Master's Programme in ICT Innovation

Target Detection, Indoor Scene Classification, Visual and Three-Dimensional Mapping for Service Robots in Healthcare.

Aalto University School of Electrical Engineering

Ambikeya Pradhan

(Major : Autonomous Systems)

Master's Thesis
2021

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Ambikeya Pradhan

Title of thesis Target Detection, Indoor Scene Classification, Visual and Three-Dimensional Mapping for Service Robots in Healthcare

Programme Master's Programme in ICT Innovation

Major Autonomous Systems

Thesis supervisor Prof. Daniele Fontanelli

Thesis advisor(s) Prof. Quan Zhou

Collaborative partner Thomas Hoglund, CEO, Done Enterprises Ab Oy

Date 22.05.2021 **Number of pages** 32 + 15 **Language** English

Abstract

This thesis details work on different techniques used to implement service robots for indoor environments. This included two machine learning techniques: Target Detection and Indoor scene classification as well as two computer vision techniques: Visual mapping and three dimensional mapping. Using these techniques, we tried to make service robots better in environments like hospitals. Assistance provided by service robots will help staff in managing tedious tasks without any problem. We used different techniques for mapping and localization so service robots can autonomously navigate from floor to floor. Depth cameras were used to make recognition and mapping better for indoor environments.

Keywords Localization, mapping, SLAM, ORB-SLAM visual odometry, object recognition, scene recognition, Visual mapping, 3D mapping

Contents

Preface

A very special thanks to Prof. Daniele Fontanelli and Thomas Höglund for their collaboration and helping me with this thesis, without their expertise this project would not have gone this smoothly! And a big gratitude to all the people who have always been there when we were in desperate need of humble support. Also thanks to the people working in similar research work, offering their expertise in all areas and providing us with service robots . And a special gratitude to everyone in the Aalto University staff that helped me out during my thesis!

Vaasa, 22 May 2021
Ambikeya Pradhan

1. Introduction

In recent years, technology has transformed everything around us. From the 1980s, first robots have emerged in the field of medicine and healthcare. They have offered surgical assistance via robotic arm technologies. Since then, Artificial Intelligence, Robotics, Data analysis and machine learning has changed the field of healthcare. Artificial intelligence (AI) - enabled machine vision based technologies have expanded the capabilities of human beings in many areas of medicine.[2]

Robots are used in operating rooms, as well as they are used in clinical settings by healthcare workers. These technological advancements have provided Doctors to look beyond the impossible, which they were not capable of doing with limited machinery and under-developed technology. During COVID-19, hospitals and clinics have realised that by deploying a much wider range of robots to accomplish tasks, they can limit the spreading of pathogens amongst their own staff. Since then, all the quarantine zones in hospitals or clinics are supervised by limiting the healthcare personnels in there. Patients are monitored through machines and in some places through automated robots. Robots are also being used to sanitize, clean and prepare patient rooms or quarantine zones in hospitals and clinics. They are also helping to limit the person-to-person contacts in admission wards. AI-enabled medicine identifier software is also used by some robots to identify, match, sort and distribute medicines to the patients admitted in the hospitals. [4]

For monitoring patients' vital statistics and alerting the nurses when human presence is needed in the room. This allows the nurses to monitor several patients in critical health at the same time. These monitoring robots can also enter the information into a patient's electronic health record which is stored in the centralised database. These data analytics features support in the hospital management. In some countries, robotic carts or carriers can be seen in the hospitals while moving supplies from one place to another. Robots are assisting surgeries that allow doctors to conduct surgeries with very high precision.

Initially, robots were focused in laboratories to take samples and analyze, transport and store them for future purpose. Then, perform respective tests on those samples and electronic data is stored in their databases for doctors to review. Robots also test, prepare and dispense medications in pharmacological labs.

In larger facilities, robot carriers or carts are used to carry bed linens, meals and other supplies from floor to floor, while going through elevators and automatic doors. Robotic assistants with "gears and wires" help paraplegics to move and they can administer physical therapies. These robotic personal assistants are built to look friendly and respond to human speech. Nowadays, these robot assistants are used for child care also. Some robotic assistants are even built to look like humanoids and used to help with personal care, socialization and for training. [1]

The robots deployed in healthcare settings are likely to rise because of their increasing technological capabilities, their reduced costs and increasing pressure to curb costs. However, robots are potentially highly disruptive innovations, and it is therefore important to the sociotechnical challenges likely to be encountered as robots are deployed to find the mitigating strategies. Sociotechnical approaches to study the implementation of technology view social and technical factors as shaping each other over time. It is assumed that technologies are shaped by their social environments (example, through designs being modified) but also that social environments are shaped by technological features.

Developments currently taking place have begun to replace individual aspects of human performance with robotic capabilities including precision (eg, surgical robots), logistic and mechanical tasks (eg, service robots) and complex cognitive tasks (eg, rehabilitation robots) Some of their different features respective to the tasks performed are shown in the table below-

Table 1 - Use of Healthcare Robotics

Type of Device	Autonomous	Semi-Autonomous	Operational	Healthcare delivery, patient- and staff-facing
Service Robots (eg, Stock control, cleaning, delivery, sterilization)	✔	✔	✔	✔
Surgical Robots		✔		✔
Telepresence Robots		✔		✔
Companion Robots	✔			✔
Cognitive Therapy Robots	✔			✔
Robotic limbs and exoskeletons		✔		✔

1.1. Robotic Systems in Healthcare

Several robotic systems are already deployed in healthcare settings to enable a high level of patient care, efficient operability in clinical settings and a safe environment for both patients and healthcare workers. Medical robots support minimally invasive procedures, customized and frequent monitoring of patients with chronic diseases or in critical condition, intelligent therapeutics and social engagement with elderly patients. In addition, robots alleviate workloads, nurses and other caregivers can offer patients empathy and human interaction, which can promote long-term well-being. [3]

a) Service Robots

Service robots streamline monotonous routine tasks, reduce physical demand of human workers, and support throughout the day with multiple shifts. Maintenance cost is low once they are installed and trained to operate within the premises. Service robots can also take work for admission wards for queries, booking appointments and feedback. Service robots are mostly deployed as carriers or robotic carts for transporting meals, other supplies, equipment and medicines from floor to floor, through elevators and automatic doors.They can also transport bed linens and other clothes to and from laundry facilities. They can also operate as telepresence robots in quarantine areas in hospitals or clinics, either for collecting data from patients or daily communication between nurses and patients for regular checkup. Many of these robots function autonomously and can send a report when they complete a task. These robots are also used to set up patient rooms, track supplies and file purchase orders, and restock medical supply cabinets. By efficiently handling routine tasks by service robots gives healthcare workers more time to focus on immediate patient needs. [4]

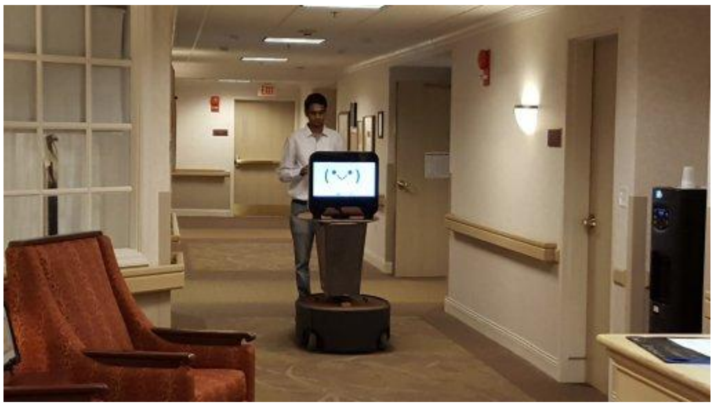


Figure 1: Use of service robots in elderly care homes

Service robots can also be converted into cleaning and disinfection robots which can operate in the same corridors and common areas through which their map is defined. These cleaning and disinfection robots allow hospitals to be sanitized and readied for incoming patients quickly. Cleaning and disinfection robots can limit pathogen exposure while helping reduce hospital acquired infections (HAIs).

b) Surgical-assistance Robots

These robots help surgeons perform complex micro-procedures without making large incisions. Over time, as motion control technology has advanced, these surgical assistance robots have become more precise. The surgical assistance robots are still evolving, eventually AI-enabled robots have started using computer vision to navigate through specific areas in the body while avoiding nerves and other obstacles. Some surgical robots are even able to complete the surgery autonomously, allowing doctors and surgeons to oversee procedures through the console.

The surgical assistance robotic field is evolving to make better use of Artificial intelligence. Surgical robots enabled with computer vision help them to differentiate between different types of issues in their field of view. For example, now they have the ability to help surgeons avoid nerves and muscles during procedures. High definition 3D computer vision can provide surgeons with detailed information and enhanced performance during procedures. Eventually, robots will be able to take over small subprocedures, such as suturing or other defined tasks under the observation of the surgeon.

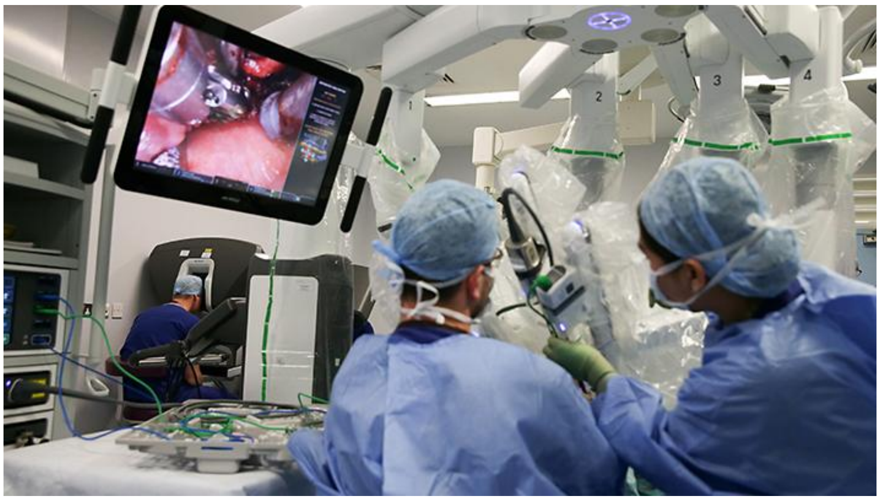


Figure 2: Surgical Robots assisting during surgery

c) Modular robots

Modular robots can be used to enhance other systems and configure to perform multiple functions. In healthcare, modular robots are included in therapeutic exoskeleton robots and prosthetic robotic arms and legs.

Most of the therapeutic robots can help patients with rehabilitation after strokes, multiple sclerosis, traumatic brain surgeries or paralysis. These robots are equipped with artificial intelligence and depth cameras. On their human-interaction module, there can be some kind of input/output device for example touch screen, monitor, different sensors like thermal imaging camera and gesture sensors. Modules enabled with a processing system can use computer vision to detect, classify and store data for learning or improving data segmentation. Modular robots can also be self-powered and charged once a day for all day use. Batteries powering the human-interaction module and processing unit module can also power the mobility module on the bottom, so that the robots can move around on their own.

The therapeutic modular robots can monitor patients' forms as they go through prescribed exercises. They can measure degrees of motion in different positions and track progress which is more precise than the human eye. They can also share improvements after analyzing data with the patient and keep a record which doctors and nurses can go through after the therapeutic period. Their interaction with the patients help them to provide coaching as well as encouragement.

d) Social Robots

Social robots are designed to directly interact with humans. These "friendly looking" are mostly designed to be used in long-term care environments. In these environments, their practical function is for socially interacting and monitoring. Their practical functions can change depending upon the care environment in which they are used, for example elderly care homes, children's wards, waiting areas etc. They can encourage patients to comply with their treatment regimens, provide analyzed data of their progress and promote daily activities to work with during post-treatment therapy. They can provide patients with cognitive engagement and keep them to stay alert and positive. Service robots can be converted into social interaction robots that can be used to offer visitors and patients direction inside the hospital environment.

During COVID-19, social robots were used as telepresence robots and were promoted by many doctors and surgeons. Telepresence robots helped doctors and their patients to interact through video conferences. This avoided person-to-person contact and saved time for both doctors and patients. Regular check ups and appointments were chosen through this telemedicine method. Healthcare workers could also contact patients in quarantine areas through telepresence robots to avoid multiple visits for routine check ups.

In general, social robots help healthcare workers by reducing their workload and improving their patients' well-being.

e) Mobile Robots

Mobile robots can be used inside any premise of the hospital or clinic. They are operated using a cable wire following or predefined tracks. They are being used for a wide range of purposes - either moving heavy machinery, helping to transport patients, or sanitizing/disinfecting rooms. Mobile robots used as cleaning or disinfection robots can use air filtration, hydrogen peroxide vapors, or ultraviolet (UV) light to sanitize reachable places in a uniform way and help reduce infection.

f) Autonomous Robots

Autonomous robots have built-in light detection and ranging (LiDAR) systems, visual computing with thermal imaging cameras or depth cameras, and mapping capabilities which the robots can use to self-navigate to patients in hospital or exam rooms. This allows doctors and other healthcare workers to interact with patients from afar. These robots are controlled by remote specialists or other healthcare workers (who are trained on the system), so they can also accompany doctors as they make hospital rounds and do their daily routine checkups with the patients. The remote specialist can also contribute to the checkups with on-screen consultation regarding patient diagnostics and care. These robots can keep track of their own batteries and make their way back to the charging station when necessary.

Some autonomous robots can also perform cleaning and disinfecting, while navigating through the operating rooms, laboratories, infectious disease wards and public hospital spaces. For example, a startup Akara has developed an autonomous robot prototype which is being tested for disinfecting contaminated surfaces in hospitals and clinics using UV light. Its goal is to help hospitals sanitize rooms and equipment, aiding in the fight against COVID-19.

1.2. Use of Service Robots

Shortages of doctors is a global phenomena. The World Health Organisation (WHO) has estimated that there is a worldwide shortage of physicians, nurses, and other healthcare workers which range up to 4.3 million. We will never be able to train as many doctors and other healthcare workers as we need. Service Robots with telemedical devices would be able to merge this gap. They will certainly appear more and more in hospitals and clinics, while it is going to be a common element to see around.

Service robots aim to offer also as a robot companion. They offer solutions in order to enable elderly and other people and other people without the necessary social support to connect with the world. There are various types of robot companions - human or animal shaped, smaller or bigger, but they all share one thing: their goal is to make life more enjoyable and easier.

They also offer patients in remote areas or people who are not able to travel have access to high-quality emergency consultations for stroke, cardiovascular, and burn services exactly when they need it. Moreover with telehealth, medical professionals in rural towns and remote areas also have access to specialty services, while patients can be treated in their own communities.

In healthcare, doctors and nurses are using service robots as telepresence robots to extend their reach to monitor and consult with patients in the hospital, skilled nursing facility and in the home. Family members are also using telepresence robots to visit loved ones when they can't be there in person.

Service robots used as carriers, robotic carts or autonomous mobile delivery robots which are able to carry around a multitude of racks, carts or bins in the form of medications, laboratory specimens or other sensitive materials. These carriers can be sent or requested using a touch screen interface and upon completing its "mission", it returns to the charging dock while it is loaded for the next job.

Service robots are also used as a robotic medical dispenser system, with a built-in dispensing capacity range, which helps any given facility (suitable to the system) for its volume. It is also designed with robust data mining capabilities, so the pharmacy can gain valuable insights about its efficiency all the time.

1.3. Different features of Service Robots

Service robots are being used for daily purposes ranging from security guard service robots to tour guides. They are used in malls as an order payment method. These service robots have most of the technical features in common. All the features can be upgraded remotely and hardware can be replaced during the time of their maintenance period. Some of these are-

- Autonomous driving (Virtual fence)- Robots can recognise the driving environment, detect obstacles and avoid driving.
- RMS remote monitoring and control- Provision of robot control platform (RMS) for multi control and service-scenario scheduling.

- Guidance service for guiding and exhibits- It is possible to provide directions and exhibit information services to escort users to specific destinations within the facility.
- Emergency alert notification- Emergency alert notification in the event of the emergency such as leakage or fire hazard.
- Data analysis using robot- Statistical analysis of usage data of robot users based on the cloud service platform (Paas).
- Customer-specific guided content- Continuous content update is possible through the application of RUX (operational location and customer-specific content) development. Various information such as location and usage of various facilities can be provided.
- Security Surveillance IP camera (CCTV)- Real time control to the central control room of surveillance cameras.
- Video call service- Emergency call and video call service in the case of emergency.
- HRI-based face avatar- They provide HRI-based face avatars with various emotion expressions, and if necessary, character avatars can be developed and applied according to customer needs.
- Natural Face Movement- Increased intimacy and attention to the robot by expressing the delicate movement of the neck joint of the robot head.
- Customized Exterior design- Customizable exterior design.
- Robot Management System (RMS)- It is possible to provide an RMS system that can remotely monitor and control the robot operation status from anywhere.
- Screen display- Equipped with a touch display that enhances the visibility of guided content.
- Removable replacement battery- The battery can be detached at any time to replace the spare battery, allowing continuous operation time to be extended.
- Thermal imaging camera- Support for installing thermal imaging cameras to prevent the spread of infectious diseases in multiple uses facilities.
- Unmanned payment (experience based service)- Provides unmanned payment linkage service that increases the value of customers' purchasing experience.

Individual robot platforms can be configured for a wide range of applications: a mobile information center in museums, DIY stores and airports, for collection and delivery services in homes and offices, for security applications or as museum robots at attractions.

1.4. Organization of this work

This paper is organised as follows. Chapter 2, 3, 4 and 5 define the Literature review, Research materials created and Methods performed to solve various problems (Target Detection, Indoor Scene Classification, Visual Mapping and Localization and three-dimensional mapping and localization) for our service robot. In Chapter 2, we reviewed all the related literature for our case problems. Target Detection includes all the techniques that have been used for object recognition. We performed indoor scene classification using the method of Faster R-CNN. We tested visual mapping and localization or Visual-SLAM based on the concept of ORB-SLAM and used the generated maps for autonomous navigation. We have described three-dimensional mapping or 3D-SLAM using 3D LiDAR and using the SLAM algorithm.In Chapter 3, we conducted the experiments for all our case problems. In Chapter 4, we have described all the obtained results and thoroughly discussed each one of them. In Chapter 5, we concluded our whole work.

2. Literature review

2.1. Target Detection

Object Detection is a computer vision technique that is used to identify and locate an object or a group of objects within an image or image frames of a video. Object detection draws a bounding box around the detected objects and signifies them separately as the class of the given object. This allows us to locate where said objects are in (or how are they moving through)in the given scene.

Object Detection can be split into machine-learning based approaches and deep-learning based approaches. In the ML-based approach, the computer vision uses different features of an image, for example color histogram, edges, group of pixels that belong to the object. These features are fed to the regression model which predicts the location of the object and its label in the image. Deep learning based approaches use convolutional neural networks (CNNs) for performing end-to-end, unsupervised object detection. This object detection doesn't require features to be defined and extracted separately.

Object Detection is indistinguishably connected to other computer vision techniques like image recognition and image segmentation. It helps in analyzing and understanding scenes in images and videos. There are some important differences amongst them. Image recognition is used to only give output a class label for an identified object and image classification is used for creating pixel-level understanding of the scene's elements. Object Detection is different from both of these tasks and it has the unique ability *to locate objects in images and videos*. This allows us to track and detect those objects in the consecutive images or image frames of a video.

Image Classification predicts the class of one object in the image. *Object localization* identifies the location of one or more objects in an image and draws a bounding box around them. *Object detection* combines these two tasks, localizes and classifies one or more objects in an image frame. We can differentiate among these three computer vision tasks as follows-

- *Image Classification*: It predicts the class or type of an object in an image.
    - *Input*: Image with a single object
    - *Output*: A class label (example, mapped integers to class labels)
- *Object Localization*: It locates the presence of an object in an image and indicates them with a bounding box.
    - *Input*: Image with one or more objects
    - *Output*: One or more bounding boxes (example, defined by point, width or height)
- *Object Detection*: It locates the presence of objects with bounding boxes and classes or types of the located objects in the image.
    - *Input*: Image with one or more objects
    - *Output*: One or more bounding boxes (example, defined by point width or height) and a class or type label for the bounding boxes.

Another extension of these computer vision tasks is *object segmentation*, also known as semantic segmentation or object instance segmentation, where recognized objects' instances are highlighted objects instead of creating bounding boxes around them. From figure 3, object detection can be displayed as a part of this suit of challenging computer vision tasks.

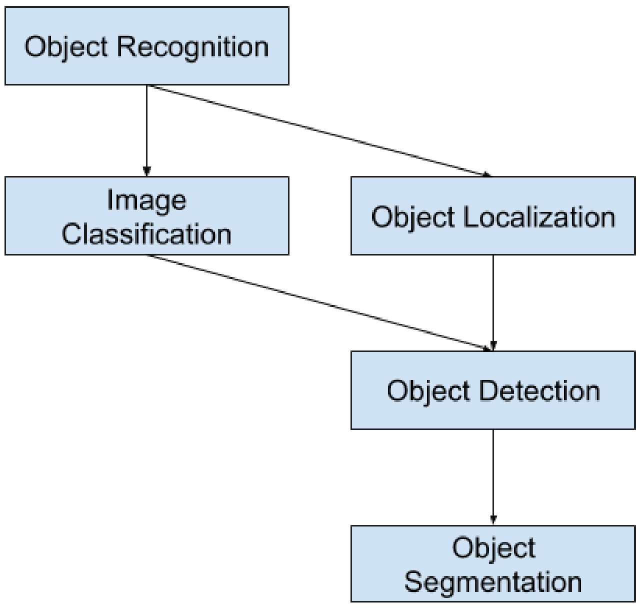


Figure 3: Overview of Object Detection Computer Vision Tasks

There is another method of *Single-Object Localization*, which is a simpler version of the more properly defined “Object Localization”. It constraints the localization tasks to objects of one type or class within an image. Single-Object Localization is an algorithm that produces a list of categories of objects that are present in the image. These lists are available along with an axis-aligned bounding box which indicates the position and scale of one instance from each object category.

Below in figure 4, is an example from a paper which compares single object localization and object detection. The difference in ground truth expectations in each case can be seen.

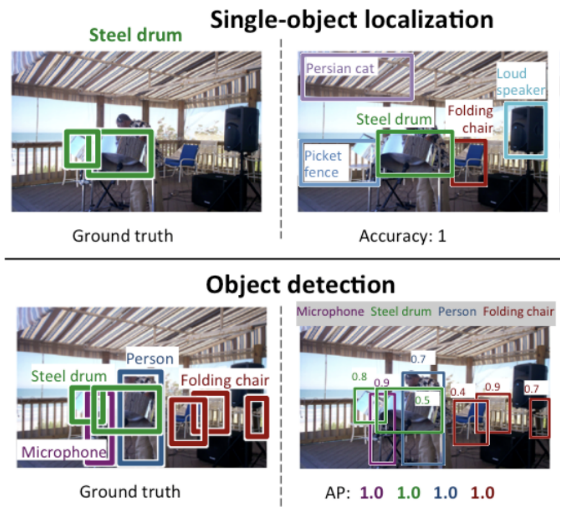


Figure 4: Comparison between Single Object Localization and Object Detection

Using mean classification error across the predicted class labels, the performance of a model for image classification is evaluated. The distance between the expected and predicted bounding box for the expected class is used for evaluating the performance of a model for single object localization. Whereas, the precision and recall across each of the best matching bounding boxes for the known objects in the image are used for evaluating the performance of a model for object detection. Now that we are familiar with object detection and localization, let's start with how object detection works.

2.2. Indoor Scene Classification

Indoor scene classification or Automatic scene classification (sometimes also known as scene recognition or scene analysis) is a well established research problem tool of computer vision. It comprises assigning labels such as 'bedroom', 'kitchen' or simply 'indoor or outdoor' to the images given as input, on the basis of images' overall contents. [15]

In this work, we will focus only on indoor images, since our application is based on the service robot which is used only for indoor scene environments. Hence, all the experiments and their results will be performed on a public MIT-67 dataset. The dataset consists of 67 indoor categories which is a total of 15620 indoor images. The number of images are varying with different categories and every category has at least 100 images. Some of the examples of images from different categories are shown in Figure 5 below.

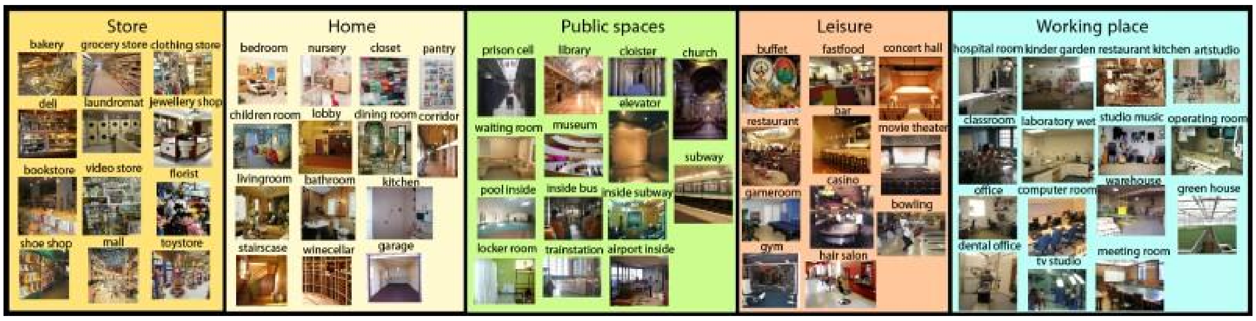


Figure 5: Examples of indoor scene images from different categories of public MIT-67 dataset

Indoor scene images are also more difficult to predict class than for outdoor scenes. The indoor scenes have various characteristics including all content complex attributes makes it tough for any deep learning-based model to keep everything in account. The processing for training on indoor scene datasets is heavy and time-consuming in comparison to outdoor scene datasets. The outdoor scene recognition methods use different supervised classification techniques for scene analysis in different environments. Several feature detectors and local descriptors used for supervised classification are like: Scale Invariant Feature Transform (SIFT), Histogram of Oriented Gradients (HOG), Bag of Visual Words (BoVW), Spatial Pyramid Matching (SPM), Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Binary Robust Independent Elementary Features (BRIEF), Oriented FAST and Rotated BRIEF (ORB), Maximally Stable Extremal Regions (MSER) and Binary Robust Invariant Scalable Keypoints (BRISK).

These above mentioned methods are easily able to achieve good performance in the tasks that require outdoor scene recognition. But their accuracy drops sharply when they are applied to indoor scene images. Outdoor scene images always have similar scene attributes and their differences in image scenes are easy to detect. On the other hand, indoor scenes are more complex. Most of the time, they suffer with partial occlusion as well as different scales and change in illumination[21]. Hence, indoor scene images show characteristics of both smaller-sized inter-class variety and larger-sized inter-class variety.

Recently, hierarchical representations based scene recognition techniques have shown to be more efficient to extract low-level features from indoor scene images[18-20]. The important step for this kind of method is to design constructive mid-level features. In the paper “Indoor scene recognition” by Quattoni in his paper [17], it was suggested to propose GIST + ROI (Region of Interest) for explaining indoor scene detection. Although, heavy computational costs were needed by the method to get ROI and it was inefficient for some indoor scene image classes like malls and offices.

In recent years, deep learning models like R-CNN have become one of the most successful methods for multi-class object recognition. R-CNN first extracts region proposals in the indoor images and then all the detected proposals generated are loaded into a deep convolutional network (CNN). R-CNN[8] thoroughly combined a heavily computed detection technique and deep learning theory for effectively detecting thousands of objects in the input images at the same time. R-CNN and its follow-up methods, Fast R-CNN[10] and Faster R-CNN [11] have accomplished excellent results in multi-class object recognition of the high quality images.

According to the above illustration, in this paper, we will present a method of indoor scene recognition based on deep learning and sparse representation influenced from the paper with the same name[16]. In this method, the object-based information like class, size and position as low-level features will be extracted by multi-class objects detector, Faster R-CNN. The modified Bag-of-Words (BoW) model is designed to collect semantic and spatial information from low-level features of objects and build mid-level features. By using Faster R-CNN, we will be quickly able to collect abundant features of objects in the images. For intensifying the robustness, the learned sparse representation from mid-level features will be used to detect the class of indoor scene images. The modified BoW efficiently solves the semantic gap of elements between high-level features and low-level features. It also maintains the spatial information provided by the objects. In comparison to all the older and traditional classifiers like Bayesian, SVM and k-NN, sparse representation is robust to solve the issues like change in illumination, partial occlusion or variations in pose.

2.2.1 Conventional Scene Recognition Framework

In conventional scene recognition framework, the process is split into three modules: pre-processing of input image, feature extraction and building a deep convolutional neural network based classifier as shown in Figure 16. The scene recognition process also involves a local feature descriptor, global feature descriptor, saliency detection and Spatial Pyramid Matching (SPM)[22].

First, the image is divided at three different sublevels of resolution, so that a three-level pyramid can be constructed. The Scale Invariant Feature Transform (SIFT) operator is used for extracting features of bins under three different sublevels. All the feature vectors of different sublevels are combined to create a mid-level feature. For giving the classification results, a SVM classifier is also trained. On the basis of Bag-of-Words (BoW) model, the information of multi-scale and position is added and by adding this information, the performance of scene recognition is improved. Hence, robust feature representation building of complex indoor scene images has improved the performance of indoor scene recognition as well as it has solved the problem of semantic gap between high-level semantic features and low-level features.

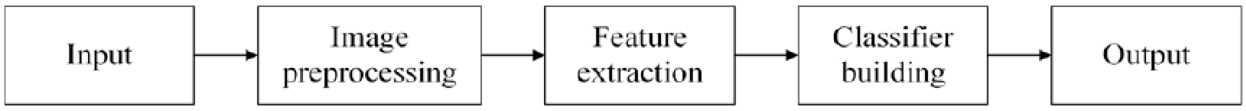


Figure 6: Conventional Scene Recognition Framework

The problems faced by previous traditional algorithms for extracting low-level features were-

- Feature extraction from low-level features is basically focused on retrieving feature information like color, texture and edge which are difficult to describe globally in the scene content. As well as these features were not enough to solve the problem of illumination change, pose variation and partial occlusion. The low-level features are very specific in scene content information hence their proper extraction is not effectively worked in earlier algorithms.
- By using a variety of low-level features for representing indoor scene images, higher feature dimensions are created and it generates heavy computational cost. This leads to slower scene recognition models and their processing and training might be high in accuracy but not a very effective indoor scene recognition method.

2.3 Visual Mapping and Localization (Visual SLAM)

Simultaneous Localization and Mapping (SLAM) is a very important field of robotics and autonomous navigation. SLAM techniques are used to build a map for an unknown environment as well as the sensor is localized in the map for real-time operation. Among all the sensors used, cameras are cheap and they provide a rich environment for accurate place recognition. For mapping closed loops (where the main sensor detects the whole mapped area and returns the starting point by correcting all environmental errors)[24], place recognition is used as a key module for SLAM. It can avoid all the tracking failures caused due to partial occlusion, system re-initialization or moving aggressively, so that the camera can re-localize. Some other issues could also be using 2D/3D sensors like LiDAR for SLAM is LiDAR's high cost. This makes it unusable for low cost systems.

Visual SLAM uses a camera as a main sensor which is considered as a major interest nowadays. Using a simple, smallest, cheapest monocular camera, we can perform Visual SLAM. But the disadvantage of using a monocular camera is that depth is undetectable for one camera. This affects estimated trajectory and scale in the map. So, a camera without multi-view or any filtering techniques cannot create an initial map which is impossible to triangulate to the first frame. Due to scale drift, a monocular camera based SLAM fails and doesn't recover to perform further mapping and localization. To avoid all these errors, stereo or RGB-D cameras replaced monocular cameras for performing Visual SLAM.

ORB-SLAM [26, 27] is a Visual SLAM method which creates a map based on the point clouds of the indoor environment using stereo, RGB-D or depth cameras. Using the point cloud, we can understand the 3D structure of the environment. The point clouds are useless for path planning and navigation if we use the algorithms which require 2D occupancy grid map as an input [28]. ORB-SLAM generated point clouds are scattered, hence it is difficult to generate an occupancy grid map from them. In this chapter, we will use ORB-SLAM and its newer version ORB-SLAM2 and build an occupancy grid map in real time from all 3D point clouds received. The grid map will be further used for navigation (navigation stack of ROS[25]) based on the real-time camera trajectory produced by ORB-SLAM.

2.4 Three-dimensional mapping and localization (3D-SLAM)

Generating three-dimensional maps and their models for the indoor environment is an interesting and popular field nowadays. This task has increased in a number of different fields. Different possible tasks require three-dimensional mapping and localization and it ranges from tasks like assisting in navigation inside big buildings like hospitals or hotels to assisting in increasing efficiency. This process, as we have already learnt about it in the last chapter, is known as Simultaneous Localization and Mapping (SLAM)[38].

Initially for plotting maps, localization and indoor navigation, two dimensional mapping methods were used. Although these methods were implemented for mobile with their focus for autonomous navigation [39], they have limitations to perform only a certain number of tasks. Then mobile robots started employing in different applications including the medical field. Service robots were required to be used in hospitals and other medical care facilities. This required incorporating mobile robots with reduction in the sizes of their sensors and electronics [40]. Other multiple applications required autonomous navigation of mobile service robots to perform in big buildings like hospitals including deploying them as carriers or robotic carts for transporting meals, other supplies, equipment and medicines from floor to floor, through elevators and automatic doors. These cases required an initial map or layout of the surrounding area as well as for its navigation a whole mapping operation is required during the operation. Many robots have either RGB-D, stereo or depth cameras, LiDAR sensors or both, so that these sensors can obtain a view for the surrounding environment in the third dimension (which is essentially needed for mapping and navigation). They are incorporated with the Inertial Measurement Unit (IMU) to provide a sense of localization (to know the movement and exact location of the robot in the map). Using IMU, the robot can provide us information about its movement and process of planning using different algorithms [41]. As we know, the Global Navigation Satellite System (GNSS) is not operational for indoor environments. So, we used the Robotic Operative System (ROS) and SLAM algorithm for data processing. Using ROS and SLAM, we performed point cloud registration and did map extraction by using Cartographer Library from Google [42]. The map extraction is modified to reconstruct the 3D image of the scenes.

The indoor 3D mapping systems produce 3D point clouds and they are not generalized and extended in research work yet to produce the accuracy tests as it has been done for 2D mapping [43]. The three approaches for existing accuracy tests for 3D points are classified in the order of approaches [44]: the control approach, the subset approach and point cloud to point cloud (p2p) approach. For the control approach, the evaluation is based on the points clouds that are under study and their distance from the corresponding point. Through these distances of the point pairs, we can understand the specific number of measurements required for the whole point cloud. For the subset approach, the new and reference points (under study) are taken from each point cloud and differences between them are evaluated. These differences are some regular variables like distances and deviations between planes. For the p2p approach, the two points in whole are evaluated. This approach is better than the rest of the two approaches as it provides the whole information about the proper quality of the points present in the three dimensions. Now all three approaches are focused for both static and mobile environments of indoor mapping. But there is one issue that is not solved by any of these three approaches. The issue of deviation in trajectory causing deformation, which has no specific measurement mentioned for it. This issue is a main factor to assess the quality of all indoor mapping systems and evaluate applied SLAM algorithms.

Through the compulsory trajectory established for the robots, we can evaluate the performed pose computation as well as the robot's tracking [45]. For indoor mapping, this accuracy is not necessary. We can accept the trajectory with partial deviations only if they are not changing the final point cloud. In this paper, we will present the mobile robot for indoor mapping. It comprises 3D LiDAR and SLAM algorithms established in ROS. For the evaluation of this methodology, we will determine its accuracy using the p2p approach. We will be using Leishen C16 3D LiDAR for collecting three dimensional point clouds with an in-built IMU for localization. It will be placed on top of a Magni mobile robot with a certain height platform to detect everything in a range of 30 degrees (+15° to -15°). Using the p2p approach we will evaluate the trajectory by measuring its effect caused on the point cloud.

3. Research material and methods

3.1. Deep learning-based approach to Object detection

In this section, we will look at different deep learning approaches to object detection and assess their basic structure that uses neural networks.

3.1.1. Basic Structure

The deep learning-based model for object detection is typically divided into two parts. First, an image goes as an input through an *encoder*, then the encoder takes it through a series of blocks and layers. After completing the block for the encoder, the algorithm will learn to extract the statistical features used to locate and label objects. The block for *decoder* will collect the passed outputs from encode. The decoder predicts the bounding boxes and labels for each object. [5]

A pure regressor is used for the simplest decoder. The regressor is directly connected to the outputs of the encoder and predicts the size and length of each bounding box around the objects. The output given by this model is the coordinate pair of the object (X,Y) as well as its defined extent in the image. Although this model is simple, it is still limited. The number of boxes are needed to be specified ahead of time. If an image has two same objects, the model is designed to detect only a single object, the other one will go unlabeled. However, if the number of objects is known ahead of time, they should be predicted in each image. For this, pure regressor-based models should be considered as a good option. [8]

Another extension of the regressor-based approach is known as the regional proposal network. In this decoder, specific regions of the image are proposed by the model, where it is believed an object might reside. The pixels of these regions are fed to classification subnetwork to either determine the label or reject the proposal. Those pixels containing regions are then passed through a classification network. The benefit of using this method is to get a better, flexible model which can estimate the arbitrary numbers of regions that might contain bounding boxes. This enhanced accuracy comes at the increased cost of computational efficiency.

In figure 7, it could be seen how image input goes through the convolutional layers to feature maps which produce regional proposal networks. Then, using classifiers network over ROI pooling, an arbitrary number of regions for bounding boxes can be proposed. [6]

Another extension for making object detection better are *Single Shot Detectors* (SSDs). Instead of using subnetworks for estimating the regions, SSD works on only predetermined regions. A grid of points known as anchor points are laid over the input image. At each anchor point, there are multiple boxes of different shapes which can serve as regions. At each anchor point with each box, the model estimates and gives the output of whether the objects exist within the regions or not. If the objects don't exist within the regions, the modifications are made to change the locations and sizes of the boxes to fit the objects in the regions properly. Most of the time, many potential detections are estimated by the SSDs that are overlapping.

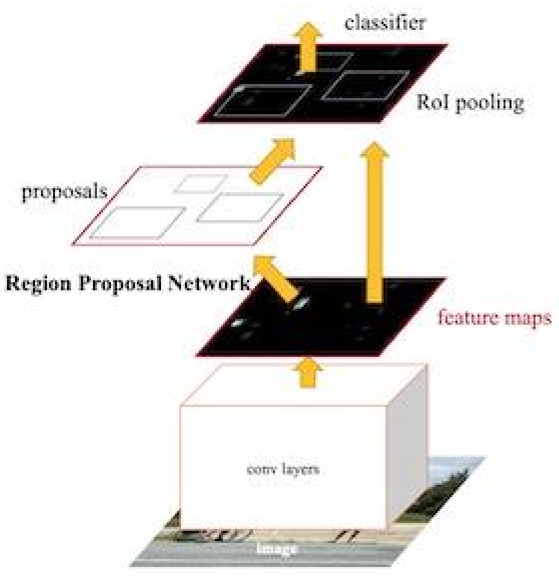


Figure 7: Region Proposal Network approach

This occurs because there are multiple boxes for each defined grid of anchor points and these anchor points can be close together. Therefore, post processing is done for the estimated overlapping to remove most of these predictions and keep the best one. One of the most popular post-processing techniques is known as *non-maximum suppression*.

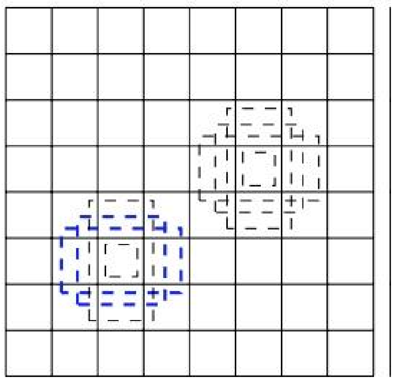


Figure 8: Single Shot Detectors

Object Detection algorithm gives the location and the class label of the objects as an output. For finding out the object's location, the most popularly known metric is *intersection-over-union (IOU)*. If the two intersecting bounding boxes are given, the area of intersection is computed and divided by the area of union. The output ranges from value 0 to 1 where 0 is no intersection and 1 is perfectly overlapping. In the labels of the bound objects, a simple "percent correct" is used.

3.1.2. Model Architecture Overview

3.1.2.1. R-CNN Model Family

Many popular object detection models are either based or belong to the R-CNN family. The regional convolutional neural network or R-CNN are the architectures that are based on the above mentioned structure of the region proposal network. This model family includes techniques like R-CNN, Fast R-CNN and Faster R-CNN which are designed and demonstrated for object detection and object localization. Let's see the highlights of each of these techniques.

- R-CNN

Regional convolutional neural networks or R-CNN can be considered one of the first largest applications of convolutional neural networks to our problem of object detection, localization and segmentation. This approach is demonstrated on the basis of benchmark datasets and achieving state-of-the-art results on the VOC-2012 dataset and the 200-class ILSVRC-2013 object detection datasets.

This approach based on the benchmark datasets has the proposed R-CNN model which is comprised of three modules, they are defined as follows-

- Module 1: Region Proposal - It generates and represents category independent region proposals, e.g. candidate bounding boxes.
- Module 2: Feature Extractor - It extracts features from each candidate region, e.g. using a deep convolutional neural network.
- Module 3: Classifier - It classifies features as one of the known classes, e.g. linear SVM classifier model.

A method to propose the candidate bounding regions or bounding boxes of the potential objects, a computer vision technique is used, known as "*selective search*", although the flexibility in its design allows the possibility to use other region proposal algorithms.

The architecture of the model is summarized in the figure 9 below.

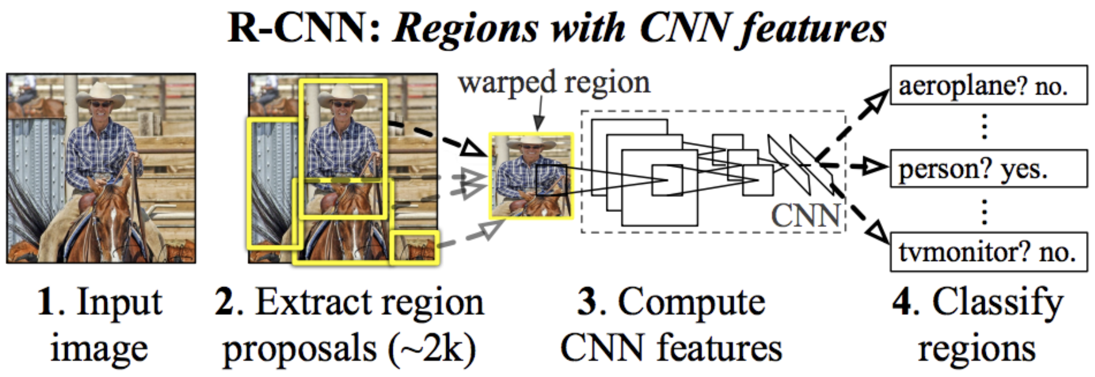


Figure 9: Summary of R-CNN model architecture taken from Rich feature hierarchies for accurate object detection and semantic segmentation

The feature extractor used by the model is based on AlexNet deep CNN. The CNN gives the output of a 4,096 element vector. The element vector is used to describe the contents of the image that is fed to a linear SVM for classification. One SVM is specifically assigned to each known class.

This application of CNNs to the problem of object detection and localization is relatively simple and straight-forward. The only disadvantage of using this application is that it is relatively slow. This requires a CNN based feature extraction that goes through each candidate region which is generated by the region proposal algorithm. The problem faced by this model is that it operates on 2,000 proposed regions per image at a time, which consumes a lot of time considering increased processing time for other modules of the model.

- <u>Fast R-CNN</u>

Based on the 2015 paper titled as “Fast R-CNN” [10] which summarizes the limitations of R-CNN as -

- *Training is a multi-stage pipeline* - training of an R-CNN model requires preparation and operation of three separate models.
- *Training is expensive in space and time* - deep CNN training on so many region proposals per image is slow.
- *Object detection is slow* - making predictions on so many region proposals per image is very slow.

Instead of using a pipeline to learn and output regions and classifications directly, Fast R-CNN is proposed as a single model. In the architecture of this model, a set of region proposals are taken over the photograph. This set is then passed through a deep convolutional neural network. VGG-16 (a pre-trained CNN) is used for feature extraction. A custom layer at the end of the deep CNN, called a “Region of Interest Pooling layer” or ROI Pooling. ROI pooling is used to extract the specific features for the given input of candidate region.

The output of the deep convolutional neural network passes through a series of fully connected layers (FCs). The output of FCs is divided into two outputs. One of them goes through class prediction via a softmax layer and another one with a linear output goes through a bounding box regressor. This process is repeated as a loop for multiple times and passed through each region of interest in a given image.The model is significantly faster but it still requires a set of candidate regions per image each time. The architecture of Fast R-CNN is shown in the figure 10 below.

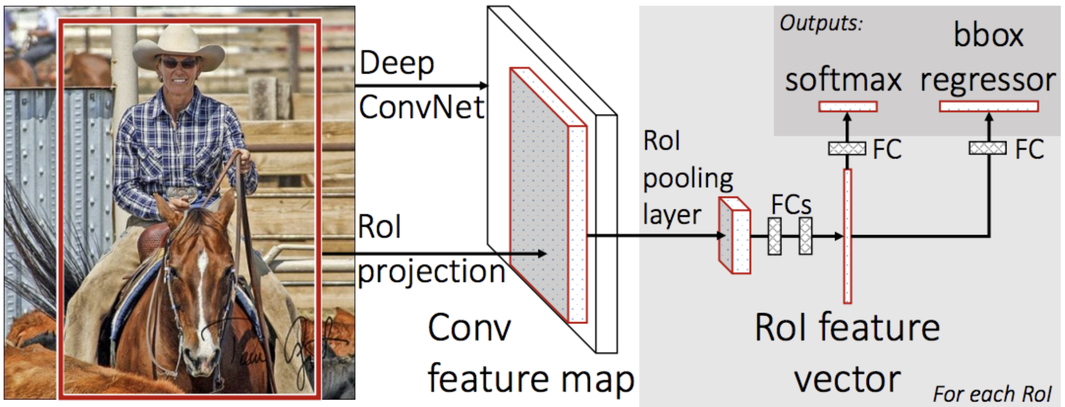


Figure 10: Summary of Fast R-CNN model architecture. Taken from “Fast R-CNN”

- <u>Faster R-CNN</u>

This model was introduced to improve speed for both training and detection. The architecture model is based on the benchmark datasets achieved from both ILSVRC-2015 and MS COCO-2015 which are used for training models for object detection and localization.

The architecture model for Faster R-CNN first introduced the concept Region proposal network or RPN which we discussed in R-CNN basic structure earlier [11]. RPN is used to propose and refine region proposals as a part of the training process. These regions are then used in agreement with single-model design prepared for the Fast R-CNN model. Using RPN, improvements are observed as the number of region proposals are reduced and the test-time operation of the model is accelerated to near real-time with then state-of-the-art performance.

The architecture of the single model is consisted of two modules-

- Module 1: Region Proposal Network - CNN to propose regions and the type of object to consider in the region.
- Module 2: Fast R-CNN- CNN to extract features from the proposed regions and output the bounding boxes and class labels.

RPN works as an attention mechanism for the Fast R-CNN model for enhancing the performance of the architecture in CNN based tasks. The architecture of the model is described in Figure 11 below.

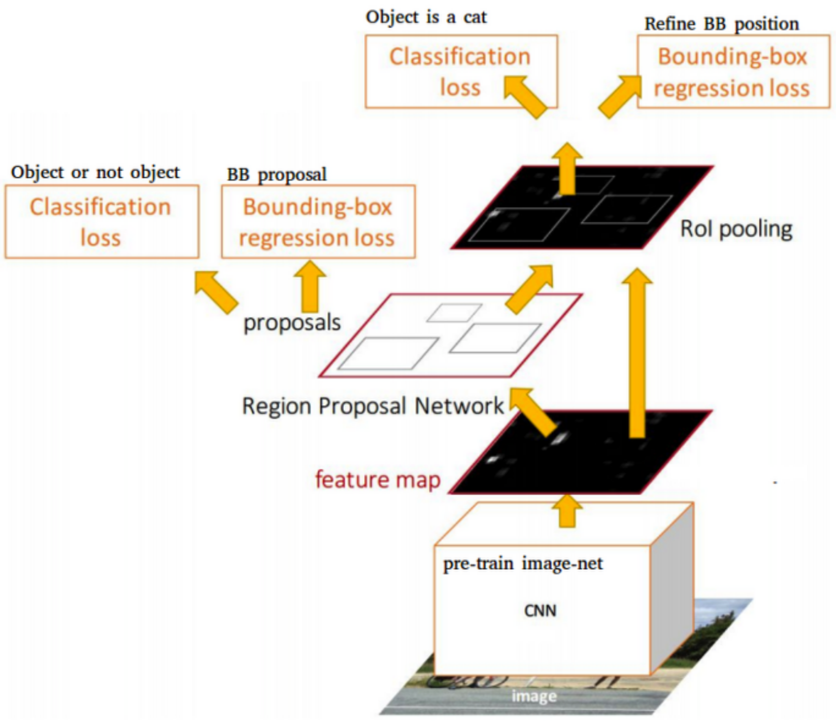


Figure 11: Summary of Faster R-CNN Model Architecture. Taken from "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"

RPN takes output from a pre-trained CNN like VGG-16 and passes over a small network for feature maps and outputs multiple region proposals. RPN checks the proposals if the potential object lies in the region proposals or not. They are passed through the next anchor box for bounding box regressor before moving to the Region of Interest pooling layer. The output of ROI pooling is moved through the anchor box for object classification and bounding box regressor until multiple region proposals are checked in the input image.

3.1.2.2. YOLO Model Family

This is another family for object detection models, known as YOLO or "*You Only Look Once*". The R-CNN model family can be considered more accurate but the YOLO model family is much faster than R-CNN. Even so, YOLO is achieving object detection in real-time.

- YOLO
  The YOLO model was first described in a 2015 paper titled as "*You Look Only Once: Unified, Real-Time Object Detection*" by Joseph Redmon [12]. In this approach, a single neural network is trained from end to end. It takes a photograph as an input, creates a bounding box around objects and then predicts class labels for each bounding box directly. Although this technique offers lower predictive accuracy(e.g. more localization errors), it is much faster than all R-CNN models. It operates at 45 frames per second and up to 155 frames per second for a better and speed-optimized version of the model. [7]

  First, the input image is split into a grid of cells. Now each cell is independently responsible for predicting B (a bounding box) if the center of the bounding box lies inside the cell. This prediction of a bounding box by each cell involves the x, y coordinate, the width (w), the height (h) and the confidence score as well as class prediction. Each confidence is used to reflect the probability that if the bounding box contains an object Pr(Object) as well as by evaluating the overlap of the predicted box with ground truth bounding box measured by the intersection over union, its accuracy is calculated.

  Therefore, the value of confidence score is $Pr(obj) * IoU_{pred}^{truth}$.

$$P_r = \begin{cases} 1 & \text{if the object exist, the confidence score} = \text{IoU} \\ 0 & \text{otherwise} \end{cases}$$

  For example, an image is split into 7 × 7 grid of cells. Each cell in the grid is capable of predicting at most 2 bounding boxes, which will result in 94 proposed bounding boxes predictions. The bounding boxes with confidence score and the class probabilities are then incorporated together to give an output of bounding boxes and class labels as a final set. The Figure 12 below summarizes the two outputs of the model.

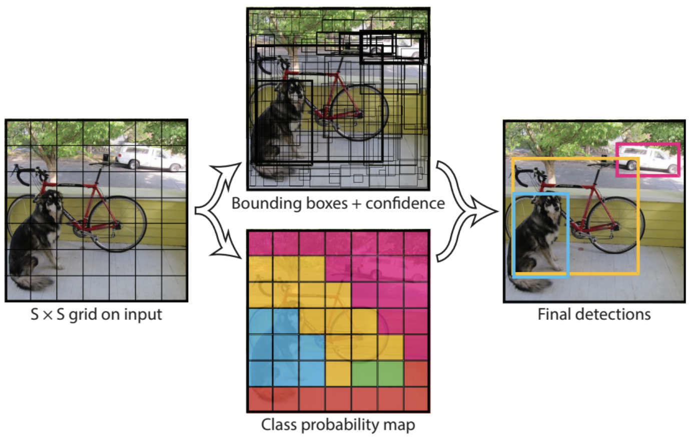


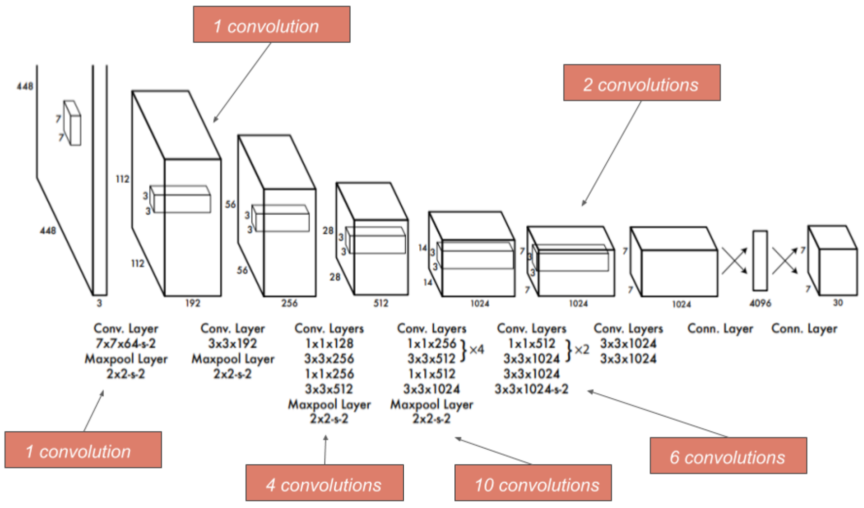
Figure 12: YOLO model summarizes all the predictions. Taken from "You Only Look Once: Unified, Real-Time Object Detection"

During test time, the model signifies the class-specific confidence score with each box and multiplies each box's confidence score with its respective class-conditional probabilities. These scores will be used to evaluate the probability of class appearing in the box and the precision of the box coordinates.

$$Pr(Class_i|Object) \times Pr(Object) \times IoU^{truth}_{pred} = Pr(Class_i) \times IoU^{truth}_{pred}$$

Architecture of YOLO is extremely based on GoogleNet. The network consists of 24 convolutional layers (used for feature extraction). It is followed by 2 fully connected layers (FCs) which are used for predicting bounding box coordinates and their respective object probabilities. Apparently, YOLO replaces the inception modules which are used in GoogleNet with 1 ✕ 1 convolution layers, so that the depth dimensions of the feature maps can be reduced.

The training part in YOLO comprises of 2 steps:
● The network is pre-trained so that classification can be performed with 224 ✕ 224 resolution on ImageNet. It uses only the first 20 convolution layers which are followed by an average pooling and a fully connected layer.
● For training the network for detection, the four convolution layers and two fully connected layers are added. A 448 ✕ 448 resolution inputs are trained as it is observed that gradually training on high resolution images considerably increases the accuracy. This is quite evident that visual information is better for detection.
The architecture for YOLO is shown in Figure 13 below.

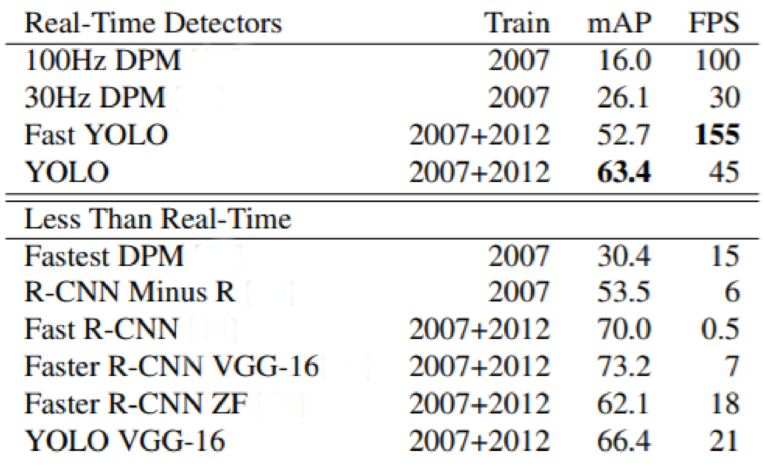


1 convolution
2 convolutions
448
7
7
448
112
3
3
112
56
3
3
56
28
3
3
28
14
3
3
14
7
3
3
7
7
7
7
7
3
192
256
512
1024
1024
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1 convolution
4 convolutions
10 convolutions
6 convolutions

Figure 13: Architecture of YOLO model. Taken from "You Only Look Once: Unified, Real-Time Object Detection"

A modified version of YOLO known as "FastYOLO" is also introduced. It is a comparatively smaller network of 9 convolution layers. Yet, this oversimplified network structure contributes to an impressive speed of 155 fps when recorded for PASCAL VOC detection. Even with an accuracy of 52.7% mAP, it has twice better accuracy than any other real-time detector. The values in the table below shows the real-time systems in PASCAL VOC 2007 and compares their accuracy and speed with each other.

Table 2: Real-Time systems on PASCAL VOC 2007



Real-Time Detectors	Train	mAP	FPS
100Hz DPM	2007	16.0	100
30Hz DPM	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM	2007	30.4	15
R-CNN Minus R	2007	53.5	6
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ZF	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

To further examine the difference between YOLO and earlier developed detectors, a detailed breakdown is done in the paper. YOLO is compared with Fast R-CNN because Fast R-CNN is one of the highest performing detectors on PASCAL as well as its detections are publicly available. The methodology and tools of Hoiem et al have been used. The top N-predictions of each category at test time are observed. Each prediction is either correct or it is classified on the basis of the type of error-

- Correct: correct class and IOU > .5
- Localization: correct class, .1 < IOU < .5
- Similar: class is similar, IOU > .1
- Other: class is wrong, IOU > .1
- Background: IOU < .1 for any object

As the localization errors are much higher for YOLO in comparison to all other detectors combined. Fast R-CNN performed better with localizing objects correctly, but background errors are much higher than what YOLO reported. As shown in figure 14 below, 13.6% background error for Fast R-CNN means Fast R-CNN is 3 times more likely to predict background detections than YOLO.

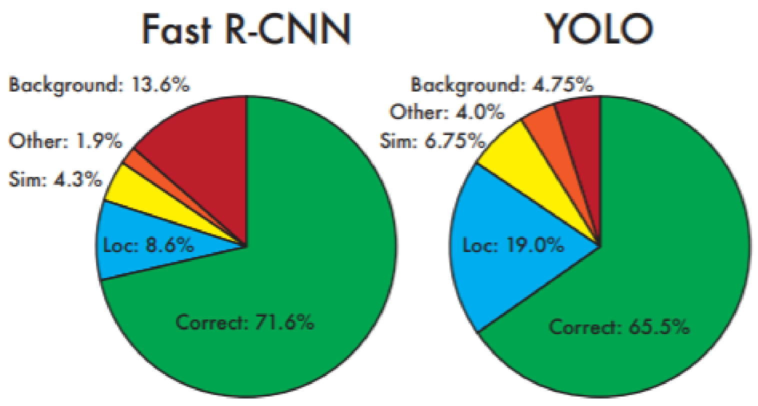


Figure 14: Error Analysis: Fast R-CNN vs YOLO

- <u>YOLOv2 (YOLO9000) and YOLOv3</u>

The updated model was designed by Joseph Redmond and Ali Farrahadi in order to improve the performance of the model as mentioned in the paper titled as "*YOLO9000: Better, Faster, Stronger*". [13]

This updated model is able to train on two object detection datasets in parallel, which makes it capable of predicting 9,000 object classes at a time. Hence YOLOv2 is also known as YOLO9000. A lot of changes related to the training method and architecture were made in this model. For example, high-resolution input images and the use of batch normalization.

Like Faster R-CNN, YOLOv2 model has also used pre-defined bounding boxes with useful shapes and sizes as well as use of anchor boxes during training. The images are pre-processed with use of k-means analysis on the training dataset for choosing the bounding boxes.

To have the less influencing effect on the predictions, the bounding boxes' predicted representation is modified to make small changes. This will result in a more stable model. Instead of choosing prediction values directly for position and size, predictions are done for moving and reshaping offsets for the pre-defined anchor boxes which are relative to a grid cell and their values are dampened by the logistic function.

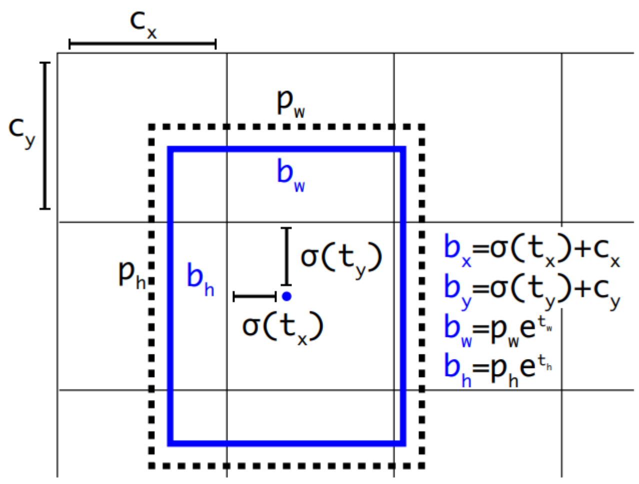


Figure 15: Example of chosen representation when predicting bounding box position and shape. Taken from "YOLO9000: Better, Faster, Stronger"

The improvements proposed for YOLOv3 in the 2018 paper titles as "YOLOv3: An Incremental Improvement" were really minor including minor representational modifications and deep feature detector network. [14]

3.2. Proposed Indoor scene recognition method

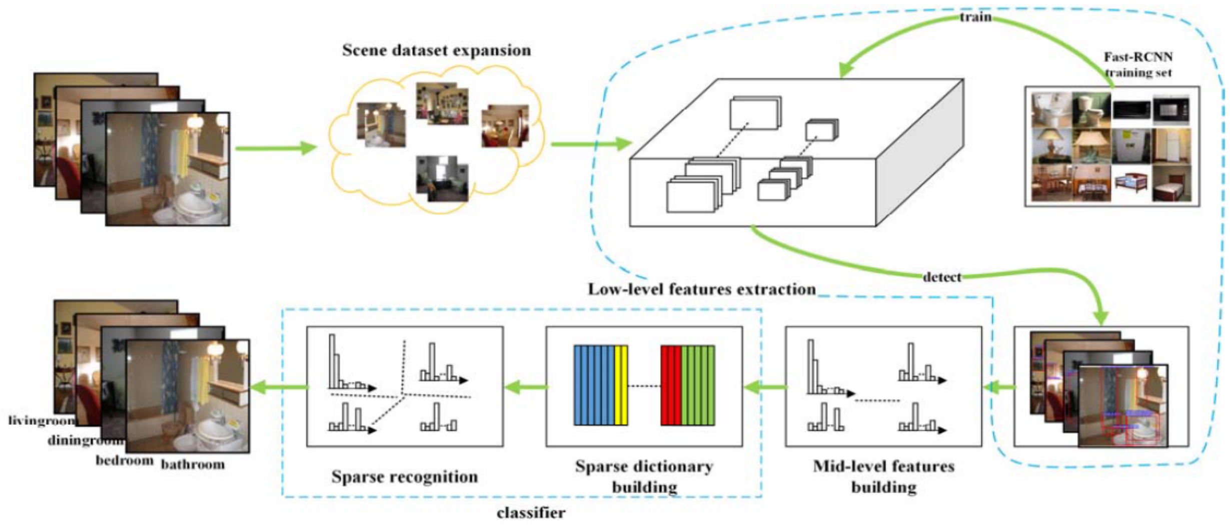


Figure 16: The framework of the proposed indoor scene recognition method

The proposed indoor scene recognition method is shown above in Figure 16. As we discussed in the above section, to overcome the problems faced by the traditional scene recognition methods, a new method has been proposed. This indoor scene recognition method will be based on deep learning and sparse representation. This method comprises five parts: scene dataset expansion, low-level features extraction, mid-level features building, sparse dictionary building and sparse recognition.

In scene dataset expansion, we increased the MIT-67 datasets by adding their rotated and flipped versions, so that we can fine-tune the Faster-RCNN multi-class detector. These edited images will increase the limited images in the different categories of MIT-67 dataset. All the information related to scene images like category, score of salient objects and position will be received by the detector. An improved BoW model is designed to create mid-level features. These mid-level features will have the semantic and spatial information from object-based low-level features. Using the spatial information from object-based low-level features, a spatial-pyramid with multiple layers is built. In each division of all layers, the feature histograms are concatenated as the mid-level feature. Finally, for generating the final recognition results, a SRC based classifier is trained on these mid-level features. Sparse representation is highly effective for object recognition under heavy partial occlusion and disturbance. Using SRC as a high-level semantic classifier, the robustness of indoor scene recognition can be improved further. Also chances of errors from low-level feature extraction and building mid-level semantics will be reduced.

3.2.1. Low-Level Feature Extraction

We have chosen Faster R-CNN multi-class object detector for extracting the low-level features from the input image scenes. Faster R-CNN is a better deep convolutional neural network architecture than R-CNN and Spatial pyramid pooling networks (SPPnets)[9]. Pre-trained Faster R-CNN model for multi-class object detector is fine-tuned by transfer learning. Faster R-CNN training set samples for indoor scenes are collected from ImageNet, as shown in Figure 17 below.

Figure 17: Faster R-CNN training set

Since we lack enough indoor image scenes for training dataset, so for enlarging it we utilized data augmentation by flipping and rotating the training dataset and added them in it. Faster R-CNN based multi-class object detector pipeline has been shown in Figure 18.

As discussed in Section 3.1.2.1, the architecture of Faster R-CNN will be divided into two parts: Region Proposal Networks and detection networks. Both parts will share feature extraction. Input image scenes pass through a deep convolutional neural network and all the Region of Interests (RoIs) of objects will be marked. All extracted RoIs from RPN are pooled into fixed-size feature maps. The feature maps are then mapped to the feature vector by the fully-connected layers. The network received two output vectors per RoI: bounding-box regressor offsets per class and softmax probabilities. This architecture is much faster to train and test than all the former versions of the R-CNN model family. This architecture also improved the performance of recognition significantly.

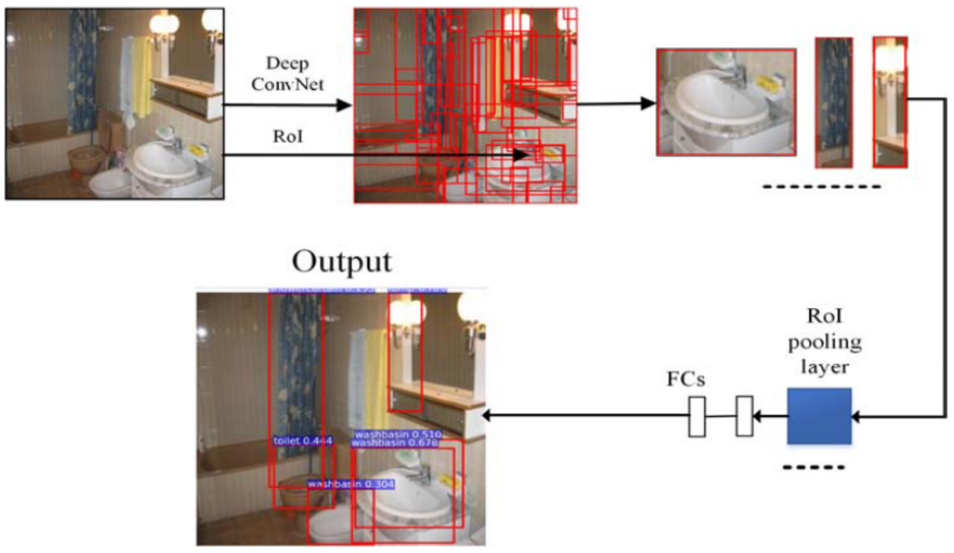


Figure 18: Faster R-CNN model architecture

3.2.2. Mid-Level Features Building

In the general procedure for BoW, a local operator has extracted the low-level features from the input images and has obtained the visual vocabulary vector. To build a dictionary (word bank) from all the low-level features, K-means clustering is used for computing the mean value of k cluster-center. All the final features are counted for generating feature histograms. Then the final classifier is trained on some learning method like SVM or KNN.

Here, we are using the BoW model to build a mid-level feature. Various partitions on different scales of the rectangle are divided from the indoor scene image. A spatial pyramid is extracted from these partitions of images and feature histograms are extracted from each partition. The accuracy of indoor scene recognition can be improved by preserving the spatial information of the low-level features. Building of SPM and its implementation plays an important effect on the final results. Larger partitions often show difficulty for providing accurate information about the object's scale and position. While smaller partitions have introduced the problem of showing more noise and leading this to a higher feature dimension. In a traditional SPM method, the partition of an image is equally subdivided layer by layer. In our case, we subdivided the image into different scales of rectangle by splitting it across its cross-sectional area. The SPM is designed by building 5 layers for the mid-feature of indoor scene image. Layer0 is the full image. Layer1 is split into two parts: vertically division and horizontal division. Similarly, Layer2, Layer3 and Layer4 are further into 4, 9 and 16 equal parts, respectively. This division method is shown in Figure 19 below.

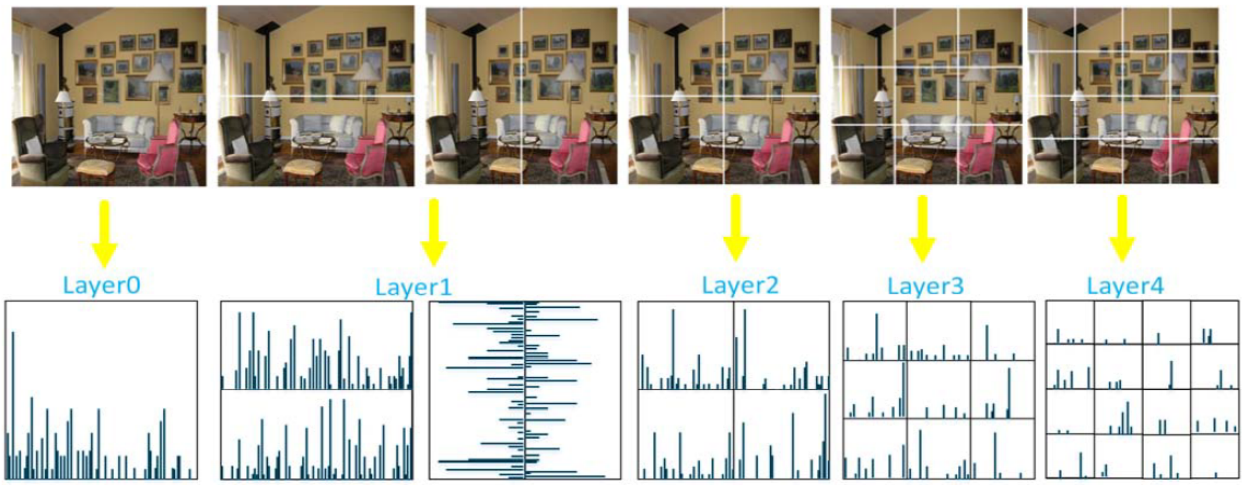


Figure 19: The division method for the proposed mid-level feature building

The process for building mid-level feature is shown as follows:

1) The low-level features of the input image are extracted $F_l(c_i, p_i, s_i)$, where c_i, p_i, s_i are the category, position and score of the ith object;
2) The feature histogram is created for each partition, designed only for counting pixel-by-pixel which is based on low-level features. The pixels accounted for are the only ones which are being overlapped by the multiple objects of the same category and their only high scores are considered. The histogram of the nth partition in the mth layer is defined as h_n^m. It is calculated by counting the sum of the score for each certain category.
3) The final mid-level feature histogram is obtained by concatenation of all the histograms h_n^m, $H = concat[h_1^0, ...,h_{M0}^0 , ..., h_{M4}^4]$, where M_0, ...$M_4$is denoted as the number of divisions in Layer0 to Layer4 respectively.

3.2.3. Indoor Scene Recognition based on Deep Learning and Sparse Representation

Sparse Representation-based Classification[23] is used for increasing performance of the scene recognition under the influence of extreme occlusion and heavy noise. To further increase the performance of recognition, SRC is trained on already received mid-level features for determining the final results achieved for scene recognition. The algorithm of our whole procedure for this method is shown in Table 3 below. The object-based low-level features are quickly and accurately extracted by the Faster R-CNN multi-object based detector. The mid-level features with a modified SPM are obtained for the BoW model. The SPM will preserve spatial information of all the low-level features. Finally, the classifier is trained on sparse representation for increasing the performance of the proposed method.

Table 3: Algorithm of the procedure for the proposed method

Indoor scene recognition based on deep learning and sparse representation
Step1: extract the object-based low-level feature $F_l(c_i, p_i, s_i)$ of input image;
Step2: build mid-level features: $H = concat[h_1^0, ...h_{M_0}^0, ..., h_{M_4}^4]$;
Step3: build sparse dictionary: $A = [H_1, ..., H_k]$, where k is the number of train samples;
Step4: calculate sparse coefficient: $x_1 = \arg\min \|x\|_1 \ subject \ \ to \ \ \|Ax - y\|_2 \ll \varepsilon$;
Step5: classify by minimum residual: $\min r_i(y) = \|y - A\delta_i(x_1)\|_2$;
Step6: determine the final recognition result: Class = identity(y);

3.3. Visual SLAM method

3.3.1. Installation of ORB-SLAM

In this part, we will install ORB-SLAM2 based on the instructions provided on Github page[30]. The current version of ORB-SLAM2 required additional installation of Eigen 3.3.3 and OpenCV 3 on ROS Melodic for Ubuntu 18.04.

3.3.2. Calibration of Camera

This part included instructions for the ROS calibration tutorial [31] used for calibrating the MYNT-EYE-D depth camera that we will use in generating our testing sequence (Section 4.2.6). The camera calibration we received is shown in Table 4.

Table 4: Camera calibration values

Camera Matrix	fx	fy	cx	cy	
	644.996428	649.647201	318.918209	225.367132	
Distortion Coefficients	k1	k2	p1	p2	k3
	-0.073281	0.052634	-0.006255	0.003086	0.000000

3.3.3. Reproduction of results on KITTI and TUM Datasets

In this part, we downloaded two sequences of the KITTI dataset: sequence 00 and 05 [32]. From the TUM dataset we download fr3_walking_halfsphere [33]. We tried all three sequences by running them on the installed ORB-SLAM2 following the instruction from the Github page [30] to reproduce results. The point clouds created as shown in the screenshots in Figure 20 below. The commands for running all of the three sequences on ORB-SLAM2 are shown below:

```
./Examples/Stereo/stereo_kitti Vocabulary/ORBvoc.txt Examples/Stereo/KITTI00-02.yaml
    ./KITTI/00
./Examples/Stereo/stereo_kitti Vocabulary/ORBvoc.txt Examples/Stereo/KITTI04-12.yaml
      ./KITTI/05
./Examples/Stereo/stereo_tum Vocabulary/ORBvoc.txt Examples/Stereo/TUM3.yaml
      ./TUM-RGBD/rgbd_dataset_freiburg_walking_halfsphere
```

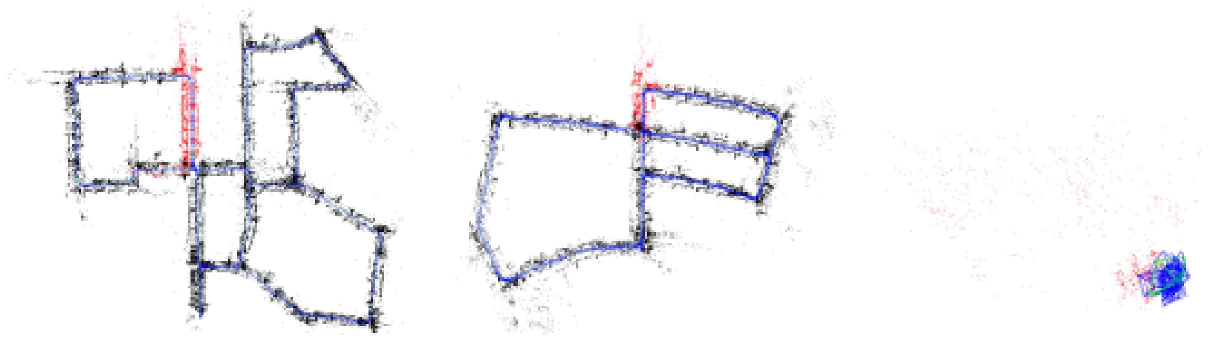


Figure 20: The point clouds generated by running the ORB-SLAM2 for KITTI datasets and TUM dataset. From left to right: KITTI00, KITTI05 and TUM fr3_walking_halfsphere.

3.3.4. Production of 2D Grid Map using ORB-SLAM map points

For generating a 2D occupancy grid, we processed all the map points (point clouds) and the keyframes generated by ORB-SLAM only after all the frames are included in the sequence. We modified ORB-SLAM stereo application Examples/Stereo/stereo_tum to give an output of all the keyframes consisting of 3D poses including map points in each frame to a text file. The script generating 2D occupancy grid will store the key frames, camera poses and the included map points of each frame into the form of dictionary structure. This will help in projecting all the values on the XZ plane. It is quite understood that these projections could only be obtained by only removing Y coordinate. The ORB-SLAM coordinate values are stored in meters while finer grid resolution is calculated by the product of all positions to inverse of desired resolution chosen as a

scaling factor. For example, if the needed resolution is 0.1m/cell, then the scaling factor is 10.
The following steps are applied to all the points in each keyframe and keyframes are then processed one at a time:

1) Camera position ray casting is set to all visible points by the method of Bressenham's line drawing algorithm [34].
2) For each point along the ray, a visit counter is incremented. For corresponding the locating of the map point, an occupied counter is incremented.

Then visit and occupied counters will be defined as integral arrays which have the same sizes (for the range of x and z locations of camera and map points after being scaled) and then they will be stored. For ORB-SLAM, we have assumed the XZ plane as the horizontal frame, therefore y will be considered as the height. After keyframes are processed, we will calculate the occupancy probability of each cell for the grid map as:

$$p_{free}(i,j) = 1 - \frac{occupied\ (i,j)}{visit\ (i,j)}$$

Where *visit (i,j)* and *occupied (i,j)* are the respective entries we received for visit and occupied counters and p_{free} is the probability of the respective cell that it is not occupied.
The ternary cost map is converted from the probability cost map using occupied_thresh and free_thresh so that particular cell will be considered free only if its p_{free} is greater than free_thresh, occupied if its value is less than occupied_thresh. Otherwise their value will be unknown.

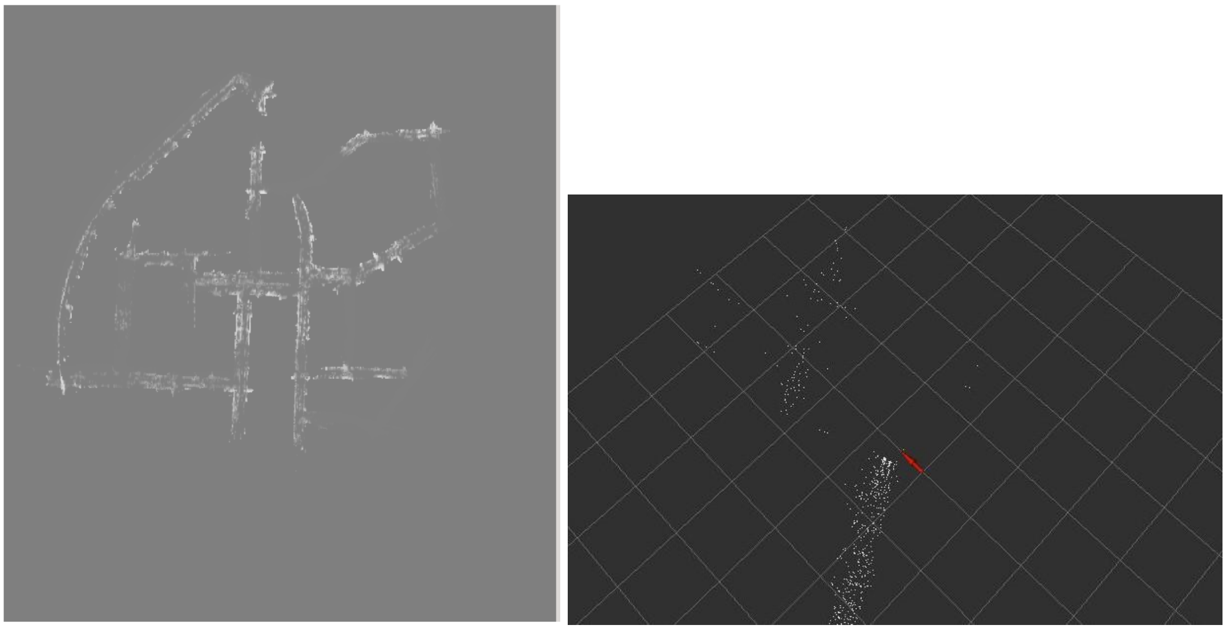


Figure 21: Probability Map generated before thresholding for KITTI00 sequence (left) and our local map (right)

3.3.5. Map visualization and robot navigation in Rviz

In this part, we used a simulation of TurtleBot in an empty world using the Gazebo simulator [35]. We applied Monte Carlo localization (AMCL) for navigation of TurtleBot [36] and its visualization on Rviz [37]. The following commands will be used to run these nodes:

```
roslaunch turtlebot_gazebo turtlebot_world.launch
     world_file:=/opt/ros/indigo/share/turtlebot_gazebo/worlds/empty.world
roslaunch turtlebot_gazebo amcl_demo.launch map_file:=./grid_map.yaml
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

In Figure 22 below, we have shown the map generated and the navigation path produced by AMCL when we ran these command lines:

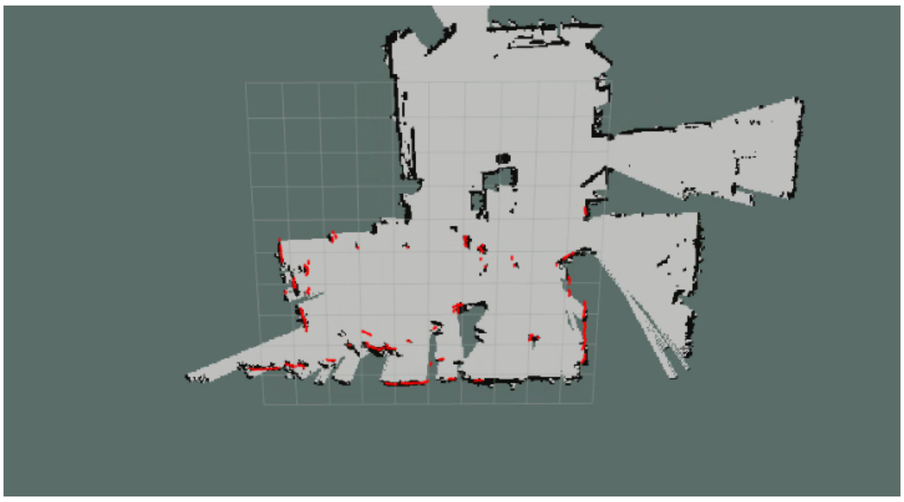
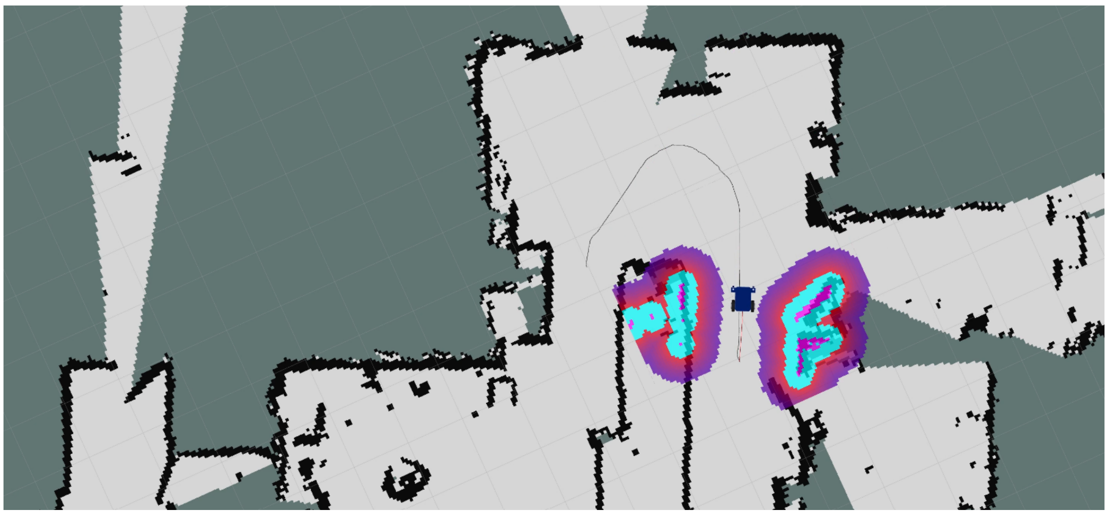
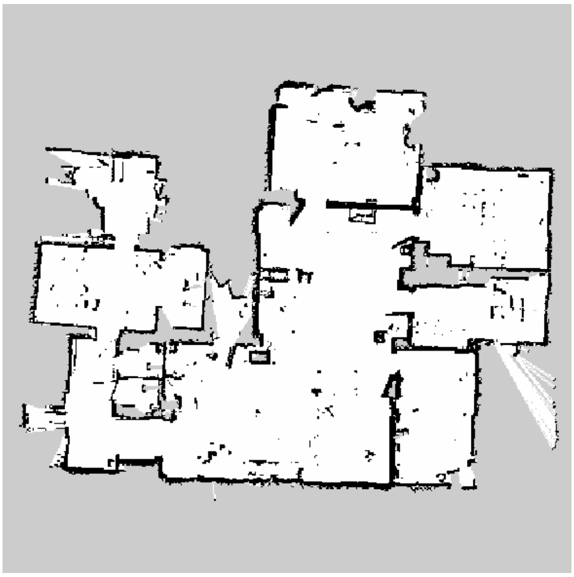



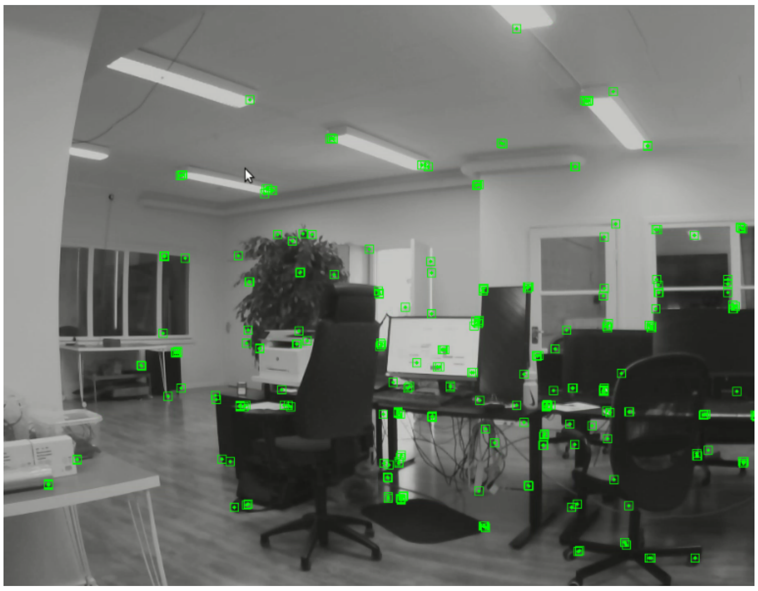
Figure 22: Map Visualization and robot navigation

3.3.6. Evaluation and comparison of results with already available methods

As we have already said earlier, it is not possible to create 2D occupancy grid maps using 3D points generated by Visual SLAM using methods like ORB-SLAM. Therefore, we needed data from 2D LiDARs so that we can compare it without results. We used RPLiDAR A3M1 for plotting 2D SLAM. The visualization of the map is shown in Figure 23 (a). As well as for comparison, we recorded our own data using the calibrated stereo camera MYNT-EYE-D for ORB-SLAM. Figure 23 (b) shows point clouds frames for indoor scene environments.

Next, using the 2D LiDAR SLAM generated map, we created a similar ground truth map for our sequence. Figure 23 (c) shows 3D point clouds of our sequence created after the completion of mapping. We used two measures for evaluating this comparison: a completeness sore and accuracy measure. The completeness score shows that out of the whole truth map how much of the map was able to generate. The accuracy measure shows how correct the map is in comparison to the ground truth map. For comparing two maps, both of them were aligned and then we used the following formulations to compute the evaluation matrices.





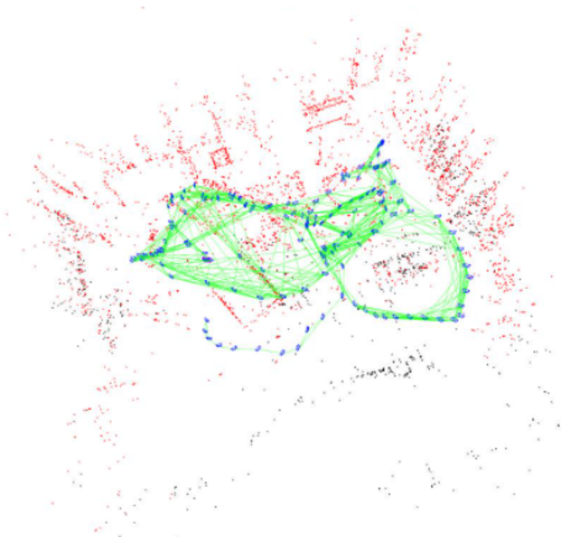


Figure 23: (a) The map visualization using 2D LiDAR; (b) The points clouds frames generated in a 3D indoor scene environment; (c) 3D point cloud produced by ORB-SLAM on this sequence

3.3.7. Novel Components

For this part, we are building a grid map in real time based on the results of the previous section and then use it to follow the camera trajectory for navigation. The novel components are as follows:

3.3.7.1. Online Variant

First, to process all the keyframes and map points in real time, we converted the python scripts into C++ code to work in an incremental manner. We created two ROS nodes for the modification of stereo ROS nodes Examples/ROS/ORB_SLAM2/Stereo. The first node is known as Stereopub. It will be used to publish each keyframe's poses which are added along to the map with all the map points already visible in that keyframe. It also performs the detection whenever loop closure takes place and then publishes it along with the map points already added so far. Stereopub is also modified to take images from live stereo cameras as an input. Now, we will run this node on three sequences (KITTI00, TUM fr3_walking_halfsphere and our live stereo camera) and to publish images (ROS node to publish images to /usb_cam/image_raw topic). The commands are as follows:

```
rosrun ORB_SLAM2 Stereopub Vocabulary/ORVvoc.txt Examples/Stereo/KITTI00-02.yaml ./KITTI/00
0
rosrun ORB_SLAM2 Stereopub Vocabulary/ORBvoc.txt Examples/Stereo/TUM3.yaml
      ./TUM-RGBD/rgbd_dataset_freiburg3_walking_halfsphere
rosrun ORB_SLAM2 Stereopub Vocabulary/ORBvoc.txt Examples/Stereo/stereo.yaml 0
rosrun ORB_SLAM2 Stereopub Vocabulary/ORBvoc.txt Examples/Stereo/demo_cam.yaml -1
      /usb_cam/image_raw
```

The second node we built is known as Stereosub and it will be used to subscribe to all the pose data that has been published by Stereopub. So, whenever it will obtain a single keyframe, it will process it using the same method as mentioned in Sec. 3.3.4. There will be several additional steps that will be included in this method and these steps are described in subsequent sections. The visit and occupied counters are being added to create a map in increments. The map this process has generated is just an approximation of the true map because co-visibility for different keyframes between them is not taken into account (even if the counters are being updated independent of each frame. Although, it turned out to show quite accurate values and it is sufficient for our purpose of navigation.

The counters have ended and recomputed for all the keyframes once these keyframes are obtained after the loop closure. All the keyframes and map points have been received from the publisher simultaneously. Now we can check all the inaccuracies that might have been collected over the time. Stereopub can still publish all the keyframes and map points periodically even if the loop closure has not been detected. Although, the no detections of loop closure are few and far apart, so their chances of occurring are very less. Map should be reset from time to time even if it is a time consuming process.

For achieving the navigation goals in AMCL and Rviz, Monosub publishes the map that has been generated as well as the meta data in process with the camera trajectories. There are a lot of parameters that has to be adjusted in Monosub, the command line for it is as follows:

```
rosrun ORB_SLAM2 Stereosub <scale_factor> <resize_factor> <cloud_max_x>
<cloud_min_x> <cloud_max_z> <cloud_min_z> <free_thresh> <occupied_thresh>
<use_local_counters> <visit_thresh> <use_gaussian_counters> <use_boundary_detection>
<use_height_thresholding> <normal_thresh_deg> <canny_thresh>
<enable_goal_publishing> <show_camera_location> <gauss_kernel_size>
```

For example, using different parameters we can fine tune two nodes on KITTI00 and our local map sequences. The commands are as follows:

```
rosrun ORB_SLAM2 Stereosub 10 1 31 -22 46 -12 0.65 0.60 1 5 1 0 1 80 300
rosrun ORB_SLAM2 Stereosub 40 1 12 -16 23 -12 0.55 0.50 1 10 1 1 1 35 350
```

Note that for the first command, the resolution is 1/10 m/cell and for the second one it is 1/40 m/cell. Under this node, we can adjust parameters at runtime.

3.3.7.2. Local and Global Counters

An issue with projection of 3D points as 2D points onto the XZ plane which is used for ray casting arises the idea of creating local and global counters. For example, let's consider a situation where we are getting the projections of multiple collinear points in 2D in a single keyframe (as shown in Figure 24). In this case, for generating the 2D map, we are using only one global counter. This will lead to misrepresentation of the available information. The value of the visit counter is incrementing when we have been decreasing the occupied ratio parameter for all the points that have been laid along. This is occurring even when the middle point is already occupied. So if we use one global counter, we can replace some of the occupied cells with the free ones.

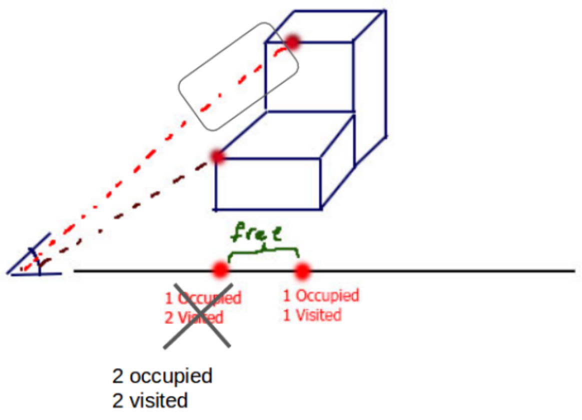


Figure 24: Problem of corrupting counter when points are collinear (in 2D)

3.3.7.3. Visit thresholding

To measure the confidence score of a cell, we have been using the visit count of that cell. In this condition, if the total visit count of the cell is higher than the threshold, only then that grid cell can be occupied or left empty. This lets us remove all the outliers that have been generated by wrong triangulations occurring in ORB-SLAM, particularly for those which were far away from the camera. We have been using the default visit threshold value of 5 in our tests.

3.3.7.4. Height thresholding

According to Goeddel et al, if the points that are lying within a specified range of y coordinate (as a height) over the XZ plane will be considered as obstacles. Therefore, changed the points back to the default camera coordinate frame and then started imposing a threshold on their height. If the points which have y-coordinate lying below this threshold will be considered as free space instead of either removing or counting their cells occupied. They will change as a free space by incrementing their visit counter.

3.3.7.5. Gaussian smoothing of counters

To avoid the problem of small distance separated two cells marked occupied causing the whole range of cells occupied, gaussian smoothing of local counters (both occupied and visited) is employed. Using this method, the value of any cell will be influenced by its nearest neighboring cell. So the cells that are empty will take the value of their neighbor's cell in both counters. This will generate a probability map much smoother. Based on this, we used a kernel size of 3 for our tests.

3.3.7.6. Canny boundary detection

As we have already employed so many methods for removing false obstacles, the problem is occurring where real obstacles are also being removed. We observed that this is usually occurring near the boundary obstacles, where they are lying somewhere between free space and unknown space. This will help us to bring all those obstacles lying between those spaces. Using canny detection, we set all the pixels of the detected edges as 0 in the probability map, so they can be marked occupied. Figure 25 below shows an approximate way to show the contours for the outer boundaries by eliminating all false obstacles from inside. Hence, we set the value of 300 for the canny threshold for our tests. We found that all the false edges have been eliminated on the interior side of the boundary without making any changes to these outer boundaries.

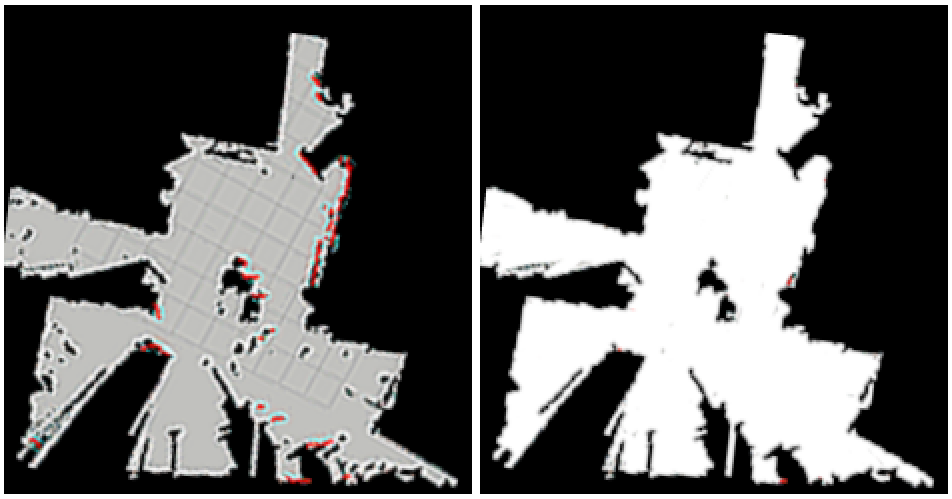


Figure 25: Finding contours for canny boundary detection

3.3.7.7. Slope thresholding

We encountered a problem where the points which are lying relative to the horizontal planes are not considered as obstacles. So we estimated the plane in which the point is lying by determining its two nearest points and then computing the best plane which can fit all these three points in the least square sense. If the angle between the normal to this plane and XZ plane is exceeding the threshold, then this plane will be declared as horizontal and the point will correspond to the nearest free cell. By checking this condition with our results, we found a threshold of 80 degrees would be correct for estimating the best fit plane.

3.4. Materials and methods for 3D-SLAM

In this section, we will describe the indoor mobile robot for 3D SLAM and testing its system for performance.

3.4.1. Indoor mobile robot for 3D-SLAM

The platform that we selected is Magni by Ubiquity Robotics. It is capable of carrying 100 Kg. It can run for 8 hours on a 10 Ah battery, and 24 hours if we upgrade it to 32 Ah battery. It comes with some sensors like Raspberry Pi Camera, sonar board etc and additionally we can connect it with other sensors like 2D LiDAR, monocular camera, depth sensors using Raspberry Pi 3 (Ubuntu 16.04) installed on board. We can use it for navigation purposes and further use it for building service robot on top of it. The magni platform is shown in Figure 26 (a) below. There is space for storing batteries and running Magni for teleoperation nodes. We have designed a support with some height for 3D LiDAR on top of Magni, so we can use teleoperation nodes for 3D SLAM. The increase in height over the mobile robot for 3D LiDAR will help us to scan all the point clouds from the ground to the ceiling, provided that angle measure is only 30° vertically (Table 5) 3D LiDAR sensor we are using is a Leishen C16 LiDAR, which we have chosen for collecting point clouds and on this mobile platform for its light weight and smaller dimensions (in comparison to other 3S LiDARs). In every scan, the sensor receives the 3D coordinate of the point which is detected by LiDAR's 16 rays. This will help to create a scattered 3D point cloud from all the points the robot has gone through.

Table 5: Technical characteristics of Lesihen C-16 provided by the manufacturer.

Leishen C-16	
Weight	800 grams
Laser rays	16 channels
Range	100-120 m
Acquisition rate	300,000 points per second
Point accuracy	3 cm
Field of View	360° (H) × 30° (V)

The information from the 3D LiDAR can be seen on the windows-based software provided by Leishen (connected using an adapter box with ethernet cable as shown in Figure 26(b). We can use this stored information (performed measurements for 3D point cloud) as a time-stamped, binary file which has structure and code provided by ROS, this format is known as bag (rosbag) [46]. It also allows us to reproduce the acquired results in the simulation.

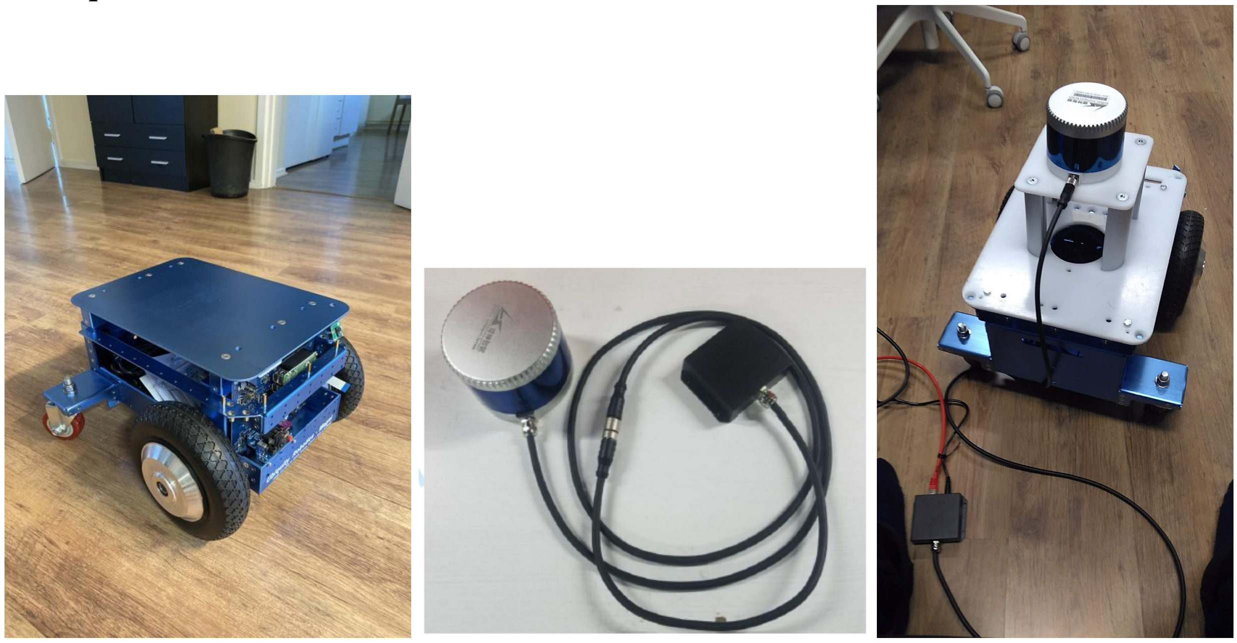


Figure 26: (a) Magni platform; (b) Connection between adapter box and LiDAR; (c) 3D LiDAR mounted on top of Magni for real-time indoor 3D mapping.

3.4.2. Windows software display and LiDAR configuration

The point cloud display software provides parsing data package and Device package information, and displays 3D point cloud data. Through the visual window, we can reset LiDAR parameters. LiDAR IP and port are shown in the table below:

Table 6 LiDAR Default Network configuration

	IP Address	UDP Equipment Port	UDP Parsing Port
LiDAR	192.168.1.200	2368 (fixed unmatchable)	2369 (fixed unmatchable)
Computer	192.168.1.102	2369	2368

When using a connecting device, it is necessary to set the computer IP to the same network segment as the device, for example, IP: 192.168.1.x, and subnet mask: 255.255.255.0. If the device's network configuration information is unknown, wireshark is used for the connecting device to capture the device's ARP packet for analysis after the LiDAR is powered on.

3.4.3. Communication Protocol

The LiDAR data output and configuration use the 100M Ethernet UDP / IP communication protocol. There are 3 UDP packet protocols with a packet length of 1248 bytes (42 bytes Ethernet header and 1206 bytes payload). The LiDAR supports unicast, broadcast, and multicast communications.

1) MSOP(Main data Stream Output Protocol), Output data include : measured distance, angle, intensity and other information；
2) DIFOP(Device Information Output Protocol), Output data include: LiDAR configuration information；
3) UCWP(User Configuration Write Protocol), Setting LiDAR Configuration parameter

Table 7: UDP Protocol

(Protocol/packet) Name	**Abbreviation**	**Function**	**Type**	**Packet Size**	**Transmission Interval**
Main data Stream Output Protocol	MSOP	Outputting scanned data	UDP	1248bytes	1.2ms/0.6ms
Device Information Output Protocol	DIFOP	Outputting device information	UDP	1248bytes	1.2ms/0.6ms
User Configuration Write Protocol	UCWP	Inputting user configured device parameters	UDP	1248bytes	INF

3.4.3.1. MSOP Protocol

The data of the packet is little-endian mode. MSOP Packet data format structure of the LiDAR includes frame header, sub-frame and frame tail. Each packet has 1,248byte: 42byte for UDP packet overhead, 1,200byte for sub-frame data packet interval (a total of 12 data blocks), 4byte for timestamp, and 2byte for frame tail factory.

3.4.3.2. Device Information Output Protocol (Device Packet Protocol)

The device package outputs read-only parameters and status information such as version number, Ethernet configuration, motor speed and running status, and fault diagnosis. The data of the device package uses big-endian mode.

The device packet includes 42-byte Ethernet header and a 1206-byte payload with a length of 1248 bytes. The payload consists of an 8-byte frame header, 1196-byte data Data and a 2-byte frame tail.

3.4.3.3. User Configuration Write Protocol (Configuration Packet Protocol)

The configuration packet protocol configures the LiDAR's Ethernet, PPS alignment angle, motor and other parameters. The data of the configuration packet adopts big-endian mode.

The configuration packet includes a 42-byte Ethernet header and a 1206-byte payload with a length of 1248 bytes. The payload consists of an 8-byte header, 1238-byte Data, and a 2-byte Tail.

3.4.4. Time Synchronization

There are 3 ways to synchronize LiDAR with external devices: GPS synchronization, NTP synchronization, and external PPS synchronization. If there is no external synchronization input, timing information is generated inside the LiDAR. The absolute precise time of the point cloud data is obtained by adding a 4-byte time-stamp (accurate to microseconds) of the data packet and a 6-byte UTC time (accurate to seconds) of the device packet.

3.4.4.1. GPS Synchronization

The LiDAR receives the PPS second pulse, the LiDAR uses microseconds as the unit, and the time value is output as the timestamp of the data packet. The LiDAR extracts UTC information from GPS's $ GPRMC information as the UTC time output of the device package, with accuracy to seconds.

3.4.4.2. NTP

The LiDAR periodically obtains the NTP server time ,The time is used as the timestamp of the data packet, and the extracted UTC time is output as the UTC (GMT) time of the device envelope. The LiDAR sends a time request to the NTP server every 4 seconds.

After receiving the request, the server sends time information to LiDAR according to the NTP protocol.

3.4.4.3. External Synchronization

The LiDAR obtains PPS signal input by the external device, the LiDAR uses microseconds as the time unit, and time is output as the time stamp of the data packet. At this time, there is no UTC time reference. If UTC time is required, it must be written through the configuration package; otherwise, the UTC time output information of the device package is invalid.

The PPS level of the external synchronization signal is 3.3 ~ 5V, which is triggered by the rising edge of the LiDAR. The PPS high pulse width is required to be greater than 40 ns.

3.4.5. LiDAR data calculations of Angles and Coordinates

3.4.5.1. Vertical Angle

See table 8 below for the vertical distribution.

Table 8: Vertical Angle Distribution of 16 Laser Channels

UDP Package Channel	Vertical Angle
Channel 0 Data	-15°
Channel 1 Data	1°
Channel 2 Data	-13°
Chanel 3 Data	3°
Channel 4 Data	-11°
Channel 5 Data	5°
Channel 6 Data	-9°
Channel 7 Data	7°
Channel 8 Data	-7°
Channel 9 Data	9°
Channel 10 Data	-5°
Channel 11 Data	11°
Channel 12 Data	-3°

Channel 13 Data	13°
Channel 14 Data	-1°
Channel 15 Data	15°

Table 9: Vertical Angle Distribution of C16-laser channels

UDP Package Channel	Vertical Angle
Channel 0 Data	-10°
Channel 1 Data	0.665°
Channel 2 Data	-8.665°
Channel 3 Data	2°
Channel 4 Data	-7.33°
Channel 5 Data	3.33°
Channel 6 Data	-6°
Channel 7 Data	4.665°
Channel 8 Data	-4.665
Channel 9 Data	6°
Channel 10 Data	-3.33°
Channel 11 Data	7.33°
Channel 12 Data	-2°
Channel 13 Data	8.665°
Channel 14 Data	-0.665°
Channel 15 Data	10°

By querying the table above, we can get the vertical angle of the 16-channel data.

3.4.5.2. Cartesian Coordinate Representation

To obtain the vertical angle, horizontal angle, and distance parameters of the LiDAR, and convert the angle and distance information in polar coordinates to the x, y, and z-coordinates in the right-hand Cartesian coordinate system. The conversion relationship is shown in the following formula:

$$\begin{cases} x = r\cos\alpha\cos\theta; \\ y = r\cos\alpha\sin\theta; \\ z = r\sin\alpha \end{cases}$$

The r is the distance, α is the vertical angle, θ is the horizontal rotation angle (the horizontal correction angle needs to be considered in the calculation), and x, y, and z are the coordinates of the polar projection onto the x, y, and z axes.

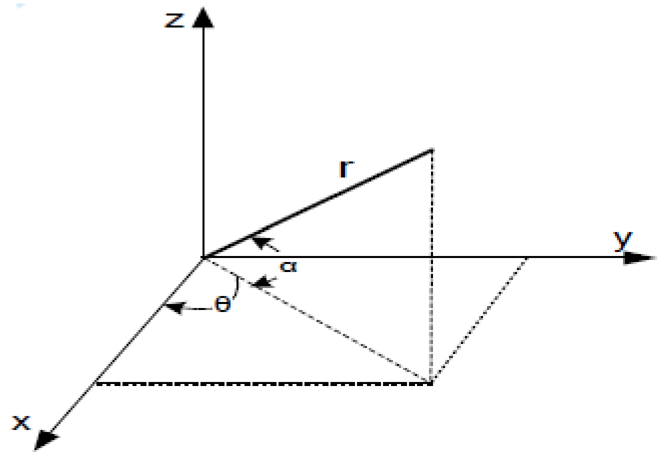


Figure 27: Cartesian Plane coordinates

3.4.6. Point cloud Data Time Calculation

To accurately calculate the time of point cloud data, we need to obtain the data packet timestamp and device package UTC time output by the LiDAR. The timestamp and UTC time come from the same synchronization source, such as a GPS or NTP server. The measurement time interval of a group of data in each data block of 16-LiDAR is 50us. The data packet has 12 data blocks, and one data block contains two groups of 16-channel data.
A data packet has (16 * 2) * 12 = 384 channels of data in total, and the packet packing time is about (50us * 2) * 12 = 1.2ms. The data rate is 1s / 1.2ms = 833.3 data packets / second. Double-echo mode data rate doubles.

3.4.6.1. GPS Time Calculation

The timestamp in the data packet is a relative time of microseconds, which is defined as the packaging time (data packet end time) of the last channel of laser measurement data in the data packet, which is less than 1 second, So to calculate the absolute time of the end of the data packet, we need to first Get the 4-byte microsecond timestamp in the data packet, and then get the UTC time (greater than 1 second) from the device packet. Adding the two is the exact time at which the data packet ends.
EXACT TIME = DIFOP TIME + MOSP TIMESTAMP

3.4.6.2. Channel Data Time Calculation

We have to obtain the exact time of the end of the data packet. Each UDP contains 12 data blocks. And each data block contains 2 groups of 16 channels of light emitting time and the light emitting time interval of each channel. The precise measurement time of each channel data can be calculated.

(a) Data Block Time - Each data block of C16 LiDAR contains 2 groups of 16 channel measurement data. The end time interval of each group of channels in each data block is 50us, each data block (single echo mode) or each parity block pair (Double echo mode) The end time interval is 2 * 50us = 100us. Assuming that the absolute time of the end of the data packet is TPacket_end, the steps to calculate the end time of the data block T_{Block_end} (N) are as follows:

1) Single Echo Mode

The data packet contains 12 data blocks, each data block includes two sets of single measurement data of 16 laser channels. The end time of each data block is the end time of all 2 groups of 16 channels. Calculate the end time of each data block as follows

T_{Block_end}(N) = (T_{Packet_end} – 100*(12-N))us,（N = 1,2, ...,12）

T_{Block_end}(N) represents the end time of the N th data block

2) Dual Echo Mode

T_{Block_end}(2N) =T_{Block_end}(2N-1) = (T_{Packet_end} – 100*(6-N))us,（N = 1,2, ...,6）

T_{Block_end}(M) represents the end time of the N th data block, M=2N or (2N-1)

(b) Point Cloud Data Time Calculation - The C16 LiDAR 2° / 1.33° type fixed the time interval of each channel as: T = 50us / 16 = 3.125us. There is a fixed correspondence between the lighting time and the UDP packet encapsulation order. Assuming that the lighting time of Channel 0 is T0, the corresponding lighting time of 16 channels is shown in the table below:

Table 10: C16 LiDAR Channel Glowing Time

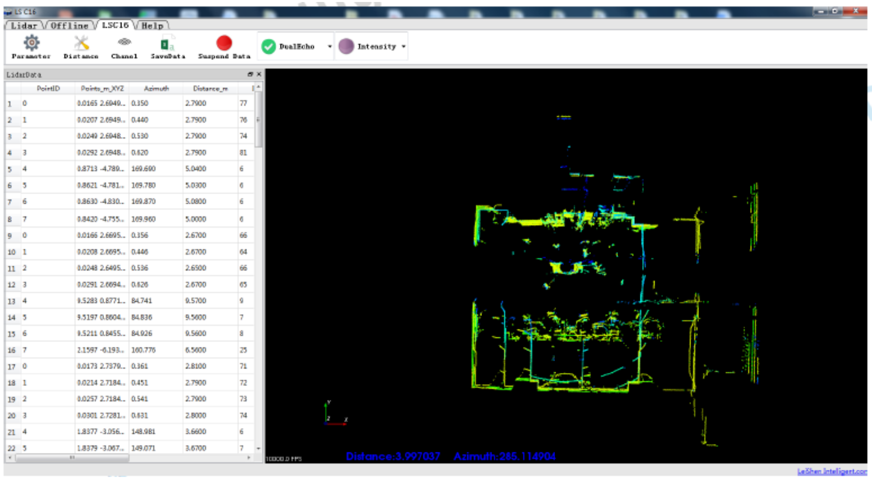
UDP (Channel)	Vertical Angle	Glowing moment（T=3.125us）
Channel 0 Data	-15°	T0
Channel 1 Data	1°	T0+(1*T)
Channel 2 Data	13°	T0+(2*T)
Channel 3 Data	3°	T0+(3*T)
Channel 4 Data	-11°	T0+(4*T)
Channel 5 Data	5°	T0+(5*T)
Channel 6 Data	-9°	T0+(6*T)
Channel 7 Data	7°	T0+(7*T)

Channel 8 Data	-7°	T0+(8*T)
Channel 9 Data	9°	T0+(9*T)
Channel 10 Data	-5°	T0+(10*T)
Channel 11 Data	11°	T0+(11*T)
Channel 12 Data	-3°	T0+(12*T)
Channel 13 Data	13°	T0+(13*T)
Channel 14 Data	-1°	T0+(14*T)
Channel 15 Data	15°	T0+(15*T)

After the end time of each data block is obtained, the precise measurement time of the point cloud data of each channel in the data block can be calculated according to the correspondence between the channel data packing order and the light emission time in the table above.

3.4.7. Implementation in Software Interface

In the Leishen's Windows software, we can connect an ethernet cable to the 3D LiDAR, we will start getting the 3D point clouds of the indoor environment as shown in Figure 28(a) and (b). We can change different filters and also record the point cloud data either in the form of excel sheet, rosbag or video.



	PointID	Points_m_XYZ	Azimuth	Distance_m	
1	0	0.0165 2.6949...	0.350	2.7900	77
2	1	0.0207 2.6949...	0.440	2.7900	76
3	2	0.0249 2.6948...	0.530	2.7900	74
4	3	0.0292 2.6948...	0.620	2.7900	81
5	4	0.8713 -4.789...	169.690	5.0400	6
6	5	0.8621 -4.781...	169.780	5.0300	6
7	6	0.8630 -4.830...	169.870	5.0800	6
8	7	0.8420 -4.755...	169.960	5.0000	6
9	0	0.0166 2.6695...	0.356	2.6700	66
10	1	0.0208 2.6695...	0.446	2.6700	64
11	2	0.0248 2.6495...	0.536	2.6500	66
12	3	0.0291 2.6694...	0.626	2.6700	65
13	4	9.5283 0.8771...	84.741	9.5700	9
14	5	9.5197 0.8604...	84.836	9.5600	7
15	6	9.5211 0.8455...	84.926	9.5600	8
16	7	2.1597 -6.193...	160.776	6.5600	25
17	0	0.0173 2.7379...	0.361	2.8100	71
18	1	0.0214 2.7184...	0.451	2.7900	72
19	2	0.0257 2.7184...	0.541	2.7900	73
20	3	0.0301 2.7281...	0.631	2.8000	74
21	4	1.8377 -3.056...	148.981	3.6600	6
22	5	1.8379 -3.067...	149.071	3.6700	7

Distance:3.997037 Azimuth:285.114904

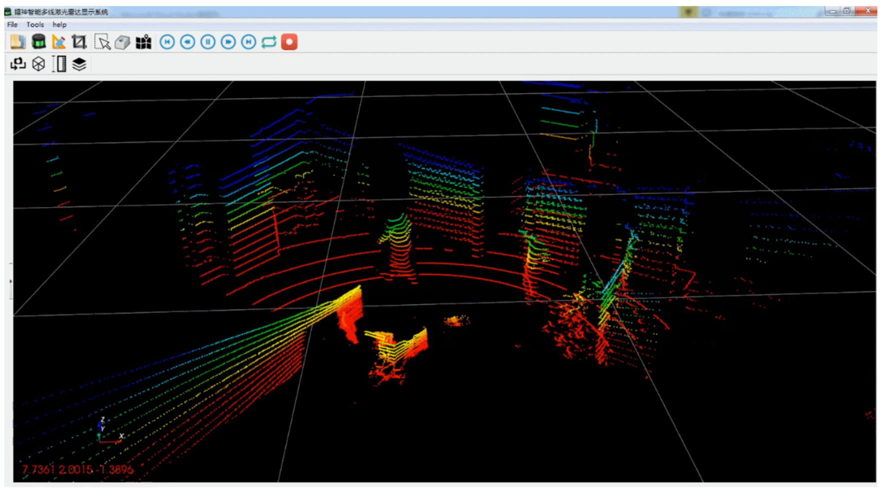


Figure 28: (a) Top View and (b) Side view of 3D LiDAR point cloud scan

3.4.8. Implementation in Simulation

We have collected the rosbags that we have generated from the windows software, now we will try the LiDAR using ROS Melodic and try to get output for SLAM in simulation Rviz.

3.4.8.1. Read the data from LiDAR

When the LiDAR is connected and is ready to use, if we already have the data permissions port:

```
$ ls -l /dev/tty
```

We should see a new item labelled *ACMX* or *USBX*, *X* being a figure equal or higher than zero (depending on how many ports are already in-use).
Our output will be in the form of:

```
$ crw-rw-XX- 1 root dialout 166, 0 2016-09-12 14:18 /dev/ttyACM0
```

or

```
$ crw-rw-XX- 1 root dialout 166, 0 2016-09-12 14:18 /dev/ttyUSB0
```

- If XX is rw, then the laser is configured properly.
- If XX is –, then the laser is not configured properly and we need to change permissions like below:

```
$ sudo chmod a+rw /dev/ttyACM0
```

or

```
$ sudo chmod a+rw /dev/ttyUSB0
```

Once the permissions are configured, we have to download the package of the LiDAR manufacturer.

For downloading LiDAR package from GitHub in the src folder of the ROS environment, the commands are:

```
$ cd ~/your_workspace/src
$ git clone #github link of the manufacturer#
$ cd ..
$ catkin_make
$ source devel/setup.bash
```

Inside the launch folder, launch with the same name as the LiDAR:

```
$ roslaunch lslidar_c16_decoder lslidar_c16.launch
```

To check that the LiDAR is publishing to /scan, we will use

```
$ rostopic list
```

All active topics will be listed, check that /scan is present. Next, check the messages being published to /scan by using:

```
$ rostopic echo /scan
```

We will be successfully able to stream data from the LiDAR. We can visualize the data on RViz with the command line below.

```
$ rosrun rviz rviz
```

3.4.8.2. Implementing LiDAR processed data

hdl_graph_slam is an open source ROS package for real-time 6DOF SLAM using a 3D LIDAR. It is based on 3D Graph SLAM with NDT scan matching-based odometry estimation and loop detection. It also supports several graph constraints, such as GPS, IMU acceleration (gravity vector), IMU orientation (magnetic sensor), and floor plane (detected in a point cloud). We tested this package with our Leishen C16 LiDAR. Output is shown in Figure 38.

4. Results and Discussion

4.1. Target Detection

We implemented YOLO using MYNT-EYE-D camera and the result is shown in Figure 29 below.

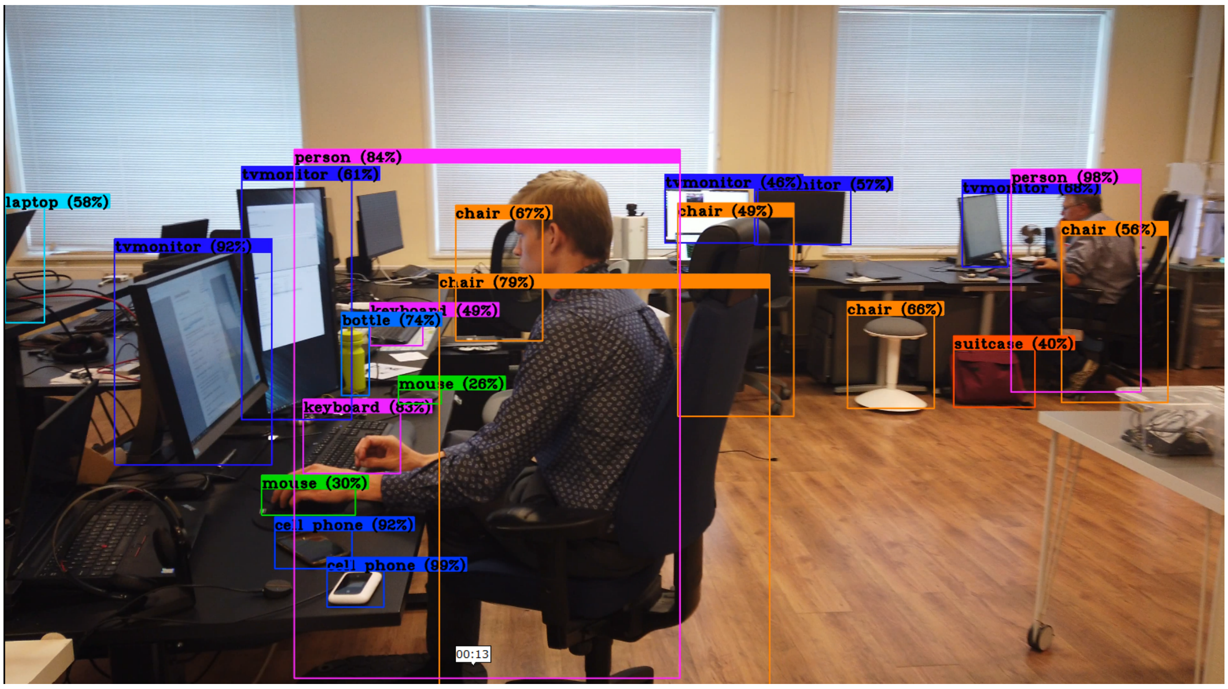


Figure 29: Image scene from a object detection video sequence

4.2. Indoor Scene Classification

The experiments consist of three modules. All the experiments including training and testing are performed on the MIT-67 dataset as mentioned earlier. The five categories that we have chosen for the experiments are: bathroom, bedroom, living room, kitchen and dining room. In total 600 image scene samples of different categories of MIT-67 dataset were taken for the experiment. The sample images are shown in Figure 21. For achieving better accuracy in our proposed method, we have used 5-fold cross-validation for testing the accuracy. Faster R-CNN detector was developed by using Caffe+Python in the JupyterHub Notebook and the rest of the parts were tested in Matlab. The experiments were conducted on an AMD Ryzen 7 processing station equipped with Radeon 2.5 GHz 8 core CPU and NVIDIA Jetson Nano 2 GB GPU.

Figure 30: Some indoor scene image samples from MIT-67 dataset

4.2.1. Evaluation of different parameters

We are using two key parameters for evaluating our method: layer combination and threshold of the Faster R-CNN detector. For layer combination, Level n is denoting the combination of any Layer n to Layer 0 . For example, Level 3 is a layer combination of Layer 0, Layer 1, Layer 2 and Layer 3. For the second parameter, threshold T of the Faster R-CNN based object detector will be assessed. The performance of the detector will be assessed on the basis of different four values of T. If the value of T is smaller then high values of false positives will be generated, while if the value of T is higher then needed objects will be missed in detection.

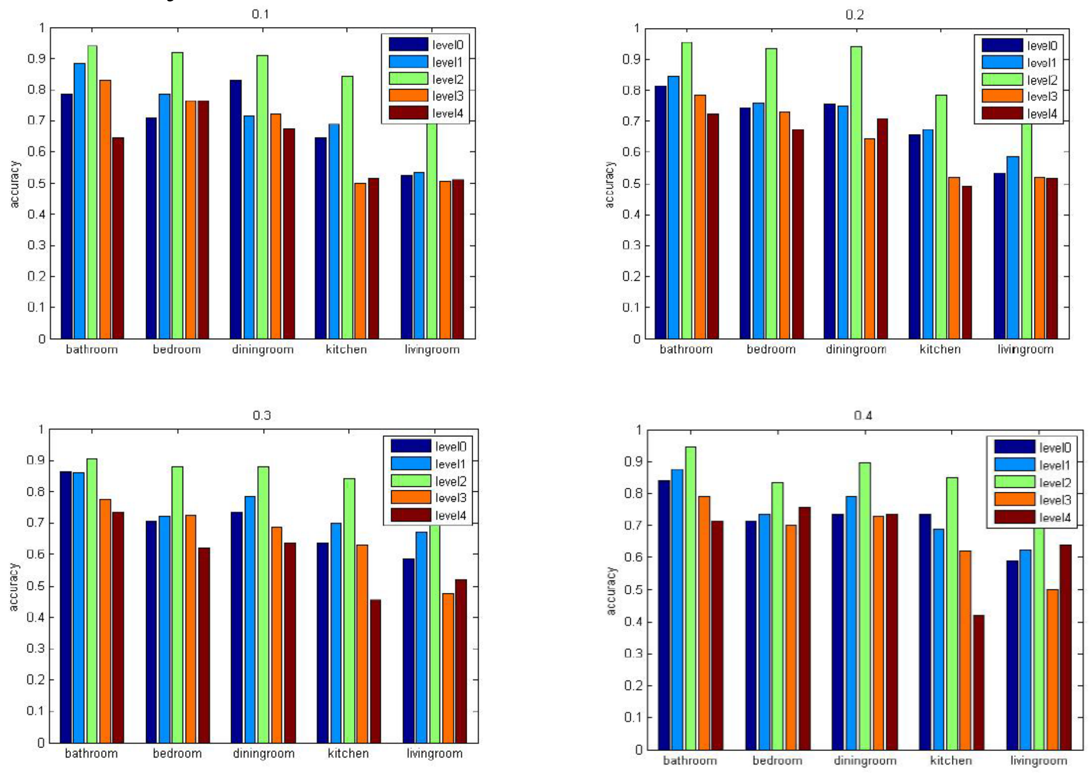


Figure 31: The Class-specific recognition accuracy of our proposed method under various layer combination and detector threshold

Using various layer combinations and detector threshold, the class-specific recognition accuracy for our method is shown in Figure 31. The complete recognition accuracy for our method using different parameters is also shown in Table 11. When we add more layers into the layer combination, the recognition accuracy increases and the maximum value achieved is at Level2. If we keep increasing layers, the accuracy starts decreasing sharply. This happens because the highly-fine partition increases the chances of adding more noise as well as higher values of feature dimensions. If *T* is increased the results do not show any proper trend for that. So from the experiments, it is observed that Level2 and T=0.3 are the selected parameters for our proposed method.

Table 11: Complete recognition accuracy of our proposed method for different parameters.

	T=0.1	T=0.2	T=0.3	T=0.4
Level0	0.688	0.697	0.687	0.704
Level1	0.722	0.723	0.747	0.743
Level2	0.877	0.866	**0.881**	0.855
Level3	0.664	0.64	0.658	0.668
Level4	0.622	0.623	0.593	0.639

4.2.2. Evaluation on different methods for Feature Extraction

Using different methods or schemes of feature extraction, we will evaluate the performance of our proposed method. Instead of using mid-level retrieved from object-based low-level features, we will use three other different kinds of features: LBP, Gabor and saliency map. The class-specific recognition accuracy achieved using four different methods on different categories of samples is shown in Figure 32. In comparison to three mentioned methods, our method achieved better performance. While our method has shown accuracy of 88.1%, other methods' accuracy was quite lower (45.3% by LBP, 21.4% by Gabor and 41.9% by Saliency). So, it is observed that our novel BoW model for mid-level feature building has shown better performance indoor scene recognition.

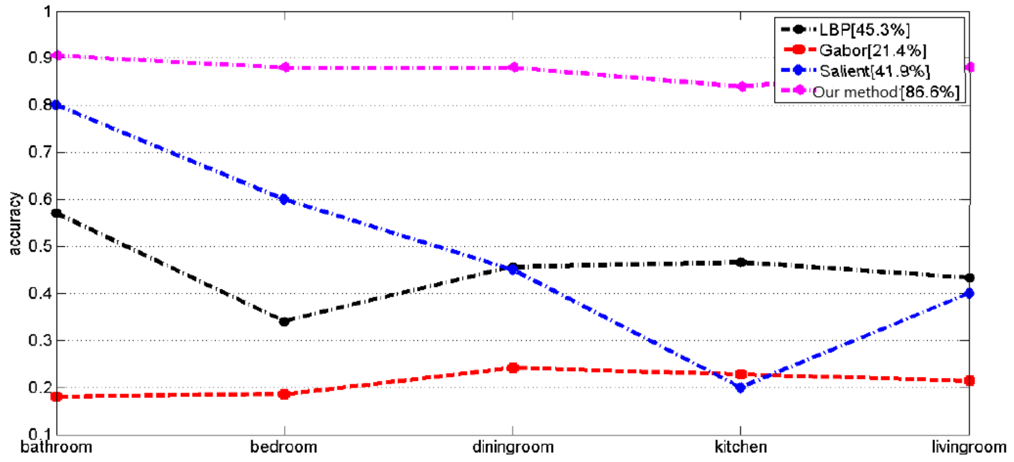


Figure 32: The class-specific recognition accuracy achieved using four different methods on different categories of samples

The confusion matrices are created for the four methods is shown in Figure 33. While the Gabor method barely recognised indoor scene images, LBP and saliency maps were more selective than Gabor. They performed better for most indoor scene images. Our method attained top accuracy in all four schemes for all the categories of indoor scene images.

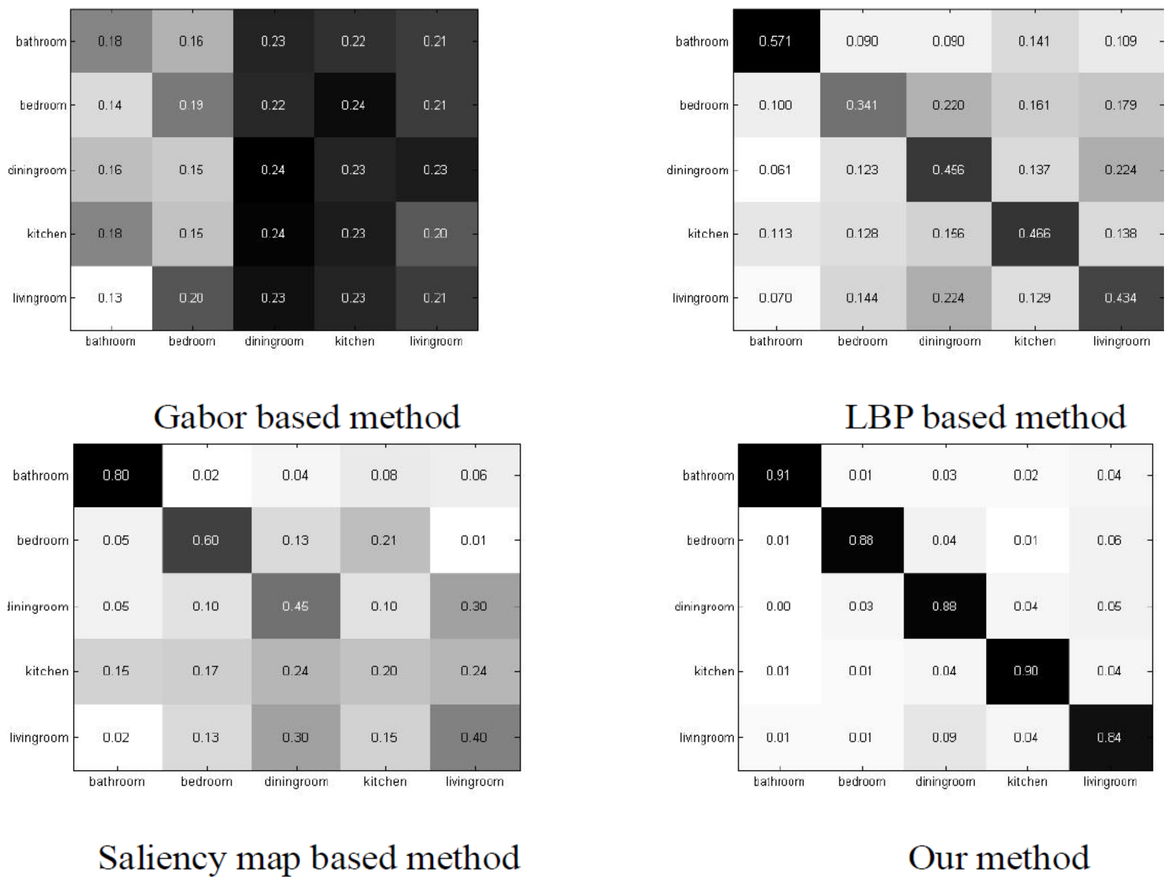


Figure 33: Comparison of our method with 3 other methods using confusion matrix for feature extraction based scene recognition

4.2.3. Evaluation on different classifiers

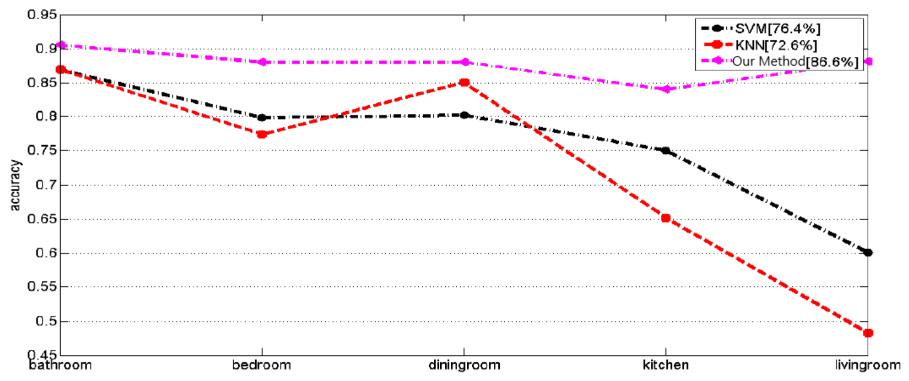


Figure 34: The class-specific recognition accuracy of our method in comparison to other classifier

By comparing our method of SRC classifier with different classifiers of SVM and KNN, we will evaluate the performance of our method for feature extraction. The comparison of class-specific recognition accuracy and confusion matrices of these three classifiers are shown in Figure 34 and 35. Our proposed method of SRC is better than SVM and KNN as the total accuracy achieved by three classifiers are: 88.1%, 76.4% and 72.6% respectively. It is proved from these results that our method of sparse representation is more efficient and effective for indoor scene recognition. In comparison to SVM and

KNN, it has been proved that SRC is more effective towards partial or full occlusion, changes in illumination and pose variation. In confusion matrices, SRC easily achieved better accuracy of 84% in comparison to SVM (60%) and KNN (46%). This showed that in even similar image samples from categories like living room and dining room, SRC is very accurate in recognizing scene images.

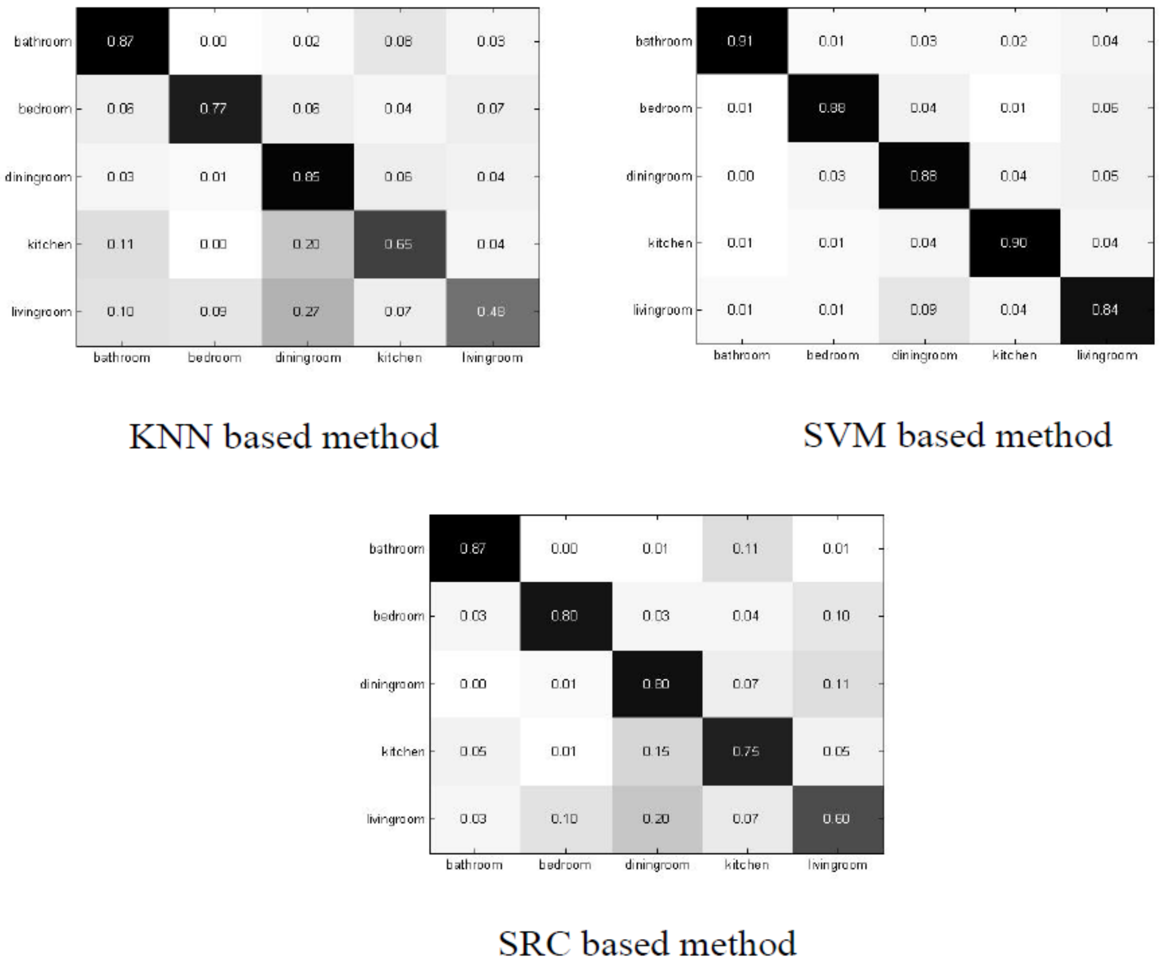


Figure 35: Comparison of confusion matrices of our method of SRC to different classifiers: SVM and KNN

4.3. Visual SLAM

In Figure 36 below, two screenshots from the KITTI00 sequence have been shown. It consists of the grid map and its navigation result in Rviz.

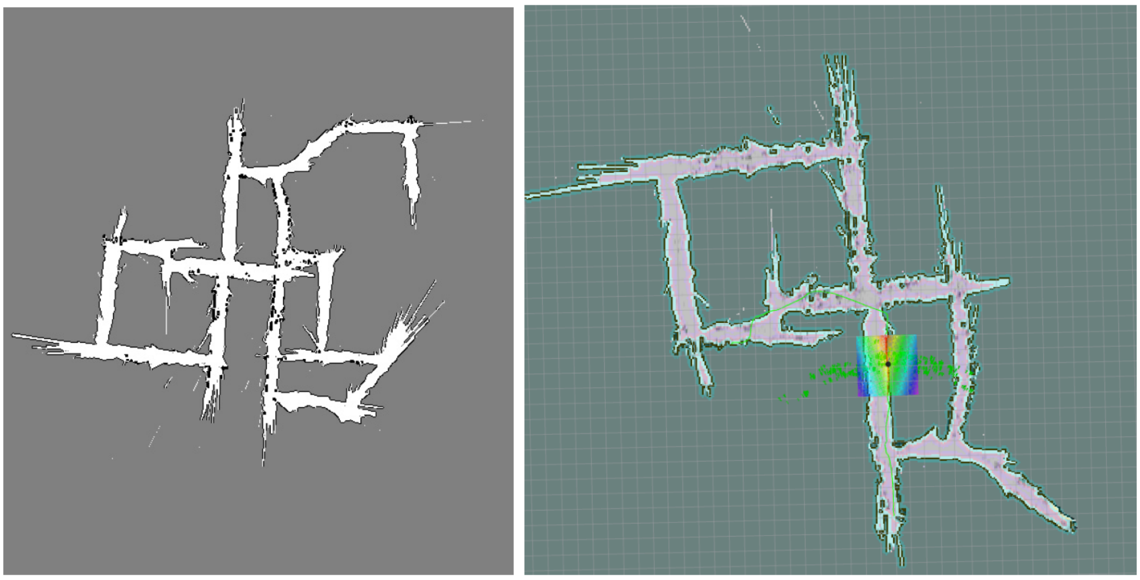
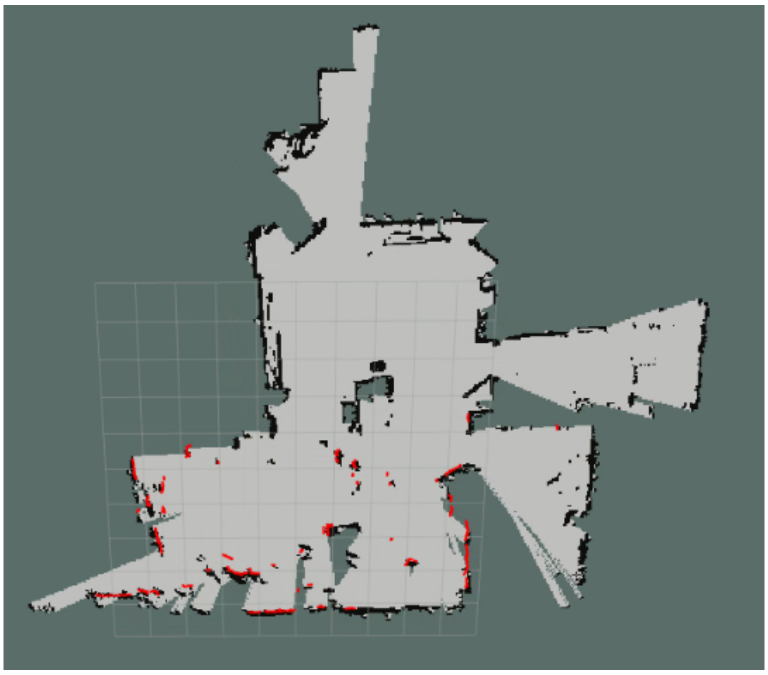


Figure 36: 2D Occupancy grid map and its navigation result for KITTI00

In Figure 37, we have shown a 2D occupancy grid map and navigation results for our sequence.



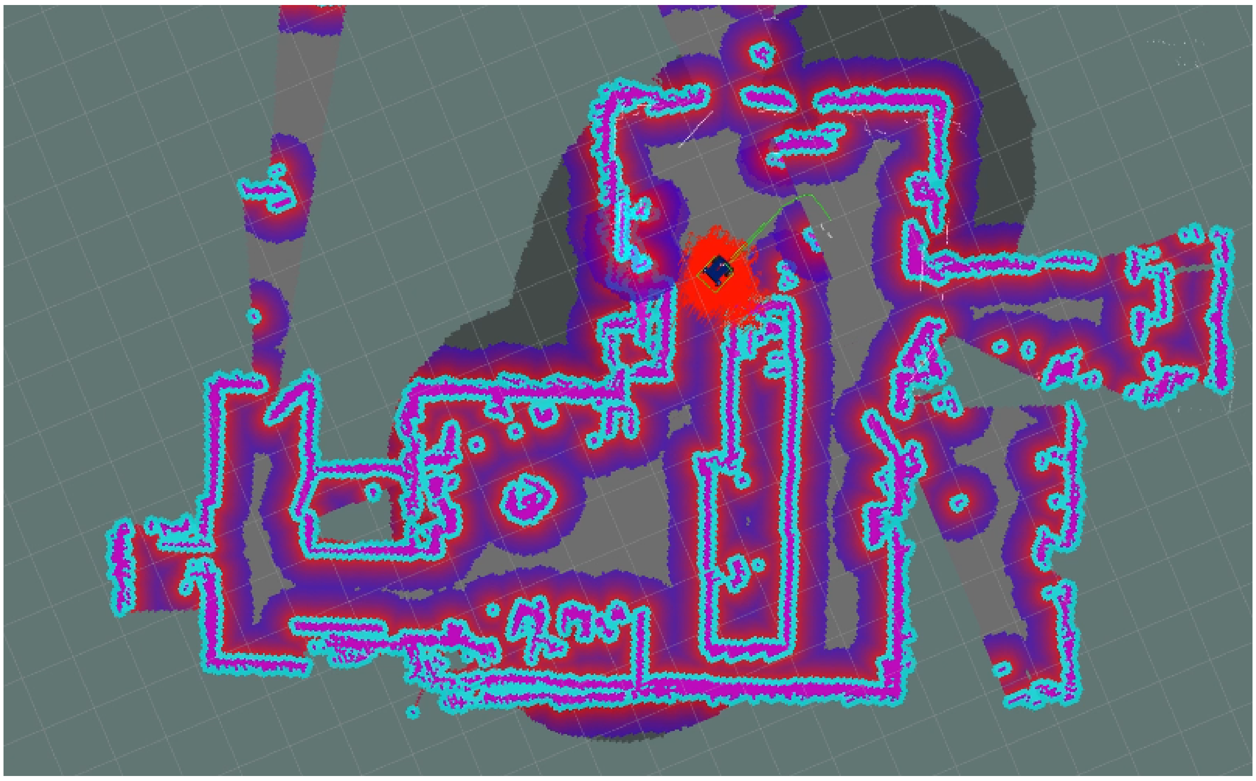


Figure 37: 2D occupancy grid and navigation result for our sequence

4.4. 3D-SLAM

Using *hdl_graph_slam*, we performed 3D SLAM and created a 3D occupancy grid map (as shown in Figure 38), which we can use for navigation also.

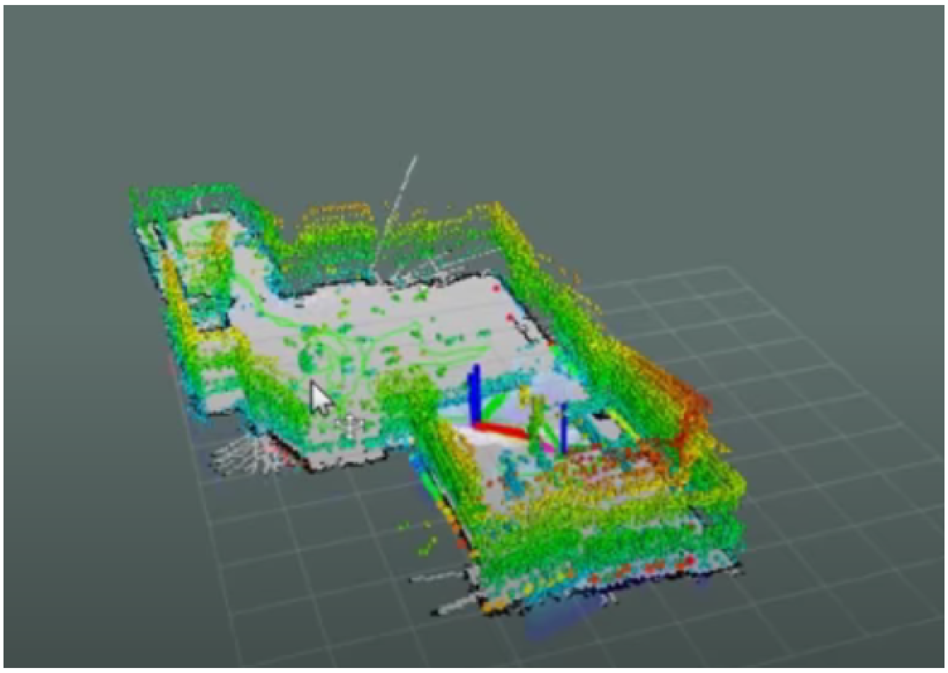


Figure 38: 3D SLAM generated occupancy grid map

5. Conclusions

We built a service robot incorporated with different sensors to perform many different purposes for indoor environments. We described Target detection strategy to solve the problem of real-time sequence. We described indoor scene classification to perform scene recognition to solve the problem of difficulties to recognize indoor scenes. We performed Visual-SLAM using ORB-SLAM so we can accumulate 3D point clouds to generate 2D occupancy grid maps. Finally we performed 3D-LiDAR based SLAM to solve the issue of mobile robots to navigate through a 3D environment by creating a three dimensional map using 3D point clouds. This concludes that an efficient service robot is created with many capabilities to perform inside indoor environments.

References

[1] K. Cresswell, S. Cunningham-Burley and A. Sheikh, "Health Care Robotics: Qualitative Exploration of Key Challenges and Future Directions", *Journal of Medical Internet Research*, vol. 20, no. 7, p. e10410, 2018. Available: 10.2196/10410.

[2] “Robotics in Healthcare: The Future of Medical Care – Intel", *Intel*, 2021. [Online]. Available: https://www.intel.com/content/www/us/en/healthcare-it/robotics-in-healthcare.html. [Accessed: 24- May- 2021].

[3] "How are Robots Changing Healthcare? - Healthcare Administration Degree Programs", *Healthcare Administration Degree Programs*, 2021. [Online]. Available: https://www.healthcare-administration-degree.net/faq/how-are-robots-changing-healthcare/. [Accessed: 24- May- 2021].

[4] "Benefits of Robotics in Healthcare: Tasks Medical Robots Will Undertake", *The Medical Futurist*, 2021. [Online]. Available: https://medicalfuturist.com/robotics-healthcare/. [Accessed: 24- May- 2021].

[5] J. Brownlee, "A Gentle Introduction to Object Recognition With Deep Learning", *Machine Learning Mastery*, 2021. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/. [Accessed: 24- May- 2021].

[6] "Object Detection Guide | Fritz AI", *Fritz.ai*, 2021. [Online]. Available: https://www.fritz.ai/object-detection/. [Accessed: 24- May- 2021].

[7] "Manal El Aidouni", *Manal El Aidouni*, 2021. [Online]. Available: https://manalelaidouni.github.io/manalelaidouni.github.io/Understanding%20YOLO%20and%20YOLOv2.html . [Accessed: 24- May- 2021].

[8] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142-158, 2016. Available: 10.1109/tpami.2015.2437384.

[9] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904-1916, 2015. Available: 10.1109/tpami.2015.2389824.

[10] R. Girshick, "Fast R-CNN", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1504.08083. [Accessed: 24- May- 2021].

[11] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1506.01497. [Accessed: 24- May- 2021].

[12] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1506.02640. [Accessed: 24- May- 2021].

[13] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1612.08242. [Accessed: 24- May- 2021].

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1804.02767. [Accessed: 24- May- 2021].

[15] "Scene Classification Using Deep Learning", *Medium*, 2021. [Online]. Available: https://medium.com/mathworks/scene-classification-using-deep-learning-853c64318f6b#:~:text=Automatic%20scene%20classification%20. [Accessed: 24- May- 2021]

[16] "Indoor scene recognition based on deep learning and sparse representation", *Doi.org*, 2017. [Online]. Available: https://doi.org/10.1109/fskd.2017.8393385. [Accessed: 24- May- 2021].

[17] "Recognizing indoor scenes", *Doi.org*, 2009. [Online]. Available: https://doi.org/10.1109/cvpr.2009.5206537. [Accessed: 24- May- 2021].

[18] "A Discriminative Representation of Convolutional Features for Indoor Scene Recognition", *Doi.org*, 2016. [Online]. Available: https://doi.org/10.1109/tip.2016.2567076. [Accessed: 24- May- 2021].

[19] "Orientational Pyramid Matching for Recognizing Indoor Scenes", *Doi.org*, 2014. [Online]. Available: https://doi.org/10.1109/cvpr.2014.477. [Accessed: 24- May- 2021].

[20] "Indoor Scene Recognition Based on the Weighting Spatial Information Fusion", *Doi.org*, 2012. [Online]. Available: https://doi.org/10.1109/isdea.2012.564. [Accessed: 24- May- 2021].

[21] L. Li, H. Su, Y. Lim and L. Fei-Fei, "Object Bank: An Object-Level Image Representation for High-Level Visual Recognition", 2013. .https://doi.org/10.1007/s11263-013-0660-x

[22] "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories", *Doi.org*, 2006. [Online]. Available: https://doi.org/10.1109/cvpr.2006.68. [Accessed: 24- May- 2021].

[23] "Robust Face Recognition via Sparse Representation", *Doi.org*, 2008. [Online]. Available: https://doi.org/10.1109/tpami.2008.79. [Accessed: 24- May- 2021].

[24] "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age", *Doi.org*, 2016. [Online]. Available: https://doi.org/10.1109/tro.2016.2624754. [Accessed: 24- May- 2021].

[25] *Ai.stanford.edu*, 2009. [Online]. Available: http://ai.stanford.edu/~mquigley/papers/icra2009-ros.pdf. [Accessed: 24- May- 2021].

[26] "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", *Doi.org*, 2015. [Online]. Available: https://doi.org/10.1109/tro.2015.2463671. [Accessed: 24- May- 2021].

[27] "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/tro.2017.2705103. [Accessed: 24- May- 2021].

[28] "Using occupancy grids for mobile robot perception and navigation", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/2.30720. [Accessed: 24- May- 2021].

[29] "FLAT2D: Fast localization from approximate transformation into 2D", *Doi.org*, 2016. [Online]. Available: https://doi.org/10.1109/iros.2016.7759305. [Accessed: 24- May- 2021].

[30] "raulmur/ORB_SLAM2", *GitHub*, 2021. [Online]. Available: https://github.com/raulmur/ORB_SLAM2. [Accessed: 24- May- 2021].

[31] "camera_calibration/Tutorials/MonocularCalibration - ROS Wiki", *Wiki.ros.org*, 2021. [Online]. Available: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration. [Accessed: 24- May- 2021].

[32] "Are we ready for autonomous driving? The KITTI vision benchmark suite", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/cvpr.2012.6248074. [Accessed: 24- May- 2021].

[33] "A benchmark for the evaluation of RGB-D SLAM systems", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/iros.2012.6385773. [Accessed: 24- May- 2021].

[34] "The Bresenham Line-Drawing Algorithm", *Cs.helsinki.fi*, 2021. [Online]. Available: https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html. [Accessed: 24- May- 2021].

[35] "gazebo - ROS Wiki", *Wiki.ros.org*, 2021. [Online]. Available: http://wiki.ros.org/gazebo. [Accessed: 24- May- 2021].

[36] "amcl - ROS Wiki", *Wiki.ros.org*, 2021. [Online]. Available: http://wiki.ros.org/amcl. [Accessed: 24- May- 2021].

[37] "rviz - ROS Wiki", *Wiki.ros.org*, 2021. [Online]. Available: http://wiki.ros.org/rviz. [Accessed: 24- May- 2021].

[38] "A solution to the simultaneous localization and map building (SLAM) problem", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/70.938381. [Accessed: 24- May- 2021].

[39] "Constructing maps for indoor navigation of a mobile robot by using an active 3D range imaging device", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/iros.1990.262380. [Accessed: 24- May- 2021].

[40] "New technologies and applications in robotics | Communications of the ACM", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1145/175247.175253. [Accessed: 24- May- 2021].

[41] J. Jung, S. Yoon, S. Ju and J. Heo, "Development of Kinematic 3D Laser Scanning System for Indoor Mapping and As-Built BIM Using Constrained SLAM", 2021. .

[42] *Int-arch-photogramm-remote-sens-spatial-inf-sci.net*, 2021. [Online]. Available: http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W3/543/2017/isprs-archives-XLII-2-W3-543-2017.pdf. [Accessed: 24- May- 2021].

[43] "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/roma.2016.7847825. [Accessed: 24- May- 2021].

[44] V. Lehtola et al., "Comparison of the selected state-of-the-art 3D indoor scanning and point cloud generation methods", *Mendeley*, 2021. [Online]. Available: https://www.mendeley.com/catalogue/06ab8db8-4e0f-3693-b8eb-0210593cca74/. [Accessed: 24- May- 2021].

[45] "MSGD: Scalable back-end for indoor magnetic field-based GraphSLAM", *Doi.org*, 2021. [Online]. Available: https://doi.org/10.1109/icra.2017.7989444. [Accessed: 24- May- 2021].

[46] 2021. [Online]. Available: http://wiki.ros.org/Bags/Format. [Accessed: 24- May- 2021].