HYBRID ALGORITHM FOR FAST DETECTION AND RECOGNITION OF SIGNAGE AND OBSTACLE AVOIDANCE FOR ROBOT NAVIGATION

Noah Diriba

A thesis submitted to the department of Computer Science and Engineering, school of Electrical Engineering and Computing in partial fulfillment of Master's degree in Computer Science and Engineering

Office of Graduate Studies
Adama Science and Technology University
Adama, Oromia, Ethiopia

Adama
September, 2018

A HYBRID ALGORITHM FOR FAST DETECTION AND RECOGNITION OF SIGNAGE AND OBSTACLE AVOIDANCE FOR ROBOT NAVIGATION

By Noah Diriba

Advisor: Prof.Yun Koo Chung

A thesis submitted to the department of Computer Science and Engineering, school of Electrical Engineering and Computing in partial fulfillment of master's degree in Computer Science and Engineering

Office of Graduate Studies
Adama Science and Technology University
Adama, Oromia, Ethiopia

Adama
September, 2018

ADVISOR'S APPROVAL SHEET

To: Computer Science and Engineering program

Subject: Thesis submission

This is to certify that the thesis entitled "A hybrid algorithm for fast detection and recognition of signage and obstacle avoidance for robot navigation" submitted in partial fulfillment of the requirements for the degree of Masters in Computer Science and Engineering, the Graduate program of the Computer Science and Engineering Program, and has been carried out by Noah Diriba Id. No GSR/0058/09, under my supervision. Therefore, I recommend that the student has fulfilled the requirements and hence hereby he can submit the thesis to the department.

Yun Koo Chung (PHD)
(Advisor) _____ _____
Signature Date

DECLARATION

I hereby declare that this MSc thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis have been duly acknowledged.

Noah Diriba Debar

_____

(Signature)

This MSc thesis has been submitted for examination with my approval as university advisor.

Yun Koo Chung (PHD)

(Advisor) _____

(Signature)

Submission Date: _____

APPROVAL OF THE BOARD OF EXAMINERS

We, the undersigned, members of the Board of Examiners of the final open defense by Noah Diriba have read and evaluated his thesis entitled "A hybrid algorithm for fast detection and recognition of signage and obstacle avoidance for robot navigation" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the degree of Masters in Computer Science and Engineering (CSE).

Yun Koo Chung (PHD) (Advisor)	Signature	Date
(Chairperson)	Signature	Date
(Internal Examiner)	Signature	Date
(External Examiner)	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.

A big thank you to my friends and family, for being a source of encouragement. A special thanks to My sister Hidare Diriba for her commitment to spent all her resources up on my education and also for her encouragement.

Table of Contents

List of Figures

List of Tables

Acronyms

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

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 sign recognition and obstacle avoidance for the navigation of the robot has been increasing due its importance in many sectors such as for autonomous car, for military surveillance robot, for indoor surveillance robot and for assisting visual impaired people with robot. Therefore, ability to observe sign and understand the meaning by the robot is essential part for navigating the robot safe and to reach the desire end without any difficulty. There has been very little work directed toward assuring safe condition for navigation of the robot using computer vision and also current robot navigate through the unknown environment by using sensing mechanisms. So introducing the mechanism that the robot uses vision techniques will improve the performance of the robot to navigate through the environment better. Therefore, integrating fast vision algorithm that can enable the robot to understand sign and avoid obstacle make the robot to navigate safe within unknown indoor environment. This study enables the robot to detect and recognize sign with high speed and enable the robot to avoid any obstacle. This study integrates fast recognition algorithm with obstacle avoidance mechanism to make the robot to navigate by understanding the environment it exists and to plan paths accordingly by avoiding obstacles. To give solution for the problem first current existing v ision algorithms is studied and the best in computation time and reliable in detection have been chosen then simulation for executing the proposed algorithm is selected by their computability time, flexibility and by their feature factor which are best are selected. Because of robot requires fast and accurate response various vision algorithm have been studied and due to the problem of large computation and slow speed, a fast matching algorithm based on the combination of ORB feature point and SURF descriptor is proposed. Experiments show that compared to the classic SIFT and SURF matching algorithm, the method is very good to achieve the goal of fast matching, in addition to the speed also improves the accuracy of the matching. The navigation is done by identifying the indoor sign and making the decision to either to re arrange its path again or to stop its motion based on the sign it observes. A proof of concept is presented using TurtleBot3, a robot kit, which runs on robot software development framework Robot Operating System (ROS).

CHAPTER 1
Introduction

1.1. Background of the Study

Existing robotic systems are already able to perform various services such as delivery, education, providing tele-presence, cleaning, or entertainment. Furthermore, there are prototypes of autonomous wheelchairs and intelligent service 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 or disabled people. The sign recognition and obstacle avoidance for robot navigation is highly applicable for autonomous car and for visually disable people.

The design and functioning of the road transport system is intended to ensure socio-economically efficient and sustainable transportation. Normally the drivers need to be very vigilant and cautious to identify road signs at the right time. But to identify road signs especially in poor weather conditions depends very much on the physical and mental health of the drivers. The visual perception abilities can be affected by stress, tension, physical illness or toxicities. Therefore, predictive techniques are being developed to maximize transportation safety and efficiency by introducing fast sign recognition and obstacle avoidance system in which the traffic signs detection and recognition holds the most important position. According to the National Highway Traffic Safety Administration (NHTSA), the main cause (94%) of vehicle crashes is human error [1] [2]. Among all possible types of driver errors, recognition errors, decision errors, and performance errors are the most frequent driver-related critical reasons for accidents. Based on this investigation, we conclude 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 sign complicate Addis Ababa traffic problems [3]. Yet among the various reasons of major traffic problems in the city the weak traffic sign identification and usage are identified as a major one by different scholar studies in the area. According to study in Addis Ababa there are total number of 940 or 1.5% of fatal injury, property damage, serious injury and minor injury occurred by violating traffic sign [4]. This motivates us to study highly optimized computer vision Algorithms that can inform the user to identify sign with high speed and which can take action for the user accordingly.

since Ethiopia became one of developing country in urbanization there are more number of ongoing standard buildings and infrastructures but for People with visual disabilities, *i.e.*, partially or totally blind, are often challenged by places that are not designed for their special condition. Examples of these places are bus and train terminals, public offices, hospitals, educational buildings, and shopping malls. Several objects that are present in most built environments become real obstacles for blind people, even putting at risk their physical integrity. Simple objects such as chairs, tables and stairs, hinder their movements and can often cause serious accidents. Current global estimates indicate that blindness affects close to 48 million people [5]. Ninety percent of the blind live in developing countries where eye diseases and blindness are major public health problems. According to recent studies, about 4 Million out of the 75 Million inhabitants of Ethiopia live with visual impairment: 1.6% is blind, 3.7% suffer from low vision, women generally are more affected by visual impairment and blindness then men and 6% of the blind population are children [6]. Therefore, several proposals have tried to address this challenge in indoor and outdoor environments. However most of them have limitations, since this challenge involves many issues (e.g., accuracy, time, coverage, usability and interoperability) that are not easy to address with the current technology. Therefore, this can still be considered an open problem. So this proposed system can provide, in real-time, useful navigation information that enables a user to make appropriate and timely decisions on which route to follow in an indoor space by quickly identifying sign and avoiding obstacle with high range. In order to provide such information, the system must take into account all the objects in the immediate physical environment which may become potential “obstacles” for blind people.

Object 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, background clutter or sensor noise [7]. Because of this, there have been numerous methods for object recognition developed 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 resolution which mounted at the top of the robots, LIDAR which is a low-cost 360 degree 2D Laser Scanner (LIDAR) system which can perform scanning with a frequency of 5.5Hz when sampling 360 points within 6 meter range which emits modulated infrared laser signal and the laser signal is then reflected by the object to be detected, a computer running a software application that coordinates the whole system, and a wireless module that delivers the navigation information to the robot through messages.

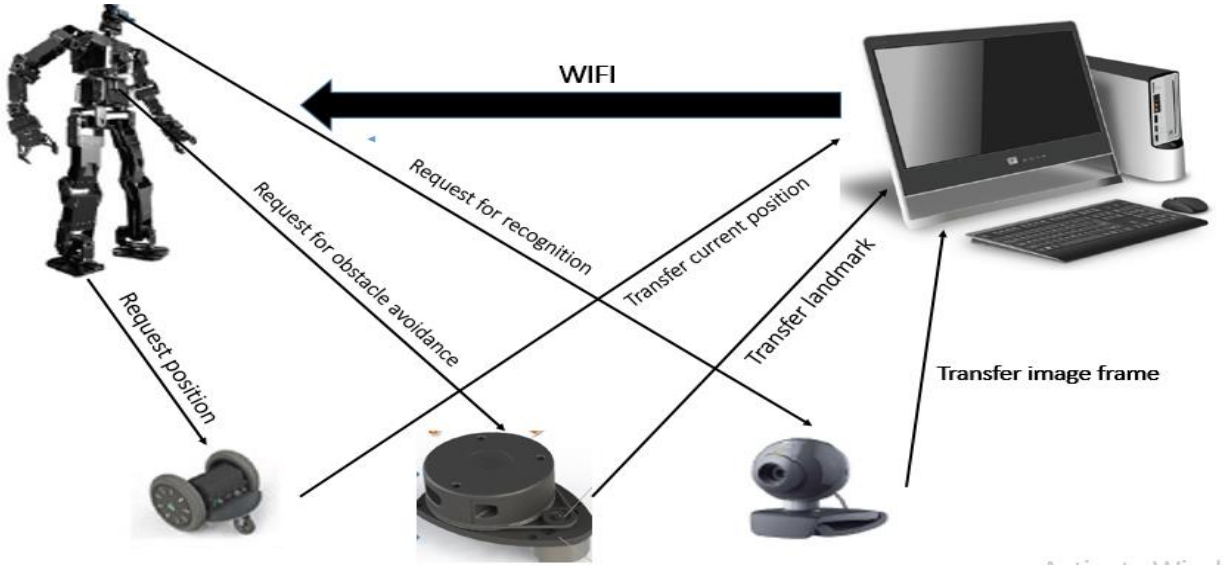

Figure 1. Component of the navigational system

When the robot request navigation information, it activates the LIDAR sensor and Odometery sensor 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 signage is existing in front of the robot. The robot position and movement are detected by the LIDAR and odometery wheels. These 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.

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 indoor signs. 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 signs. 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. 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 sign 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 signs.

1.3. Statement of the Problem

Signage 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]. Since the research has been done in the real time environment for indoor sign recognition, there are three utmost aspects considered, to be achieved by the real-time sign recognition system:

1. Speed: Processing Time in Sign recognition.
2. Efficiency: Overall System's Response Time.
3. Reliability: Accuracy in Sign Recognition.

As well as for obstacle detection and avoidance distance for obstacle detection is considered. So generally, the problem of existing system can be summarized as follow.

- Difficulty in detecting and recognition of signage with lower time
- Difficulty in detecting and recognition of signage 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

The aforementioned problems are solved by the following research questions:

RQ1. How can the robot detect and recognize sign with high speed or with lower required time?

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

RQ3.How can the robot can detect obstacle with long distance and can avoid obstacle without collision?

1.4. Objective of the Study

1.4.1. General Objective

The general objectives of the research are to develop a robot that can recognize sign with less required time and that can detect and avoid obstacle from long distance 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 Signage 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 signage
- 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 signage 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 obstacle avoidance, detection and recognition of signage. It is focus on improving the speed in which the robot can recognize sign and the mechanism to detect and avoid obstacle. The speed for recognition of the sign 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 LIDAR laser scanner (3D) based perception of road surface and obstacles. This is the main sensor for obstacle avoidance and environment perception. The classification into traversable road surface and non-traversable road surface.
- 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.

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

1.7. Organization of the Thesis

The organization of this thesis is as follows chapter one introduces about robot navigation and about signage 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 signage detection and recognition system, state the merit of signage 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

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 it's 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 processes that the mobile robot builds its Owen map and know it's current position by using the built map [4]. In SLAM both the trajectory of the platform and the location of all landmarks are estimated online without the need for any a priori knowledge of location. 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 unfamiliar 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.

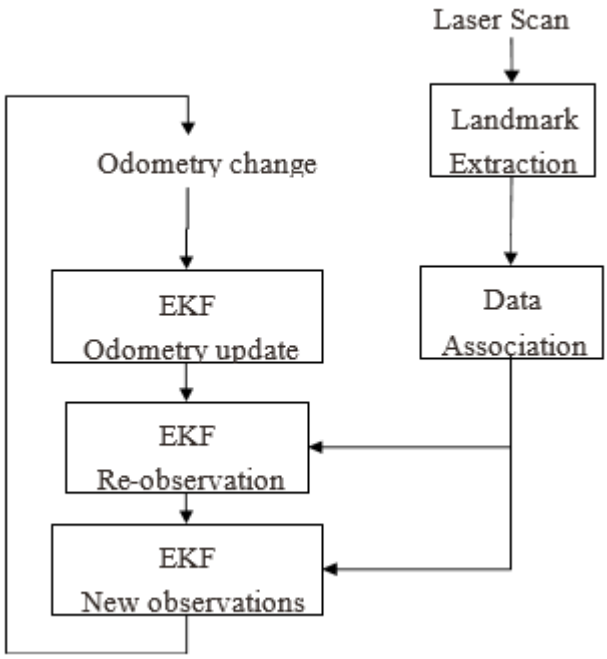

Figure 2. 1 Overview of the SLAM process [5]

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 distinguished from the environment [5]. 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 spikes and RANSAC [5].

2.1.1.2. Spikes land mark extraction

These kind of land mark extraction used for landscape changing between two laser beam or used for extracting landmark for non-smooth surface by using extreme value in which from two laser scan reading differ by certain values.

2.1.1.3. RANSAC land mark extraction

In this kind of land mark extraction is mainly used for smooth surface this types of land mark extraction used for extracting line from the laser scan by using least square approximation. After collecting the laser reading point RANSAC checks how many laser readings lie close to this best fit line if the number is greater than some thrash hold value which is called consensus it can assume that it finally gets line [5].

2.1.1.4. Data association

In this process after the landmark extraction it will be stored in database. After it stores the extraction point, the robot starts for scanning other land mark using the laser scan and the data association start to associate the newly detect reading from already stored data in the database. If the newly read data is new, then it will be stored as newly observed landmark in database to make it ready for the next association. By setting a constant, an observed landmark is associated to a landmark if the following formula holds [4].

$$V_i^T S_i^{-1} V_i \leq \lambda$$

Where vi is the innovation and Si is the innovation covariance

2.1.1.5. EKF (extended kalman filter)

EKF is used to estimate the current position of the robot from the odometry reading and from the landmark observation [5]. 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.

$$\begin{matrix} x + \ \Delta t\ cos\theta + q\Delta cos\theta \\ y + \ \Delta t\ sin\theta + q\Delta sin\theta \\ \theta + \Delta\theta + q\Delta\theta \end{matrix}$$

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:

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

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.

$$K = P * HT * (H * P * HT + V * R * VT)-1$$

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.

2.2. Vision based navigation

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.

2.2.1. Object Detection and Recognition

Object detection is detecting an object in image or from sequence of images or is part of computer vision and image processing that used to detect the required or intended object of a certain class (such as faces, signs, buildings, etc) in digital images and videos captured by cameras. Object detection plays a very important role in many applications such as image retrieval, surveillance, robot navigation, way finding, etc [13]. According to the study the human visual system is powerful, selective, robust and fast [2]. It is not only selective in order to classify among the similar object but also robust enough to classify with large variance of the same category objects. Object recognition in human visual system is very fast taking up to 100 to 200ms [2]. However, it is difficult in order to build robust and selective visual algorithm which can handle very similar object with large variation. Although there are various object detection and recognition algorithm proposed among them SURF is the one.

2.2.1.1. SURF

Speeded UP robust Features "SURF" algorithm is considered a robust local feature detector and extractor algorithm used in many computer vision applications like object recognition, track objects, face recognition and to make 3D scenes. Because of its high speed of operational activity than SIFT (scale invariant feature transform) SURF is became the chosen algorithm for those problems that require fast computation. SURF's detector and descriptor are not only faster, it is also more distinctive [14].

There are three steps that this algorithm goes in order to detect and recognize the object the First one is called interest point detection, in this 'interest points' are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. The second step is interest point description, in this step the neighborhood of every interest point is represented by a feature vector. This descriptor has to be distinctive and at the same time robust to noise, detection displacements and geometric and photometric deformations. Finally, the third step is matching, in this step the descriptor vectors are matched between different images. The matching is based on a distance between the vectors. [8]

The fast computation time secrete for SURF algorithm is that SURF use one of the well-known techniques used by viola jones in face detection and recognition algorithm that is called integral image which used to compute total summation of pixel intensity with much fast time and SURF uses Box Filter instead of LoG (Laplacian of Gaussian). One of the big advantage of this approximation is that, convolution with box filter can be easily calculated with the help of integral images and Also the SURF rely on determinant of Hessian matrix for both scale and location [15]. The difference between SIFT and SURF can be summarized as follow.

Table 2. 1 Difference between SIFT and SURF

	SIFT	***SURF***
Scale Space	Difference of Gaussian (DoG) is convolved with different size of images with same size of filter. Fix filter is convoluted with down sampling images	Different size of box filter(Laplacian of Gaussian (LoG)) is convoluted with integral image. Fix image is convoluted with up sampling filters
Key point detection	Using of local extrema detection, apply Non	Determine the key points with Hessian matrix

	maxima suppression and eliminate edge response with Hessian matrix	and Non Maxima suppression
Orientation	Image gradient magnitude and orientations are sampled around the key point location, using the scale of the key point to select the level of Gaussian blur for the image. Orientation of histogram is used for same.	A sliding Orientation window of size $\pi/3$ detects the dominant orientation of the Gaussian weighted Haar Wavelet responses at every sample point with in a circular neighborhood around the interest points
Descriptor	The key point descriptor allows for significant shift in gradient positions by creating orientation histograms over 4 x 4 sample regions. The figure shows 8 directions for each orientation histogram, with the length of each arrow corresponding to the magnitude of the histogram entry.	An orientation quadratic grid with 4x4 square sub regions is laid over the interest point. For each square, the wavelet responses are computed from 5x5 samples.
Size of descriptor	128 bits	64 bits

2.3. Related Work

There is some related work that use signage for robot navigation. In Signage system for the navigation of autonomous robots in indoor environments by Ana Corrales proposed that in their system the robot navigate through unfamiliar indoor environment through accessing the released signal. This signal hold sufficient information like the current position, the connection that can take to the intended destination directly or indirectly and the action that the robot must execute to reach to the required destination. So by using RFID tag attach on the wall and RFID receiving antenna which attached on the robot (UHF RFID readers) the robot can receive the stored information about the indoor environment and use that information in order to reach the intended destination [16]. But the shortcoming of this proposed system is that the distance RFID transmitting signal are minimum and the capability to accesses the signal from RFID transmitter tag is only one robot can and also the signal will be interrupt when there is metal object or other material exists in between the transmitter and receiver.

In Robot Navigation Using Camera by Identifying Arrow Signs by Jongan Park proposed that first they build track where the robot moves then they put an arrow sign at each turn. 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 recognize weather the arrow sign is left or right after recognition the arrow 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 [17].

However, the proposed system will take much time when it tries to send the image wirelessly to computer module and try to identify the signage by image processing. Segmenting 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.

While observing other related papers which mainly emphasized on the object detection and recognition algorithms to find fast and robust techniques that can recognize objects we found some related papers for object recognition system that works on agriculture, text recognition and system that recognize sign for visual impaired people some of the related papers are reviewed as follow.

Recognition of Amharic Braille by Teshome alemu proposed system which is used to recognize Braille to print document. In this mechanism they took two kind step the first one recognition of Braille character in this phase braille character is trained for testing the braille print document. First the image is binarized by binarization techniques which means the image is converted to black and white. Then the image is segmented by mesh-grid technique. With mesh, dots are extracted following the vertical and horizontal grid line. Having done this, in feature extraction the system once again takes advantage of the mesh. Finally, the character is recognized by the rule defined as context analysis. The second phase was to classify those detected and recognized trained data with braille print document. However, segmentation 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 [18].

Automatic Recognition and Counterfeit Detection of Ethiopian Paper Currency by Jegnaw Fentahun Zeggeye and Yaregal Assabie proposed the matching mechanism for automatic recognition of Ethiopian currency. The system usually checks whether the given currency is genuine or not. In this proposed system the image which is token by scanner or input camera processed by the combination characteristics feature of currency and local feature descriptor to be classified in to four level classifier. Then finally the categorization component classifies the currency notes in to their respective denomination [19].

Ancient Ethiopic Manuscript Recognition Using Deep Learning Artificial Neural Network by Siranesh Getu Endalamaw proposed the recognition system for handwritten Ethiopic Geez scripts documents, which aims at transforming written text into machine encoded text. In this recognition process first they get digital image of manuscript pages for input to the system through digital camera by the process called image accusation. Then they proceed to preprocessing stage to make the input image to enhances suitable for segmentation by the processes binerization to convert the input image black and white or one and zeroes pixel, Skew detection and correction to align the document page correctly and noise reduction to reduce unwanted noises from the image. Then they goes to segmentation the preprocessed image by the process of line segmentation, word segmentation and character segmentation to make the separation between the individual characters of an image. After segmentation they extract the feature to retrieve the most important data from the raw data. Finally, they Classify and recognize the text through artificial neural network [20].

Computer Vision for Ethiopian Agricultural Crop Pest Identification By Dagnachew Melesew Alemayehu, Abrham 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) [21].

In Real time text localization for Indoor Mobile Robot Navigation by Kazem Ghazanfari proposed text recognition from the signage for robot navigation through unfamiliar environment. In their system from the taken image by the camera of the robot first they detect weather the taken image contains the text or not then if it contains they processed to localize the text from the given image. After the localization they processed to extracting the text using OCR system. In this system they used morphological operator which is used to extract the text with fast, invariant to geometrical transformations and scales. After the text extracted it goes to SVM classifier in order to classify the text then finally they recognize the extracted text with OCR [22].

Detecting signage and doors for blind navigation and winding by shuihua wang,xiaodongyang and yingli tian proposed that an effective mechanism for detecting signage and doors . They proposed framework by combining two methods the first is that signage detection by combining saliency map with bipartite graph the second one is for detecting doors based on geometrics door framework by combing edges and corners. During the sign detection they first employs the saliency map to detect the intended area by using color intensity and orientation then after detection the signage or the intended area the scaled pattern is detected within attended area using bipartite graph matching [2]. But the proposed algorithm would fail if the signage were text and also the time required for recognition is high.

Ying li tian , xiadong yang chucai yi on their paper of toward a computer vision based wayfinding aid for visual impaired persons to access unfamiliar indoor environments uses context information in order to find and extract a text from the given signage then extracting the text from region of interest that are most likely to contain text information. Due to the uniform color of the text in the signage they perform color decomposition to obtain identical color from different color then perform color quantization to reduce number of color and keep essential information for text extraction and segmentation. Then they perform edge detection, second non edge pixel is sampled, third grouping pixel of similar pixel, fourth performing mean color values of each color after color decomposition they obtain region with similar color now they extract text region based on the structure of text characters that are composed of strokes and major arcs are aligned in the direction whatever text string they belong to. Due to might be involvement of

textured non text region in the extracted region of interest it will be required to further filter out the non-text region they use text structure model to identify letters and numbers and also they further filter out the non-text bounding boxes. They first measure the distance variation among three neighboring bounding boxes in accordance with their central points. If one of them is larger or smaller than its two neighbors that have similar foreground area, it will be filtered out. Finally, in text recognition after filtering out they perform text recognition by employ off the shelf OCR software to translate the text image into character codes [23].

However, the draw back from the proposed paper is that during the detection process the time to detect the signage or doors and the detection rate for multiple signage and for recognition is high and also the proposed system only concentrates on the text area it would be failed if the signage is not text**.**

There are also other related papers that works on detecting obstacle for robot navigation some of the related papers are Moving obstacle avoidance of a mobile robot using a single camera by Jeongdae Kim, Yongtae do proposed that to detect the moving obstacle by using single camera during the navigation of robot. In this proposed system the moving obstacle is found by using BBME (block based motion estimation). In this method they first divide the image frame in to blocks then the block with best match from previous divided block taken at different moment is detected. Then the detected block is converted to gray scale so as the relative speed between the moving object and the robot increase the brightness of the object increases. After gray scale conversion they processed for further evaluation of the block with histogram process by giving threshold and binary image is projected vertically finally the position of the moving object is estimated [24].

In fast goal navigation with obstacle avoidance using a dynamic local visual model by Juan Fasola proposed a technique by which an AIBO robot can visually navigate to globally-defined goal points on the Soccer field while avoiding obstacles. In this process they detect and try avoid those obstacles by analyzing the segmented image with an algorithm called visual sonar. In this algorithm it detects the obstacle and calculates the distance between the robot with the obstacle then location for the open area and obstacle are stored in ego centric local model. The vision system for the robot labels each pixel with different color for the free space as well as for obstacle [25].

Therefor what the proposed system intended to do is to take the best result of each of the above aforementioned works and synthesize them in to one holistic system. Apart from this by improving and developing a new system upon their short comings the proposed system find way to tackle the problem that arisen with in previous systems. Thus the proposed system is more beneficiary and useful for robot navigation through unfamiliar environment with the ability of detecting and recognizing signage and avoiding obstacle with much high speed and better length of detection respectively .

Table 2. 2 comparison of different proposed solution

Reference no	***Proposed mechanism for signage detection and recognition***	***Proposed mechanism for avoidance obstacle***	***Pros***	***Cons***
14	Use RFID signal that contain visual information	No mechanism is proposed	Advantageous for time consumption	RFID short distance working, destructed with metal objects, only work for single robot and little information can hold
15	Segmentation for detection	IR signal	Advantageous for time consumption	Segmentation is shape dependent and variant for the orientation and sizes.IR usually perfectly work for short distance and distract with metal objects
28	OCR with SVM classification for text holding signage	No mechanism proposed	High Accuracy in classification	Time consuming and only work for text holding signage not directional signage
29	R-CNN for classification and recognition of exit signage	LIDAR	High range for detecting and avoiding obstacles	Time consuming in recognition of signage

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 pycharm, ROS (robotic operating system), UML modeling, and other relevant tools. These are described below:

3.3. 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].

3.4. 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 reusability

through use of standard tools and interfaces [17] and the design tool that ROS provide to design and implement the robot like Rviz, gazebo we choose ROS as simulator software for robot.

Table 3.5 Characteristics of Open-Source Robotic Simulation Software [30]

	Player/Stage	**Gazebo**	**ROS**	**Simbad**	**CARMEN**	**USARSim**	**MRDS**	**MissionLab**
OS	Linux, Mac, Win.	Linux	Linux, Mac, Win.	Linux, Mac, Win.	Linux	Linux Win.	Win.	Linux
Simulator Type	2-D	3-D	2-D, 3-D	3-D	2-D	3-D	3-D	3-D
Programming Language	Player(any) Stage(C, C++, Python, Java)	C, C++, Python, Java	C++, Python, Octave, LISP, Java, Lua	Java	C, Java	C, C++, Java	VPL, C#, Visual Basic, JScript, IronPython	VPL
Documentation	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 (using Player)	Yes	Yes
Sensors	odometry, range	odometry, range, camera	odometry, camera, range	vision, range, contact	odometry, range, GPS	odometry, range, camera, touch	odometry, range, camera	odometry, range
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 alsko 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 posibility 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.

3.4.1. 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 [17][18]. 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 [17][18].

Scripting languages such as Python and Ruby makes it easier to accomplish many software tasks easily compared to others

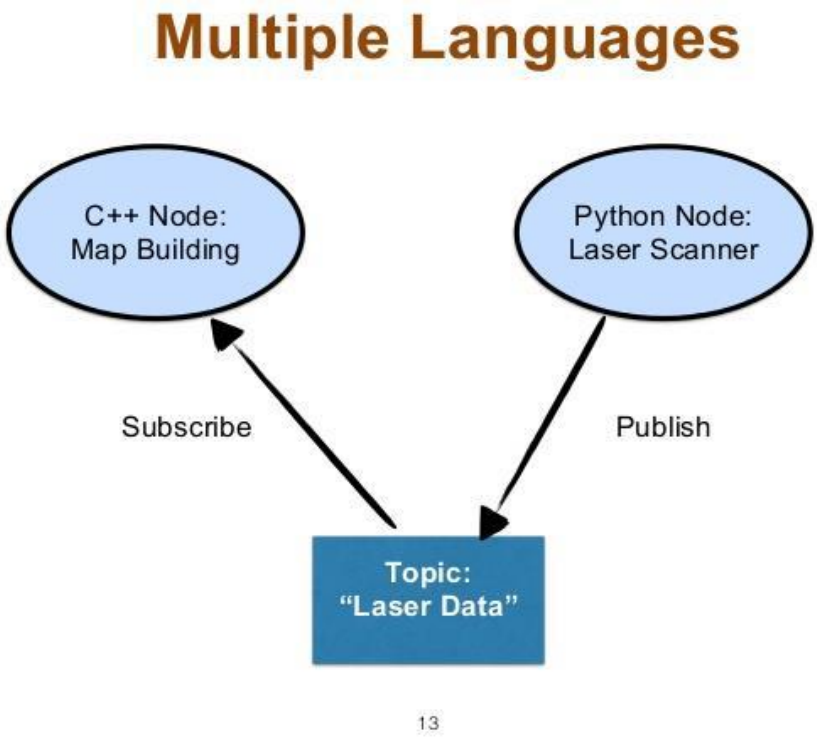

Figure 3. 1 multiple language supporter by meta operating system ROS [18]

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 [18].

3.4.2. 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.

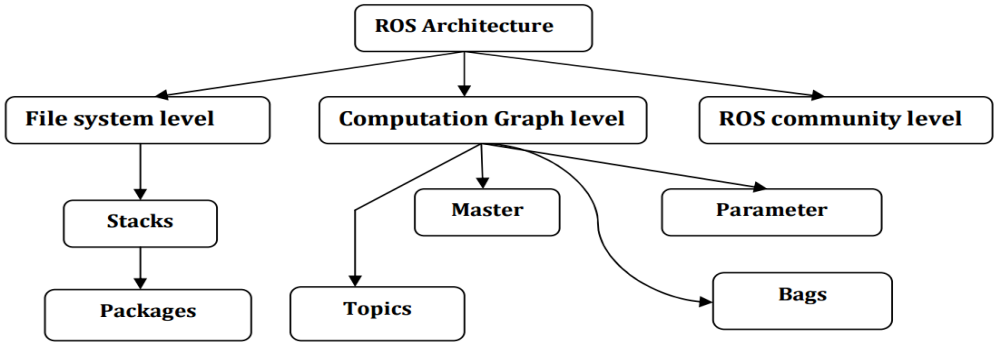

Figure 3. 2 ROS architecture with their respective section and level concept [18]

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.

3.4.3. 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) [17].

3.4.3.1. 3D 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.

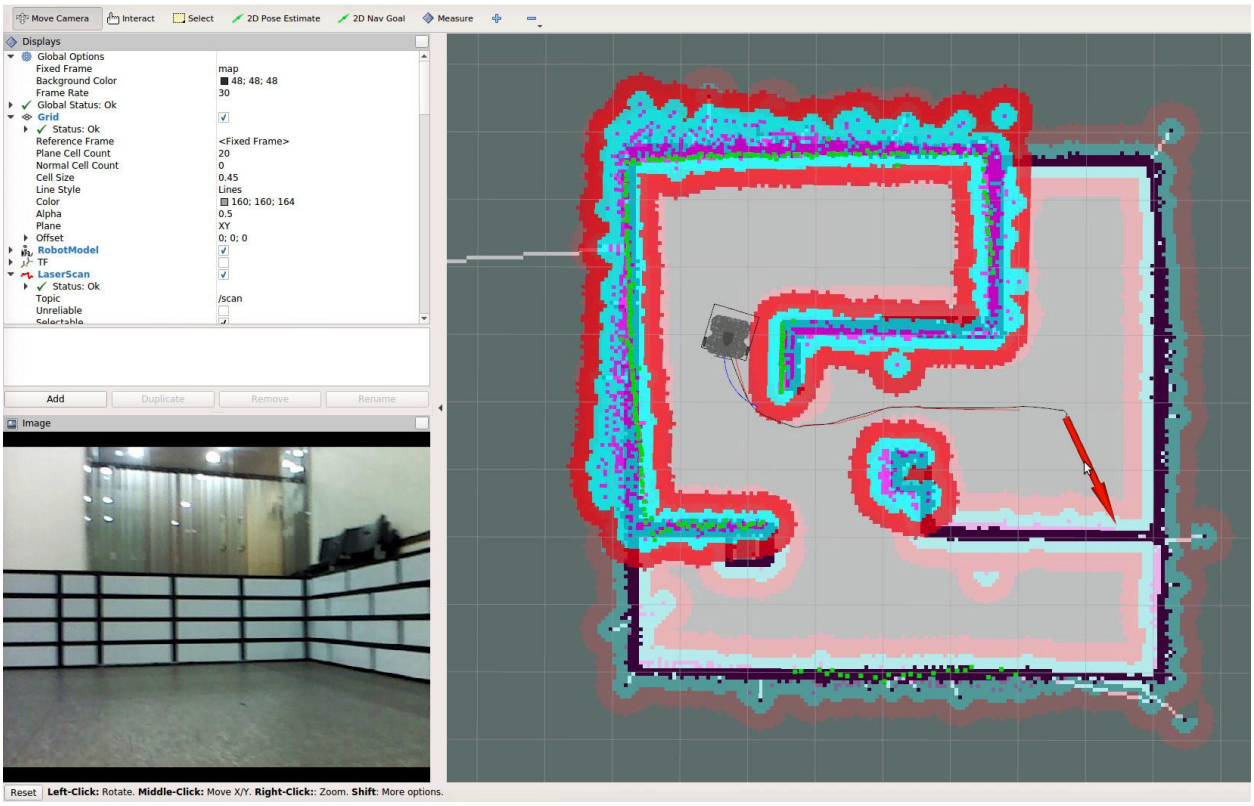

Figure 3. 3 Over view of RVIZ simulator [17]

3.4.3.2. 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 [17].

3.4.3.3. 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 [19]. 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.

3.6. 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.

3.6.1. Dataset content and sampling techniques

The query image dataset was created by grabbing 480 * 360-pixel indoor sign images from the websites. 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 as text sign and symbol sign because most indoor sign are sign with text and also there are sign with arrow or symbols. As figure below shows from those classified sign we classified each group in to motion and path sign. Finally, from the image 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 repective dataset. The following subsections provide more information regarding each of the data sets.

- **Recognition I: Lighting and Orientation**

The purpose of this data set is to test the robustness of the sign recognizer with respect to various illumination changes and in plane rotations. Frontal images of signs 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 the sign at different position. This second database contains ten images of 10 different signs 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 sign image is exists. This process is mainly useful to test the recognition capability from long distances.

Figure 3.4. Some of the sample image stored in dataset with their respect variation of intensity rotation and scale

Using the process described above, a database with sign images collected. The dataset consists of 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 sign, which is nearly impossible, but it does cover a good selection of signs. 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 sign.

3.6.2. Training and testing techniques

Training data were retrieve from the database which is stored for the testing purpose. In this process images with size of 480*360 pixel were stored and trained by the algorithm for testing with the test image. The test image is the image which is going to be tested with the trainee image of their correspondence. 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 it will be matched with correspondence point in the trainee image. If the match greater than threshold matched will be found.

3.8. Proposed System Design Method

The system design is designed for continuously to take image and features (landmark) dataset to make able to recognize the traffic sign and to be able to build the map respectively. Specifically, the designed system detects and recognizes indoor sign with fast image processing algorithm and detects an obstacle for safe and reliable robot navigation with in indoor environment.

3.9. Proposed System Implementation Method

The proposed system has two important parts navigational phase and computer vision phase. in the first part the robot builds its map for indoor environment and navigate through it by avoiding obstacles and by path planning. During navigation it has the vision part so that the robot detects

and recognizes indoor sign from the camera mounted on the robot. Generally, the designed system has been implemented using ROS but for testing and evaluating the proposed image processing algorithm we used pycharm.

3.10. 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 signs. Since the speed for recognition of the sign 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. Some of the papers observed are described as follow.

From the existing matching algorithm SURF, SIFT, ORB, FREAK, DAISY, MSER BRISK and FAST are the chosen algorithm for matching between two images.

SIFT (scale invariant feature transform) algorithm is one of the well-known algorithm for feature selection and matching of different objects which have been widely employed for detecting and recognizing of objects due to its robustness in scale variation, translation, rotation, illumination and 3D affine transformation [9]. SIFT algorithm have GOOD recall rates (have high accuracy), it also included in Opencv library, it's features are robust to occlusion and clutter and relatively efficient compared to older algorithm. However, SIFT is mathematically complicated and computationally heavy and SIFT is based on Histogram of gradient that why the gradient of each pixel in patch need to be computed and this computation cost time therefor it is quit slow [9].

So the other version of SIFT called SURF (Speeded up robust feature) is chosen for image processing unit. SURF algorithm provides similar performance while running faster. Due to its accuracy, robustness in scale variation, translation, to occlusion, to clutter and its fast characteristics in recognition of the object SURF algorithm has been chosen. However, since the

image processing unit of the proposed system should be fast and accurate ORB and SURF is integrated and tested for their performance.

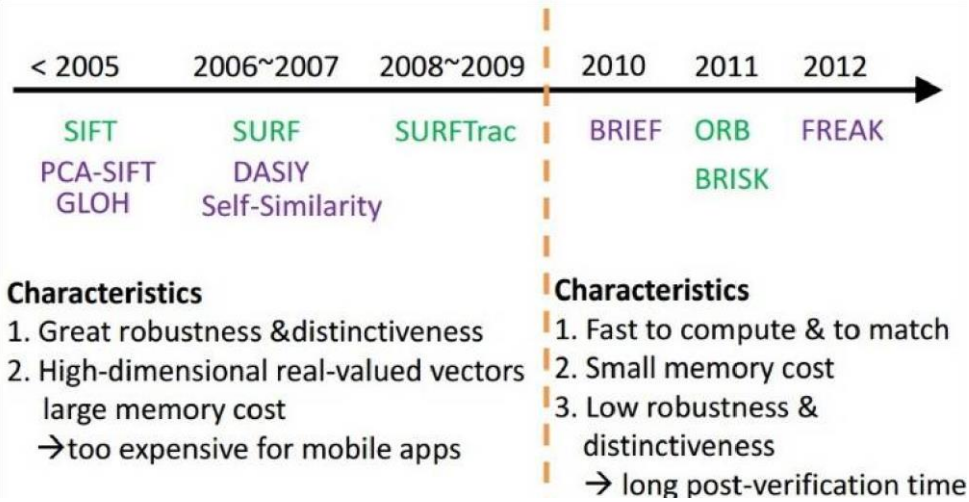

Figure 3.4 Development and bottlenecks of existing local features [9]

Table 3.6: Comparing Results

Algorithm	**Ranking** (1 for higher and 9 for lower)	**Accuracy Percentage**	**Speed Coefficient**	**Description**
SIFT	9	_	_	• It includes the four major stages. • DoG applied instead of the Laplacian to identify points of potential interest. • Hessian matrix is applied to calculate the main curve and eliminate the key points.
MSER	8	_	_	• A method for detecting bubbles in the images.
SURF	7	85%	1	• SURF is inspired by SIFT, designed with an emphasis on speed. • A Haar estimate wavelet that applies bubble detector based on Hessian determinant is applies in SURF.
FAST	6	_	_	• FAST is based on the SUSAN corner detector. • FAST algorithm applies a Bresenham of a circle 3.4 diameter of pixel as the mask in testing.

BRIEF	5	_	2	• This descriptor applies binary strings. • This descriptor is resistant against common photometric classes and geometric conversions of image.
DAISY	4	_	_	• A local image descriptor, efficient in dense computation. • Efficiency of DAISY is due to that most calculations consist of discrete convolutions.

BRISK	3	87%	_	• BRISK is a method for the detection, description and comparison of key points. • This method combines DAISY and BRIEF.
ORB	2	_	2	• This is a standard method for oriented FAST and rotated BRIEF.
FREAK	1	91%	3	• It is a new key location descriptor inspired from the human visual system. • The Difference between FREAK and BRISK sample pattern is the exponential change in size and the overlapping receptive fields. • Saccadic search through the indices to multistep analysis, be imitated.

From the above comparative study ORB,BRIEF and BRISk which is classified as one group due to their year of proposed they are best in matching and computation time on the other hand the left group SIFT,SURF and SURF trac is robust and they are characterized by their distinctiveness.

FREAK is the fast algorithm for detection and description but it is weak in real-time occupation and filling the computational and memory bandwidth operation. The system performance is assessed when the features extracted are weak and sensing devices are of low quality. BRIEF on the other hand it is fast but it is not designed to be stabilized against the rotation. There exist many views on the performance of these feature detection algorithms. The SIFT has the best strength with respect to rotation and changes in scale, while the issue of time is confirmed again. BRIEF and FAST yield better results in increasing brightness and ORB is better in reducing brightness. ORB also shows good performance in blurred images.

All of the above algorithm goes in detection and description of interest point depending on weather they are corner point, T-junction or blob and depending on their orientation respectively. In SIFT for detection of interest point it applies a series of Different-Gaussian (DOG) filter for multiple scales, and it get DOG filtered and down sampled version of the original image then goes for detecting those interest point. For descriptor SIFT use a composed form of 4x4 array of gradient orientation histogram weighted by the gradient magnitude [32]. In SURF for detection it approximates the determinant of hessian matrix which will give a local maximum [20].

For description SURF uses 64 dimensional vectors which is obtained by summing up the Haar wavelet coefficient over 4x4 pixel around the key point. FAST algorithm is usually used to detect corner point by using circle consisting of 16 pixels around candidate corner pixel to determine whether they are corner point or not. By comparing intensity values of those 16 pixel to threshold it determines as interest point. BRISK is descriptor algorithm which is based on the pairwise intensity comparison. ORB algorithm is which uses FAST as detector of interest point and BRISK as descriptor for those detected interest point [20].

According to Ebrahim Karami and silva Prased in their paper of comparative study for image matching for distorted image with SURF, SIFT, ORB, they proposed that after Appling different type of image transformation, ORB is the fastest algorithm for matching but SIFT is the best for matching rate. In this paper they indicate the performance by evaluation parameter such as the number of key points in images, the matching rate, and the execution time require for each algorithm [33].

On the other hand, a comparative study which is did by Maridalia Guerrero Peña on the three image matching algorithm SIFT, SURF and FAST indicates that SIFT have great number of feature detection as well as matching rate while SURF holds the second places but FAST algorithm detect features with more accurate than SIFT and SURF [34].

From the above comparative study, the 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. Although ORB is fast and accurate the algorithm it uses for description is variant for different image transformation so instead for descriptor SURF is used. Finally, for matching purpose FLANN based algorithm is used.

3.11. Evaluation Method

In order to verify the effectiveness of the matching 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-juncton or it should be blob point.

Chapter Four

Proposed Solution

4.1. Overview of the Proposed Solution

This chapter describes the procedure followed in 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 signage detection 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 sign and the time to take action based on the sign 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. So ORB for detection interest point, SURF for description and FLANN for matching are selected.

SURF algorithm goes in the task of finding points correspondences between two images (Training image and test image) of the same scene or object. The search for discrete image point correspondences can be divided into three main steps. The first step is detecting an interest point from the given image, second one is describing those image from the given image then finally matching with corresponds image.

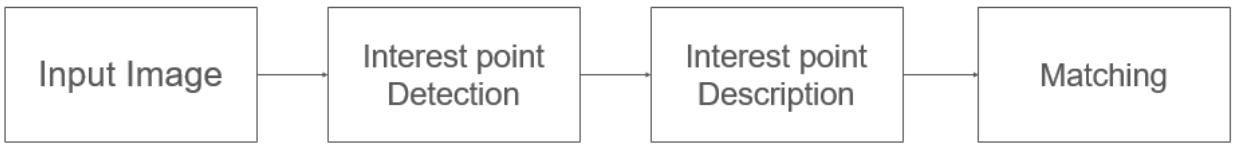

Figure 4. 1 Flow of object recognition by SURF

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 SURF algorithm. In this process instead of using SURF interest point detector it uses ORB detector and for descriptor process it uses SURF descriptor finally for matching it uses FLANN based matching. ORB algorithm is one of the fastest algorithms having rational invariance and reducing sensitivity to noise. This algorithm is based on the improved FAST algorithm for detecting an interest point from the image and BRIEF algorithm for description process [35]. However, since BRIEF descriptor is variant to the image rotation it doesn't used the descriptor part of the ORB.

In detection part ORB used FAST algorithm in this algorithm it begun to search for pixel B which is called as FAST corner point when the neighborhood around the pixel B has enough pixel in different gray area with the pixel B [23]. In this process first it chooses an arbitrary pixel point as center to form circular area and it consider the circular areas as the pixel point's neighborhood. If from the pixel point from the surrounding the center point pixel from n consecutive pixel satisfies the following equation 1, it considers the point pixel as candidate feature point.

$$|Ix-Iy|>t \quad \text{……equation 1}$$

Where t is the threshold

Ix is the gray value of consecutive n pixels

Iy is the gray values of the point p.

To achieve with fast time and to overcome the invariance in scale it assigns n as 9 and it implement scale pyramid. Then to obtain the candidate feature point it carries out on FAST testing in each layer. Finally ORB use Harris algorithm to detect the interest point from those detected candidates in each layer. In this process it sorts the detected point and takes N point which are most suitable. The flow of the ORB detection process can be summarized as follow.

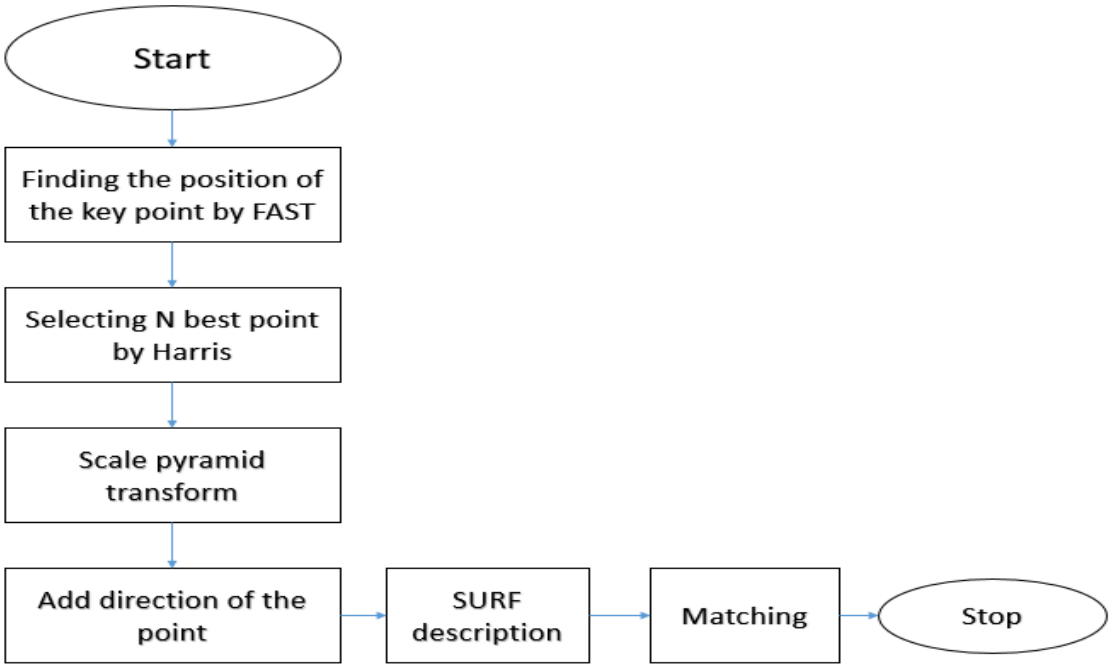

Figure 4. 2 flow in which ORB uses for detection of interest point from the image

Pseudocode for interest point detection

Selection of the center point

Loop for each 16 surrounding points

Loop for n consecutive point

calculate Ix,Iy

Normalize Responses

If(Ix-Iy>threshold)

Store point as interest point at scale

End if

End loop

End Loop for Each point

Suppress non maximum interest points interpolate interests point Between octaves

Interpolate interest points between octaves

Output: Interest points

4.2.2. The Description Step

The second step is describing those interest point found in step one with respect to its orientation and direction because the description of those interest point should be invariant of the image rotation. To do that first it applies window size of 20s around the point to calculates the Haar wavelet responses in x and y direction within a circular neighborhood of radius 6s around the interest point and the responses are represented as points in a space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate. Then the horizontal and vertical responses within the window are summed to yield a local orientation vector. Longest such vector over all windows defines the orientation of the interest point[8].

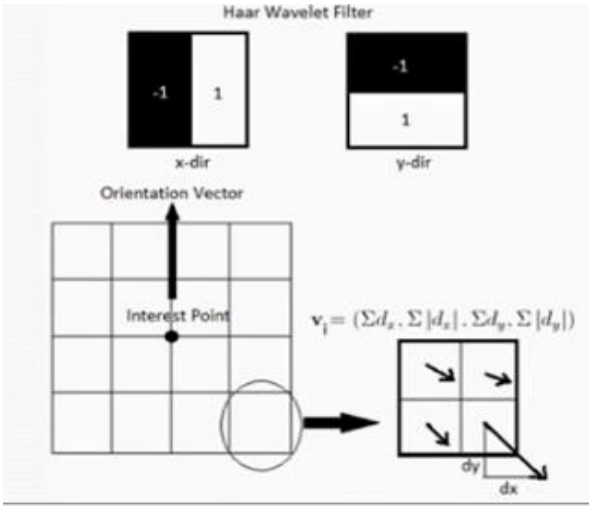

Figure 4. 3 description of the detected interest point with respect of their orientation

Pseudocode for SURF description of interest point detected by ORB

For each interest point detected from ORB output

Calculate orientation

Window area of 20s around point

Divide window into 4x4 subareas

For each subarea

Calculate Haar Wavelet

Smooth with Gaussian

Form feature vector

End for each subarea

Store feature vector

End for each interest point

Output: feature vector

4.2.3. Matching Step

The third and the final step is the matching step. In this process after detecting the interest point from the given image and describing them with respect to their direction and orientation characteristics, the final step that remains is matching those interest points of the trainee image with the interest point it founds at test image with the same procedures mentioned earlier. For this step it uses FLANN based matcher. FLANN stands for fast library for approximate nearest neighbors which work with faster time. It contains algorithm which is useful for fast nearest neighbor search in large dataset and for high dimensional features [36]. In this algorithm it takes one feature template from the descriptor and looks for nearest neighbor pixels which is extracted from the detected interest point. Once obtained the list matches, it calculates the distance between such key points

$$dk(p,q)=|\omega k(p)- \omega k(q)\ |$$ where dk is the distance q and p is the interest point

Then it finds the number of good match where their distance is less than a threshold in which the threshold can be calculated as

$$threshold = max(2*min\_dist,0.02)$$

However, during matching process there are some observed points during the matching process which is missed match from the point which is not it's match. In order to eliminate this mismatching it apply RANSAC algorithm. RANSAC is one of the most efficient robust estimation algorithms that used to eliminate the wrong matching point. In this algorithm the missed matched points are fitted by the least square method, in order to meet the prices matching [24]. In this algorithm it selects random sample from the point detected and described and form model parameter to calculate the parameter model to determine the number of points from all points that are in accordance with prescribed tolerance. If the number of the correct points in the point set is more than the predetermined threshold value, it uses the determined interior points to

re-estimate the model parameters. Otherwise, it will repeat the above steps up to K times until the number of correct points is more than the predetermined threshold values.

The way the algorithm chooses K is by finding the probability of success atleast once after applying the algorithm starting from intial trails. In this first the probability of real inliers obtained then from the number of samples drawen in each iteration K is obtained

$$P = 1 - (1 - p^n)^k \quad \text{.....equation 2}$$

$$k = \frac{\log(1-P)}{\log(1-p^n)} \quad \text{.....equation 3}$$

P is the probability of one success after applying K

P is the probability of real inliers

n is the number of samples drawn in each iteration

Flow of the RANSAC algorithm can be summarized as follow

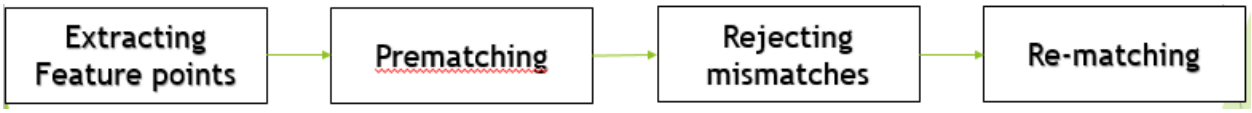

Figure 4. 4 application RANSAC for the elimination of missed match from the matching process

Finally, after founding the good match, it counts the number of good much it found from the above algorithm then compare it with the threshold. In this case, to say the image is matched the number of match should be grater then certain threshold, in our case the number of good much should be more than five. The reason to choose five is that the probability the algorithm get the good match will decrease to the actual match it found in the matching step.so to reduce the probability for no match found by still having the match we reduce good match to be more than five.

Generally, in this process first it divides the collected images as training image and test image. The training image is the one, which is, stored in database and extract all the feature and information which it going to use it in order to predict the test image. Test image is the one

which it uses to test the prediction. So the image which have high resolution and clear from any noise are trained by the algorithm and stored as trainee data in the database. For test purpose any image that is going to be classify with different resolution and size are chosen. First, the camera will start recording. From the image that is recorded by the camera, interest point are detected and describe thoes interest point by ORB and SURF algorithm respectively. Then the FLANN matcher tries to match with overall image trained and stored in the database. If the matching number is grater than the threshold, the system assumes the image is found in the database and the one that is compared in the previous step will be put as an output. If the matching number is lower than any image stored in the database the system assumes no match is found so it goes back to retrieve other image from the camera. The flow of the computer vision pipeline is looked like the following

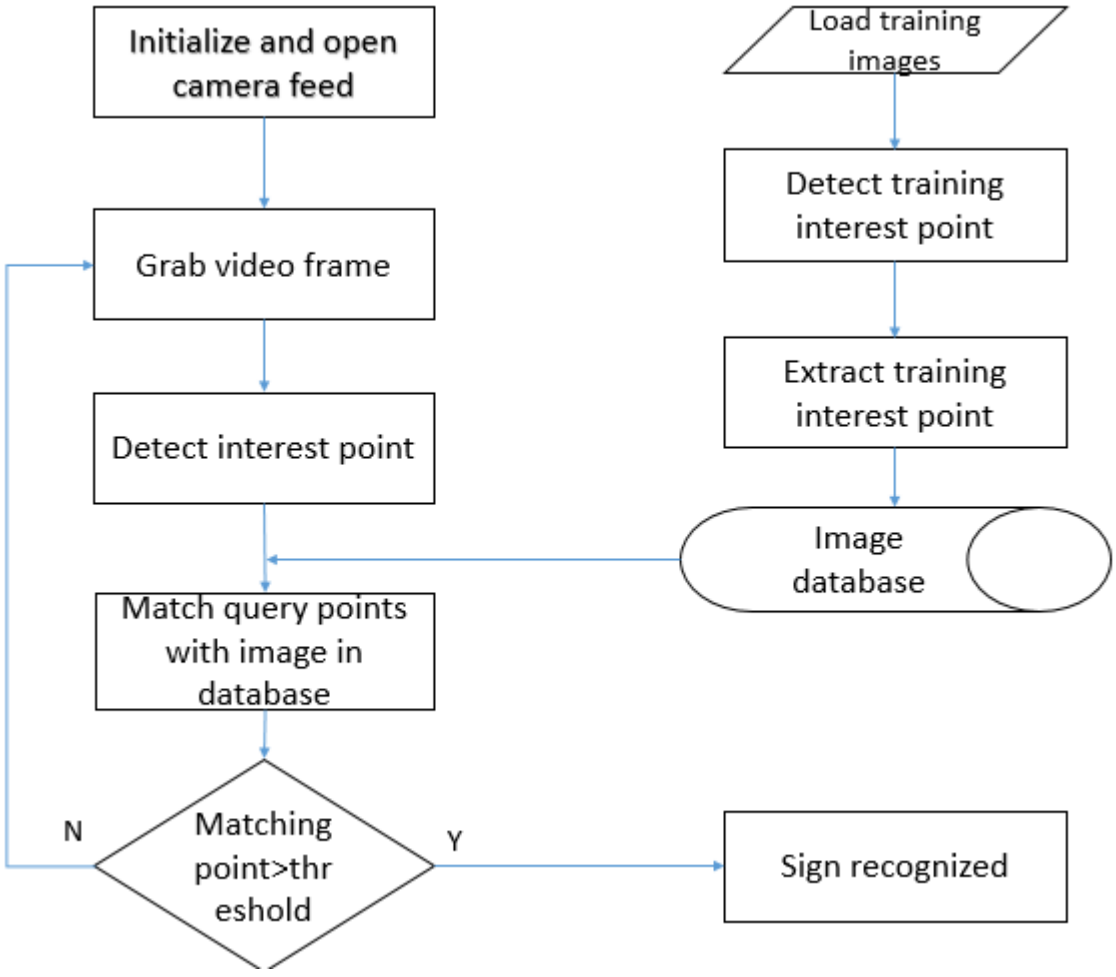

Figure 4. 5 the flow for matching between trainee mage with test image

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.

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 [37].

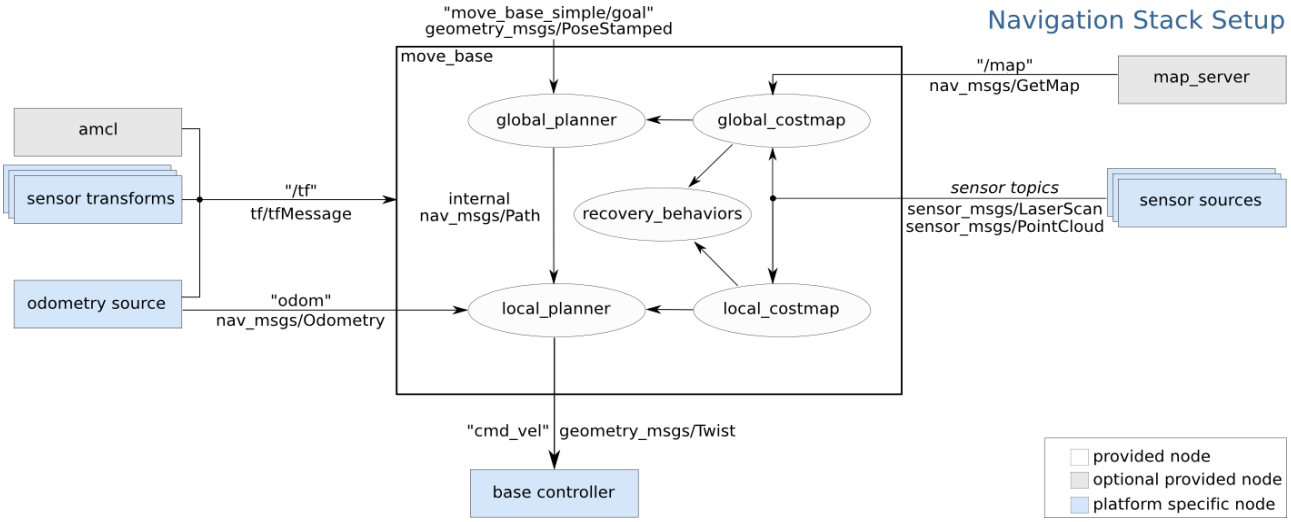

Figure 4. 6 general sachem move_bas path planner

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, odometry data, and sensor transforms. Sensor data is used to generate local cost maps while odometry data is used for relative position estimation. The sensor transforms contain geometric information of the robot such as the relative position between the mobile base and the LIDAR sensor. 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.

Since robot navigation focus on generating a global path to navigate the robot from source to destination it need to create 2D geometric representation of its environment which is called map using the data provided from the LIDAR sensor. So using the method SLAM and Gmapping package provided by ROS a 2D base map was created by using LIDAR laser and odeometery data. First Rviz simulator are opened and make wireless communication between the turtlebot3 and master pc, the robot LIDAR start to collect localization data by using 360 degree routable LIDAR sensor and start to build map by using teleoperation command. In the process of SLAM there are five vital steps the robot has to take in order to build the map and make it ready for autonomous navigation. The first step is the robot takes landmarks as an input data using 360 degree routable LIDAR sensor. Landmark are features which can be re observed and differentiate easily from the surrounding and is used by the robot to estimate its own position [5].

The second step is from the data input the features are extracted using spike landmark and RANSAC depending on the type of landscape (smooth and non-smooth). If the surface is not smooth it uses spike landmark to extract the feature point by finding values in the range of a laser scan where two values differ by more than a 0.5m (e.g corner point). This means that to say there is landmark behind other landmark the laser have to return from the front landmark 0.5 m difference from the land mark existing behind []. while RANSAC used for landscape which are smooth by finding lines from the extracting land marks and by randomly taking sample of laser reading it finds the best fit line that run through this reading by least square approximation method and checks for laser reading that lies close to this lines (e.g. edges).

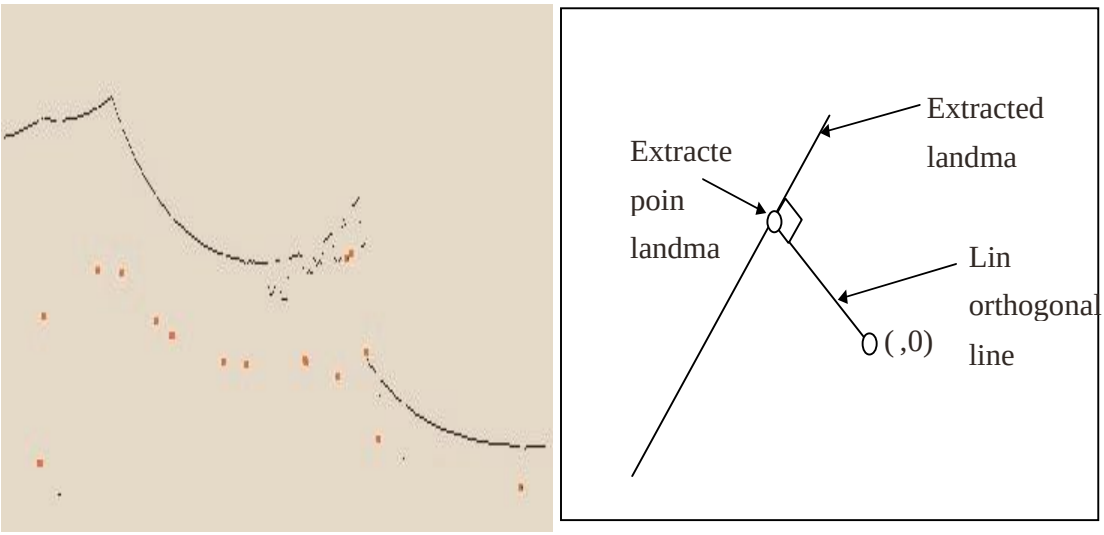

Figure 4. 7 spikes and RANSAC extraction

Finally, after 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.

Generally, to create a map using SLAM first by using LIDAR distance value have to be obtained from the X Y plane of the model object which is already built. Second the position of the robot have to be obtained by the principle of odometry sensor.

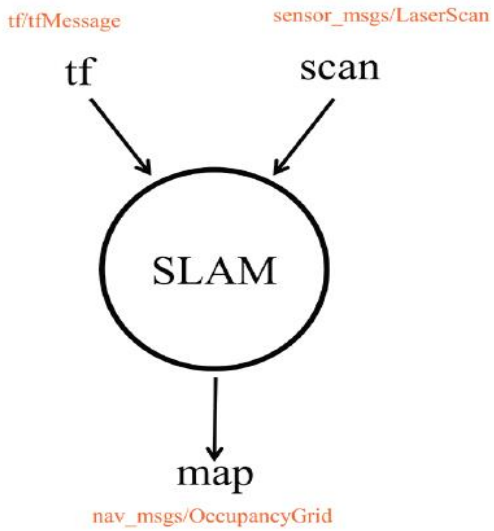

Figure 4. 8 map build by the robot using Laser scan (scan) and odometry (tf)

The flow of SLAM is described in figure 4.9. In this flow each node are launched when launching the existing Launching SLAM file. Sensor_node it runs the LIDAR sensor and sends the scan information for slam_gmapping node. In turtlebot3_teleop is teleop keyboard node that can receive the keyboard input and control the robot motion. This node sends the rotational and transitional speed command to the turtlebot3_core. Turtlebot3_core receives the transitional and rotational speed command and move the robot. In this node it frequently publish the odom data

which is going to be converted to relative coordinate in tf form. In turlebot_slam_gmapping node creates the map based on the scan information from the distance measuring sensor by LIDAR and the tf information. Finally the map_saver node create map.pgn and map.yaml file which include the information file for the map.

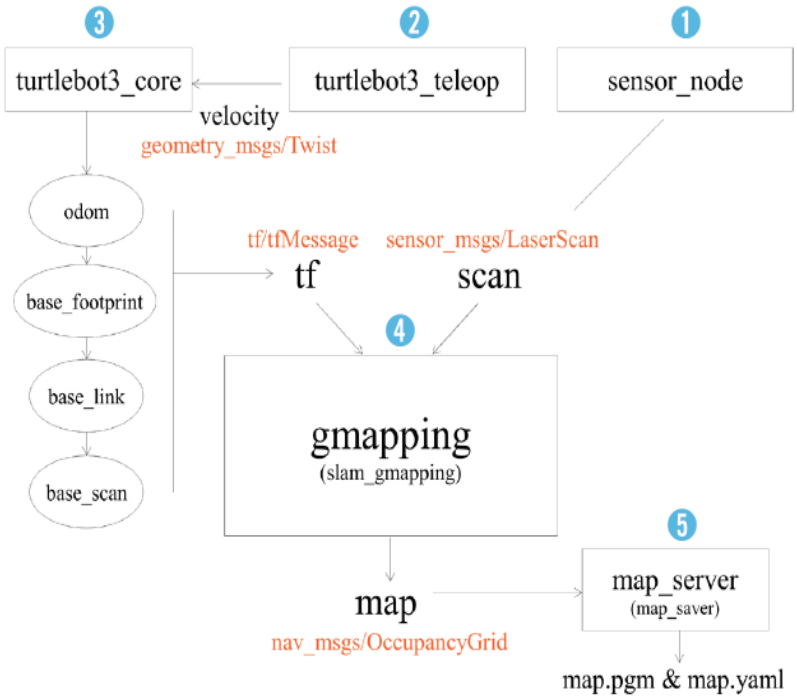

Figure 4.9 flow chart for creating the map using SLAM

4.3.2. 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 sign 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 re pathing (Danger sign, wet floor, no trans pass or under construction) and Motion sign (stop sign). The controller waits for receiving valid message from the computer vision pipeline publisher message.

Once the message is received, it will decide if it can see a re pathing sign or a motion signs. If re pathing sign is detected, the controller tries to move the robot close to it and activate the path planning algorithm in ROS (cost map) to change the path previously it planned to move to

another path so that it can't reach the area which is covered by the re pathing sign. If a motion sign is detected, the controller tries to make the robot to stop its motion. This process repeats till it reaches the destination which is given by the user.

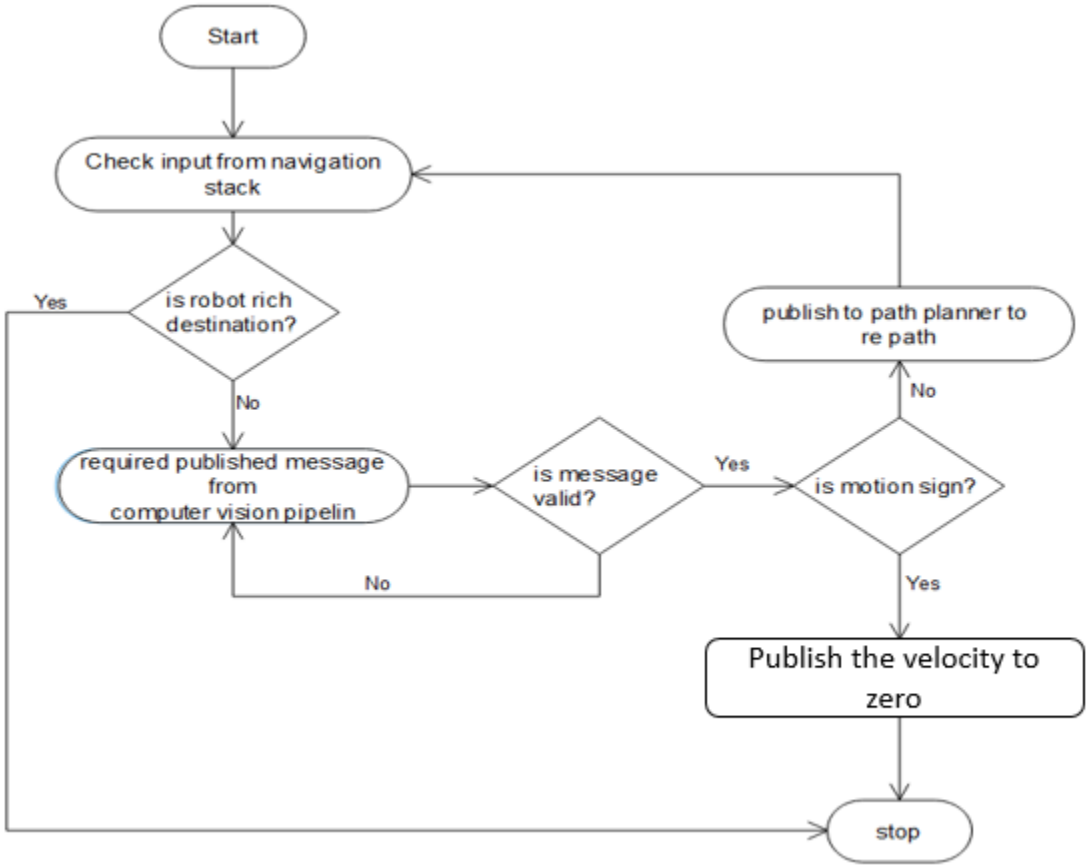

Figure 4. 9 flow chart

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 [25]. 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.

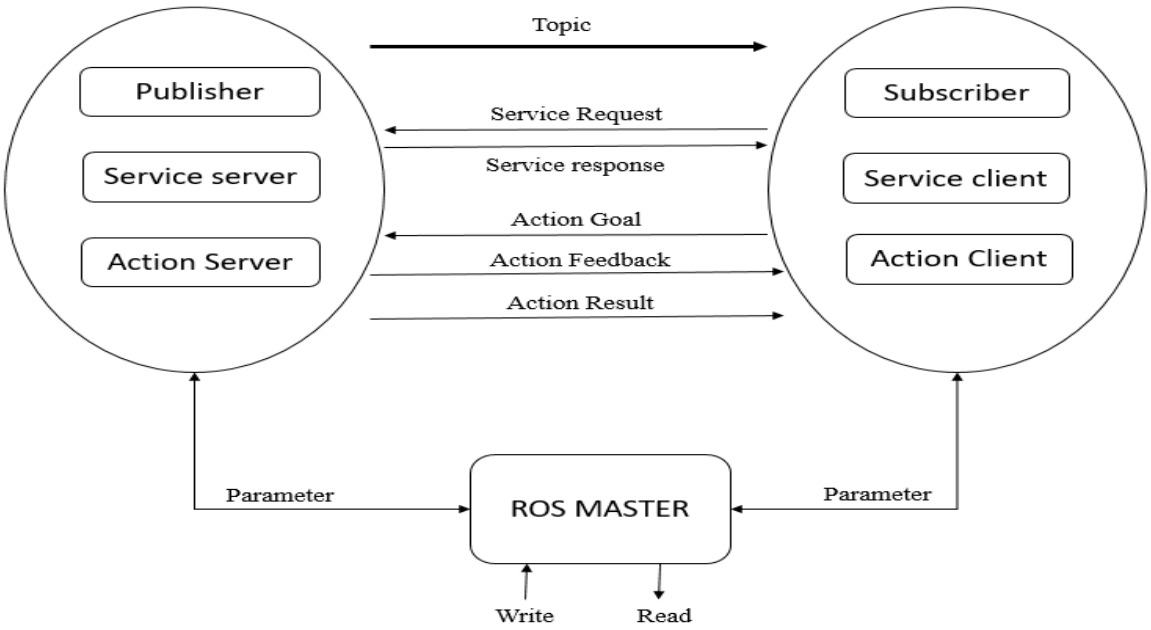

Figure 4.10 Message Communication Architecture

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

5. 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

Since the robot have to detect and avoid obstacle, the robot should have Laser scanner that used for collecting landmark and measuring reflecting distance. So LIDAR is 360-degree routable laser scanner with frequency of 5.5Hz with sampling of 360 point within 6-meter range that used to build 2D map [38] . LIDAR based on the laser triangulation ranging principle it emits infrared signal and the emitted signal will be reflected by the object to be detected then the reflected signal will be sampled by vision acquisition system in LIDAR.

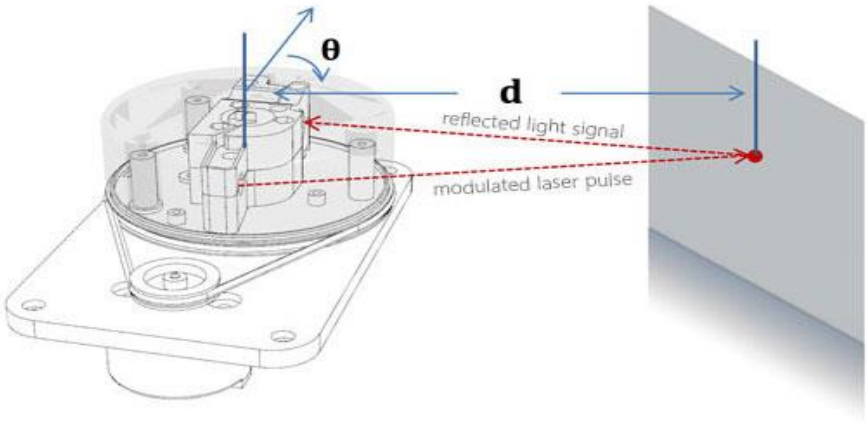

Figure 5. 1 LIDAR [37]

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 landmarks. And also we can integrate camera to raspberry pi3 with initial port for transferring the image and be able to processed 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 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 also some LIDAR driver 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. In this DYNAMIXEL wheel are used that can measure the distance and the position through the rotation of it. The laser scan is done by Laser scan mounted on the top of the robot by the device called LIDAR. Due to laser ray should not distracted by any object the laser scanner is mounted at the tp of the robot. This device is responsible for collecting the landmark and useful to avoid obstacles.

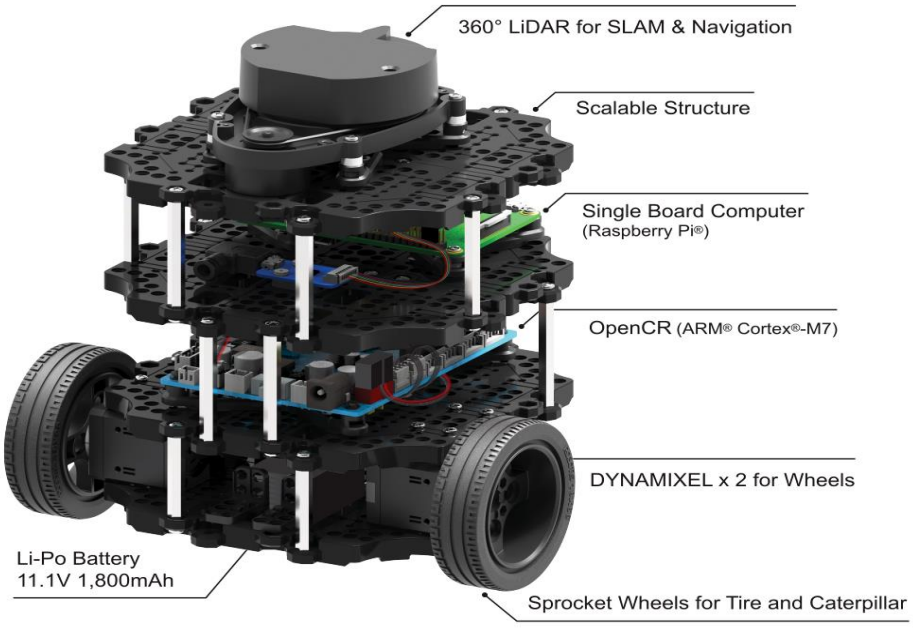

Figure 5. 2 Turtlebot3

5.2. Recognition Step

In this step Pycharm 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 python language were used. In this step first different type of traffic sign image were collected and resized to 350x251 to reduce the complexity of processing the image. 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. In this process corner point are detected as interest point by choosing pixel from those consecutive chosen pixel surrounding the arbitrary pixel by using Harris algorithm. Sample code for the conversion to gray scale image, Gaussian blurring and detection of interest point is listed as below. Image 1 is a test image that is going to be tested while image 2 is the image that is stored in the database as trainee.

- 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 SURF description algorithm. In this algorithm it calculates Haar Wavelet response of x and y direction by applying a window size of 20s around the point to yield orientation vector. The sample code for the description of interest point by using SURF described as follow.

- SURF = cv2.xfeatures2d. SURF_create()
- KPTS1, description_image1=SURF.compute (Blur1, KPTS1)
- KPTS2, description_image2=SURF.compute (Blur2, KPTS2)

After description, matching those described interest point of trainee with described interest point of test image were proceeded by using FLANN based matcher algorithm. In this process first from the detected point it searches for nearest point and made a list for calculating the distance between the lists. Then threshold is given as 0.7 to found the good match. The reason to choose 0.7 is that we tried to give varied threshold value starting from 0.2 up to 0.9. From 0.2 to 0.65 the number of missed match is increases this lead to problem in the accuracy. The threshold starting from 0.75 to 0.9 also lead to increase missed match and number of no match is increase. Therefore, for accurate measurement we choose 0.7. If the distance is less than the given threshold it found the good match else, it decides as no match were found.

Good_match =[m1 for(m1,m2) in matches f m1.distance < 0.7 *m2.distance]

Finally, after it found number of good matches, it proceeds to count and compare the number of those good match with the threshold given. In this case threshold are given as 5 and if the number of good match is greater than the threshold it made an output with matching line between the trainee and test image to indicate match is found else it made an output test image is not matched with the trainee. However, to reduce the probability of finding missed match RANSAC algorithm for filtering missed match was implemented.

```
If len (good_match) > 5

        M, mask =  cv2.findHomgraphy(src_pts, dst_pts,cv2.RANSAC, 5.0)

        Cv2.polylines (canvas, [np.int32(dst), True, (0, 255, 0), 3, cv2.LINE_AA)

        Cv2.imshow ("matched found", matched)

        Pub_traffic_sign(msg_sign)

Else

        Print ("Not matched is found")
```

5.4. 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 odometery data by SLAM algorithm. In this process turtlebot3 is drove around side of the wall of the built map using teleoperation to create a map. While driving the turtlrbot3 around the side wall, turtlebot3_lds node is launched for the robot to start to uses LIDAR sensor in order to receive the localization data and to send the data to turtlebot3_slam_gmapping node. After sampling it start to process the sampled data to make output of distance and angle value between the object and the robot through the USB port interface to the raspberry pi3 by the embedded DSP in LIDAR [26].

LIDAR based on the laser triangulation ranging principle it emits infrared signal and the emitted signal will be reflected by the object to be detected then the reflected signal will be sampled by vision acquisition system in LIDAR. After sampling it start to process the sampled data to make output of distance and angle value between the object and the robot through the USB port interface to the raspberry pi3 by the embedded DSP in LIDAR.

Turtlebot3_teleop_keyboard node is launched next to drove the robot all around the side wall of the model. Within this turlebot3_core node is launched in order to collect the commanded given by the teleop_keyboard in order to receive the translation and rotational speed from the

keyboard to publish the Odom data to turtlebot3_slam_gmapping node. Finally, turtlebot3_slam_gmapping node creates map and map saver node is launched to save the created map by the slam_gmapping node.

5.5. Obstacle Avoidance

The ROS kinetic package for raspberry pi3 contain gmapping node that is used by the robot to make a map from the environment and saving it. The 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 between the robot and obstacle is obtained from the LIDAR sensor then the position of the robot, obstacle information and occupancy grid map obtained from SLAM 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. In this first the time required for each distance is calculated by maximum velocity the robot have 0.18 m/s with until 6-meter range of the LIDAR. Then from the different range cost_scaling_factor (0.5) is added on each distance and the one which have lower time than the radius of the robot times the area of the currently the robot exists is chosen as the distance that inform the robot weather it is free space or obstacle exist space. 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)

So, first distance is set in local_costmap_params.yaml which is used for path planning and obstacle avoidances in limited area. Some of the aforementioned factors code are:

Indicate the object as an obstacle when the distance between the robot and obstacle is within this range.
obstacle_range: 2.5

```
# sensor value that exceeds this range will be indicated as a free space
raytrace_range: 3.5
# external dimension of the robot is provided as polygons in several points
footprint: [[-0.110, -0.090], [-0.110, 0.090], [0.041, 0.090], [0.041, -0.090]]
# radius of the robot. Use the above footprint setting instead of robot_radius.
# robot_radius: 0.105
# radius of the inflation area to prevent collision with obstacles
inflation_radius: 0.15
# scaling variable used in costmap calculation. Calculation formula is as follows.

# scaling

# exp(-1.0 * cost_scaling_factor *(distance_from_obstacle - inscribed_radius)) *(254 - 1)

cost_scaling_factor: 0.5
# select costmap to use between voxel(voxel-grid) and costmap(costmap_2d)
map_type: costmap
# tolerance of relative coordinate conversion time between tf
transform_tolerance: 0.2
# specify which sensor to use
observation_sources: scan
# set data type and topic, marking status, minimum obstacle for the laser scan
scan: {data_type: LaserScan, topic: scan, marking: true, clearing: true}
```

Finally, after costmap calculate the obstacle area, possible collision area and robot movable area the DWA (Dynamic window approach) method select the appropriate speed for the robot that can quickly reach the target point by avoiding collision from the obstacle. Some DWA published the speed command to the robot in dwa_local_param.yaml are

```
# robot parameters
max_vel_x: 0.18 # max velocity for x axis(meter/sec)
min_vel_x:-0.18 # min velocity for x axis (meter/sec)
max_vel_y: 0.0 # Not used. applies to omni directional robots only
```

min_vel_y: 0.0 # Not used. applies to omni directional robots only

max_trans_vel: 0.18 # max translational velocity(meter/sec)

min_trans_vel: 0.05 # min translational velocity (meter/sec), negative value for reverse

trans_stopped_vel: 0.01 # translation stop velocity(meter/sec)

max_rot_vel: 1.8 # max rotational velocity(radian/sec)

min_rot_vel: 0.7 # min rotational velocity (radian/sec)

rot_stopped_vel: 0.01 # rotation stop velocity (radian/sec)

acc_lim_x: 2.0 # limit for x axis acceleration(meter/sec^2)

acc_lim_y: 0.0 # limit for y axis acceleration (meter/sec^2)

acc_lim_theta: 2.0 # theta axis angular acceleration limit (radian/sec^2)

5.6. Mapping Graph

Since 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. After building the packages and launching each nodes, each nodes begun to communicate with the master and create the whole system that operate the navigation, path planning, obstacle avoidance, sign recognition, controller the action of the robot based on the sign, current position estimate etc. through communicating and exchanging information with their required node until termination of the whole system. The whole node communication made by the ROS to make single operating robot made by the Rqt_view_node is shown as below figure.

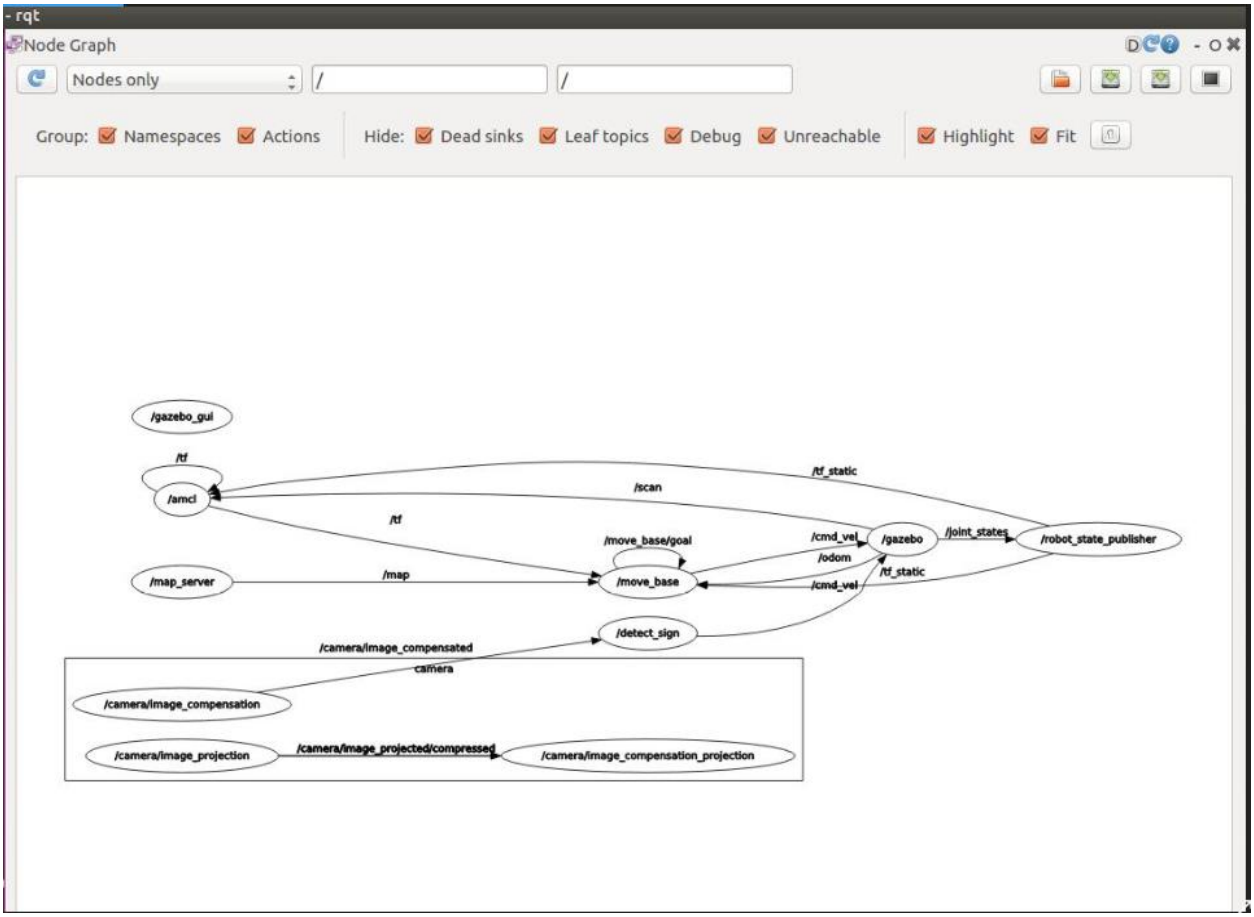

Figure 5.3 Node graph made by Rqt that show the connection and exchange of message between subscriber and publisher node with their respective types of topic

Chapter Six

6. Result, Evaluation and Discussion

This chapter presents the results of this work. It includes comparisons with other literature.

6.1. Recognition of sign

To verify the proposed hydride algorithm is feasible, the paper compares the classical SIFT algorithm and SURF algorithm with proposed algorithm separately for image recognition. In order to verify the effectiveness of the matching 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. So counting those point will lead to found the number of correct matching point. 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 or it should be blob point. If the detected feature point is in flat surface or other places, those point will be invalid. On the other hand, the number of matches it found also measured will be used to compare those algorithms with the proposed algorithms. The threshold is 0.7 which controls the match points and the experimental environment is pycharm and OpenCV 3.4. This paper will take two groups of images for validation and evaluate the performance of the algorithm. The first set of pictures called Danger sign from the typical images, the size is 480 * 360 and with JPG format; the second set of pictures similar with the first but different in the scale, rotation and intensity.

Figure 6. 1Two original image of signage

From the input image key point detection by SIFT, SURF and ORB+SURF obtained as follow

Figure 6. 2 Key point which detected by SIFT

Figure 6. 3 Key point detected by SURF detector

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 [40].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 texturless patches are nearly impossible to localize they can lead for increase the number of missed matched [28]. From the above image the key point which is detected by SURF and SIFT there are point which is on flat surface that lead it for missed match. There are more number of key point that are detected in flat surface in the key point which is detected by SURF and SIFT.

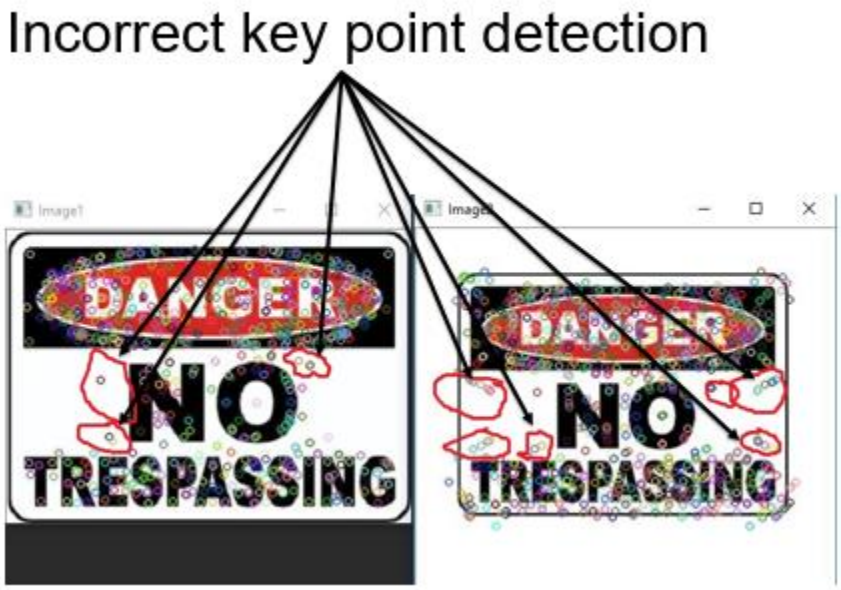

Figure 6. 4 Key point that is detected by SURF and SIFT

However, the proposed algorithm detects key point only at the edges and corner for it reduce the number of missed match by reducing the incorrect detection of key point.

Figure 6. 5 Key point detected by ORB+SURF

From the above output picture ORB+SURF reduce the key point which is on the flat surface almost to zero.

Matching correctness and matching efficiency for SURF, SIFT and ORB+SURF have been evaluated through different image transformation like scaling, rotation, intensity and selectiveness characteristic of the algorithm.

6.1.2. In the case of Intensity

The images with varying intensity and color composition values were used to compare the algorithms and results are presented in Table 6.1 figure 6.6. From the given output the number of key point detected by ORB+SURF is best and the time it spent to detect all those point is minimum that make it also the best in detection rate. In the matching process ORB+SURF have high matched number with number of missed match zero. But SIFT has fast matching rate because it has lower number of matching and 4 missed match points. However, from the result obtained ORB+SURF perform accurate, more numbers of matching point and fast detection rate for the given image which varied with the intensity.

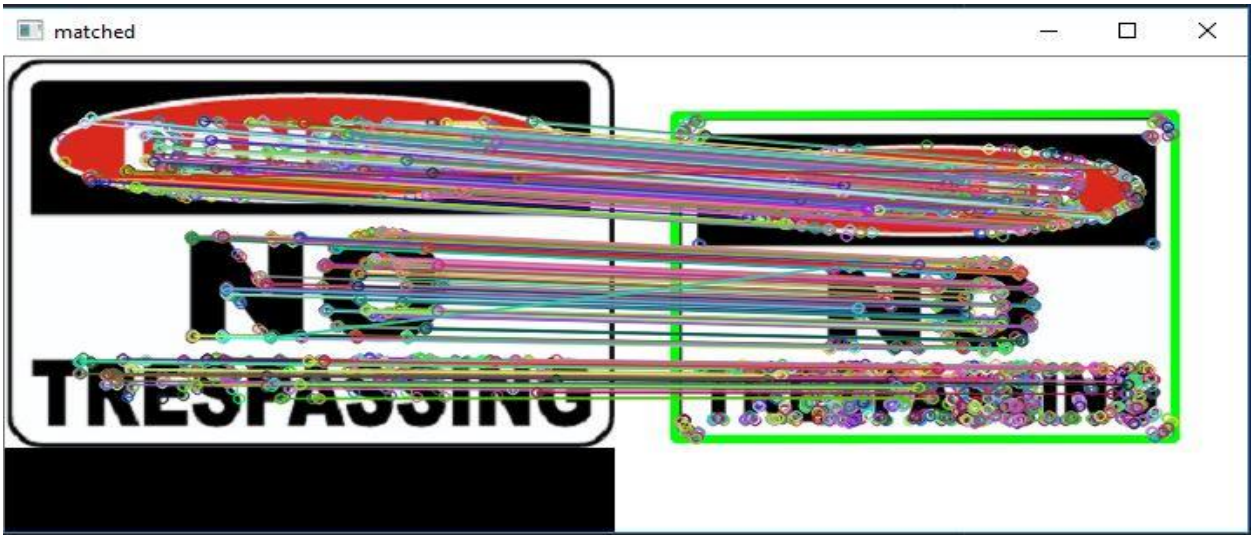

Figure 6. 6 Matched by ORB+SURF algorithm

Table 6. 1 Feature detection and matching capability comparison on time of change in intensity

	No Kpt1	***No kpt2***	***Time for detection***	***Matching number***	***Missed matched number***	***Time for matching***	
SIFT	358	347	0.031(sec)	208	4	0.061(sec)	
SURF	669	746	0.046(sec)	279	6	0.093(sec)	
ORB+SURF	914	1405	0.031(sec)	346	0	0.13(sec)	

6.1.3. In case of Rotation

It has been considered here a rotation of 45 degrees to the image to be matched. The results are given in the Table 6.2 and Figure 6.7. With rotated image, as one can see from Table 6.2, ORB+SURF have high detection speed than the other algorithm. On the other hand, it has high number of detected interest point and matching pointes with zero number of missed matched. However due to lower matching number sift has lower matching time but ORB+SURF have clarity in the matching and have compatible matching time with its matching number.

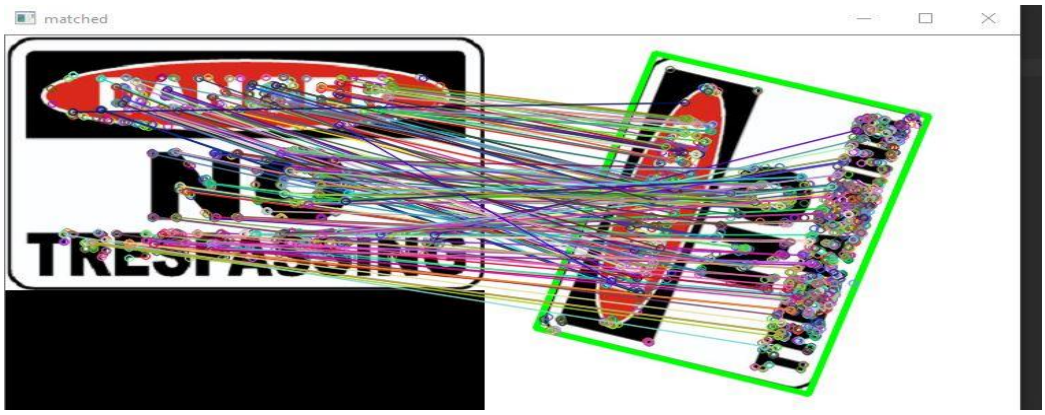

Figure 6. 7 Rotated image matched by ORB+SURF

Table 6. 2 Feature detection and matching capability comparison on time of variation in Rotation

	No Kpt1	***No kpt2***	***Time for detection***	***Matching number***	***Missed matched number***	***Time for matching***
SIFT	358	359	0.046(sec)	197	10	0.061(sec)
SURF	669	1728	0.046(sec)	196	31	0.13(sec)
ORB+SURF	914	1783	0.031(sec)	428	0	0.22(sec)

6.1.4. For case of Scaling

It has been considered here scale down to the image to be matched. The results are given in the Table 6.3 and Figure 6.8. With scaled down image, as one can see from Table 6.3, ORB+SURF have high detection speed than the other algorithm. On the other hand, it has high number of detected interest point but low in matching pointes with zero number of missed matched. When the distance of the image increases the description for those detected point reduces and more number of missed match point will be eliminated so the matching number will decrease. With. In this matching evaluation with SURF and SIFT we can see that ORB+SURF is reliable matcher between trainee and test image but have lower capability detection for small size image so it is not reliable for long distance detection. SIFT is reliable for long distance matching between trainee and test images.

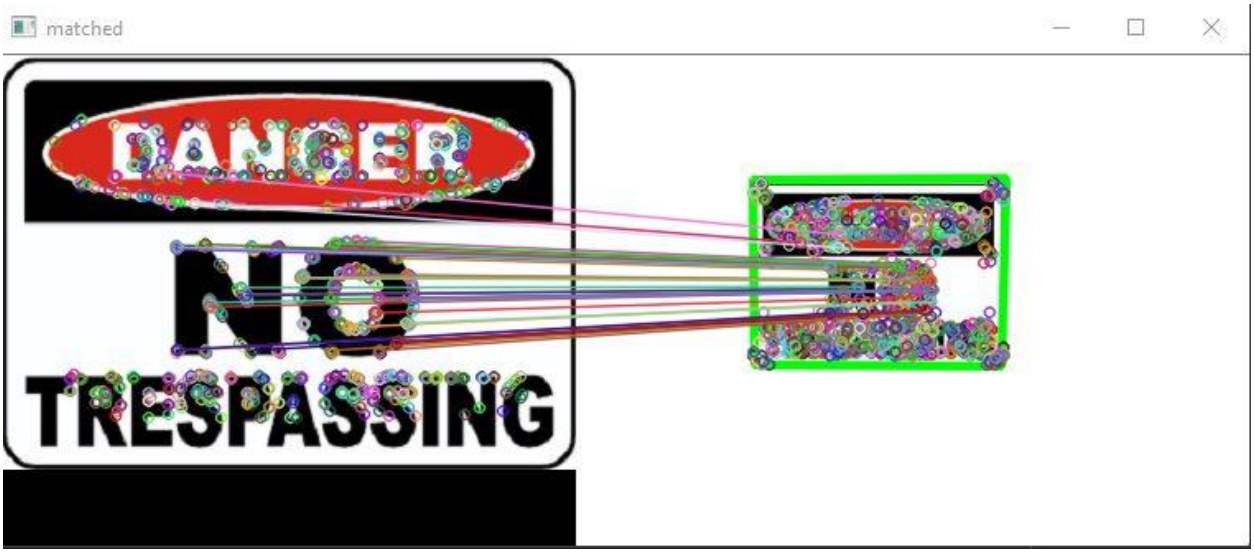

Figure 6. 8 Low scaled image matched by ORB+SURF

Table 6. 3 Feature detection and matching capability comparison on time of change in scale

	No Kpt1	***No kpt2***	***Time for detection***	***Matching number***	***Missed matched number***	***Time for matching***
SIFT	358	235	0.03(sec)	146	10	0.06(sec)
SURF	669	328	0.05(sec)	86	31	0.07(sec)
ORB+SURF	914	1293	0.01(sec)	40	0	0.2(sec)

6.1.5. For case of Selectiveness

In this case an image which have various similar images are used as test image to compare the algorithm of their selectiveness characteristic with low number missed match as figure 6.9. This evaluation insures as the weather the proposed algorithm is reliable detection or not in situation where similar but different indication sign exists within indoor environment. From the figure 6.9 one can observe that two similar sign but having different indication is included. ORB+SURF detect and match similar point from both sign because DANGER and N character are included in both image. Even the algorithm match similar character from both images it doesn't indicate that both test images are equal with trainee image. In this case the algorithm observes for the image which have greater number of correct match than threshold (in our case 9). So, for matching number higher than the threshold is the one that is similar with the trainee image. But in case for SIFT and SURF one can observe that the matching image include not similar character this make to increase the number of missed match. From the given output in table 6.4 the number of key

point detected by ORB+SURF is best and the time it spent to detect all those point is minimum that make it also the best in detection rate. In the matching process ORB+SURF have high matched number with number of missed match zero. The matching time almost similar to SURF but SIFT have lower matching time. From the evaluation we can observe that ORB+SURF is reliable and selective for the intended image and fast for detection the intended sign from other related images.

Figure 6. 9 Detecting the required image from different image by SURF+ORB

Table 6. 4 Selective capability from merging similar image signs

	No Kpt1	***No kpt2***	***Time for detection***	***Matching number***	***Missed matched number***	***Time for matching***
SIFT	358	999	0.08(sec)	128	6	0.1(sec)
SURF	669	3792	0.09(sec)	167	5	0.2(sec)
ORB+SURF	914	4000	0.03(sec)	171	0	0.2(sec)

From the above evaluation in different image transformation that have been done between ORB+SURF, SIFT and SURF we can conclude that ORB+SURF is reliable, fast, robust and

selective algorithm for short distance detection of sign that can be used for navigation of the robot.

6.2. Computer vision pipeline analysis

After analyzing the proposed algorithm separately on pycharm simulator, implementing the recognition algorithm in ROS-kinetic was carried out. In this process the video which is recorded by raspberry pi v1 camera, which is mounted on turtlebot3, was transferred to the master computer through Wi-Fi and conversion the image to compressed image was carried out to make the input image ready for recognition. An example of the output from the computer vision pipe line is shown in Fig. 6.7.

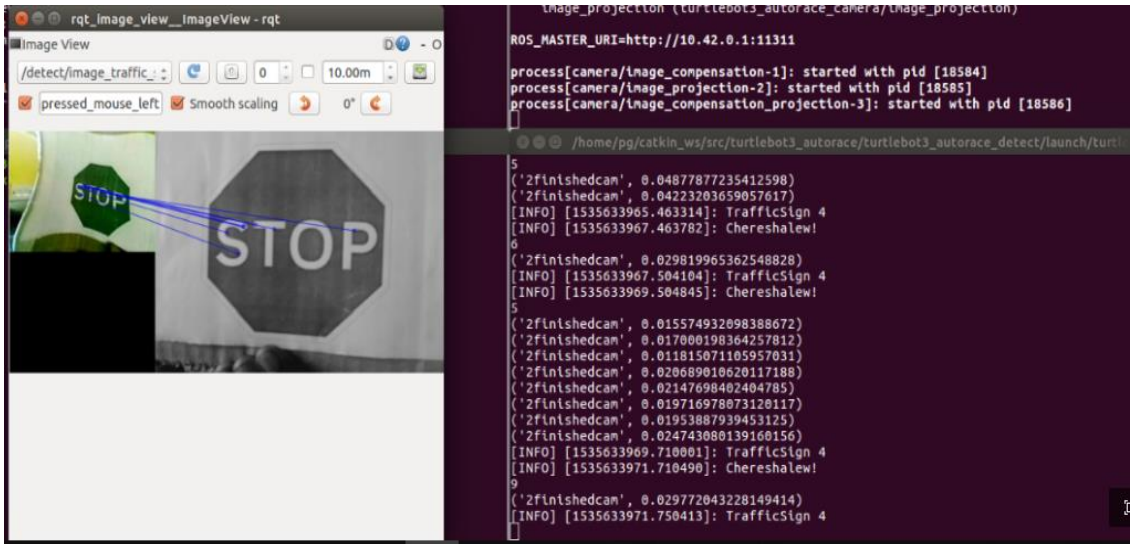

Figure 6. 10 When Turtlebot3 detect the Stop sign

Figure 6.7 shows matching line which interconnects the key point stored from trainee image with the key point which is detected and described from the test image along with the time it cost for the computation. In this process after recognizing the image Turtlebot3 decide if it can see a re pathing sign or a motion signs. If it is re-pathing sign it activate the path planning algorithm in ROS (cost map) to change the path previously it planned to move to another path so that it can't reach the area which is covered by the re pathing sign. If a motion sign is detected, the controller tries to make the robot to stop its motion. fig below show the motion plan when it detects the sign from riViz simulator.

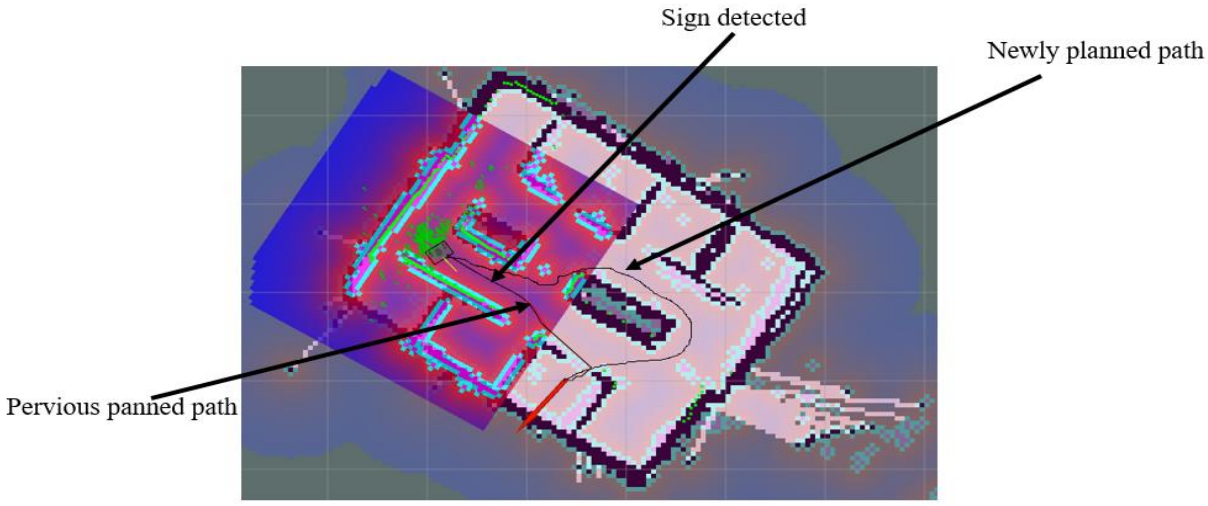

Figure 6. 11 Re-planning the path after detecting the sign

Red arrow shows the destination that the user gives to make the robot able to go. The red line from the robot to the destined arrow is the path which is planned by the robot to reach in to the destination. After it detects the No trans pass sign the robot will change the first path in to other path that can take the robot to the required destination.

To evaluate the performance of the proposed algorithm, other algorithm was implemented and compared with proposed algorithm with respect to time, number of detection per minute, number of missed detection per minute and their maximum distance which the algorithm detects the sign with the same resolution of the camera have been evaluated.

Table 6. 5 Detection and matching capability by the camera of the turtlebot3 movement

	Time(in second)	Number of detection per minute	Number of missed detection per minute	Maximum distance it can detect
SIFT	0.048	18	1	50 cm
SURF	0.032	21	70	45 cm
ORB+SURF	0.019	16	0	34 cm

From the above table as one can see is that the number of detection by SIFT and SURF after repetitiously putting the sign in front of the robot in different scale and distance for minute is better than ORB+SURF but SURF have more number of missed detection than correct detection for this reason SURF is less reliable for the navigation of the robot using sign. Although SIFT perform in detection better than ORB+SURF it has one missed match. But, ORB+SURF have no missed match and the time it cost for detection is better than the other two algorithms this make the proposed algorithm reliable and fast in detection. Regarding on distance SIFT have a higher capability for detecting sign from long distance than the other two algorithms but ORB+SURF is the least in distance.

6.3. Map building

Since robot navigation focus on generating a global path to navigate the robot from source to destination it need to create 2D geometric representation of its environment which is called map using the data provided from the LIDAR sensor. So using the method SLAM and Gmapping package provided by ROS a 2D base map was created by using LIDAR laser and odometry data. In this process first testbed which the turtlebot3 travel have been built as figure 6.12 then using the LIDAR sensor and odometry, localization have been collected and build the map on the riviz simulator which used by the robot for navigation as figure 6.12.

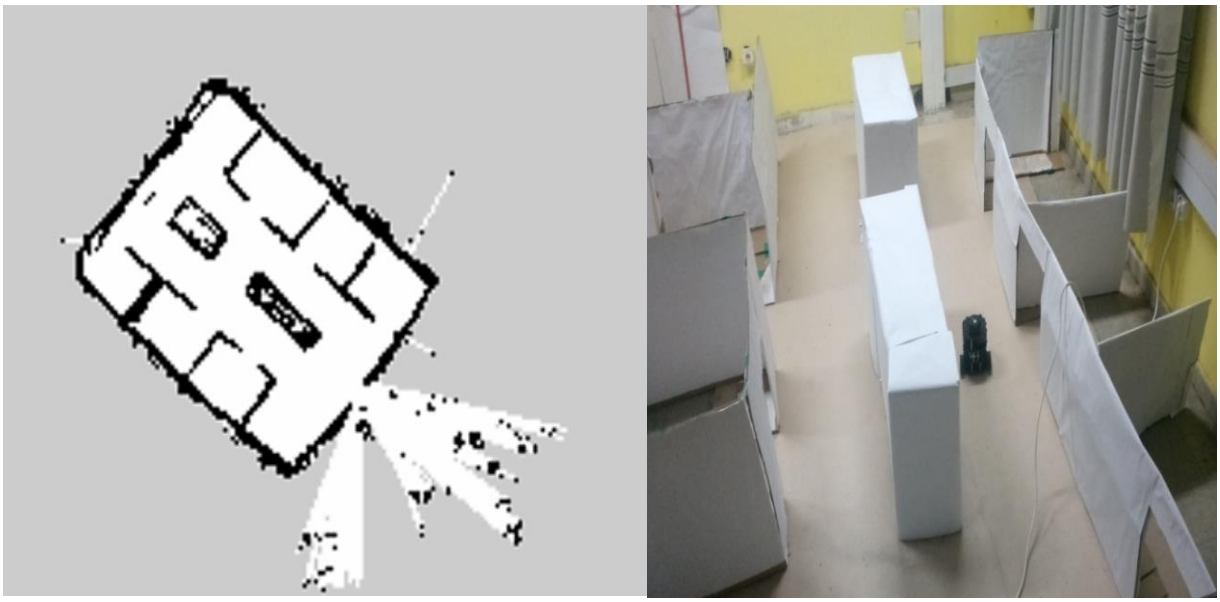

Figure 6. 12 Model built on real environment and the map built using Turtlebot3 LIDAR on rviZ

6.4. Obstacle detection and avoidance analysis

A Program was written to send simple navigational goals to the ROS navigational stack. These navigational goals are defined by distances in X and Y axes and the Z axis rotation. This allows the Turtlebot3 to reach its goal by avoiding any obstacles in its path. From the fig below shows built map by driving the turtlrbot3 around the side wall in order to receives the localization data by using LIDAR sensor. Even if the map is built, the LIDAR of the robot detect any obstacle within 6 meter ranges. The images in Fig. 6.13 are screen shots of Rviz visualization software taken while the TurtleBot3 moved towards its goal. The blobs of pink, blue and purple in the images represent the costmap created by the Turtlebot3. This costmap contains the real boundaries of objects in pink and the inflated space around it in blue. Inflated space is the minimum distance to be maintained between the object and the center of the robot, so that its edges do not collide with objects.

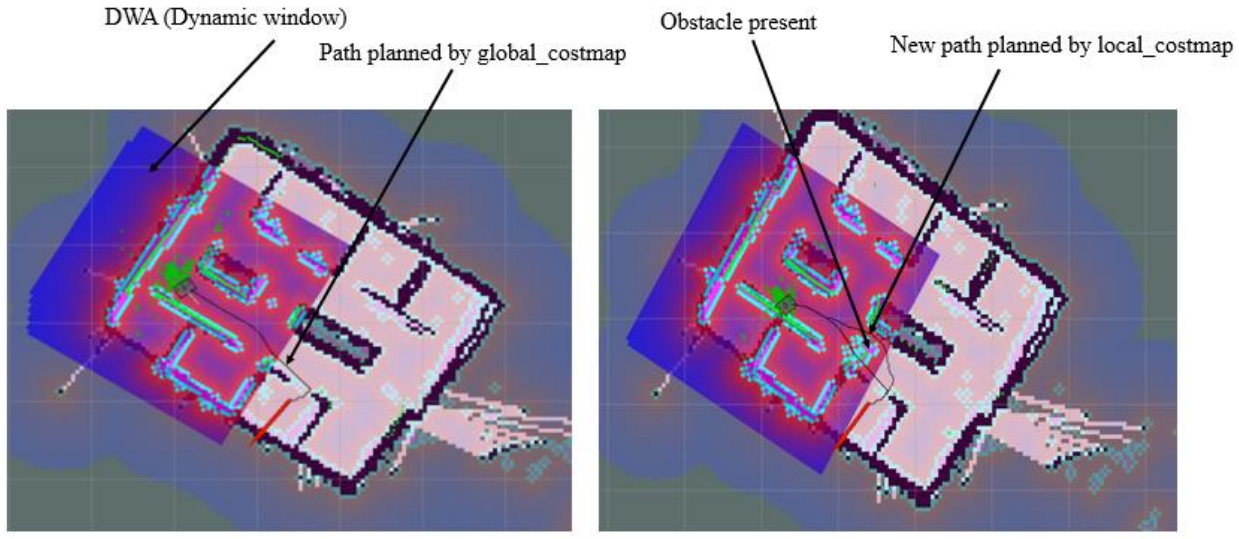

Figure 6. 13 Range covered by the Turtlebot3 DWA and when the robot try to pass by avoiding the obstacle

The screen shots were captured at different stages of the robot's movement to its goal. In the first image the Turtlebot3 is facing upwards 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 costmap, which is represented by a thin black line. First distance between the robot and obstacle is obtained from the LIDAR sensor then using position of the robot, 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 any collision 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 other algorithm. After implementing with robot the proposed algorithm performs dramatic improvement in detecting sign with lower time. 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. Since navigating the robot through unfamiliar environment requires understanding of signage and other useful visual information, the robot has to have a method for detecting recognizing signage with high speed through different difficulties. This work is focused in this type of detection and recognition of the signage and detection of obstacle to make the robot able to navigate through unfamiliar environment with high speed and safe.

A variety of hardware and software systems were integrated to create a working prototype robot that used to detect and navigate indoor sign for robot 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 characteristics in recognition of the object SURF algorithm has been chosen. However, since the image processing unit of the proposed system should be fast and accurate ORB and SURF 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 SURF were used. Finally, for matching purpose FLANN based algorithm is used.

The experimental results are compared with those of SIFT and SURF algorithm with different image transformation. SIFT have better matching number for the scaled dawn image this means it have high range of capability to detect with greator distance. From the robot experiment we can see that SIFT can detect sign that exists at 50cm ranges. This means SIFT have the capability to detect sign or object that exists at higher distance. ORB+SURF algorithm have high number

of key point detection than SIFT and SURF with minimum required time. The detection of key point by ORB+SURF is more on the edges and corner point and missed match rate is decrease than SIFT and SURF. However, the matching time for SIFT is higher than both SURF and ORB+SURF because of low in detected key point. But implementing the proposed algorithm in ROS the time required for recognition in the robot is less than SURF and SIFT. And also it has zero missed match capability when it compared with SIFT and SURF this make the ORB+SURF better for minimum distance with quality and time.

The second is for avoidance of the obstacle LIDAR was used.in this process Program was written to send simple navigational goals to the ROS navigational stack that allows the TurtleBot3 to reach its goal by avoiding any obstacles in its path.

7.2. Future work

This study was dedicated on detection and recognition of sign and avoidance of obstacle for navigation of the robot in safe paths using combination of ORB+SURF algorithm and DWA+Djkstrass path planning respectively. The researcher recommended the following future works to be continued for any new researcher who have motivated. Since time limitation the distance that the algorithm can detect can be improved using different techniques. In this proposed algorithm usually the SURF algorithm used to calculate the detected feature point by ORB, therefore SURF took long time due to the number of feature point detected by ORB is greater than SURF can describe them with less time. So SURF took much time to describe all of them. So, by reducing the complication of the method in which the SURF descriptor uses we can modify the description time as well as the matching time requires to calculate for more images. For obstacle avoidance, due to its superior performance the DWA method doesn’t yield a satisfying speed for avoiding and speed for the robot to pass the obstacle after avoiding it. So this method can be replace by other method that enable for the robot to avoid obstacle with less time requisition.

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".

[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]. Available: www.ncbi.gov.

[8] Luis A. Guerrero 1, Francisco Vasquez 2 and Sergio F. Ochoa, "An Indoor Navigation System for the Visually Impaired," 2012.

[9] ANdersen,Jens Christan: Ravn, Ole, "Mobile Robot Navigation," *DTU Liberary (Technical Information Center of Denmark),* pp. 1-191, 2007.

[10] H. Duarrant, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE,* vol. l, no. 1-9.

[11] Søren Riisgaard and Morten Rufus Blas , SLAM for Dummies, By the ‘dummies’ .

[12] "Wikipedia," Encyclopedia, 24 may 2018. [Online]. Available: https://en.wikipedia.org/wiki/Robot_navigation. [Accessed 15 jun 2018].

[13] Shuihua Wang, Xiaodong Yang, and Yingli Tian, "Detecting Signage and Doors for Blind Navigation and Wayfinding," *IEEE,* pp. 1-15, 2016.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded Up Robust Features," *IEEE*, p. 14.

[15] Darshana Mistry and Asim Banerjee , "Comparison of Feature Detection and Matching Approaches: SIFT and SURF," *Global Research and Development Journal for Engineering* , vol. ll, pp. 1-13, 2017.

[16] Ana Corrales Paredes, Maria Malfaz and Miguel A.Salichs , "Signage System for the Navigation of Autonomous Robots in Indoor Environments," *IEEE*, vol. 10, pp. 1-688, 2014.

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

[18] T. Alemu, "Recognition of Amharic Braille," Research gate, 2009.

[19] 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.

[20] S. G. Endalamaw, "Ancient Ethiopic Manuscript Recognition Using," A.A.U, Adiss Ababa, 2016.

[21] Dagnachew Melesew Alemayehu, Abrham Debasu Mengistu, Seffi Gebeyehu, "Computer Vision for Ethiopian Agricultural Crop Pest Identification".

[22] Kazem Ghazanfari, Saeed Shiri , "Real time text localization for Indoor Mobile Robot Navigation," pp. 1-5.

[23] YingLi Tian, Xiaodong Yang, Chucai Yi, and Aries Arditi, "Toward a Computer Vision-based Wayfinding Aid for Blind Persons to Access Unfamiliar Indoor Environments," 24 may 2012. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3636776/. [Accessed 15 Sepmtember 2017].

[24] JeongdaeKim and YongtaeDo, "Moving Obstacle Avoidance of a Mobile Robot Using a Single Camera," *Procedia Engineering* , vol. 41, pp. 1-6, 2012.

[25] 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.

[26] E.M., "Professionaly and communicate with essential Edraw Solution," *Edraw soft*, vol. 8, 2016.

[27] "Wikipedia," Encyclopedia , 30 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Robot_Operating_System. [Accessed 23 september 2018].

[28] "ROS," ROS, [Online]. Available: ros.org. [Accessed 12 Jun 2018].

[29] R. I. Rasel, "Obstacle Detection for Indoor Navigation of Mobile Robots," pp. 1-85, 2017.

[30] Aaron Staranowicz, Gian Luca Mariottini, "A survey and comparsion of commercial and open source robotics simulatores," *IEEE,* 2016.

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

[32] Ertugrul Bayraktar and Pinar Boyraz, "Analysis of Feature Detector and Descriptor Combinations with a Localization," *IEEE,* pp. 1-12.

[33] Ebrahim Karami, Siva Prasad, Mohamed Shehata, "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images," vol. l, pp. 1-20, 2017.

[34] M. Guerrero, "A Comparative Study of Three Image Matcing algorithm:SIFT, SURF and FAST," pp. 1-105, 2011.

[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] Alexander Mordvintsev & Abid K., "Opencv-Python Tutorials," 2013. [Online]. Available: https://opencv-python tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html. [Accessed 12 August 2018].

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

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

[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] R. Szeliski, Computer Vision:Algorithms and Application, 2010.

[41] S. Wang, "Object Detction and Recognition for Visually impaired people," vol. volum 1, pp.

1-42, 2012.

[43] S. Zabihi, "Detection and Recognition of Traffic Signs Inside," 2017.

Appendix I

Sample code for recognition

```
import rospy
import numpy as np
import math
import sys
import time
import os
import timeit
import tf
import cv2
from enum import Enum
from std_msgs.msg import UInt8
from sensor_msgs.msg import Image, CompressedImage
from cv_bridge import CvBridge, CvBridgeError
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
import actionlib
from actionlib_msgs.msg import *

class DetectSign():
```

```
    def __init__(self):
        self.fnPreproc()

        self.sub_image_type = "raw" # you can choose image type "compressed", "raw"
        self.pub_image_type = "compressed" # you can choose image type "compressed", "raw"

        if self.sub_image_type == "compressed":
            # subscribes compressed image
            self.sub_image_original = rospy.Subscriber('/detect/image_input/compressed',
CompressedImage, self.cbFindTrafficSign, queue_size = 1)
        elif self.sub_image_type == "raw":
            # subscribes raw image
            self.sub_image_original = rospy.Subscriber('/detect/image_input', Image,
self.cbFindTrafficSign, queue_size = 1)

        #self.pub_traffic_sign = rospy.Publisher('/detect/traffic_sign', UInt8, queue_size=1)
        self.pub= rospy.Publisher('cmd_vel', Twist, queue_size = 10)

        if self.pub_image_type == "compressed":
            # publishes traffic sign image in compressed type
            self.pub_image_traffic_sign = rospy.Publisher('/detect/image_output/compressed',
CompressedImage, queue_size = 1)
        elif self.pub_image_type == "raw":
            # publishes traffic sign image in raw type
            self.pub_image_traffic_sign = rospy.Publisher('/detect/image_output', Image, queue_size
= 1)
        self.cvBridge = CvBridge()
```

```
    self.TrafficSign = Enum('TrafficSign', 'divide stop parking tunnel')

    self.counter = 1

def fnPreproc(self):
    # Initiate SIFT detector
    self.sift = cv2.xfeatures2d.SIFT_create()
        self.surf = cv2.xfeatures2d.SURF_create()
        self.orb = cv2.ORB_create(nfeatures=200)
        self.fast = cv2.FastFeatureDetector_create()
        self.brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()
    dir_path = os.path.dirname(os.path.realpath(__file__))
    dir_path = dir_path.replace('turtlebot3_autorace_detect/nodes', 'turtlebot3_autorace_detect/')
    dir_path += 'file/detect_sign
        self.img2 = cv2.imread(dir_path + 'tunnelo.png', 0)         # trainImage1
    self.img3 = cv2.imread(dir_path + 'park.png', 0)      # trainImage2
    self.img4 = cv2.imread(dir_path + 'stopm.png', 0)       # trainImage3

        #self.blur2 = cv2.GaussianBlur(self.img2,(15,15),0)
        #self.blur3 = cv2.GaussianBlur(self.img3,(15,15),0)
        #self.blur4 = cv2.GaussianBlur(self.img4,(15,15),0)


    start2 = timeit.default_timer()
    self.kp2 = self.orb.detect(self.img2,None)
```

```
        self.kp3 = self.orb.detect(self.img3,None)
        self.kp4 = self.orb.detect(self.img4,None)

        self.kp2, self.des2 = self.surf.compute(self.img2,self.kp2)
    self.kp3, self.des3 = self.surf.compute(self.img3,self.kp3)
    self.kp4, self.des4 = self.surf.compute(self.img4,self.kp4)
        stop = timeit.default_timer()
    excution_time = stop - start2
        print ("2finishedpic", excution_time)
    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
    search_params = dict(checks = 500)

    self.flann = cv2.FlannBasedMatcher(index_params, search_params)

  def fnCalcMSE(self, arr1, arr2):
        squared_diff = (arr1 - arr2) ** 2
        sum = np.sum(squared_diff)
        num_all = arr1.shape[0] * arr1.shape[1] #cv_image_input and 2 should have same shape
        err = sum / num_all
        return err

  def cbFindTrafficSign(self, image_msg):
    # drop the frame to 1/5 (6fps) because of the processing speed. This is up to your
computer's operating power.
```

```
if self.counter % 3 != 0:
    self.counter += 1
    return
else:
    self.counter = 1
if self.sub_image_type == "compressed":
    #converting compressed image to opencv image
    np_arr = np.fromstring(image_msg.data, np.uint8)
    cv_image_input = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
elif self.sub_image_type == "raw":
    cv_image_input = self.cvBridge.imgmsg_to_cv2(image_msg, "bgr8")

MIN_MATCH_COUNT = 9
MIN_MSE_DECISION = 50000

# find the keypoints and descriptors with SIFT
    start1 = timeit.default_timer()
    kp1 = self.orb.detect(cv_image_input,None)
kp1, des1 = self.surf.compute(cv_image_input,kp1)
stop = timeit.default_timer()
    excution_time = stop - start1
    print ("2finishedcam", excution_time)
matches2 = self.flann.knnMatch(des1,self.des2,k=2)
matches3 = self.flann.knnMatch(des1,self.des3,k=2)
matches4 = self.flann.knnMatch(des1,self.des4,k=2)
```

```
image_out_num = 1


    start1 = timeit.default_timer()
good2 = []
for m,n in matches2:


     if m.distance < 0.65*n.distance:
        good2.append(m)
if len(good2)>MIN_MATCH_COUNT:
    src_pts = np.float32([ kp1[m.queryIdx].pt for m in good2 ]).reshape(-1,1,2)
    dst_pts = np.float32([ self.kp2[m.trainIdx].pt for m in good2 ]).reshape(-1,1,2)
    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
    matchesMask2 = mask.ravel().tolist()
    mse = self.fnCalcMSE(src_pts, dst_pts)
    if mse < MIN_MSE_DECISION:
        msg_sign = UInt8()
        msg_sign.data = self.TrafficSign.stop.value
        #self.pub_traffic_sign.publish(msg_sign)
        rospy.loginfo("TrafficSign 2")
       image_out_num = 2


else:
    matchesMask2 = None


good3 = []
```

```
for m,n in matches3:
    if m.distance < 0.65*n.distance:
        good3.append(m)

if len(good3)>MIN_MATCH_COUNT:
    src_pts = np.float32([ kp1[m.queryIdx].pt for m in good3 ]).reshape(-1,1,2)
    dst_pts = np.float32([ self.kp3[m.trainIdx].pt for m in good3 ]).reshape(-1,1,2)

    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
    matchesMask3 = mask.ravel().tolist()

    mse = self.fnCalcMSE(src_pts, dst_pts)
    if mse < MIN_MSE_DECISION:
        msg_sign = UInt8()
        msg_sign.data = self.TrafficSign.parking.value
            rospy.loginfo("TrafficSign 3")
        #self.pub_traffic_sign.publish(msg_sign)

        #tell the action client that we want to spin a thread by default
            #self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
            #rospy.loginfo("wait for the action server to come up")
            #allow up to 5 seconds for the action server to come up
            #self.move_base.wait_for_server(rospy.Duration(5))

            #we'll send a goal to the robot to move 3 meters forward
```

```
            #goal = MoveBaseGoal()
            #goal.target_pose.header.frame_id = '/map'
            #goal.target_pose.header.stamp = rospy.Time.now()
            #goal.target_pose.pose.position.x = 1.5 #1.3 meters
            #goal.target_pose.pose.position.y = 0.5 #1.3 meters
            #goal.target_pose.pose.orientation.w = 1.0 #go forward

            #start moving
            #self.move_base.send_goal(goal)

            #allow TurtleBot up to 60 seconds to complete task
            #success = self.move_base.wait_for_result(rospy.Duration(0))
            #rospy.loginfo("1.3 meters in y direction")


        image_out_num = 3

else:
    matchesMask3 = None

good4 = []
for m,n in matches4:
    if m.distance < 0.65*n.distance:
        good4.append(m)
if len(good4)>MIN_MATCH_COUNT:
```

```
        src_pts = np.float32([ kp1[m.queryIdx].pt for m in good4 ]).reshape(-1,1,2)
        dst_pts = np.float32([ self.kp4[m.trainIdx].pt for m in good4 ]).reshape(-1,1,2)

        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
        matchesMask4 = mask.ravel().tolist()

        mse = self.fnCalcMSE(src_pts, dst_pts)

        elif self.pub_image_type == "raw":
            # publishes traffic sign image in raw type
            self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_imgmsg(cv_image_input,
"bgr8"))

    elif image_out_num == 2:
        draw_params2 = dict(matchColor = (0,0,255), # draw matches in green color
                    singlePointColor = None,
                    matchesMask = matchesMask2, # draw only inliers
                    flags = 2)

        final2 =
cv2.drawMatches(cv_image_input,kp1,self.img2,self.kp2,good2,None,**draw_params2)

        if self.pub_image_type == "compressed":
            # publishes traffic sign image in compressed type
```

```
            self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_compressed_imgmsg(final2,
"jpg"))

        elif self.pub_image_type == "raw":
            # publishes traffic sign image in raw type
            self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_imgmsg(final2, "bgr8"))

    elif image_out_num == 3:
        draw_params3 = dict(matchColor = (255,0,0), # draw matches in green color
                    singlePointColor = None,
                    matchesMask = matchesMask3, # draw only inliers
                    flags = 2)

        final3 =
cv2.drawMatches(cv_image_input,kp1,self.img3,self.kp3,good3,None,**draw_params3)



            if self.pub_image_type == "compressed":
            # publishes traffic sign image in compressed type
            self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_compressed_imgmsg(final3,
"jpg"))

        elif self.pub_image_type == "raw":
            # publishes traffic sign image in raw type
            self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_imgmsg(final3, "bgr8"))
```

```
        elif image_out_num == 4:
            draw_params4 = dict(matchColor = (255,0,0), # draw matches in green color
                                singlePointColor = None,
                                matchesMask = matchesMask4, # draw only inliers
                                flags = 2)

            final4 =
cv2.drawMatches(cv_image_input,kp1,self.img4,self.kp4,good4,None,**draw_params4)



            if self.pub_image_type == "compressed":
                # publishes traffic sign image in compressed type
                self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_compressed_imgmsg(final4,
"jpg"))

            elif self.pub_image_type == "raw":
                # publishes traffic sign image in raw type
                self.pub_image_traffic_sign.publish(self.cvBridge.cv2_to_imgmsg(final4, "bgr8"))

    def main(self):
        rospy.spin()

if __name__ == '__main__':
    rospy.init_node('detect_sign')
    node = DetectSign()
```

```
node.main()
```