM can reconstruct a sparse 3D structure of the environment only, the third thread can be used to reconstruct a dense 3D structure of the environment [20, 21].

2.1.2.6 ORB-SLAM

ORB-SLAM is a versatile and accurate SLAM solution for Monocular, Stereo and RGB-D cameras. It is able to compute in real-time the camera trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments, ranging from small hand-held sequences of a desk to a car driven around several city blocks. It is able to close large loops and perform global relocalisation in real-time and from wide baselines. It includes an automatic and robust initialization from planar and non-planar scenes. Demonstrating videos, code and related publications are shown below.

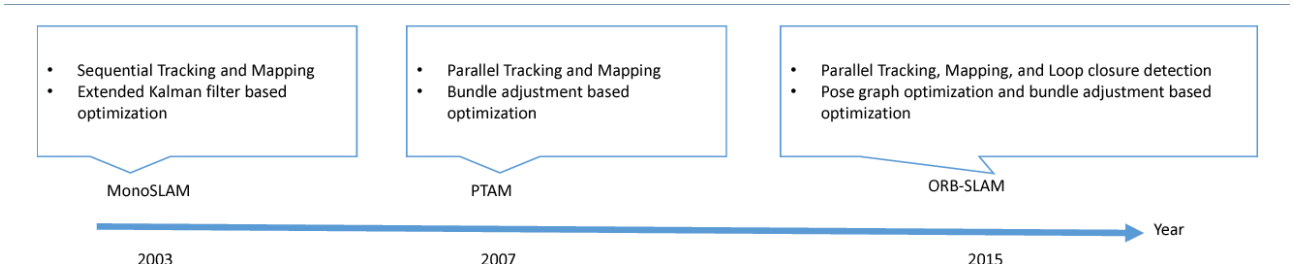


Figure2.2 Summary of feature-based methods [28]

On other hand Computer Vision for Ethiopian Agricultural Crop Pest Identification By Dagnachew Melesew Alemayehu, Abraham Debasu Mengistu, Seffi Gebeyehu proposed a hybrid algorithm for recognition of pest in agricultural crops. In this process they first convert the input video in to images to apply per-processing for the acquired image which is used for removing low frequency background noise, normalize the intensity of the individual particles on a given image, removing reflection and masking portion of image. Then they used adaptive median filtering methods for reducing noises. Finally, they processed to segmenting the pre process image using Kmeans techniques and classify them using the hybrid of RBF and SOM (Radial basis function and Self organizing map)

In Robot Navigation Using Camera by Identifying maps by Jongan Park proposed that first they build track where the robot moves. When the robot begins to move on the tack it begin to capture image frame by frame with 320 x 240 resolution image and send back to computer by using wireless communicator. The control module running on the computer receives the frames and passes the image to image processing module then the computer begins to process the image. When processing the image, it first converts the image in to binary image after converting it filter the image from noises then goes to segmentation after the segmentation it sends back to the robot through the wireless module then finally the robot receives the information and initiates its algorithm to either turn left or turn right. And also they used IR sensor in order to avoid obstacles

After I listed all the above techniques there is something I want to listed which I will combine with ORB-SLAM that will be minutiae extraction technique which is mostly bio-metric technique

2.2 Minutiae extraction

S. Kim et al. [23] presented an efficient minutia post processing algorithm which is based on minutiae distance, connectivity of ridges as the key factor. After applying per-processing procedures such as local ridge orientation calculation, image enhancement and thinning, minutiae points are extracted [24] and template is generated. Still the minutiae points extracted contain many false minutiae points which degrades fingerprint identification execution by expanding FRR and FAR.

Numerous Untrue minutiae constructions are broken ridge, Bridges, Short ridges, Holes and triangle. So, to remove false minutiae while holding true minutiae points, the proposed method consists of steps such as (i) Discovery of broken ridge structure (ii) Discovery of Bridge Structure (iii) Discovery of Short Ridge Structure (iv) Discovery of Hole structure (v) Threshold and False Minutiae Removal. So, selection of minutiae points [18] is crucial to remove fictitious minutiae even though holding real minutiae, incorrect minutiae are perceived as well as removed in precise directive are: (I) broken ridge (ii) bridge (iii) Short Ridge and lastly (iv) Hole.

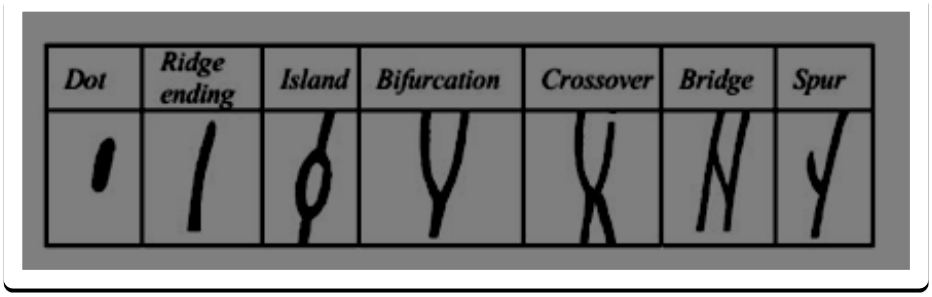


Figure2.3 various types of minutiae

M. Bandur et al. [25] proposed an improved extraction technique for minutiae by enhancing fingerprint images; they portrayed carefully the skeleton based algorithm. The obtained recognizing qualities are utilized together in the manual and programmed fingerprint matching method for identification tenacities. Proposed upgrade algorithm, using multi scale introduction field estimation alongside Log-Gabor [26] channels in recurrence area, essentially enhances unique finger impression picture quality, bringing about reduced number of erroneously minutiae excavated [27].

They comprised an algorithm for minutiae extraction with a few adjustments in orientation module of fingerprint image (where the application of multi-scale approach comes [27]), thinning module as well as binarization. When fingerprint picture has been procured, minutiae finding algorithm is performed through a few stages:

- Pre-processing (segmentation and normalization);
- Positioning and occurrence approximation
- Improvement
- Banalization
- Dilution
- Discovery of minutiae as well as sifting of it [28].

The outcome of tests performed on finger impression pictures indicates an aggressive accomplishment of projected algorithm.

2.3.Related Work

According to [41] vision sensors are utilized in various robotic systems like object recognition, obstacle avoidance, topological and global localization. The reason for this is because vision sensor over the other sensors are potable, less expensive, compact, precise, low-priced, non-invasive and pervasive.

There is some related work that use VSLAM for robot navigation. Jihong Min, Jungho Kim, Hyeongwoo Kim, Kiho Kwak and In So Kweon on proposed 2014 VSLAM with moving object tracking, (VSLAMMOT) approach. This approach tightly combines two key methods: a super pixel based segmentation to detect moving objects and a Rao-Blackwellized Particle Filter to estimate a stereo-vision-based SLAM posterior. Most successful methods perform vision based SLAM (VSLAM) and track moving objects independently. However, both VSLAM and moving object tracking as a single correlated problem to leverage the performance. the approach estimates the relative camera motion using the previous tracking result, and then detects moving objects from the estimated camera motion recursively. Moving super pixels are detected by a Markov Random Field (MRF) model which uses spatial and temporal information of the moving objects. They demonstrate the performance of the proposed approach for VSLAMMOT using both synthetic and real datasets and compare the performance with other methods.[40]

They presented a probabilistic solution to stereo-based VSLAMMOT with a sequential Bayesian filtering framework. To achieve the goal, they estimate the hybrid camera path consisting of the camera path and labels of moving objects. The samples for the camera path are generated by using the data-driven proposal distribution. Moving objects are detected by using the proposed MRF energy function. The proposed superpixel-wise MRF framework ensures the spatio-temporal consistency of the detection results through the image sequence. Which demonstrated that the proposed method outperforms other VSLAMMOT approaches in terms of the localization accuracy and the performance on moving object detection.[40]

However, the proposed system will take much time when it tries to send the image wirelessly to computer module and try to identify the environment by image processing. The given image during image processing do not detect clearly when there are more similar object exists, variant for the orientation of the image and due to there are many type of segmentation, depending on the shape of the object to be segmented, the algorithm to detect the object is highly shape dependent i.e only known shape of objects can be segmented.

A proposed method by Al-Mutib, Khalid N., Et Al. "Stereo vision SLAM based indoor autonomous mobile robot navigation." Robotics and Biomimetics (ROBIO on Stereo Vision Simultaneous Localization and Mapping (VSLAM) for autonomous mobile robot navigation in an indoor location. The objective was to design a system in which an autonomous robot would exclusively utilize a vision sensor for acquiring data and navigating the environment. Their navigation system comprised of navigation and self-localization. The overall navigation hierarchy contained of localization, Perform the Region of Interest (ROI), Region of Interest (ROI) Sub Screening, grid mapping optimal path search and path planning as illustration. The routine activities into their Visual Simultaneous Localization and Mapping (SLAM) navigation system was to achieve 3D depth calculation of the location, scene analysis, optimal path search, real time path planning and motor speed control.[43]

After the experiment the results showed their proposed system was capable of navigating in a pre-planned paths environment positively. However, the data association on the mobile robot became a challenge due to variations in illumination, specular reflection in environment and inconsistent point clouds which happened because of the variation in viewing angles of the camera.

In the study conducted by Brand, C., Schuster, M. J., Hirschmüller, H., & Suppa, M. (2014, September). Stereo-vision based obstacle mapping for indoor/outdoor SLAM., for Visual Simultaneous Localization and Mapping (VSLAM) based Indoor/ Outdoor environment obstacle avoidance. The purpose of the study was to create an on-broad stereo-vision based mapping system that would be used for path planning and local obstacle avoidance in a search and rescue mission. Their system design consisted of three layers which was perception, mapping as well planning and control. The perception layer included dense stereo matching, visual odometry computation and sensor fusion. The mapping layer included negative edge detection which assisted in identifying negative slopes and edges in the environment, like cliffs or stair-heads; to further improve on their technique, the adaptive step and slop detection was utilized to convert the depth image into a point-cloud that was associated to the local tangent plane; Outlier Filtering was deployed to reduce the standardization, propagated and stereo correlation error in a 3D positions of the detected obstacles; Time-Based probabilistic Integration was employed to surpass the small aperture angle, and Rao-Blackwellized Particle Filters (RBPF) algorithm it was used to accurately calculate the factorization of jointly probability of the robot route and the environment model into distribution possible robot paths or route. The results of localization and model calculation would be employed by planning and control layer for robot control, local path planning and obstacle avoidance[43].

The experiments of the proposed system design were performed in an indoor, outdoor and mixed Indoor & outdoor environment. In outdoor set-up the narrow-angle and wide angle Vision Simultaneous Localization and Mapping (VSLAM) system depicted an above precision of 0.22m

mean fault, having an improved deviation of less than 0.08% and 0.06% of the full course meaning that the Vision Simultaneous Localization and Mapping (VSLAM) system was capable to handle inaccuracies, drifting fusion calculations and to rectify the drifts as well. A Vision Simultaneous Localization and Mapping (VSLAM) system also showed favorable progress in closing the large loop in an experimental set-up. The Vision Simultaneous Localization and Mapping (VSLAM) system demonstrated that it could manage with changing light mixed Indoor & Outdoor set-up. However, the system couldn't perform a quantitative evaluation of obstacle map because of lack of ground truth data, the un-textured objects like white walls, regular patterns and reflective surfaces became a problem for the system architecture in an outdoor Environment.

A proposed framework byWang, Y., Huang, S., Xiong, R., & Wu, J. (2016). A framework for multi-session RGBD SLAM in low dynamic workspace environment was presented on Visual Simultaneous Localization and Mapping (VSLAM) in low dynamic workspace environment using a Red Green Blue Depth (RGBD) Sensor, the aim of the framework was to update a map by keeping track on latest changes on environment as the autonomous robot was navigating the location. The proposed framework had two components which were multi-Session Visual Simultaneous Localization and Mapping (VSLAM) and graph management. The multisession Visual Simultaneous Localization and Mapping (VSLAM) module had a graph model with each landmark being a position and each edge being a restriction, by combining previous sessions with the current information and recent session to keep the model of the environment in one global coordinates. The graph management was responsible for updating the graph model and without increasing difficulties employing the out-of-dated scan identification module and redundant scan identification module. [44]

The experimental results showed that the proposed framework was able to update the map in a dynamic or static environment without increasing difficulties and at an acceptable error level, however they were setback on the system for out-of-dated scan identification tested in the lab and industrial workspace the small object captured by Red Green Blue Depth (RGBD) sensor with low quality, were considered as noise by the algorithm, two object were regarded as one object and when the object is removed and the new one is added in the position and such change cannot be reflected on the geometric shape.

In reference to a study done by Gao, X., & Zhang, T. (2015). Robust RGB-D simultaneous localization and mapping using planar point features for Robust Visual Simultaneous Localization and Mapping (VSLAM) using planar point's features with the aid of a Red Green Blue Depth (RGBD) sensor. The objective of the study was to propose a method that would improve the accuracy in capturing and recording point's features into the Visual Simultaneous Localization and Mapping (VSLAM). Their proposed method utilized a planar feature to align key frames, the proposed method would start by removing planes points from cloud obtained the Red Green Blue Depth (RGBD)

sensor, then attempts improve the quality of the mined planes, then depth values of planes would be modified according to the parameter of plane models and Random Sample Consensus (RANSAC) framework is then employed to sense and compare point features on individual pair of planes. According to [45] Random Sample Consensus (RANSAC) is an overall method for correcting model in the existence of outliers. Their proposed method also contained a graphic end part which executed image process ego-motion calculation and pose optimization part was for executing global pose graph optimization. [46]

The experimental results of a proposed method on open datasets showed tracking trajectory is fairly accurate, high accuracy in motion of the two fames with less failure rate. However physical experiment results showed repetitive structure on floor, daylight from the ceiling disrupted the matching process and sunshine from window disrupts the illumination conditions.

A proposed algorithm by Lin, L. H., Lawrence Robust outdoor stereo vision SLAM for heavy machine rotation sensing for robust outdoor stereo Vision Simultaneous Localization and Mapping (VSLAM) for heavy machine rotation sensing. The purpose of the algorithm would be to calculate a mining rope shovel's rotation about its vertical axis. Extended Kalman Filter (EKF) was selected the main algorithm in this study because of its abilities to close loop successfully. It defines Extended Kalman Filter as an algorithm that is utilized in a number of nonlinear estimation and machine learning systems as well as calculating the state of a nonlinear dynamic application, calculating parameters for nonlinear system identification when system dynamic and observation models are linear. For every camera frame, based on Extended Kalman Filter (EKF) Simultaneous Localization and Mapping (SLAM).[47]

According to the study, the proposed algorithm also employed the two methods to increase Visual Simultaneous Localization and Mapping (VSLAM) effectiveness for outdoor environment. The techniques were Harris corner namely locally maximal feature and Feature Cluster landmark. Locally maximal feature chose features more evenly across the image and senses features that consistently in a non-uniformly illuminated scene, Whereas Feature Cluster landmark acquired each landmark in a cluster of 3D feature from a single camera frame, this will permit extremely consistence and robust landmark measurement. As corners were designated based corner threshold within an image. The study also used a percent threshold method to measure the pixel locations of edge that surpasses the corner threshold within an image.[47]

The experiment of a vision algorithm was performed using three separate video sequence, at round 2900, 8000 and 6600 frames respectively. The maximum error for all video sequence settled within 1o. When compared two methods used in a vision algorithm, it was a discovered that the Local maximum method was steadier than Percent Threshold method detecting frame with direct sunlight or shadows illumination.

The above comments and experiment most of them are not my experiment. The exprement was taken by different organization like IEEE and by Jabulani K. Makhubela, Tranos Zuva and Olusanya Yinka Agunbiade. [48]

Chapter Three

Methodology of the Study

3.1 Overview

In this section, it is described the flow of the research methodology examining the current system in order to get deep understanding of proposed development method, compare and contrast available development tools in order to select appropriate tools for the proposed system, defining the design of the proposed system method and describing proposed system testing method.

3.2. Development Tools

Development tools discuss the tools that provided to develop the proposed approach. There are different types of tools used for this research to design and implement the proposed system. These are database of Computer Vision Group TUM Department of Informatics Technical University of Munich, pycharm, ROS (robotic operating system) for simulation, UML modeling, and other relevant tools. These are described below:

3.2.1 Design Tools

Edraw Max tool is type of software design tool which is used to design the proposed system. This design tool is more popular and user friendly software design tool. This design tools provides many features that could not be found in other diagram software's such as workflow, flowchart, UML diagrams, Business diagram, database and ERD [26].

3.2.2 Simulator Selection

There are a lot of tools available for the simulation in computer vision as well as robot. To select the best fit simulation, we have gone through some identified simulation tools. We compared these simulation tools. Since the robot which we uses only operates on ROS, ROS provides node to node features which reduces complexity, reduce the development time and complexity by providing device drivers, libraries, visualizers, message-passing, package management, access to third party tools, hardware abstraction [27] [28] [29], code can be reuse for the robotic developer and researcher in a common and generalized platform, its language-independent architecture (C++, python, lisp, java, and more), scalable platform (ARM CPUs to Xeon Clusters) and intent to enable researchers to rapidly develop new robotic systems by increasing the re-usability through use of standard tools and

interfaces and the design tool that ROS provide to design and implement the robot like Rviz, gazebo we choose ROS as simulator software for robot.

So from the table 3.1 listed below open source robot simulator software one can observe that ROS uses more number of programming language than the other simulators and the documentation we can uses on ROS is high level and also tutorial for using ROS can be found easily for developing robot. Player/Stage /Gazebo and ROS. These two simulators, in fact, exhibit unique features, such as accurate 3-D environment modeling, as well as the possibility to interface with real robots and sensors. The ROS tools allow a easy usage of packages and stacks. The advantage of ROS is code reuse and sharing. Code sharing allows everyone to have a common basis and helps with replicating and testing the source code. By using the ROS tools, implementation is efficient and simplifies some of the complex tasks needed to execute the code but this creates a dependency to ROS and reduces the mobility of the code. ROS can connect to real robots and use simulated robots in ROS's Gazebo, Stage, and Rviz. However, control code written for Player's Stage and Gazebo will have to be modified to work in Gazebo/Stage under ROS.

Table 3.1 Characteristics of Open-Source Robotic Simulation Software

	Player	Gazebo	ROS	simbad	CARMEN	USARSim	MRDS	MissionLab
OS	Linux win	Linux	Linux	Linux	Linux	Linux win	win	Linux
SimulatorType	2-d	3-D	3-D,2-D	3-D	2-D	3-D	3-D	3-D
Programming language	C++ python java	C,C++,Python,Java	C,C++, Python, Java	java	C,java	C,java c++	VPL, C#, Visual Basic, JScript, IronPython	VPL

Documentashion	Low level	Low-Level	Highlevel	High-Level	Low-Level	High-Level	High-Level	High-Level
Tutorial	Yes	Yes	Yes	Limited	Limited	Limited	Yes	Limited
Portability	Yes	Yes	Yes	Limited	Yes	Yes	Yes	Yes
Sensors	Odometry range	Odometry range camera	Odometry range camera	vision,range	Odometry,rang,GPS	Odometry,rang,camera,touch	Odometry,rang,camera	Odometry,rang
Debugging/ Logging	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Graphical User Interface	NO	No	No	No	No	Yes	Yes	Yes

So from the above open source robot simulator software one can observe that ROS uses more number of programing language than the other simulators and the documentation we can uses on ROS is high level and also tutorial for using ROS can be found easily for developing robot. Player /Stage /Gazebo and ROS. These two simulators, in fact, exhibit unique features, such as accurate 3-D environment modeling, as well as the possibility to interface with real robots and sensors. The ROS tools allow a easy usage of packages and stacks. The advantage of ROS is code reuse and sharing. Code sharing allows everyone to have a common basis and helps with replicating and testing the source code. By using the ROS tools, implementation is efficient and simplifies some of the complex tasks needed to execute the code but this creates a dependency to ROS and reduces the mobility of the code. ROS can connect to real robots and use simulated robots in ROS's Gazebo, Stage, and Rviz. However, control code written for Player's Stage and Gazebo will have to be modified to work in Gazebo/Stage under ROS.

3.2.2.1. ROS

ROS is an open source, meta operating system for our robot that provides us with service that we would expect from an operating system including hardware abstraction layer [19]. It is clearly understandable that to implement an indoor navigation system in a mobile robot platform, the developer must use robot, different type of sensors, devices and also will implement some algorithm, logical program. But the first issue is to put all the things together so that the system works and therefore a platform or framework is required. ROS provides wide range of libraries and tools to help software developers creating innovative robot applications in an organized pattern. Some the features ROS have are:

a. Peer to Peer

ROS consists of small programs which are connected to each other and they also exchange messages. These messages are sent within the ROS framework and the travel directly from one program to another [30]. They are not routed through any central routing service. This will be an advantage to scale up the system when the amount of data increases.

b. Multiple Language Support

ROS supports a number of programming languages such as C, C++, Java, Python, etc. ROS client libraries exist for C++, Python, Java, JavaScript, Ruby, OpenCV and many more for example

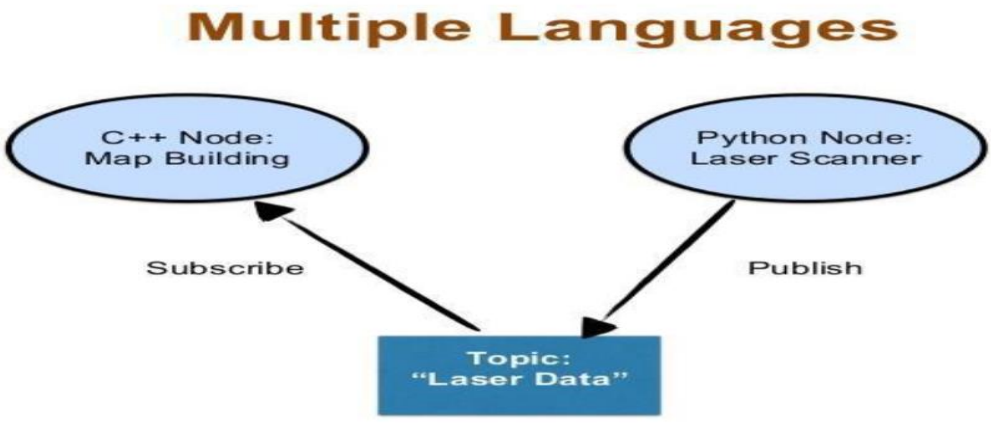


Figure3. 1 multiple language supporter by meta operating system ROS

c. Free and Open Source

ROS is released and licensed by BSD (Berkeley Software Distribution) which can be used for commercial and non-commercial purposes. People who use ROS for commercial purposes can do their development behind a firewall and still have their closed source modules communicating with large number of open source modules

3.2.2.2 ROS Architecture

ROS architecture is designed and divided into three sections and levels of concepts. In the figure below shows the diagram of ROS architecture and below the main levels is briefly described.

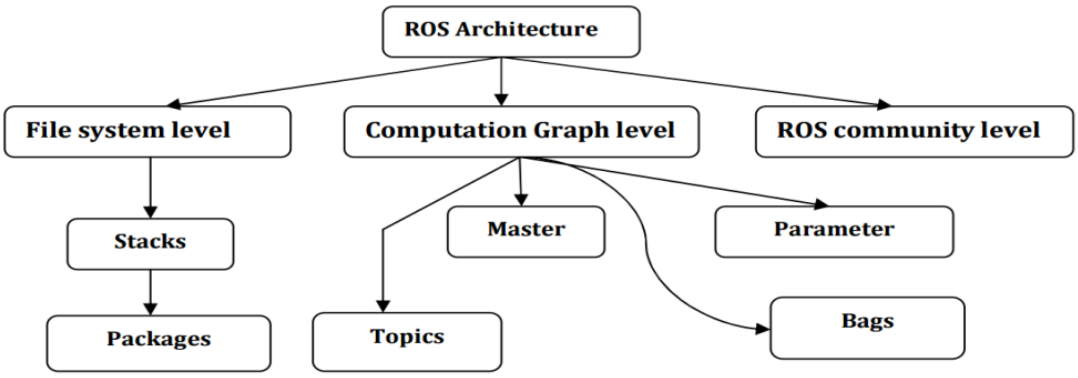


Figure3. 2 ROS architecture with their respective section and level concept

The File system level is the first level which is used to explain ROS folder structure, how ROS internally works, and the minimal files needed for it to work. The Computation Graph level is the second level in which communication between processes and systems happen. If there is more than one computer ROS sets up systems which can handle all the processes which can communicate with computers on the network. The Community level is external to the ROS in the sense that is has nothing to do with operation of ROS [31]. This level is important in the context of helping ROS to grow quickly.

Beside the above reason the other reason which we choose ROS is that ROS does not require one to develop the existing system and programs all over again, but can rather easily turn a non-ROS system to a ROS-system by simply inserting a few standardized codes. Furthermore, ROS provides various tools and software that are commonly used, and it allows users to focus on the features that

they are interested in or would like to contribute in, which ultimately reduces the development and maintenance time.

3.2.2.3 ROS Tools

There are variety tools that is provided by ROS one of the tools which is discussed is RVIZ 3D visualization tool, rqt Qt-based ROS GUI development tool, rqt_image_view Image display tool (a type of rqt), rqt graph A tool that visualizes the correlation between nodes and messages as a graph (a type of rqt) [31].

3.2.2.4. Data visualization Tool (RVIZ)

RViz is the 3D visualization tool for ROS which is used to show ROS message in 3D, allow us to visually verify data like the distance from the sensor of the laser distance sensor(LDS) to an obstacle, the image value obtained from the camera and many more without having to separately develop the software [17]. It also supports various visualization using user specified polygons, and Interactive Markers allow users to perform interactive movements with commands and data received from the user node.

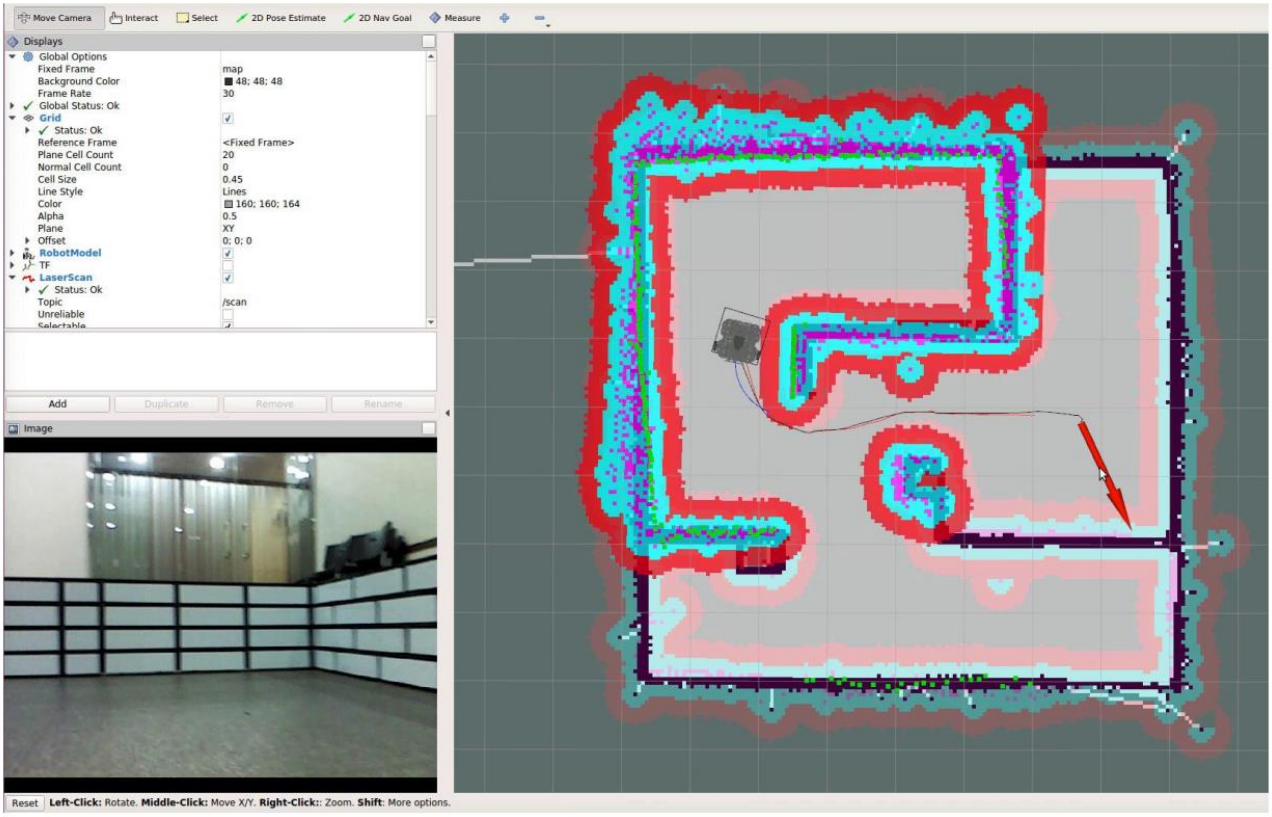


Figure3. 3 Over view of RVIZ simulator

3.2.2.5. ROS GUI Development Tool (rqt)

Beside RVIZ ROS provide various tools for robot development such as graphical tools that shows the hierarchy of each nodes as diagram to show the status the current node and topic, and plot tool that used to show the message passage in a 2D graph. rqt was developed based on Qt, which is a cross platform framework widely used for GUI programming, making it very convenient for users to freely develop and add plugins. Among the rqt there is rqt graph which is a tool that shows the correlation among active nodes and messages being transmitted on the ROS network as a diagram. This is very useful for understanding the current structure of the ROS network.

3.2.2.6. Gazebo

Gazebo simulator is 3D simulator that integrates within ROS to provide 3D dynamic simulation. Gazebo simulator provides efficient and accurate simulation for robot procedure in complex outdoor and indoor environment and used to test robotics algorithm, robot design and executing testing in truthful situation [31]. Gazebo simulator useful to design map for navigation of the robot and simulation observation for turtlebot3. In this simulation after each node runes we can observe the 3D output while performing the required task.

3.3. Data collection method

When doing computer vision projects, training data and test data is essential. If algorithms that train themselves based on images are used, the need of training data is obvious. However, also simpler systems need training data that the algorithms can run on, to asses and adjust their performance. A test set is necessary for evaluating system performance. It is important that the system is not tested on the same data as it is trained on, since that cannot tell if the system generalizes to other data. This section describes both considerations about the actual data content, and the technical means with which it was collected. During this section there will be no distinction between training and test data. In the end, splitting the dataset into separate training and test pools is discussed. Websites and literature review are the main sources of data for this research in order to collect relevant images and information's respectively. It has been collected information as much as possible using the recent information from the websites and ROS official website.

3.3.1. Data set content and sampling techniques

The query image and video dataset was created by grabbing different pixel indoor images from different websites and capture them manually. In addition, the dataset contains the template images that will be matched with the query images. These template images were chosen from the

query image dataset to form a reliable subset of ground truth images. In this process we used stratified sampling techniques. First, we divide each image and video because we have to understand difference of the outcome from each algorithm and combined each other. Finally, from the image and video collected we rearrange them in to different shapes, intensity and sizes for testing purpose. In this rearrangement process similar images with different intensity, shapes and sizes are stored with their respective dataset. The following subsections provide more information regarding each of the data sets, because most of research and VSLAM survey research papers suggested this[20]

Recognition I: Lighting and Orientation

The purpose of this data set is to test the robustness of the recognizer with respect to various illumination changes and in plane rotations. Frontal images of environment were taken at different intensities variation to test the reliability of the algorithm of different light changes.

Recognition II: Viewing Angle

We compiled a second recognition data set to test the robustness of the recognizer with respect to different viewing angles. In this the images are rotated 45 degrees to verify the recognizer algorithm is robust to detect at different position. This second database contains ten images of 10 different place under 45 degree viewing angles.

Recognition III: Scaling

In this process a third data set is formed to test the robustness of the recognizer with respect to different size of the test images. In this data set small size image is exists. This process is mainly useful to test the recognition capability from long distances.

Using the process described above, a database with video and images collected. The dataset consists of 20 video 30 images and contains 10 classes with different transformation. The resolution of the full captured frames varies from 480x360 to 1024x522 pixels. All the images are in color. The dataset does not cover the full indoor and outdoor environment, which is nearly impossible, but it does cover a good selection of environment . Given that the dataset has been collected from hours of real driving through indoor environments, it should represent the real-world class distribution well. Thus, the dataset reflects reality, and if it forces researchers to focus on shape-based detectors, that is only good, because those are the detectors that can be implemented in today’s recognition of environment.

3.3.2. Training and testing

Training data were retrieve from the database which is stored for the testing purpose. In this process images with size of 650*480 pixel and video were stored and trained by the algorithm for testing with the test image. The test image and video is the image which is going to be tested with the trainee image and video of their target. The trainee image is loaded frequently for retrieving the detected features and describing them to know their real position in the images pixels within stored dataset. After describing them the trainee image the detected and described data is stored in the database. After detecting and describing, the test image and video will be matched with correspondence point in the target outcome. If the match greater than threshold matched will be found.

Format of the testing data

- The color images are stored as 640x480 8-bit RGB images in PNG format.
- The depth maps are stored as 640x480 16-bit monochrome images in PNG format.
- The depth images are scaled by a factor of 5000, i.e., a pixel value of 5000 in the depth image corresponds to a distance of 1 meter from the camera, 10000 to 2 meter distance, etc. A pixel value of 0 means missing value/no data.

You can access all of this thesis work database in this link:
(https://vision.in.tum.de/rgbd/dataset/freiburg1/rgbd_dataset_freiburg1_xyz.tgz)

3.4. Proposed System Implementation Method

The proposed system has two important parts navigational phase and computer vision phase. In the first part we extract both core value and extraction key frames with the proposed algorism after that there will be matching each frame and figure out matches and form map.

after that also try simulate using ROS(gmapping) with combination that on but for testing and evaluating the proposed image processing algorithm we used pycharm.in detail description it look like :

four threads that run in parallel: minutia extraction tracking, local mapping and loop closing. The tracking is in charge of localizing the camera with every frame and deciding when to insert a new keyframe and available core values(minutia) will be added as keyframes. We perform first an initial feature matching with the previous frame and optimize the pose using motion-only BA with the matching minutia. If the tracking is lost (e.g. due to occlusions or abrupt movements), the place recognition module is used to perform a global relocalization. Once there is an initial estimation of the camera pose and feature matchings, a local visible map is retrieved using the covisibility graph of keyframes then matches with the local map points are searched by

reprojection, and camera pose is optimized again with all matches. Finally the tracking thread decides if a new keyframe is inserted. The local mapping processes new keyframes and performs local BA to achieve an optimal reconstruction in the surroundings of the camera pose. Parallel computation of the two models: Compute in parallel threads a homography Hcr and a fundamental matrix

$$\mathbf{F}cr:$$
$$\mathbf{x}c = \mathbf{H}cr\ \mathbf{x}r\ \mathbf{x}T\ c\ \mathbf{F}cr\ \mathbf{x}r = 0$$

with the normalized DLT and 8-point algorithms respectively as explained in [12] inside a RANSAC scheme. To make homogeneous the procedure for both models, the number of iterations is refixedand the same for both models, along with the points to be used at each iteration, for the fundamental matrix, and for the homography. At each iteration we compute a score SM for each model M (H for the homography, F for the fundamental matrix):

$$SM = \mathrm{X}$$
$$i$$
$$\rho Md2\ cr(\mathbf{x}i\ c;\ \mathbf{x}i\ r;\ M) + \rho M(d2\ rc\mathbf{x}i\ c;\ \mathbf{x}i\ r;\ M)$$
$$\rho M(d2) = (\Gamma 0 - d2 \text{ if if } d\ d2\ 2 < T \geq TM\ M$$

where d2 cr and d2 rc are the symmetric transfer errors [12] from one frame to the other. TM is the outlier rejection threshold based on the $\chi 2$ test at 95% (TH = 5:99, TF = 3:84, assuming a standard deviation of 1 pixel in the measurement error).[15] Γ is defined equal to TH so that both models score equally for the same d in their inlier region, again to make the process homogeneous. We keep the homography and fundamental matrix with highest score. If no model could be found (not enough inliers), we restart the process

3.5. Algorithm Selection for Image Processing Unit

The main objective of the work is to design navigation system for autonomous robot in order to navigate within unknown environment by detecting indoor map. Since the speed for recognition of the map within indoor environment should be compatible with the speed of the robot. So a robust, fast and selective image processing algorithm is required. Feature detection and matching is one of the well-known matching algorithm that used for matching similar images by finding correspondents point between two images. Its characteristic in matching with high speed and reliability this technique is best compared other techniques requires segmentation and

computational qualifications. Papers have been reviewed to find fast, robust and selective algorithm that have been developed in order to adopt and modify for the proposed work. From the existing matching algorithm MonoSLAM, ORB, Minutiae extraction FAST are the chosen algorithm for getting match details.

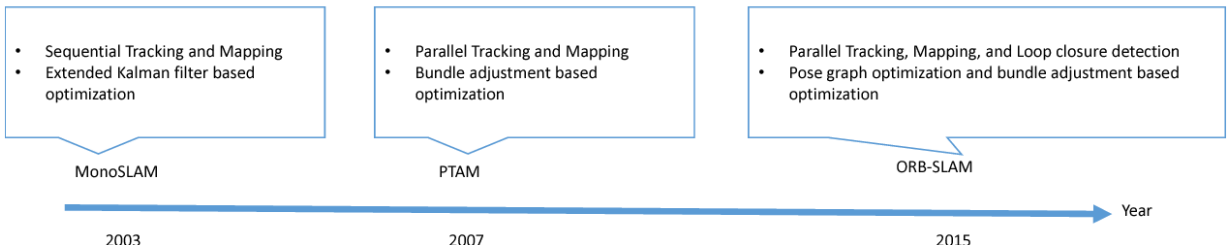


Figure3.1 Summary of feature-based methods

There are common VSLAM algorithms:

	Method	Map density	Global optimization	Loop closure
Mono-SLAM	Feature	Sparse	No	No
PTAM	Feature	Sparse	Yes	No
ORB-SLAM	Feature	Sparse	Yes	Yes
DTAM	Direct	Dense	No	No
LSD-SLAM	Direct	Semi-dense	No	Yes
SVO	Semi-direct	Sparse	No	No
DSO	Direct	Sparse	No	No
KinectFusion	RGB-D	Dense	No	No
Dense visual SLAM	RGB-D	Dense	Yes	Yes
ElasticFusion	RGB-D	Dense	Yes	Yes
SLAM++	RGB-D	Dense	Yes	Yes

3.5 Evaluation Method

In order to verify the effectiveness of the detecting algorithm, experiments were carried out from three aspects: matching correctness, extracting feature points and matching efficiency through different transformation of the images like intensity variation, rotation, scaling and selective characteristic by merging different images all together. The matching correctness is evaluated as the probability of the point that should exits at the same position of the point existing in the corresponding image. When evaluating the algorithm using extracting feature point it means that

the feature point that detected by the algorithm should be at corner, T-junction or it should be blob point.

Chapter Four

Proposed Solution

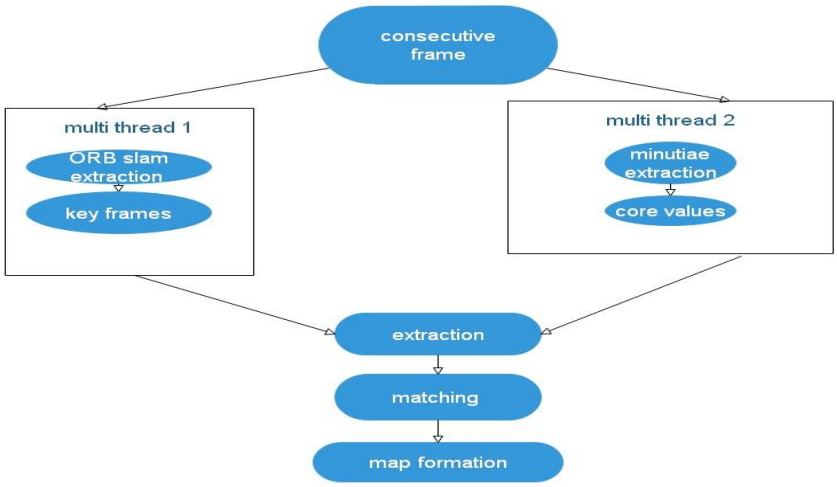
4.1. Overview of the Proposed Solution

This chapter describes the procedure followed n order to give solution for the described problem. This include the following: make the robot to be able to understand the environment by building map and simultaneously being able to understand its location which is called navigational control system, identification of the obstacle and recognition process (computer vision pipeline).

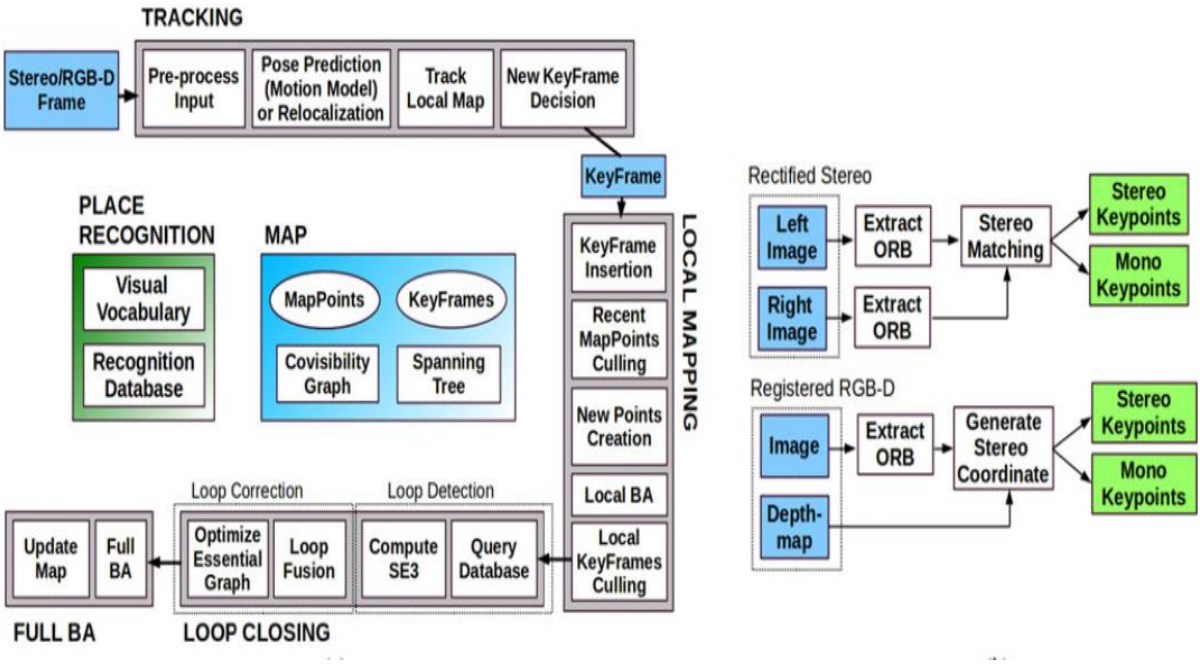
4.2 Computer Vision Pipeline

Since it uses robot that navigate through indoor environment, the time that the algorithm used to make the robot able to observe and recognize any details and the time to take action based on the outcome should be compatible with the speed of robot. To do so from different comparative study the best algorithm for detection and matching is selected and integrated for better result.

ORB-SLAM on figure show us to do the tracking to localize the camera with every frame by finding feature matches to the local map and minimizing the reprojection error applying motion-only the local mapping to manage the local map and optimize it, performing local BA; and 3) the loop closing to detect large loops and correct the accumulated drift by performing a pose-graph optimization. This thread launches a fourth thread to perform full BA after the pose-graph optimization, to compute the optimal structure and motion solution.



ORB-SLAM



(a) (b)

Fig. 4.1. ORB-SLAM is composed of three main parallel threads: tracking, local mapping, and loop closing, which can create a fourth thread to perform full BA after a loop closure. The tracking thread preprocesses the stereo or RGB-D input so that the rest of the system operates independently of the input sensor. Although it is not shown in this figure, ORB-SLAM also works with a monocular input as in . (a) System threads and modules. (b) Input preprocessing [31]

MINUTIAE EXTRACTION

The purpose of minutiae extraction s to identify the diplomat features called minutiae and to extract them from the input images. Extraction consists of three main steps, and they are

- Preprocessing.
- Minutiae extraction.
- Post-processing.

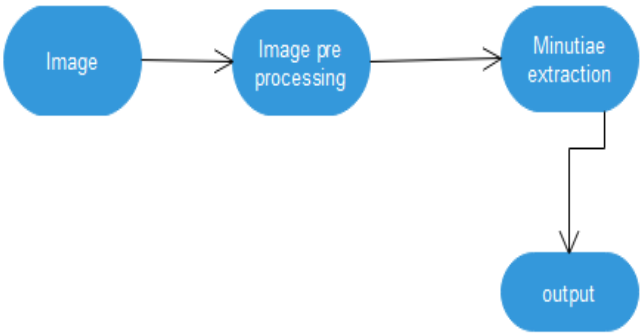


Figure4.2 simple steps of minutiae extraction

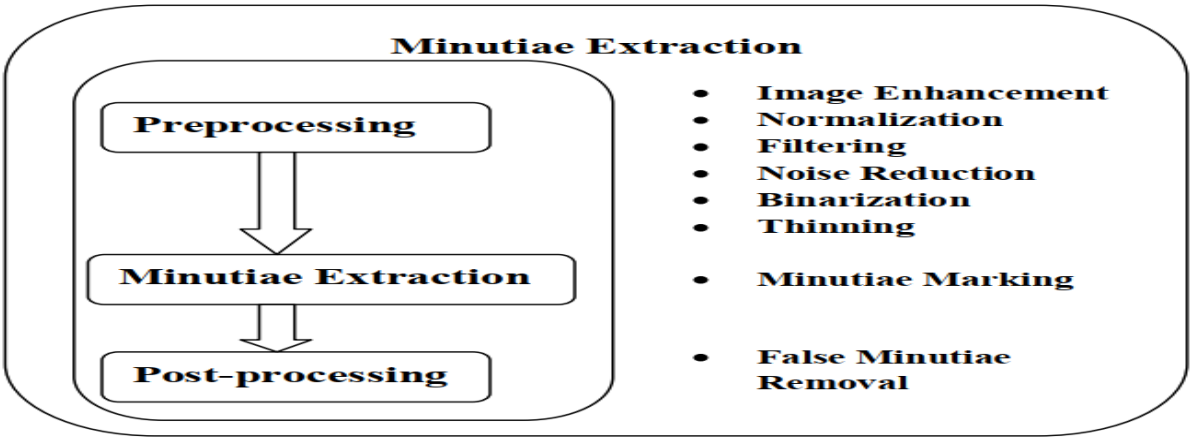


Figure4.3 extended steps of minutiae extraction

4.2.1. Detection Step

The first step is detection of interest point from the given image. In order to increase the speed of the detection process we integrate ORB with Minutiae algorithm with thread management for each additional key points. The purpose of minutiae extraction is to identify the diplomat features called minutiae and to extract them from the input fingerprint images. It is very complicated for the prominent and accurate demonstration of the input images since ORB is fast parallel, we can use this algorithm with minutia. But since minutiae extraction have extended enhancement steps, we try to decrease some steps make it fast as much as possible so the steps will be

- Normalization
- Noise Reduction
- Binarization
- Thinning
- Minutiae Marking

Normalization: Normalization of an image means to spread the gray scale evenly and fill all available values instead of a part of the available gray .By means of histogram, the distribution of pixels with a certain amount of gray has been plotted. To normalize an image, the area which is to normalize has to be known. The highest and the lowest pixel values of the current image have to be identified and this is the foremost thing. Along this scale, each and every pixel is then evenly spread out. The following equation represents the normalization process.

$$Inorm\ (x,y) = \frac{I(x,y)-Imin}{\mathrm{I}\max-Imin}\ X\ M$$

Where I is the intensity (gray level) of the image. I_{min} is the lowest pixel value found in the image, I_{max} is the highest one found. M represents the new maximum value of the scale, mostly M = 255, resulting in 256 different gray levels, including black (0) and white (255). $I_{norm}(x, y)$ is the normalized value of the pixel with coordinates x and y in the original image $I(x,y)$.The normalized images are much easier to evaluate and determine quality since the spread now has the same scale. Without the normalization I would be impossible to use a global method for comparing quality. Figure 4.3 shows a sample normalized image.

Noise Reduction: In order to reduce the noise, pixel-wise wiener filtering is proposed The filter is based on the estimated local statistics from a local neighborhood of size 3 × 3 of each pixel, and is given by

$$w(m_1, m_2) = \tau + \frac{\mu^2-\sigma^2}{\mu^2}(I(m_1, m_2) - \tau)$$

xzz

Where σ2 is noise variance, and are local mean and variance, where the gray level intensity is represented (m1, m2).

Binarization: The separation of the object and background is known as Binarization. A gray scale picture is turned into a binary picture. A binary picture has two dissimilar values only. The colors black and white are represented by the values 0 and 1 respectively. A threshold value in the gray scale image is picked for binarization of an image. Everything darker than this threshold value is converted to black and everything lighter is converted to white. To find the identification marks in such as singularity points or minutiae, this method is performed. The complexity with binarization lies in finding the accurate threshold value to be able to eliminate insignificant information and improve the significant one. Finding a working global threshold value that can be used on every image is unfeasible. The deviations can be too huge in these types of images that the background in one image can be darker than the print in another image. Therefore, algorithms to find the optimal value must be applied separate on each image to get a functional binarization.so on this research we tried to do algorithm is based on global thresholds.

$$I_{new}(m_1, m_2) = \begin{cases} 1 \; if \; I_{old}(m_1, m_2) \geq Local \; Mean \\ 0 \; otherwise \end{cases}$$

Thinning: One way to make a skeleton is through thinning algorithms. The technique takes a binary image and makes the ridges that appear in the print just one pixel wide without changing the overall pattern and leaving gaps in the ridges creating a sort of "skeleton" of the image. The form is used as structural element, consisting of five blocks and each block representing a pixel. The center pixel of that element is called the origin. When the structural element overlays the object pixels in its entirety, only the pixels of the origin remain. The others are deleted. Thinning makes it easier to find minutiae and removes a lot of redundant data.

Minutiae extraction: A valid demonstration of the image is the pattern of the minutiae details. It satisfies the basic properties like compactness, agreeable to recognize, robust to noise and distortions and is easy to compute. A total of 150 diverse local ridge characteristics, called minutiae details, have been identified. Most of them are not enduring and cannot be consistently recognized, and they depend deeply on the impression conditions. We will try to recognize using direct level minutiae extraction.

The overall suggested work will look like

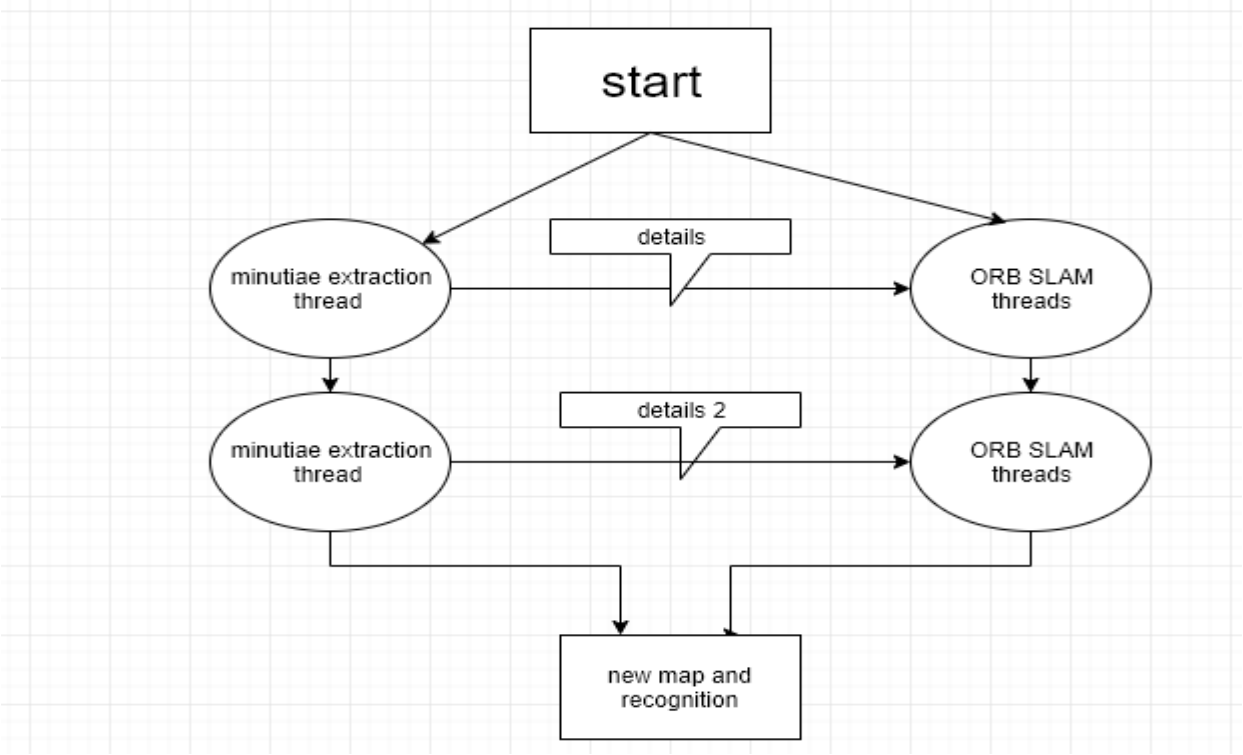


Figure4.4 flow in which ORB and minutiae extraction goes together

4.3. Navigation control system

The navigational control system is responsible for making a navigational decision for the robot when it has or does not have goals from the computer vision pipeline. The control system can be divided in to two sections: the navigational stack in ROS, and the controller algorithm in which we will see next.

4.3.1. Navigational process

ROS provides a navigational stack setup that is responsible for generating cost maps from sensor data and planning paths for the robot with these cost maps[36]

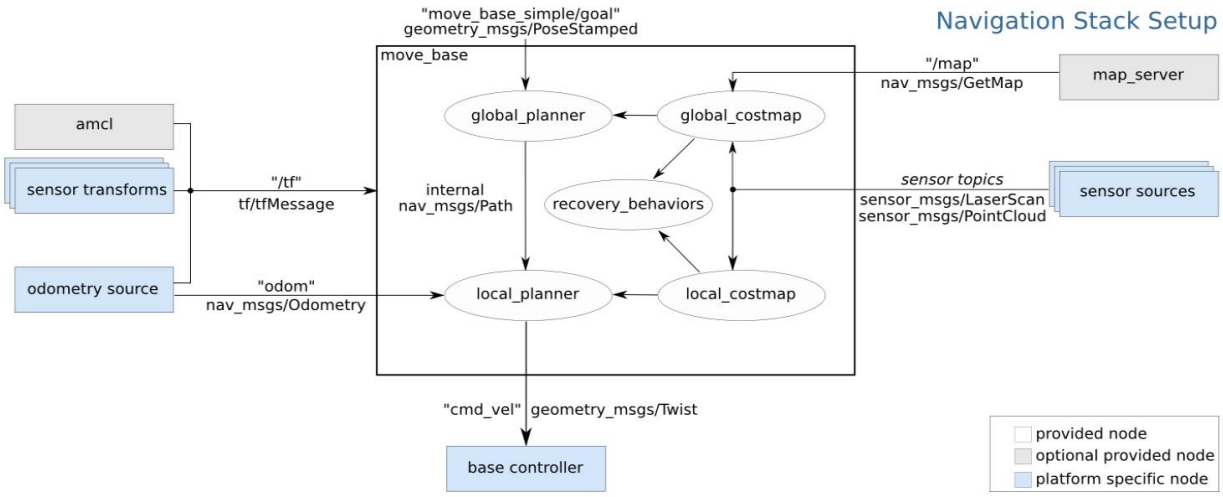


Figure4.5 general sachema move_bas path planner

Figure above shows the navigation stack setup in ROS. The stack receives inputs such as sensor data and navigational goals. It gives the base controller output commands. Inputs that are platform dependent (TurtleBot in this case) are the sensor data, odometer data, and sensor transforms. Sensor data is used to generate local cost maps while odometer data is used for relative position estimation. The sensor transforms contain geometric information of the robot such as the relative position. The stack contains a node called move_base, that accepts simple goals (X and Y coordinates and rotation angles), plans a path avoiding obstacles and boundaries and sends velocity commands to the base controller (the mobile base driver). There are some optional nodes that can be seen in Fig. above which are external maps and AMCL parameters for navigation in a pre-mapped environment.

ROS by it self extracting the key point and features it stores in the database all the extracted features. The third step is during the journey of the robot it continually take up the features and it start to associate with the already stored data in the database. During the association if it found the new features it will store in the database and if it is already existing feature it will used to estimate the position of the robot by measuring the transition in the distance and angle to the old features. Finally based on the new position and measurement data the map was updated and built. The repetition of the five step will continue until the map is finished built by using teleport command.

4.3. Controller Algorithm

Since the destination, the robot has to move is given by the user, the robot has to move to the destination automatically by observing output and avoiding any obstacles. So the controller algorithm is responsible to receive data from the computer vision pipe line and make decisions for the next move of the robot. It keeps track of goals and updates them when required. The image goals are hence divided into different decisions. The controller waits for receiving valid message from the computer vision pipeline publisher message.

4.4. Deployment model

ROS is developed in unit of nodes, which is the minimum unit of executable program that has broken down for maximum reusability [34]. Each process described in the above proposed solution like Teleop, SALM, Computer vision pipeline, the mechanism to avoid obstacle and the control algorithm are programed separately as node. Each nodes exchanges data with other nodes through messages forming a large program as a whole. The type of exchanging messages can be topic which provides a unidirectional message transmission and reception, service which provides a bidirectional message request and response and an action which provides a bidirectional message

goal/result/feedback.

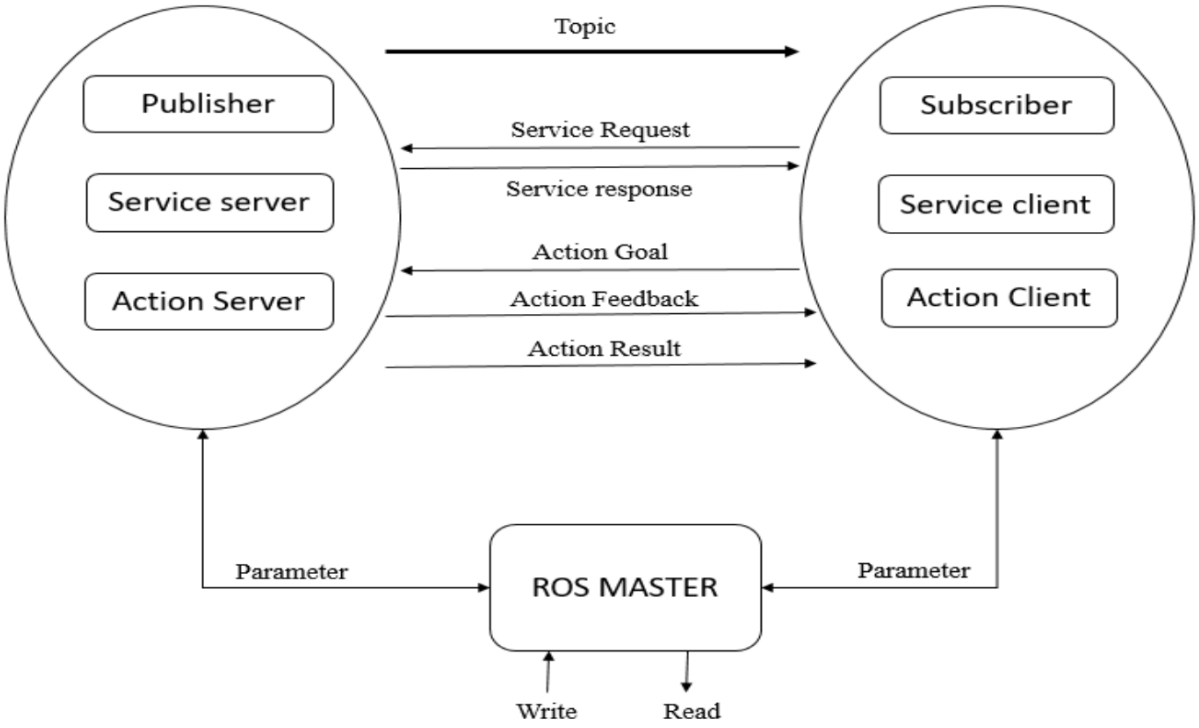


Figure4.7 Message Communication Architecture

The master manages the information of the nodes, and each node connects and communicates with other nodes as needed. After the master run the master register the name of the nodes, topic, services, action, message type, URI addresses and port for node to node connection. When the subscriber node run the node register its node name, topic name, message type, URI address and port with the master it runs with XMLRPC. At the same time the publisher node register to the master its address and name as the subscriber node. Then the master distributes information such as the publisher name, topic name, message type, URI address and the port number of the publisher to subscriber. After distribution the publisher information to the subscriber the subscriber node requests direct connection to the publisher. During the request procedure the subscriber node transmit information to the publisher node the node name, the topic name and the message type using XMLRPC communication. Finally, the subscriber node creates a client for publisher node using TCPROS and connect to the publisher node. At this time the publisher node begun to transmit predefine message to the subscriber node using TCPROS communication.

Chapter Five

Implementation of the Prototype

This chapter describes the procedure that has been followed in order to implement the proposed solution. Some of the main steps were recognition step, prototype design, map building and obstacle avoidance.

5.1 Prototype Setup

we design two ways of prototype

1 implement with ROS simulation on Turtlebot3

2.use already recorder video from monocular camera

Since the robot have to detect and avoid obstacle, we can integrate camera to raspberry pi3 with initial port for transferring the image and be able to process it on the computer. Turtlebot3 also has DYNAMIXE two rotating wheel that can measure the odometery which is useful data to measure the position of the robot by its rotation the robot should additionally Laser scanner that used for collecting landmark and measuring reflecting distance.

Turtlebot3 is robot which uses raspberry pi3 and Arduino board for operating and controlling the robot. This robot has LIDAR which is mounted at the top which is used to measure distance from reflected object as well as to useful to collect landmark But, Degraded at high sun angles and reflections, Unreliable for water depth and turbulent breaking waves, Very large datasets that are difficult to interpret it can work well integrate by camera to raspberry pi3 with initial port for transferring the image.

The initial setup includes turtlebot3, Desktop, a raspberry pi3. The desktop pc run on Ubuntu 16.04 while the raspberry pi 3 run on Ubuntu mate. ROS Kinetic full desktop version was installed on the desktop. This included all ROS libraries, visualization tools like Rviz and rqt, 2D/3D simulators, and 2D/3D perception. ROS kinetics for raspberry pi3 was installed on raspberry pi3 of the Ubuntu mate and turtlebot3 message was installed. For both the pc and the turtlebot3 network was configured in order to make both to be able to communicate and raspberry pi3 camera driver was installed on the raspberry pi3 and made calibration process to make the robot to be able to measure distance from the image. The V.12 camera with 8 mega pixel resolution is mounted in front of the robot that able the robot to record the video. The camera is connected to raspberry pi3 in order to be processed by the raspberry pi3 and to be able to be send to the computer through the wifi modem to be further processed.

.

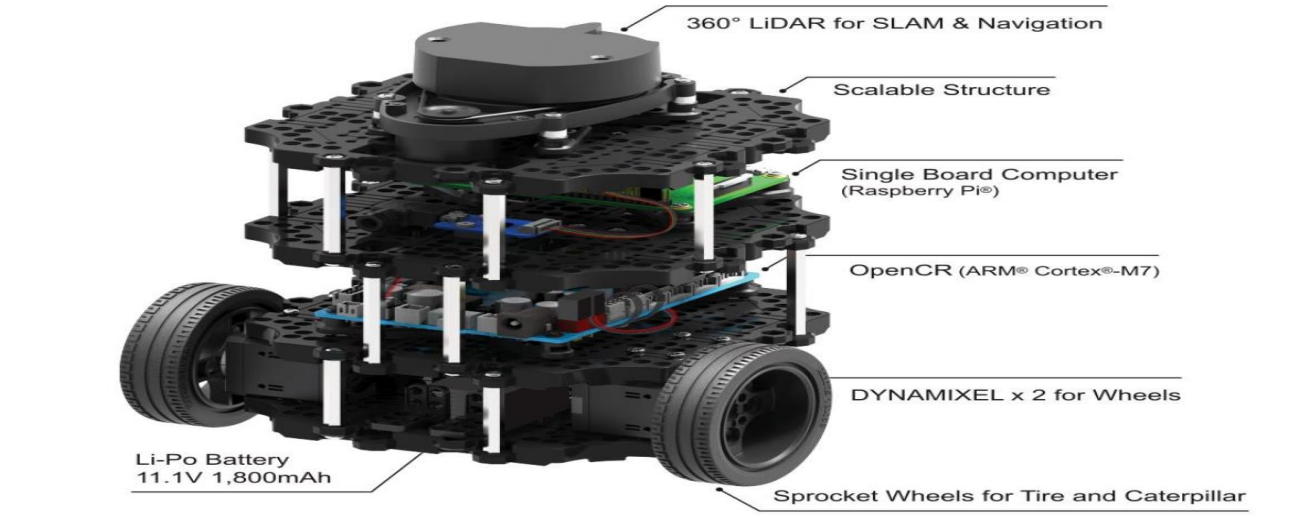


Figure5. 2 Turtlebot3

And we try to fine sample video from monocular camera and transfer it to frame by frame for testing use

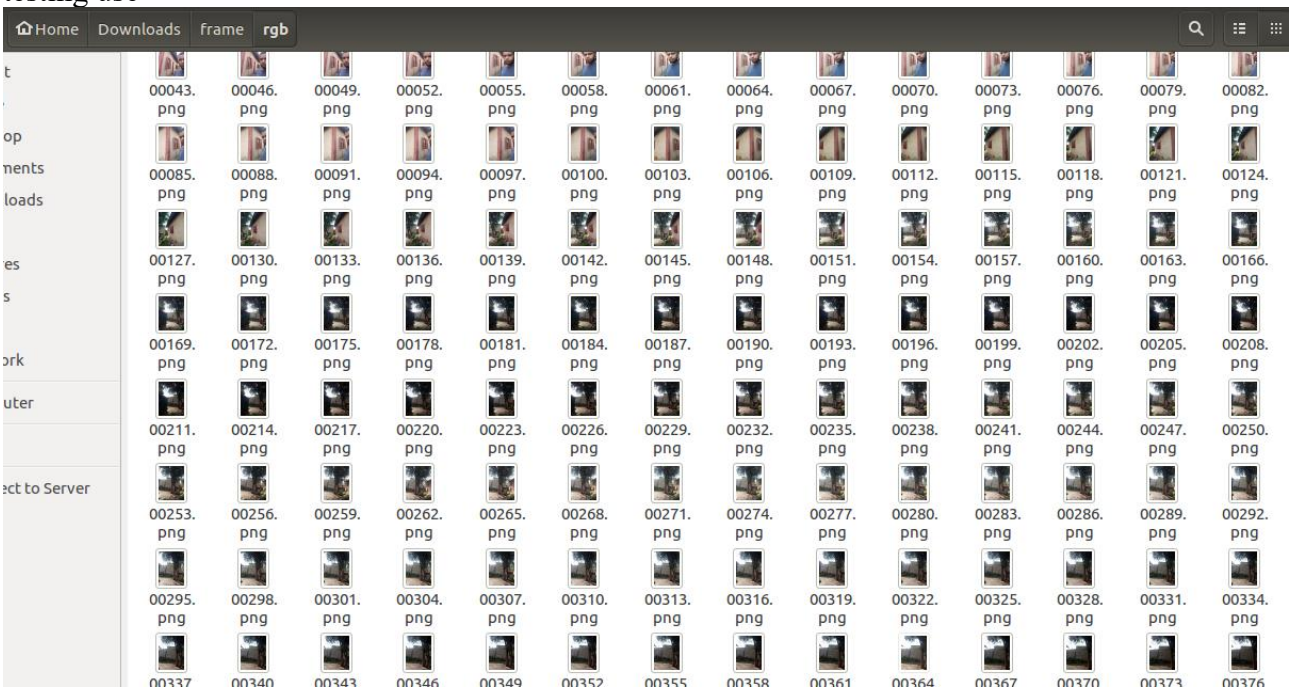


Figure5. 2.2 video changed to frame by frame

5.2. Extraction Step

In this step some IDE were used to implement and check the performance of the merged algorithm for the proposed work. In this software Opencv 3.4 were installed. Opencv is a library of

programing function mainly aimed at real time computer vision and c++ language were used. In this step trainee image were trained and stored for prediction by the test image. For the trainee and test image first due to color information doesn't help us identify important edges or other features the input image is converted to gray scale image then interest point was detected by the ORB algorithm.. Also Sample code for the conversion to gray scale image, Gaussian blurring and detection of interest point is listed as below.

.

- Gray_image1 = CV2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
- Gray_image2 = CV2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
- Blur1=cv2.GaussianBlur(Gray_image1, (15,15),0)
- Blur2=cv2.GaussianBlur(Gray_image2, (15,15),0)
- ORB=cv2.ORB_create ()
- KPTS1=ORB.detect(Blur1, None)
- KPTS2=ORB.detect(Blur2, None)

After detecting the interest point, describing those detected key point was proceeded by using minutiae algorithm identify the diplomat features called minutiae and to extract them By examining the local neighborhood of each ridge pixel using a 3x3 window, this method extracts the ridge endings and bifurcations from the skeleton image.

To extract the minutiae points, the Crossing Number (CN) method is used. By examining the local neighborhood of each ridge pixel using a 3x3 window, this method extracts the ridge endings and bifurcations from the skeleton image. The concept of Crossing Number (CN) is widely used for extracting the minutiae (Jain *et al*., 1997). The crossing number for a pixel P is

$$CN = \frac{1}{2}\sum_{i=1}^{8} | \mathrm{Pi} - \mathrm{Pi}+1 |$$

where, P_i is the binary pixel value in the neighborhood of P with P_i = (0 or1) and $P_9 = P_1$. For a pixel *P*, its eight neighboring pixels are scanned in an anticlockwise direction as follows:

P_4 P_3 P_2

P_5 P P_1

P_6 P_7 P_8

5Then the pixels are classified according to the property of their CN value. As shown in Figure 4.3, a ridge pixel with a CN of one corresponds to a ridge ending, and a CN of three corresponds to a bifurcation.

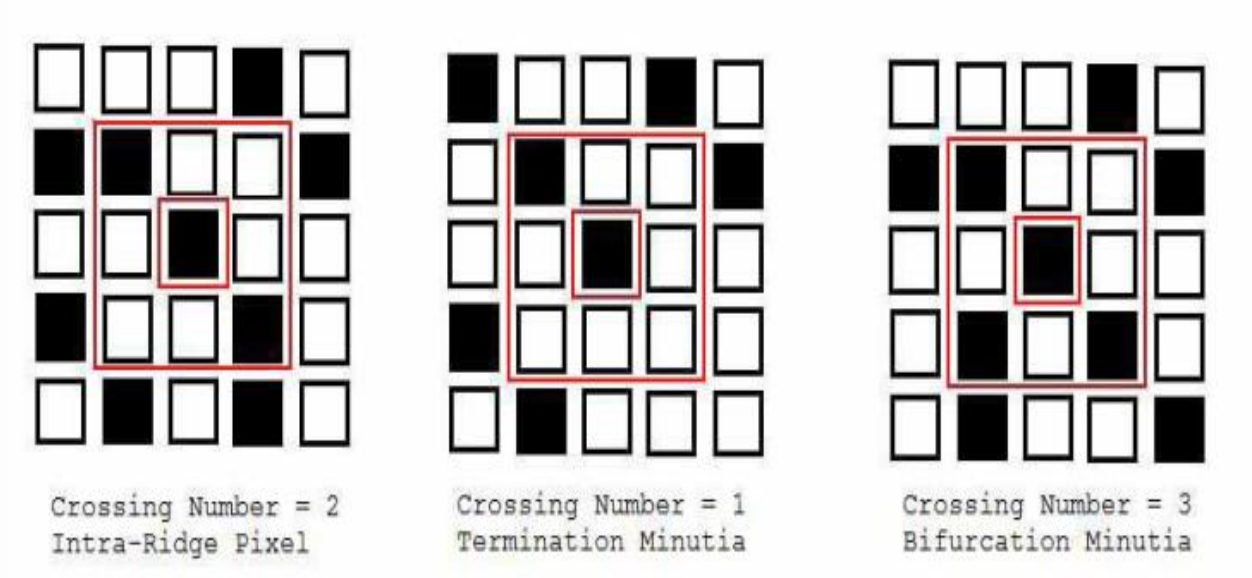


Figure5.3 Examples of a ridge ending and bifurcation pixel

so all the basic steps are :

Histogram Equalizer of Frame

Histogram is the estimation of the probability distribution of a particular type of data. An image histogram is a type of histogram which offers a graphical representation of the total distribution of the gray values in a digital image. The Variation of illumination is equalized to

equalizeHist(frame[x], frame_gray);

In a gray level image the probabilities assigned to each gray level can be given by the relation:

$$P_r(r_k) = \frac{n_k}{N} \quad 0 \le r_k \le 1, k = 0,1,2 \dots L-1$$

Extraction key frame and core value

Building the system has embedded a Place Recognition module based on DBoW2 for relocalization, in case of tracking failure (e.g. an occlusion) or for reinitialization in an already mapped scene, and for loop detection. The system maintains a covisibiliy graph that links any two keyframes observing common points and a minimum spanning tree connecting all keyframes. These graph structures allow to retrieve local windows of keyframes on the top of minute values , so that tracking and local mapping operate locally, allowing to work on large environments, and serve as structure for the pose-graph optimization performed when closing a loop The system uses the same ORB features and

minute core values for tracking, mapping and place recognition tasks. Our system handles monocular stereo key points and core values

Stereo key: points are defined by three coordinates xs = (uL; vL; uR), being (uL; vL) the coordinates on the left image and uR the horizontal coordinate in the right image. For stere cameras, we extract ORB in both images and for every left ORB we search for a match in the right image. This can be done very efficiently assuming stereo rectified images, so that epipolar lines are horizontal. We then generate the stereo keypoint with the coordinates of the left ORB and the horizontal coordinate of the right match, which is subpixel refined by patch correlation. For RGB-D cameras, we extract ORB features on the RGB image and, as proposed for each feature with coordinates (uL; vL) we transform its depth value d into a virtual right coordinate:

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

where fx is the horizontal focal length and b is the baseline between the structured light projector and the infrared camera.

As mentioned above core value or Crossing Number (CN) method is used. By examining the local neighborhood of each ridge pixel using a 3x3 window. And implementation will be satisfies in two steps below commonly known as Zhang and Suen algorithm:

Step 1:	Step2:
1) $2 \leq B(P(1)) \leq 6$ 1)	$2 \leq B(P(1)) \leq 6$
2)A(P(1)) = 1	2) A(P(1)) = 1
3) P(2) * P(4) * (P(6) = 0	3) P(2) * P(4) * (P(8) = 0
4) P(4) * P(6) * P(8) = 0	4) P(2) * P(6) * P(8) = 0

5.3. Map Building

Since navigating the robot autonomously through the indoor environment requires map, map is built. In this process simple office is built first which have one toilet, five offices and two corridors using simple materials. After building the office model the map is built using the Robot sensor and odometer data by ORB-SLAM2 algorithm. ORB-SLAM2 processes stereo and RGB-D inputs to estimate camera trajectory and build a map of the environment. The system is able to close loops, relocalize , and reuse its map in real time on standard CPUs with high accuracy and robustness.

Step to get the map :

Covisibility information between keyframes is very useful in several tasks of our system, and is represented as an undirected weighted graph as in [21]. Each node is a key point and an edge between

two keyframes exists if they share observations of the same map points, being the weight θ of the edge the number of common map points. In order to correct a loop we perform a pose graph optimization and minutia extraction [20] that distributes the loop closing error along the graph. In order not to include all the edges provided by the covisibility graph, which can be very dense, we propose to build an Essential Graph that retains all the nodes (keyframes), but less edges, still preserving a strong network that yields accurate results.[18] The system builds incrementally a spanning tree from the initial keyframe,(in our case key points) which provides a connected subgraph of the covisibility graph with minimal number of edges. When a new key point is inserted, it is included in the tree linked to the keyframe which shares most point observations, and when a keyframe is erased by the culling policy, the system updates the links affected by that keyframe. The Essential Graph contains the spanning tree, the subset of edges from the covisibility graph with high covisibility and the loop closure edges, resulting in a strong network of cameras.(Figure6.8)

5.4 Obstacle Avoidance

The ORB-SLAM2 make a map and saving it. The ROS kinetic package for raspberry pi3 gmapping node can also take in simple goals (x and y coordinates and rotation angle) and plan a path by avoiding obstacles. The process of navigating the robot through sensing the environment to make robot able to trajectory in safe path by avoiding obstacle and navigating the robot to the shortest path called global path planning [39]. First distance obstacle information and occupancy grid map obtained from ORB-SLAM2 is used by the costmap in order to calculate the obstacle area, possible collision area and robot movable area.to say obstacle is in front of the robot the distance between the robot and object should be within range of 2.5m if the distance exceeds 3.5m it will be considered as free space. The obstacle distance is chosen according to the maximum time required should be less than the radius of the robot times the area it exists. Then the costmap try to calculate the obstacle area, available pathes and tolerance by the following equation.

exp(-1.0 * cost_scaling_factor *(distance_from_obstacle - inscribed_radius)) *(254 - 1) [15]

So, first distance is set in local_costmap_params.yaml which is used for path planning and obstacle avoidances in limited area. One function of the angle calculation code:

```
 static float IC_Angle(const Mat& image, Point2f pt, const vector<int> & u_max)
{
int m_01 = 0, m_10 = 0;
const uchar* center = &image.at<uchar> (cvRound(pt.y), cvRound(pt.x));
// Treat the center line differently, v=0
```

```
for (int u = -HALF_PATCH_SIZE; u <= HALF_PATCH_SIZE; ++u)
m_10 += u * center[u];
// Go line by line in the circuI853lar patch
int step = (int)image.step1();
for (int v = 1; v <= HALF_PATCH_SIZE; ++v)
{
// Proceed over the two lines
int v_sum = 0;
int d = u_max[v];
for (int u = -d; u <= d; ++u)
{
int val_plus = center[u + v*step], val_minus = center[u - v*step];
v_sum += (val_plus - val_minus);
m_10 += u * (val_plus + val_minus);
}
m_01 += v * v_sum;
}
return fastAtan2((float)m_01, (float)m_10);
}
 const float factorPI = (float)(CV_PI/180.f);
 static void computeOrbDescriptor(const KeyPoint& kpt,
 const Mat& img, const Point* pattern,
 uchar* desc)
 {
 float angle = (float)kpt.angle*factorPI;
 float a = (float)cos(angle), b = (float)sin(angle);

 const uchar* center = &img.at<uchar>(cvRound(kpt.pt.y), cvRound(kpt.pt.x));
 const int step = (int)img.step;
 #define GET_VALUE(idx) \
 center[cvRound(pattern[idx].x*b + pattern[idx].y*a)*step + \
cvRound(pattern[idx].x*a - pattern[idx].y*b)]
```

5.5. Mapping Graph

Once we have an estimation of the camera pose and an initial set of feature matches, we can project the map into the frame and search more map point correspondences. To bound the complexity in large maps, we only project a local map by

1) Compute the map point projection x in the current frame. Discard if it lays out of the image bounds.

2) Compute the angle between the current viewing ray v and the map point mean viewing direction => n v · n < cos(60◦).

3) Compute the distance d from map point to camera center. Discard if it is out of the scale invariance region of the map point

4) Compute the scale in the frame by the ratio d = d min.

5) Compare the representative descriptor D of the MapPoint with the still unmatched ORB features in the frame, at the predicted scale, and near x, and associate the map point with the best match

After this steps the camera pose is finally optimized with all the map points found in the frame create sample map

The other step will be simulation, ROS developed in unit of nodes each response or decision the robot has to make have to be programmed separately as node. Finally, each node makes communication with each other to make the whole system work as required. So a master that manages connection information in a message communication between nodes is an essential element that must be run first in order to use ROS. Each node described in the above is programmed as node and configured the nodes as subscriber or publisher node in CMakeList.txt and Message.txt folder for the type of nodes and the message they used to exchange respectively in workspace package.

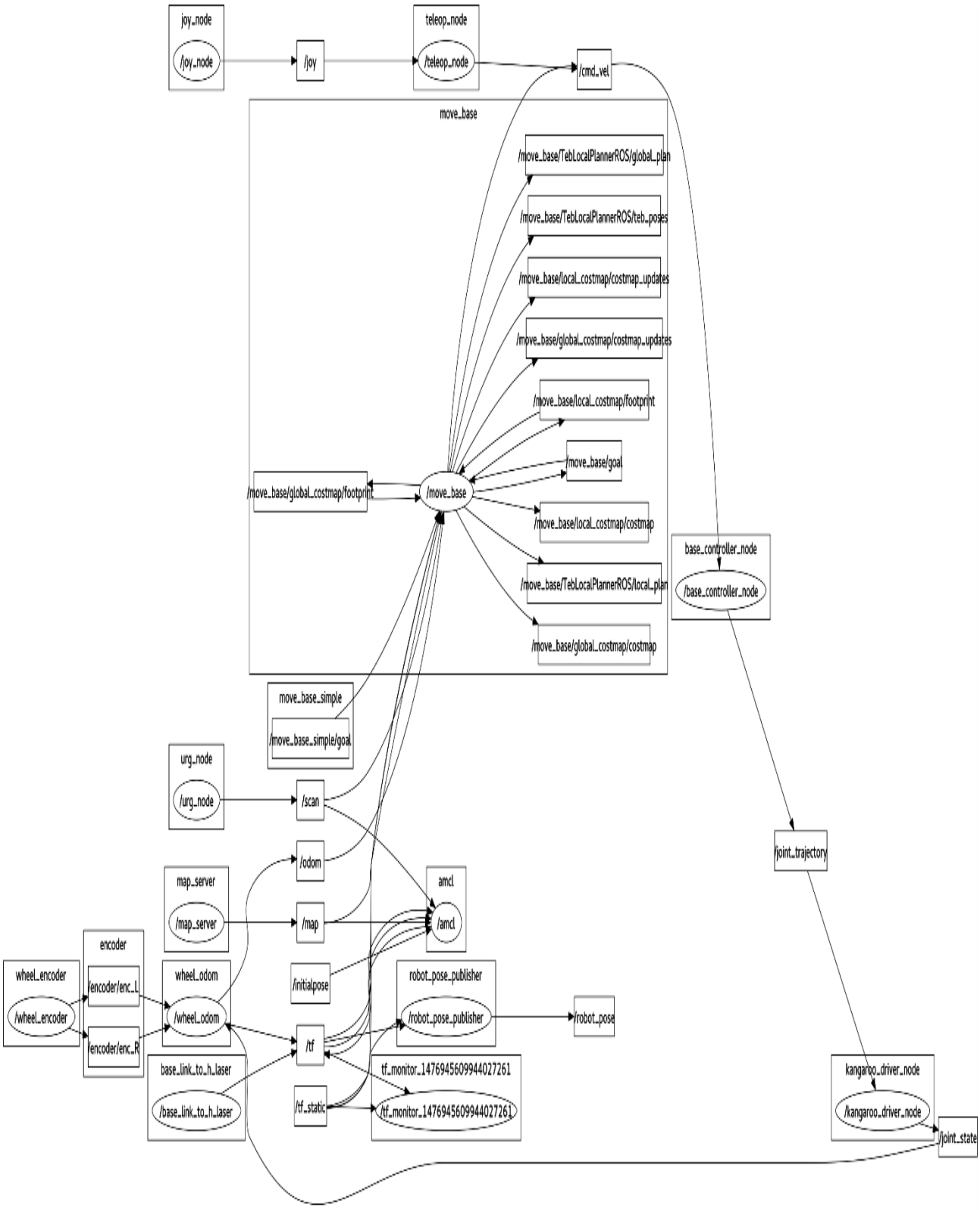


Figure5.4 Node graph made by Rqt that show the connection and exchange of message between subscriber and publisher node with their respective types of topic

Chapter six

Result, Evaluation and Discussion

This chapter presents the results of this work. It includes comparisons with other literature And show all the necessary outcomes.

6.1 Key Point extraction and formation with core value

To verify the proposed hydride algorithm is practicable and feasible, the paper compares the ORB-SLAM and ORB-SLAM with proposed algorithm separately for image recognition map building. In order to verify the effectiveness of the matching algorithm, experiments were carried out from three aspects: extracting feature points and extraction efficiency through different transformation of the images like intensity variation, rotation, scaling and selective characteristic by merging different images all together. . When evaluating the algorithm using extracting feature point it means that the feature point that detected by the algorithm should be at corner, T-juncton, ridge ending, ridge bifurcation or it should be blob point. If the detected feature point is in flat surface or other places, those point will be invalid. The threshold is 0.7 which controls the match points and the experimental environment is pycharm, clion and OpenCV 3.4. This paper will take two groups of images for validation and evaluate the performance of the algorithm. The first group of pictures called rgbd dataset from fireburg library, the size is 640 * 480 and with PNG format; the second set of pictures similar with the first but different in the scale, rotation and intensity with Additional of sample turtlebot ros robot simulation video was posted on YouTube link:

https://youtu.be/TPjxbLPhU1o

https://www.youtube.com/watch?v=vEdm16kB8b8

Step 1 .get testable environment change the video into frames

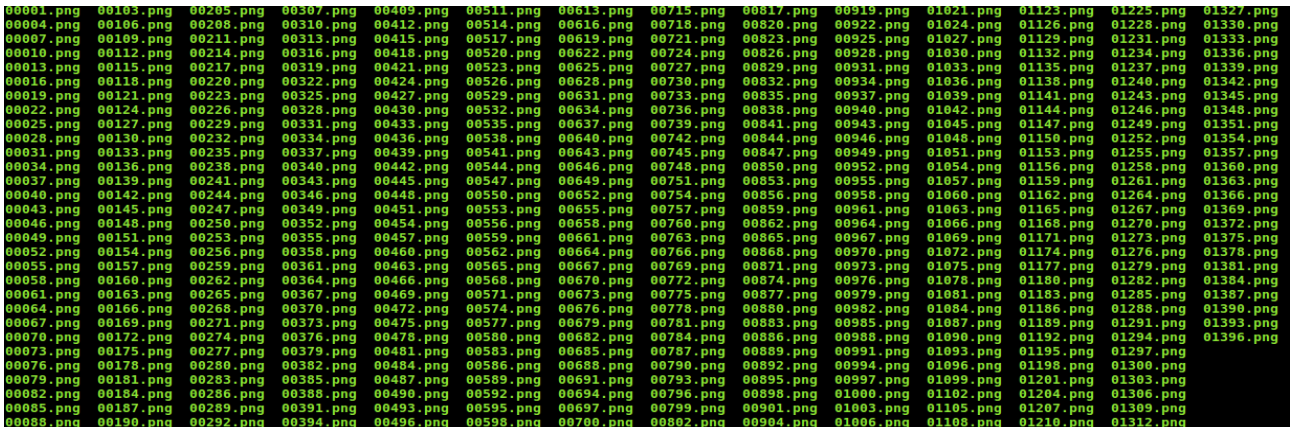


Figure6.1 the snip shoot of implementation of training image frame

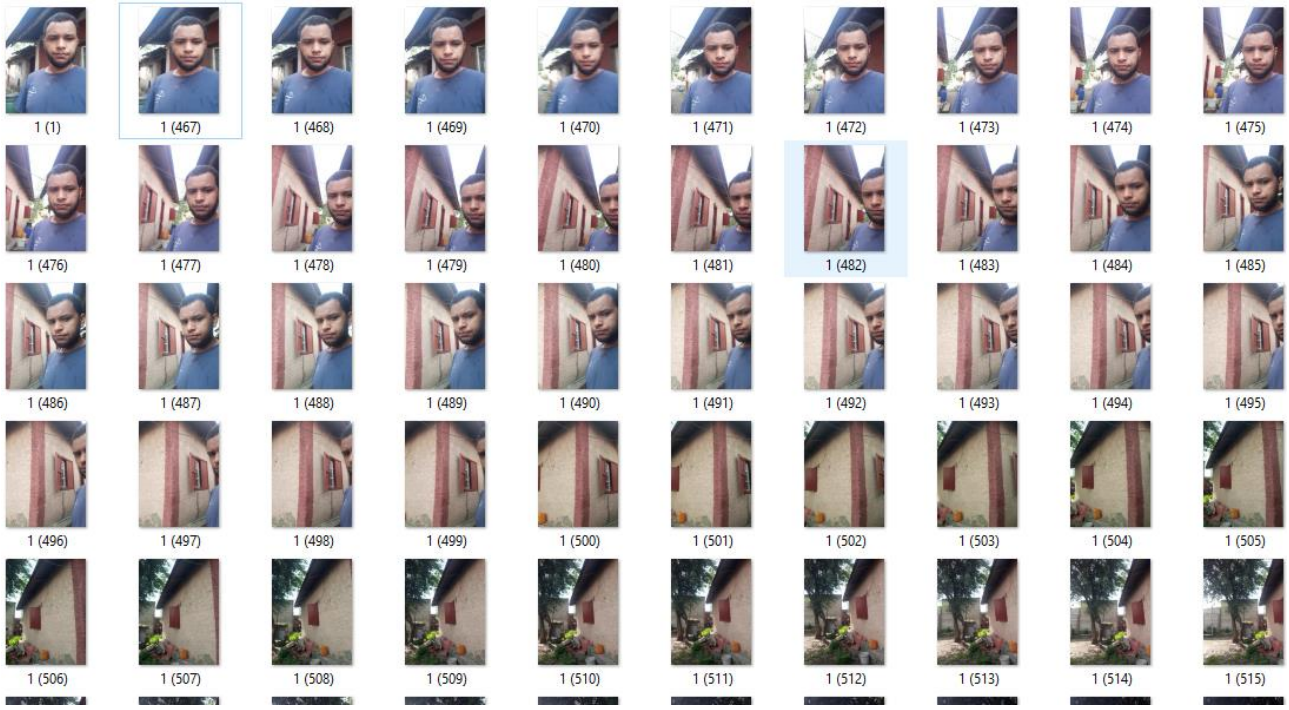


Figure6.2 the snip shoot of another implementation of training image frame

Figure6. 2Two original frame of two different video

Step 2 extraction using recommended algorithms
So we will change all image or the frame to gray for depth calculation

Figure6. 3 The process of resize and Conversion to gray scale of test frame

Key points are point that can uniquely identify the given image and used to match with their relative key point after described with respect to their orientation and vector .features such as Corners, blob and T-junction offers significant advantage for detecting corresponding image contents. Since such features can be extracted very efficient, can be accurately localized, can be remain stable over reasonably varying view points and they can allow an individual identification they are well suited for object recognition application. However flat surface and textures patches are nearly impossible to localize they can lead for increase the number of missed matched.

Figure6. 4 extraction using minutiae points (Key points) from consecutive frames

This key point matching from minutia was formed by characterized based on numerous minute peculiarities from minutiae points along with their relative position inside the consecutive frame as mention in chapter 5 on extraction step and we have to do of finding the best alignment between the compared frames before processing with the comparison step to maximize the score that quantified the quality of the matching

Figure6. 5 extraction using ORB slam (Key points)

How ORB slam get keypoint, perform first an initial feature matching with the previous frame and optimize the pose using motion-only BA. If the tracking is lost (e.g. due to occlusions or abrupt movements), the place recognition module is used to perform a global relocalization. Once there is an initial estimation of the camera pose and feature matching, a local visible map is retrieved using the covisibility graph of keyframes that is maintained by the system,

From the above image we extract key points we extract counter

```
-------
Start processing sequence ...
Images in the sequence: 798

Framebuffer with requested attributes not available. Using available framebuffer. You may see visual artifacts.New Map created with 92 points
Wrong initialization, reseting...
System Reseting
Reseting Local Mapper... done
Reseting Loop Closing... done
Reseting Database... done
New Map created with 103 points
^C
```

Figure6. 6 key points counter

Total image in sequence(number of image)	798	
	Key Point/matches	Time of detection
ORB-SLAM2	191	0.021(sec)
MINUTIAE	96(core values)	0.10(sec)
Both methods combined	220	0.028(sec)

Table 6.1 key point /matches

Sometimes even though it have some incorrect points it give more detailed information for the user reboot or any other machine.

6.1.1. In the case of high Intensity

In different intensity the minutiae and the ORB have their result as below

Figure6. 7 In different light of intensity(on high sun light around my home)

	Low intensity key point	High Intensity key point	Normal Intensity key point
ORB-SLAM2	186	185	191
MINUTIAE(core values)	82	94	96
Both methods combined	268	279	287

Table 6.2 intensity measurement

On more intensity the minutiae has low difference but when we have low brightness the minutiae need correction of noise and has low output However, from the result obtained ORB + minutiae perform accurate, more numbers of matching point and fast detection rate for the given image which varied with the intensity.

6.2. Computer vision and map formation

After analyzing the proposed algorithm separately on pycharm and Clion simulator, implementing the recognition algorithm in rgbd dataset from fireburg library was carried out. In this process the video which is recorded by rgbd camera and mounted on turtlebot3, was transferred to the master computer and conversion the image to compressed image was carried out to make the input image ready for recognition. So I little bit detail map was sated. Which have more detailed key point so by moving the RGB camera we can design more detailed output of the environment.

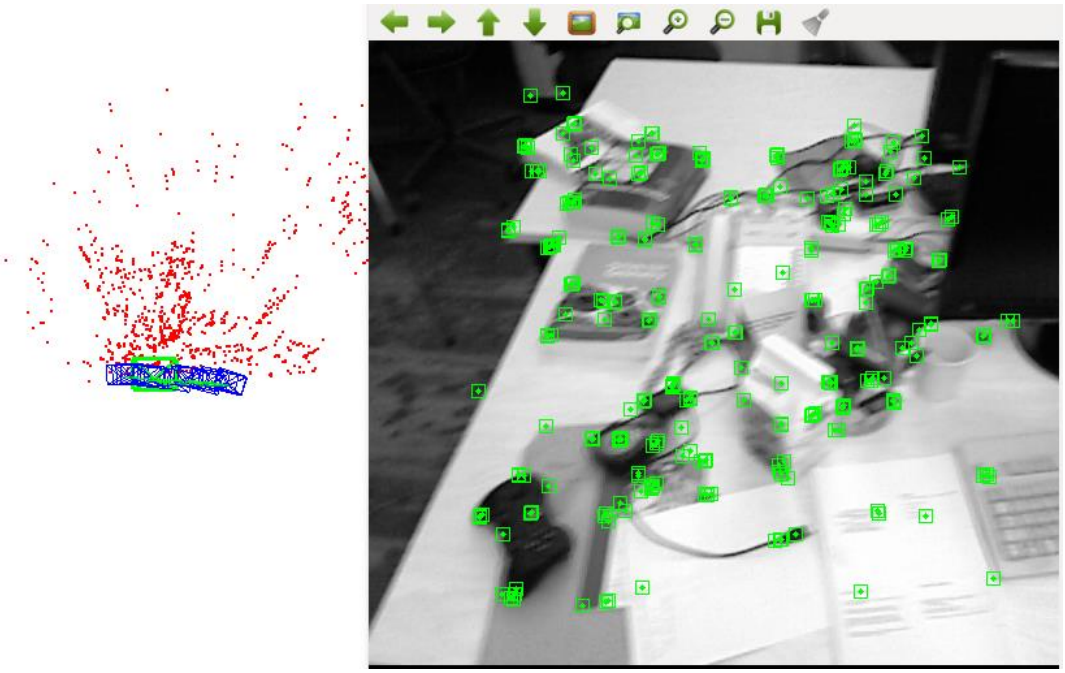


Figure6. 8 from combined key point creating estimated map

The local mapping processes new core values, keyframes and performs local BA to achieve reconstruction in the surroundings of the camera pose. New correspondences for unmatched key point in the new keyframes and core points are searched in connected in the covisibility graph to triangulate new point. covisibility graph means Each node is a keyframe and an edge between two keyframes exists if they share observations of the same key points at explain in chapter 5 on map building

6.3. Computer vision with ROS and Test Bed

In our SIG laboratory we formed sample implementation test bed from that we try to make and represent that test bed on virtual ROS to understand our hybrid system work in real time environment

Figure6.9 SIG test bed

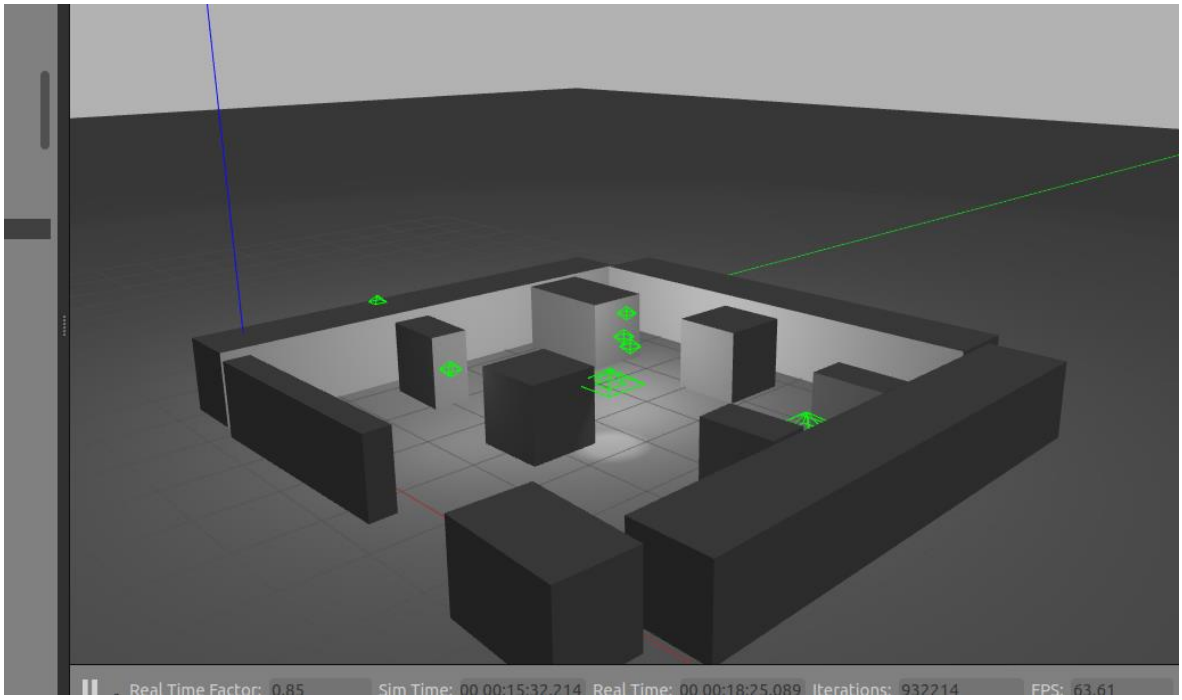


Figure6.10 ROS sample test bed on gazebo

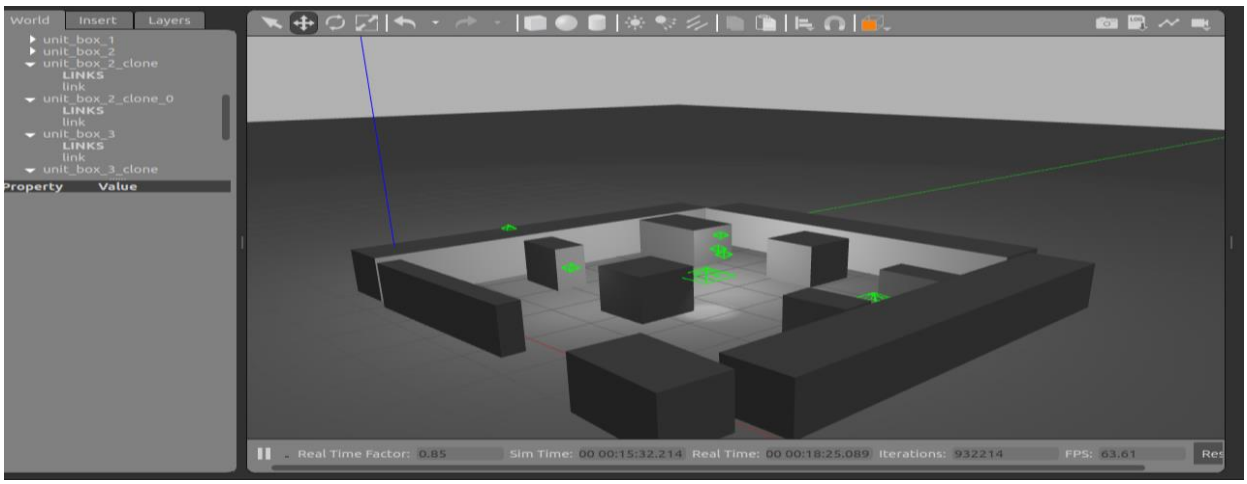


Figure6.11 sample test bed

From the above test bed we try to form and implement our hybrid algorithm with the help ROS node gmapping ,that combined both LIDAR and camera out put video based odometry data and it is equipped with a horizontally-mounted, fixed output for riviz

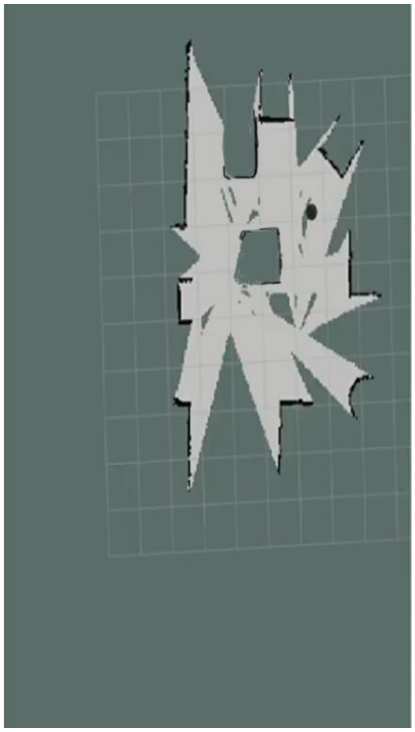


Figure6.12 rough map output with combinig LADAR in ROS with gmaping node

The screen shots were captured at different stages of the camera movement to its goal. In the first part the camera started is facing upwards left corner with respect to the frame. We can see that there is some obstacle directly a head of it (pink blob) and the goal is beyond it. Then node generates a path around this obstacle, using the information from the gmaping and costmap, which is represented by a thin black line. First distance between the robot and box is obtained from the camera sensor then

using position of the robot (camera), obstacle information and occupancy grid map obtained from SLAM, the costmap calculate the obstacle area, possible collision area and robot movable area. Finally, from the figure above using DWA publish the velocity search space with the translational velocity v and the rotational velocity ω as axes to make the robot to accelerate to destination free from anycollision by obstacle.

So from the above evaluation of the result we can discussed that the proposed algorithm has high number of reliable detected point than the ORB-SLAM algorithm alone. After implementing with robot the proposed algorithm performs dramatic improvement in detecting map with high accuracy. Even the proposed algorithm with in different image transformation is distinctive and reliable. so, from the above evaluation the proposed algorithm is fast and robust for any image transformation therefore research question one and two is answered. On the other hand, for obstacle avoidance the robot calculates the collusion area, collision free area and obstacle area by costmap and DWA choose the speed that the robot should have to move by avoiding the obstacle in order to reach to the desire destination without any collusion and without ceasing for long moment. So from this, research question three is answered

Chapter Seven
Conclusion and Future work

7.1. Conclusion

In order to navigate the robot safe, the robots, need to have a set of capabilities that allows them to move and interact with a real environment. Vision is the most important for safely moving and interacting with the world. There are mini and micro robot which only got one camera to discover the environment so such future needs to be applied. Since navigating the robot through unfamiliar environment requires understanding of details and other useful visual information, the robot has to have a method for detecting recognizing more than ORB-SLAM algorithm with high speed through different difficulties.

A variety of hardware and software systems were integrated to create a working prototype robot that used to detect and navigation within indoor environment. The work presented in this thesis is twofold: first, a model has been developed and trained using computer vision to assist with successful indoor navigation toward defined goals; in this process fast and accurate recognition algorithm have been proposed that can reduce the recognition time or that is used to increase the speed of recognition.

Due to its accuracy, robustness in scale variation, translation, to occlusion, to clutter and its fast map localization and recognition of the environment ORB-SLAM algorithm and minutiae algorithm has been chosen. However, since the image processing unit of the proposed system should be fast and accurate ORB and is integrated and tested for their performance. ORB algorithm perform fast matching rate and since it uses FAST algorithm for detecting interest point the detected feature will be accurate and fast. So for detection of interest point ORB is used and for description of those detected interest point minutiae algorithm were used.

7.2. Future work

For more than two decade different approach and algorithms are used for future based approach and Recognition. Each studies are almost works in terms of accuracy, But no more studies related to ORB-SLAM algorithm and minutiae algorithm is worked accuracy with respect to speed to be recognized.

My recommendation for Researcher the parameter of Detection and Recognition is not only Accuracy and Error. The speed of Recognition is also very necessary in terms of timeliness improvement also images in training and testing face have high noise it is difficult to recognize so enhancement improvement.need So further studies are needed to work such as:

- Including other features such as: graph map optimization enhancement in modeling the classifier to achieve a higher detection rate as well as reducing the false negative and false positive rates
- Add machine learning AI to understand the environment more
- Speed of recognition keyframe.

References

[1] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash," 2015.

[2] S. Zabihi, "Detection and Recognition of Traffic Signs Inside".,Published 2017, Engineering, Computer Science2017,IEEE Intelligent Vehicles Symposium

[3] H. Yilma, "Challenges and Prospects of Traffic Management Practices of Addis Ababa City Administration," A.A.U, 2014.

[4] T. A. Abdi, "Road Crashes in Addis Ababa, Ethiopia: Empirical Findings," AN INTERNATIONAL MULTI-DISCIPLINARY JOURNAL,, 2017.

[5] A. Aga, "The Ethiopian Journal of Health Development," Avoidable visual impairment among elderly people in a Slum of Addis Ababa.

[6] "Country Background Living with Visual Impairment in Ethiopia," [Online]. Available: http://www.together-et.org/en/. [Accessed 26 Septmeber 2018].

[7] Alejandra Carolina Hernandez, Clara Gomez, Jonathan Crespo, 28 Jul 2016. [Online].

[8] Billinghurst M, Clark A, Lee G (2015) A survey of augmented reality. Found
Trends Human-Computer Interact 8(2-3):73–272

[9]Takafumi Taketomi, Hideaki Uchiyama and Sei Ikeda “Visual SLAM algorithms: a survey from 2010 to 2016”,IPSJ Transactions on Computer Vision and Applications,2017

[10] Engel J, Sturm J, Cremers D (2012) Camera-based navigation of a low-cost quadrocopter. In: Proceedings of International Conference on Intelligent Robots and Systems. Pp 2815–2821

[11] Dissanayake, M.W.M.G.; Newman, P.; Clark, S.; Durrant-Whyte, H.F.; Csorba, M.A. Solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Autom. 2001, 17, 229–241.

[12]Davison AJ (2003) Real-time simultaneous localisation and mapping with a single camera. In: Proceedings of International Conference on Computer Vision. Pp 1403–1410

[13]Davison AJ, Reid ID, Molton ND, Stasse O (2007) Monoslam: real-time single camera SLAM. Pattern Anal Mach Intell IEEE Trans 29(6):1052–1067

[14]Bundle adjustment a modern synthesis. In: Triggs B, McLauchlan PF, Hartley RI, Fitzgibbon AW (eds) (2000) Vision algorithms: theory and practice. pp 298–372

[15]Klein G, Murray DW (2007) Parallel tracking and mapping for small AR workspace. In: Proceedings of International Symposium on Mixed and Augmented Reality. Pp 225–234

[16] Nistér D (2004) An efficient solution to the five-point relative pose problem. Pattern Anal Mach Intell IEEE Trans 26(6):756–770

[17] Williams B, Klein G, Reid I (2007) Real-time SLAM re-localization. In: Proceedings of International Conference on Computer Vision. Pp 1–8

[18]Castle R, Klein G, Murray DW (2008) Video-rate localization in multiple maps for wearable augmented reality. In: 2008 12th IEEE International Symposium on Wearable Computers. pp 15–22 do i:10.1109/ISWC.2008.4911577

[19]Klein G, Murray DW (2009) Parallel tracking and mapping on a camera phone. In: Proceedngs of International Symposium on Mixed and Augmented Reality. pp 83–86

[20] Newcombe RA, Davison AJ (2010) Live dense reconstruction with a single moving camera. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp 1498–1505

[21] Pradeep V, Rhemann C, Izadi S, Zach C, Bleyer M, Bathiche S (2013) MonoFusion: real-time 3D reconstruction of small scenes with a single web camera. In: Proceedngs of International Symposium on Mixed and Augmented Reality. Pp 83–88

[22]Whelan T, Kaess M, Johannsson H, et al. Real-time large-scale dense RGB-D SLAM with volumetric fusion[J]. The International Journal of Robotics Research, 2015, 34(4-5): 598-626.

[23]S. Kim, D. Lee, J. Kim, “Algorithm for Detection and Elimination of False Minutiae in Fingerprint Images”, International Conference on Audio and Video-Based Biometric Person Authentication, Halmstad, Sweden, vol. 2091, pp 235-240, 2001

[24] G. Panchal, A. Ganatra, Y P Kosta, “Efficient Feature Extraction and Selection Techniques for Classification” , International Journal of Computer Science and Emerging Technology (IJCSET), Vol. 2(2),2011.

[25] M. Bandur, B. Popovic, A. Raicevic, D. Randelovic, “Improving minutiae extraction in fingerprint images through robust enhancement”, International Conference on Computer and Computational Sciences (ICCCS), Belgrade, Serbia, pp. 506-509, 2015

[26] S. Chakraborty, K. Rao, "Fingerprint enhancement by directional filtering", IEEE 9th International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology, Phetchaburi, Thailand, pp. 1-4, 2012.

[27] B. Popovic, M. Bandjur, A. Raicevic, D. Randelovic, “Different methods for fingerprint image orientation estimation”, IEEE Telecommunication Forum, Belgrade, Serbia, pp.662-665, 2012.

[28]D. Maltoni, D. Maio, A. Jain, S. Prabhakar, Handbook of Fingerprint Recognition, Springer-Verlag, London, 2009

[29]Jegnaw Fentahun Zeggeye and Yaregal Assabie, "Automatic Recognition and Counterfeit Detection of Ethiopian Paper Currency," I.J. Image, Graphics and Signal Processing, pp. 1- 9, 2016.

[30]Jongan Park, Waqas Rasheed and Junquk beak , "Robot Navigation Using Camera by Identifying Arrow Signs," IEEE, pp. 1-28, 2008.

[31]F. Jusuf, "Auto-Navigation For Robots. Implementation of ROS," Metropolia, pp. 1-25,2016

[32] Raul Mur-Artal and Juan D. Tard ´ os,” ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”.IEEE 2017

[33] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim, ROS Robot Programming, 2017.

[34] Juan Fasola, Manuela Velso and Paul Rybski, "Fast Goal Navigation with Obstacle Avoidance Using a Dynamic Local Visual Model," *Research gate,* pp. 1-10, 2004.

[35] Lei Yu1, Zhixin Yu1 and Yan Gong2 , "An Improved ORB Algorithm of Extracting and Matching Features," *International Journal of Signal Processing,* vol. 8, pp. 1-126, 2015.

[36] AK Assad, Mashruf Chowdhury and Yanik Porto , Robotics Engineering 2: ROS-Turtlebot Motion Control and Navigation, 2015.

[37] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim, ROS Robot Programming, 2017.

[38] Alexander Mordvintsev & Abid K., "Opencv-Python Tutorials," 2013. [Online]. Available: opencv-pythontutroals.readthedocs.io/en/latest/py_tutorials/py_fe 2d/py_matcher/py_matcher.html. [Accessed 12 August 2018].

[39] HuiwenZhang,1,2 XiaoningHan,1,2 MingliangFu,1,2 andWeijiaZhou1 , "Robot Obstacle Avoidance Learning Based on Mixture Models," *Journal of Robotics* , vol. l, pp. 1-14, 2016

[40] Jihong Min, Jungho Kim, Hyeongwoo Kim, Kiho Kwak and In So Kweon, " Hybrid Vision-based SLAM Coupled with Moving Object Tracking ",IEEE International Conference on Robotics & Automation (ICRA),june 7,2014

[41] Moreno, F. A., Blanco, J. L., & Gonzalez, J. (2009). Stereo vision specific models for particle filter-based SLAM. Robotics and Autonomous Systems, 57(9), 955-970

[42] Al-Mutib, Khalid N., Et Al. "Stereo vision SLAM based indoor autonomous mobile robot navigation." Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on. IEEE, 2014

[43] Brand, C., Schuster, M. J., Hirschmüller, H., & Suppa, M. (2014, September). Stereo-vision based obstacle mapping for indoor/outdoor SLAM. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on (pp. 1846-1853). IEEE.

[44] Wang, Y., Huang, S., Xiong, R., & Wu, J. (2016). A framework for multi-session RGBD SLAM in low dynamic workspace environment. CAAI Transactions on Intelligence Technology, 1(1), 90-103.

[45] Gao, X., & Zhang, T. (2015). Robust RGB-D simultaneous localization and mapping using planar point features. Robotics and Autonomous Systems, 72, 1-14.

[46] Raguram, R., Chum, O., Pollefeys, M., Matas, J., & Frahm, J. M. (2013). USAC: a universal framework for random sample consensus. IEEE transactions on pattern analysis and machine intelligence, 35(8), 2022- 2038.

[47] Lin, L. H., Lawrence, P. D., & Hall, R. (2013). Robust outdoor stereo vision SLAM for heavy machine rotation sensing. Machine vision andapplications, 1-22

[48] Jabulani K. Makhubela, Tranos Zuva and Olusanya Yinka. Agunbiad eFaculty of Applied and Computer Sciences A Review on Vision Simultaneous Localization and Mapping (VSLAM) ,IEEE.2018 (1-4)

Appendix

Sample code for recognition I

Map formation

```
#include "Map.h"

#include<mutex>

namespace ORB_SLAM2
{

Map::Map():mnMaxKFid(0),mnBigChangeIdx(0)
{
}

void Map::AddKeyFrame(KeyFrame *pKF)
{
   unique_lock<mutex> lock(mMutexMap);
   mspKeyFrames.insert(pKF);
   if(pKF->mnId>mnMaxKFid)
      mnMaxKFid=pKF->mnId;
}

void Map::AddMapPoint(MapPoint *pMP)
{
   unique_lock<mutex> lock(mMutexMap);
   mspMapPoints.insert(pMP);
}

void Map::EraseMapPoint(MapPoint *pMP)
{
   unique_lock<mutex> lock(mMutexMap);
   mspMapPoints.erase(pMP);
```

```
    // TODO: This only erase the pointer.
    // Delete the MapPoint
}

void Map::EraseKeyFrame(KeyFrame *pKF)
{
    unique_lock<mutex> lock(mMutexMap);
    mspKeyFrames.erase(pKF);

    // TODO: This only erase the pointer.
    // Delete the MapPoint
}

void Map::SetReferenceMapPoints(const vector<MapPoint *> &vpMPs)
{
    unique_lock<mutex> lock(mMutexMap);
    mvpReferenceMapPoints = vpMPs;
}

void Map::InformNewBigChange()
{
    unique_lock<mutex> lock(mMutexMap);
    mnBigChangeIdx++;
}

int Map::GetLastBigChangeIdx()
{
    unique_lock<mutex> lock(mMutexMap);
    return mnBigChangeIdx;
}

vector<KeyFrame*> Map::GetAllKeyFrames()
{
    unique_lock<mutex> lock(mMutexMap);
    return vector<KeyFrame*>(mspKeyFrames.begin(),mspKeyFrames.end());
```

```
}

vector<MapPoint*> Map::GetAllMapPoints()
{
   unique_lock<mutex> lock(mMutexMap);
   return vector<MapPoint*>(mspMapPoints.begin(),mspMapPoints.end());
}

long unsigned int Map::MapPointsInMap()
{
   unique_lock<mutex> lock(mMutexMap);
   return mspMapPoints.size();
}

long unsigned int Map::KeyFramesInMap()
{
   unique_lock<mutex> lock(mMutexMap);
   return mspKeyFrames.size();
}

vector<MapPoint*> Map::GetReferenceMapPoints()
{
   unique_lock<mutex> lock(mMutexMap);
   return mvpReferenceMapPoints;
}

long unsigned int Map::GetMaxKFid()
{
   unique_lock<mutex> lock(mMutexMap);
   return mnMaxKFid;
}

void Map::clear()
{
```

```
    for(set<MapPoint*>::iterator  sit=mspMapPoints.begin(),  send=mspMapPoints.end();  sit!=send;
sit++)
        delete *sit;

    for(set<KeyFrame*>::iterator sit=mspKeyFrames.begin(), send=mspKeyFrames.end(); sit!=send;
sit++)
        delete *sit;

    mspMapPoints.clear();
    mspKeyFrames.clear();
    mnMaxKFid = 0;
    mvpReferenceMapPoints.clear();
    mvpKeyFrameOrigins.clear();
}
#include "MapPoint.h"
#include "ORBmatcher.h"

#include<mutex>

namespace ORB_SLAM2
{

long unsigned int MapPoint::nNextId=0;
mutex MapPoint::mGlobalMutex;

MapPoint::MapPoint(const cv::Mat &Pos, KeyFrame *pRefKF, Map* pMap):
    mnFirstKFid(pRefKF->mnId),          mnFirstFrame(pRefKF->mnFrameId),          nObs(0),
mnTrackReferenceForFrame(0),
    mnLastFrameSeen(0),          mnBALocalForKF(0),          mnFuseCandidateForKF(0),
mnLoopPointForKF(0), mnCorrectedByKF(0),
    mnCorrectedReference(0),      mnBAGlobalForKF(0),      mpRefKF(pRefKF),      mnVisible(1),
mnFound(1), mbBad(false),
    mpReplaced(static_cast<MapPoint*>(NULL)),      mfMinDistance(0),      mfMaxDistance(0),
mpMap(pMap)
{
```

```
    Pos.copyTo(mWorldPos);
    mNormalVector = cv::Mat::zeros(3,1,CV_32F);

    // MapPoints can be created from Tracking and Local Mapping. This mutex avoid conflicts with
id.
    unique_lock<mutex> lock(mpMap->mMutexPointCreation);
    mnId=nNextId++;
}

MapPoint::MapPoint(const cv::Mat &Pos, Map* pMap, Frame* pFrame, const int &idxF):
    mnFirstKFid(-1), mnFirstFrame(pFrame->mnId), nObs(0), mnTrackReferenceForFrame(0),
mnLastFrameSeen(0),
    mnBALocalForKF(0), mnFuseCandidateForKF(0),mnLoopPointForKF(0),
mnCorrectedByKF(0),
    mnCorrectedReference(0), mnBAGlobalForKF(0), mpRefKF(static_cast<KeyFrame*>(NULL)),
mnVisible(1),
    mnFound(1), mbBad(false), mpReplaced(NULL), mpMap(pMap)
{
    Pos.copyTo(mWorldPos);
    cv::Mat Ow = pFrame->GetCameraCenter();
    mNormalVector = mWorldPos - Ow;
    mNormalVector = mNormalVector/cv::norm(mNormalVector);

    cv::Mat PC = Pos - Ow;
    const float dist = cv::norm(PC);
    const int level = pFrame->mvKeysUn[idxF].octave;
    const float levelScaleFactor =  pFrame->mvScaleFactors[level];
    const int nLevels = pFrame->mnScaleLevels;

    mfMaxDistance = dist*levelScaleFactor;
    mfMinDistance = mfMaxDistance/pFrame->mvScaleFactors[nLevels-1];

    pFrame->mDescriptors.row(idxF).copyTo(mDescriptor);
```

```
    // MapPoints can be created from Tracking and Local Mapping. This mutex avoid conflicts with
id.
    unique_lock<mutex> lock(mpMap->mMutexPointCreation);
    mnId=nNextId++;
}

void MapPoint::SetWorldPos(const cv::Mat &Pos)
{
    unique_lock<mutex> lock2(mGlobalMutex);
    unique_lock<mutex> lock(mMutexPos);
    Pos.copyTo(mWorldPos);
}

cv::Mat MapPoint::GetWorldPos()
{
    unique_lock<mutex> lock(mMutexPos);
    return mWorldPos.clone();
}

cv::Mat MapPoint::GetNormal()
{
    unique_lock<mutex> lock(mMutexPos);
    return mNormalVector.clone();
}

KeyFrame* MapPoint::GetReferenceKeyFrame()
{
    unique_lock<mutex> lock(mMutexFeatures);
    return mpRefKF;
}

void MapPoint::AddObservation(KeyFrame* pKF, size_t idx)
{
    unique_lock<mutex> lock(mMutexFeatures);
    if(mObservations.count(pKF))
```

```
        return;
    mObservations[pKF]=idx;

    if(pKF->mvuRight[idx]>=0)
        nObs+=2;
    else
        nObs++;
}

void MapPoint::EraseObservation(KeyFrame* pKF)
{
    bool bBad=false;
    {
        unique_lock<mutex> lock(mMutexFeatures);
        if(mObservations.count(pKF))
        {
            int idx = mObservations[pKF];
            if(pKF->mvuRight[idx]>=0)
                nObs-=2;
            else
                nObs--;

            mObservations.erase(pKF);

            if(mpRefKF==pKF)
                mpRefKF=mObservations.begin()->first;

            // If only 2 observations or less, discard point
            if(nObs<=2)
                bBad=true;
        }
    }

    if(bBad)
        SetBadFlag();
```

```
}

map<KeyFrame*, size_t> MapPoint::GetObservations()
{
   unique_lock<mutex> lock(mMutexFeatures);
   return mObservations;
}

int MapPoint::Observations()
{
   unique_lock<mutex> lock(mMutexFeatures);
   return nObs;
}

void MapPoint::SetBadFlag()
{
   map<KeyFrame*,size_t> obs;
   {
      unique_lock<mutex> lock1(mMutexFeatures);
      unique_lock<mutex> lock2(mMutexPos);
      mbBad=true;
      obs = mObservations;
      mObservations.clear();
   }
   for(map<KeyFrame*,size_t>::iterator mit=obs.begin(), mend=obs.end(); mit!=mend; mit++)
   {
      KeyFrame* pKF = mit->first;
      pKF->EraseMapPointMatch(mit->second);
   }

   mpMap->EraseMapPoint(this);
}

MapPoint* MapPoint::GetReplaced()
{
```

```
    unique_lock<mutex> lock1(mMutexFeatures);
    unique_lock<mutex> lock2(mMutexPos);
    return mpReplaced;
}

void MapPoint::Replace(MapPoint* pMP)
{
    if(pMP->mnId==this->mnId)
        return;

    int nvisible, nfound;
    map<KeyFrame*,size_t> obs;
    {
        unique_lock<mutex> lock1(mMutexFeatures);
        unique_lock<mutex> lock2(mMutexPos);
        obs=mObservations;
        mObservations.clear();
        mbBad=true;
        nvisible = mnVisible;
        nfound = mnFound;
        mpReplaced = pMP;
    }

    for(map<KeyFrame*,size_t>::iterator mit=obs.begin(), mend=obs.end(); mit!=mend; mit++)
    {
        // Replace measurement in keyframe
        KeyFrame* pKF = mit->first;

        if(!pMP->IsInKeyFrame(pKF))
        {
            pKF->ReplaceMapPointMatch(mit->second, pMP);
            pMP->AddObservation(pKF,mit->second);
        }
        else
        {
```

```
            pKF->EraseMapPointMatch(mit->second);
        }
    }
    pMP->IncreaseFound(nfound);
    pMP->IncreaseVisible(nvisible);
    pMP->ComputeDistinctiveDescriptors();

    mpMap->EraseMapPoint(this);
}

bool MapPoint::isBad()
{
    unique_lock<mutex> lock(mMutexFeatures);
    unique_lock<mutex> lock2(mMutexPos);
    return mbBad;
}

void MapPoint::IncreaseVisible(int n)
{
    unique_lock<mutex> lock(mMutexFeatures);
    mnVisible+=n;
}

void MapPoint::IncreaseFound(int n)
{
    unique_lock<mutex> lock(mMutexFeatures);
    mnFound+=n;
}

float MapPoint::GetFoundRatio()
{
    unique_lock<mutex> lock(mMutexFeatures);
    return static_cast<float>(mnFound)/mnVisible;
}
```

```
void MapPoint::ComputeDistinctiveDescriptors()
{
    // Retrieve all observed descriptors
    vector<cv::Mat> vDescriptors;

    map<KeyFrame*,size_t> observations;

    {
        unique_lock<mutex> lock1(mMutexFeatures);
        if(mbBad)
            return;
        observations=mObservations;
    }

    if(observations.empty())
        return;

    vDescriptors.reserve(observations.size());

    for(map<KeyFrame*,size_t>::iterator mit=observations.begin(), mend=observations.end();
mit!=mend; mit++)
    {
        KeyFrame* pKF = mit->first;

        if(!pKF->isBad())
            vDescriptors.push_back(pKF->mDescriptors.row(mit->second));
    }

    if(vDescriptors.empty())
        return;

    // Compute distances between them
    const size_t N = vDescriptors.size();

    float Distances[N][N];
```

```
    for(size_t i=0;i<N;i++)
    {
        Distances[i][i]=0;
        for(size_t j=i+1;j<N;j++)
        {
            int distij = ORBmatcher::DescriptorDistance(vDescriptors[i],vDescriptors[j]);
            Distances[i][j]=distij;
            Distances[j][i]=distij;
        }
    }

    // Take the descriptor with least median distance to the rest
    int BestMedian = INT_MAX;
    int BestIdx = 0;
    for(size_t i=0;i<N;i++)
    {
        vector<int> vDists(Distances[i],Distances[i]+N);
        sort(vDists.begin(),vDists.end());
        int median = vDists[0.5*(N-1)];

        if(median<BestMedian)
        {
            BestMedian = median;
            BestIdx = i;
        }
    }

    {
        unique_lock<mutex> lock(mMutexFeatures);
        mDescriptor = vDescriptors[BestIdx].clone();
    }
}

cv::Mat MapPoint::GetDescriptor()
{
```

```
    unique_lock<mutex> lock(mMutexFeatures);
    return mDescriptor.clone();
}

int MapPoint::GetIndexInKeyFrame(KeyFrame *pKF)
{
    unique_lock<mutex> lock(mMutexFeatures);
    if(mObservations.count(pKF))
        return mObservations[pKF];
    else
        return -1;
}

bool MapPoint::IsInKeyFrame(KeyFrame *pKF)
{
    unique_lock<mutex> lock(mMutexFeatures);
    return (mObservations.count(pKF));
}

void MapPoint::UpdateNormalAndDepth()
{
    map<KeyFrame*,size_t> observations;
    KeyFrame* pRefKF;
    cv::Mat Pos;
    {
        unique_lock<mutex> lock1(mMutexFeatures);
        unique_lock<mutex> lock2(mMutexPos);
        if(mbBad)
            return;
        observations=mObservations;
        pRefKF=mpRefKF;
        Pos = mWorldPos.clone();
    }

    if(observations.empty())
```