UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

INDOOR LOCALIZATION AND MAPPING USING DEEP LEARNING NETWORKS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

RAVI SONI
Norman, Oklahoma
2017

INDOOR LOCALIZATION AND MAPPING USING DEEP LEARNING NETWORKS

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

_____
Dr. Sesh Commuri, Chair

_____
Dr. Andrew Fagg

_____
Dr. Choon Yik Tang

Acknowledgements

First, I would like to express my special appreciation to my advisor Dr. Sesh Commuri. He has always motivated, supported, and guided me during my graduate study. This thesis would not be possible without all his help. I specially thank him for not only being my supervisor, but also for being my mentor and for giving me various opportunities to learn from him. My gratitude goes to Dr. Andrew Fagg and Dr. Choon Yik Tang for serving on my master's committee and advising me at all stages of my graduate study. I thank Dr. Fagg for sharing his knowledge and experience in multidisciplinary research areas. My appreciation also goes to Dr. Tang for advising and mentoring me. I believe I am a better researcher and engineer because of the experience I gained.

I would like to thank my parents, my father Mr. Vipul Soni, and my mother Mrs. Jagruti Soni. Their unconditional love and confidence in me helped me stay focused and motivated during my graduate study. Finally, a special thanks to all my friends at OU and in India for their precious support, company, and friendship in my life. At the end of my academic journey as a master's student, I would like to pay my deepest respect and gratitude to all my teachers and mentors throughout my academic life.

Table of Contents

List of Tables

List of Figures

Abstract

Over the past several decades, robots have been used extensively in environments that pose high risk to human operators and in jobs that are repetitive and monotonous. In recent years, robot autonomy has been exploited to extend their use in several non-trivial tasks such as space exploration, underwater exploration, and investigating hazardous environments. Such tasks require robots to function in unstructured environments that can change dynamically. Successful use of robots in these tasks requires them to be able to determine their precise location, obtain maps and other information about their environment, navigate autonomously, and operate intelligently in the unknown environment. The process of determining the location of the robot and generating a map of its environment has been termed in the literature as Simultaneous Localization and Mapping (SLAM). Light Detection and Ranging (LiDAR), Sound Navigation and Ranging (SONAR) sensors, and depth cameras are typically used to generate a representation of the environment during the SLAM process. However, the real-time localization and generation of map information are still challenging tasks. Therefore, there is a need for techniques to speed up the approximate localization and mapping process while using fewer computational resources. This thesis presents an alternative method based on deep learning and computer vision algorithms for generating approximate localization information for mobile robots. This approach has been investigated to obtain approximate localization information captured by monocular cameras. Approximate localization can subsequently be used to develop coarse maps where a priori information is not available. Experiments were conducted to verify the ability of the proposed technique to determine the approximate location of the robot. The

approximate location of the robot was qualitatively denoted in terms of its location in a building, a floor of the building, and interior corridors. ArUco markers were used to determine the quantitative location of the robot. The use of this approximate location of the robot in determining the location of key features in the vicinity of the robot was also studied. The results of the research reported in this thesis demonstrate that low cost, low resolution techniques can be used in conjunction with deep learning techniques to obtain approximate localization of an autonomous robot. Further such approximate information can be used to determine coarse position information of key features in the vicinity. It is anticipated that this approach can be subsequently extended to develop low-resolution maps of the environment that are suitable for autonomous navigation of robots.

Chapter 1: Introduction

1.1 Motivation and Problem Statement

One of the important abilities for a mobile robot is the ability to navigate autonomously through regions of interest and carry out a variety of tasks. Such capability requires the robot to be able to determine its current location and then navigate to a desired location using maps of the environment. In unknown or partially known environments, where accurate information about the surrounding objects and free paths for navigation are not available, the robot has to be able to develop detailed maps of its surroundings and then use that information to navigate to a specified location. Given an environment for the robot to traverse, finding open space and a path to navigate through are two significant challenges in robot mobility. These problems have traditionally been discussed in the literature as 'Findspace' and 'Findpath' problems [25]. Early research on 'Findspace' and 'Findpath' can be traced back to a paper by Lozano-Pérez et al. titled 'Spatial planning: A configuration space approach' [25]. Later, researchers started exploring various other ways to find optimal trajectories for autonomous robots to navigate in an environment [40-45]. These approaches assume that detailed information about the environment is available and that the robot has precise knowledge of its location in the environment. However, tasks such as search and rescue often require the robot to operate in completely unknown environments, where such information is not available or the information is changing in a dynamic manner.

In the past, extensive work has been carried out for planning and deploying autonomous robots for various tasks [35-45]. Approaches utilizing virtual force fields, artificial potential fields, and computer vision were used to help the robot navigate

through environments. In the virtual force field method, obstacles are assumed to have fields attached to them that causes a virtual repulsive force to act on the robot, while the target has virtual attractive force that attracts the robot to the goal [26]. This method allows the robot to avoid the densely cluttered and unexpected obstacles autonomously, while navigating towards the goal. The artificial potential field was first proposed by Khatib et al. [27]. Repulsive potential fields are assigned around obstacles which need to be avoided, and attractive potential field is assign to the target. The potential energy at any location in the environment is inversely proportional to the square of the distance from the source of the field. The robot navigates along the lines of force in the direction of decreasing potential energy, and is termed as a gradient descent search method. Both of the above methods were used with an assumption that the initial position and orientation of a robot, location of the goal, and a set of obstacles located in workspace are known. Vision based approaches are therefore being proposed in the literature to address this shortcoming.

Horswill et al. were one of the first researchers to propose a vision-based behavior to help a robot traverse through an obstacle-filled environment [28]. Vision based approaches were also refined to develop maps of the environment in real-time to enable the robot to navigate to a target location. Once the map is formed, the robot can find an obstacle-free path to the target. For example, rovers used for exploration of Mars can build a map to navigate autonomously in a highly unknown environment. There are various methods to create a map and determine the location of the robot. Simultaneous Localization and Mapping (SLAM) is a popular approach to build a map in real-time by keeping the track of the position of the robot from the reference location that utilizes

sensors such as Light Detection and Ranging (LiDAR) sensors, Sound Navigation and Ranging (SONAR) sensors, and depth cameras [48]. In this approach, new spatial information is obtained as the robot traverses the environment and this information is used to build the map in an incremental manner. This method is discussed in greater detail in Chapter 2 of this thesis.

SLAM finds applications in many scenarios in which a prior map is not available and needs to be developed. The SLAM community has made significant progress over the last 30 years [29]. According to Cadena et al. [29], the first twenty years of the SLAM problem can be termed as the classical age (1986 – 2004), and the subsequent period is known as the algorithmic-analysis age (2004-2015). Probabilistic methods for SLAM was introduced at the IEEE Robotics and Automation Conference in 1986. Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte published two seminal papers on probabilistic SLAM after their discussion in that conference. The systematic historical review of the classical age of SLAM problem was covered by Durrant-Whyte and Bailey et al. in [29, 30]. The significant contribution of the SLAM community during the classical age (1986-2004) was the probabilistic foundations for SLAM, including approaches based on Extended Kalman Filter, Particle Filters, and Maximum Likelihood Estimation. The algorithmic-analysis age saw the study of fundamental properties of SLAM like convergence and consistency. Some of the SLAM techniques can increase the computational cost and can be slow depending on the complexity of the environment. Some of the prominent SLAM techniques that use advanced sensors such as Light Detecting and Ranging (LiDAR) sensor do not perform well in the presence of the glass walls in the environment, whereas other techniques using depth cameras have limitations

when used outdoors [61]. Moreover, depth cameras require significant computational resources and can slow down the real-time mapping and navigation process [61]. For the time-constrained applications such as search and rescue, these techniques may not be adequate. This thesis presents an alternative method based on deep learning and computer vision algorithms for obtaining approximate localization information in an environment.

1.2 Scope of this Thesis

This thesis aims to collect indoor images and use them to train a machine learning model to augment localization and map information in an unknown or partially known environment. The location of the robot is determined both qualitatively as well as quantitatively. The qualitative location is denoted in terms of the location in a building, floor of the building, and interior corridor. The quantitative location is reported in terms of cartesian coordinates of a reference frame. The approximate location is then used to compute the location of key features in the environment of the robot as a first step towards developing a coarse map of the environment. The following are the key steps in the process adopted for this thesis:

- Three different types of indoor scenes, namely buildings, floors, and corridors, are considered for the localization problem. Images from different floors, corridors in buildings are collected to create an initial dataset. Images taken at different time and under different lighting conditions are used to examine the robustness of the process.
- The dataset images are then divided into three parts: a training set, validation set, and test set. The training set is used to develop a machine learning model using high computing GPU nodes and this model is later validated and tested. These

images are then grouped to avoid overfitting of the model during training of the classifiers. For example, images, that are taken on days - one, two, and four, could be a part of training set; while, images taken on days - three and five, could be a part of validation and testing set respectively. There is a high probability that two consecutive frames of a video could be a part of the training and as well as the validation set if they are chosen randomly. This can result in overfitting of the machine learning models and can be avoided by appropriately grouping of the images.

- A convolutional neural network (CNN) is implemented and trained using the datasets developed in the previous step.
- The trained models will be evaluated based on the accuracy that is achieved. These models are trained using Amazon Web Services (AWS) and FloydHub virtual GPU nodes.
- An implementation of convolutional neural network will be evaluated to address its suitability to the rapid localization problem.
- Furthermore, the use of planar markers to improve the speed of the localization process will be investigated.

This thesis is presented in three parts. Different types of localization and mapping techniques, their advantages and disadvantages, and proposed improvement are addressed in the first part. The training and evaluation of a state-of-the-art of machine learning model is presented in the second part of the thesis. In the third part, the use of planar markers as landmarks/features for localizing in the environment will be considered.

1.3 Anticipated Contributions of this Thesis

One of the primary goals of this thesis is to simplify localization by using a machine learning model and a computer vision algorithm. It is anticipated that the technique proposed in this thesis will provide both qualitative and quantitative localization information of the robot. Thereafter, this information can be used to develop coarse maps where a priori information of the environment is not available. The low-resolution map can reduce the computational complexity for some tasks where low level details of surroundings are sufficient for navigation. As result, a robot can produce map representations that are flexible, and whose complexity can vary depending on the task at hand.

1.4 Organization of this Thesis

This thesis is organized as follows.

- Results from literature on localization and mapping techniques using visual and non-visual sensors are discussed, and their application to robot navigation is presented in Chapter 2.
- Convolutional neural network, a popular deep learning algorithm for visual recognition, is considered and its application to localization is presented in Chapter 3.
- Experimental results obtained using low cost low resolution monocular cameras for localization are also discussed in Chapter 4
- Conclusions and scope of the future work are covered in Chapter 5.

Chapter 2: Localization and Mapping for Indoor Autonomous Robots

2.1 Localization and Mapping Techniques

Mapping is the process of creating symbolic representations of significant features in an environment. Mobile robots can use this representation of an environment to plan, navigate and/or carry out specific tasks. These representations or maps are needed when accurate absolute position is needed. Precise location information of a robot on a map is needed to be tracked when the robot is tasked to go to the coordinates of a specific place on a map and this is known as Localization [46]. A solution to modeling an environment and tracking the location problems is to localize and map the unknown environment simultaneously. The primary objective of the Simultaneous Localization and Mapping (SLAM) is to localize the robot from an initial location and start exploring unknown environments from that reference location in pursuit of an accurate map generation. In this manner, an autonomous robot can create maps of the unknown environment through indirect interactions with its surrounding objects. Full and Online SLAM are two main forms of SLAM [52]. Full SLAM involves estimating the entire path; while Online SLAM seeks to recover present robot location, instead of the entire path [52]. Online SLAM algorithms are usually incremental and process one data item at a time. In literature, the online SLAM methods are typically implemented using filtering techniques such as Extended Kalman Filter, Particle Filter. Researchers have also classified SLAM based on the method used, for example Feature-based vs. Volumetric SLAM, Topological vs. Geometric SLAM, Static vs. Dynamic SLAM, Single-robot vs. Multi-robot SLAM, and Active vs. Passive SLAM. Brief explanation of each method is given in the next paragraph.

In Volumetric SLAM, the environment is sampled at a high resolution to allow for photo-realistic representation of surroundings [57]. Feature-based SLAM extracts sparse features or landmarks from the sensor data stream [56]. Feature-based SLAM methods tend to be efficient, but the spatial representation that is obtained may be poorer than volumetric SLAM as extraction of feature discards other useful information from the measurement [52]. Topological SLAM represents the qualitative description of the environment, and in addition to spatial information, depicts the relationship between primary features [58]. Geometric SLAM considers the quantitative description of the environment. For example, the distance between two points in a map could be in units and feet when using Topological SLAM and Geometric SLAM respectively [52]. Static environment based SLAM algorithms assume that the environment does not change over time, while dynamic SLAM assumes that changes may occur in the environment. The single-robot SLAM method considers using only one robot to construct the map of the environment. The multi-robot SLAM method considers using more than one robot to construct the SLAM map. The multi-robot SLAM method is quick but coordinating the measurements between robots is a difficult task. The multi-robot based SLAM method finds uses in large and complex spaces [59]. In passive SLAM method, a human operator or some other entity controls the robot during the map generation process. Once the map is generated, the robot can navigate autonomously. In active SLAM method, the robot controls its movement and actively explores the environment independently.

In a broad sense, the various solutions to the SLAM problem can be categorized as follows: Feature-based SLAM, Pose-based SLAM, and Appearance-based SLAM. The feature-based solution is the most popular approach to the SLAM problem [31]. It

searches for a predefined feature or landmark in an environment to estimate the current robot pose with respect to this feature or a landmark. Based on the types of sensor, various filtering techniques are used in conjunction with SLAM. As reported in Zamora et al. (2013), the estimated state in EKF-SLAM includes the position and orientation of the robot and the location of all landmarks in the environment. However, the biggest shortcoming of EKF-SLAM is that the computational time that is required is a function of the square of the number of features [1]. The estimation accuracy of Graph SLAM is better than other SLAM methods, but comes at an increased computational cost and runs slower in real time. Furthermore, Zamora et al. show that while the PF SLAM is very efficient for nonlinear and non-Gaussian systems, the computational cost is prohibitive and the results are inconsistent when used in long-term autonomous applications [1].

One of the popular techniques to address the SLAM problem models the poses of a robot as nodes in a graph and the spatial information gleaned from the sensors/observations are used to represent the edges. Graph based techniques are then used to determine paths without prior knowledge of the maps, thereby making the techniques suitable for navigation. However, the complexity of the problem increases as the number of nodes in the graph increase.

The last method, appearance-based system, uses the appearance information of the environment to localize and map the world. This information can be divided further into visual and spatial information of the environment. The visual appearance method augments the previous metric-based method by detecting the loop closure and previously visited places. While appearance-based, as opposed to feature-based, methods are able to close loops between day and night sequences or between different seasons, the resulting

loop closure is topological in nature [31]. The spatial appearance based method uses range finders (non-visual sensors) to identify the location and the loop closure. Loop closure is the problem of recognizing a previously visited location by a robot. Current approaches rely heavily on vehicle pose estimates to prompt loop closure [62]. The unclear loop closure is one of the major issues when applying this method to real-world applications. In next subsections, SLAM methods using visual and non-visual sensors are explained. The focus of the thesis is estimating the approximate location and recognizing the scene through monocular images. Later, the generated map will be used to augment high resolution map of the environment.

2.1.1 Localization and Mapping using Non-Visual Sensors

Exploring an unknown environment is important as the autonomous robot should be able to execute its tasks in a safe manner. There are several sensor technologies such as range sensors, absolute position sensors, environmental sensors, and inertial sensors that can be used detect objects and structure of an environment. With the use of these sensors, a robot can perform path planning and navigate its surroundings without altering or disturbing the environmental objects. An example of range sensor (LiDAR) is shown in Figure 2.1 (a). This sensor emits a pulsed ray of light and measures the distance based on the time-of-flight of the reflected beam from the object. Global Positioning Systems (GPS) sensor is one example of absolute position sensors that returns the absolute position of the robot in terms of longitude and latitude. The inertial sensor returns differential properties of the robot's position and orientation, for example, acceleration. The inertial sensors are used to assist an autonomous robot to control its motion through sensor feedback. These sensors are known as non-visual sensors as they return real-valued data

to portray that entire environment, and they are different from the perceived scene. This problem of recovering the environment from the sensor data can be described as an inverse problem.

(a) (b)

Figure 2.1 (a) Hokuyo LiDAR (b) Mapped environment using LiDAR [33]

A variety of sensor properties such as the speed of operation, error rate, robustness, computational requirements, power, weight and size requirements can be used to select the right sensor for a given application. Based on the application, correct sensor technologies are chosen. For example, 2D LiDAR and depth cameras do not work well outdoors; because sunlight can inundate the sensor data stream and introduce errors as they also use light rays to estimate the distance. Heavy sensors, like RGB depth cameras, are not useful for micro aerial vehicles. Therefore, there is a need for the usage of adequate sensors such as vision sensors that can sense and represent the environment better.

2.1.2 Localization and Mapping using Visual Sensors

Humans seem to navigate effortlessly with vision and odometry; it seems obvious to consider vision as a tool for navigation of autonomous robots. There are various types

of visual sensors that can be very helpful in solving the SLAM problem. As vision relates measurement to the scene structure, it seems obvious to place more emphasis on vision based sensors compared to non-vision based sensors. For example, the laser range sensors return the distance to the objects, and so the robot obtains a set of distance measurements as a function of the direction in which the laser sensor is pointed.

Many autonomous robots use a camera as one of the imaging sensors that can record its field of view. However, for a given scene, a single perspective camera is not sufficient to determine the three-dimensional measurements of objects. A camera provides a two-dimensional array of points with respect to the field of view; and this two-dimensional array can only provide the information of the direction and distance of the object from the camera. To get the full perspective and information about the third dimension, i.e. the depth of the point being measured, it is necessary to use stereo or trinocular cameras on a mobile robot. Now suppose the robot used a vision based sensor instead of a laser sensor, the perceived data is relatively easy to interpret for the human operator as can be seen in Figure 2.2. The stereo camera is being used to measure the relative depth information of the scene. This depth information can help an autonomous robot to avoid the static and dynamic obstacles in the environment effortlessly. Few drawbacks of using this stereo sensor are the amount of computation required and the high operational power requirements. Other sensors such as trinocular and omnidirectional cameras are also used to understand the environment in-detail as one of the problems with a standard monocular or stereo camera is the limited field of view that is available. Omnidirectional cameras resolve this issue and provide 360 degrees view of

an environment into a single image. However, they have similar drawbacks as stereo cameras.

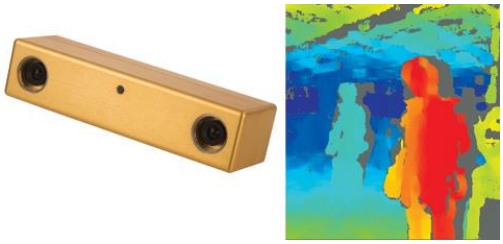


Figure 2.2 BumbleBee stereo camera and depth information [22]

Autonomous robots like CoBots, shown in Figure 2.3, are deployed for a long-term autonomy with multiple depth cameras, monocular cameras, non-visual sensors like range sensors mounted on them. This robot can be expensive to deploy commercially as these sensors cost much higher.

Figure 2.3 Autonomous Robots: CoBots, CMU [21]

In recent years, autonomous vehicles are being commercialized. This effort motivates the researchers to come up with better solutions for object detection, segmentation, localization, and approximate the relative distances using low cost monocular or stereo cameras information. Figure 2.4 shows one latest effort in this research direction. The algorithm can localize the movable and relevant objects using a

single monocular image. Segmented objects based localization and planning can be very useful for outdoor robots to navigate using a RGB image sequence. Successful results of that work motivate the use of a monocular camera for localization and mapping in this work. This thesis is focused on using monocular image and state-of-the-art computer vision and machine learning algorithms to obtain approximate location information.

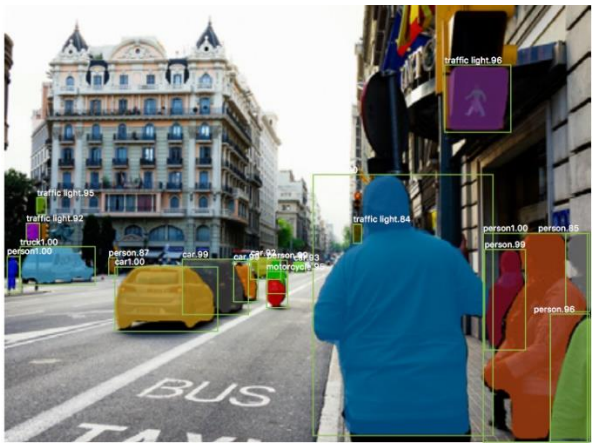


Figure 2.4 Object segmentation using single monocular image [20]

2.2 Related Work

Work has been done as for many years towards the goal of extracting the information about the environment and storing in best possible way so that systems can operate autonomously. Biswas et al. proposed localization and navigation algorithms of the CoBots for long-term autonomy [21]. The CoBots interact with the real world every day and face many challenges as mentioned by Biswas. They use different types of visual and non-visual sensors to get the best estimate of the robot's state and state of the mapped environment. CoBots are deployed with very efficient sensors such as range sensors, ultrasonic sensors, high-precision wheel encoders, stereo and depth cameras. However, as mentioned [21], the laser rangefinders and depth cameras on the CoBots are rated only for indoor use and are not rated for operation in direct sunlight. In some parts of their

mapped environment, at some particular seasons and times of the day, the CoBots do encounter direct sunlight. The direct sunlight saturates the sensor measurements, causing them to return invalid range readings. They chose to err on the side of caution in this case, and when the CoBots detect invalid range or depth readings, they come to a stop and wait [21]. If the sensors do not recover from saturation within 5 minutes, the CoBots email the research group with the last estimated location for the human support. Moreover, due to high power requirements and size requirements, this robot can only be used to validate the research algorithms and cannot be deployed commercially as a service robot. Furthermore, glass corridors and heat radiators are invisible for these CoBots which are very common issues for all the autonomous robots. The glass walls are invisible for both laser ranger finders and depth cameras as depicted in Figure 2.5. The radiators being poorly reflective, are invisible for a depth camera but partially visible to laser range sensors.

Figure 2.5 CoBots between glass walls [34]

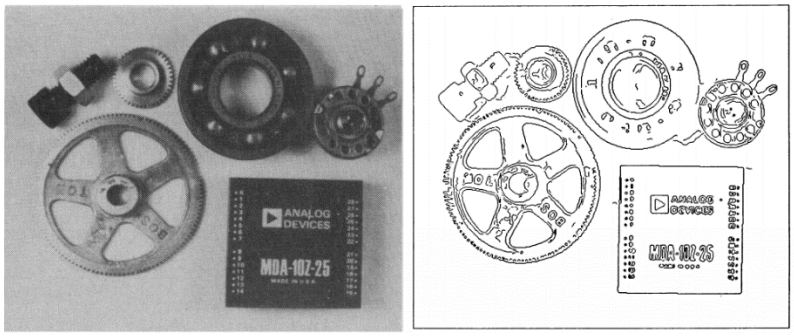
In these events, our proposed work can resolve both issues by passively interacting with these places which can determine approximate localization information.

Chen et al. describe door recognition algorithm for visual based robot navigation using the state-of-the-art machine learning technique in the paper [3]. They use this algorithm to detect the presence of doors in an indoor environment and identify five types of spatial relationship between the robot and the door [3]. Later, they used the presence of the door as the localization information to navigate the corridor. However, their model would fail in several scenarios like a fully opened door, no door corridor, transparent door, or occluded door. Therefore, this algorithm in which authors localize and navigate a robot by detecting an object like a door cannot be applied to all settings. For fully autonomous navigation systems, there is a necessity of map generation using similar imaging sensors used for the door detection. Using different type of sensors like non-visual and visual sensor while performing map building and navigation task can increase computational cost. The next chapter describes one state-of-the-art machine learning and computer vision algorithm that can be useful for monocular camera based localization, mapping, and navigation.

Chapter 3: Deep Convolutional Neural Networks for Indoor Scene Recognition

3.1 Introduction to Scene Recognition

Computer vision is an interdisciplinary field that aims to provide capabilities like the human visual system to a computer [50]. Computer vision is concerned with the extraction and understanding of useful information from an image or a sequence of images. This useful piece of information termed as a feature in the literature. These features could be a specific structure of an image like points, edges, regions, shapes, etc., or could be the color of an area within some specific boundaries. These features are used to develop object detection algorithms that recognize and localize objects in an image. For example, one of the feature detection algorithms is the edge detection algorithm [51] that is depicted in Figure 3.1. Figure 3.1 (a) shows some typical objects in the view of the robot and the edges of the objects detected by the algorithm are shown in Figure 3.1 (b).



(a) (b)

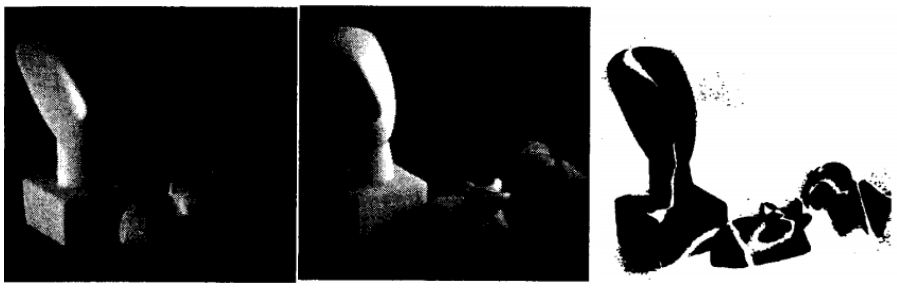
Figure 3.1 (a) Parts image (b) Canny edge detection [51]

Feature detection algorithms are also useful for scene recognition tasks. In real world images, objects are depicted in different contexts. For example, automobile showroom images and highway images contain cars but they are completely different

scenes. Therefore, recognition of the scene cannot be performed solely based on the objects the scene contains. In literature, a group of similar context images for a given application is known as a class. For the car model recognition task, Toyota Camry and Corolla are two different classes; whereas, for the car manufacturer recognition task, Toyota cars and Honda cars are two distinct categories (classes). Therefore, recognizing only objects in a scene cannot help in the classification. Furthermore, if the images are under all kinds of variations such as lighting, viewpoints, partial occlusion, then the recognition task can become significantly harder.

Scene recognition using computer vision is an open-ended and challenging problem. While there have been several technological advances in image capture, processing, and analysis, there are still many challenges to be addressed in vision-based navigation. One of those challenges is the extraction and understanding of scene and context information from images. Most context and scene understanding algorithms that work well for outdoor images perform very poorly in indoor conditions. Several researchers in computer vision have used images ranging from outdoor landscapes to indoor spaces to develop algorithms for scene recognition [5, 6, 7]. While the accuracy rate is acceptable for some outdoor space classification tasks, recognizing indoor spaces is still a challenging task due to similar image features across different classes and huge feature variation within a class [8]. The main issue is that while the high-level features can well characterize some indoor scenes (e.g. corridors, halls), others (e.g. library, laboratory, coffee-shop) are better described by the objects they contain. Furthermore, there still exists the same challenges like lighting condition, object orientation, object occlusion, camera parameters, etc. for both indoor and outdoor categories.

When a computer stores an image, it stores a 2D or 3D array of pixel values in memory. Grayscale image can be represented as a 2D array; while, color image can be represented as a 3D array. Each of the array element is given a value from 0 to 255 which describes the pixel intensity at that pixel location. One example of the effect of illumination can be seen in Figure 3.2. Figure 3.2 (a) and 3.2 (b) depict two frames from a sequence and Figure 3.2 (c) is the absolute difference between the frames (i.e., arrays). As seen in Figure 3.2 (c), there are significant image information changes in just two consecutive frames of the sequence for the same object which increases difficulty level of object detection or scene recognition task.



(a) (b) (c)

Figure 3.2 Two frames from sequence, (a) and (b), and their difference, (c) [8]

3.1.1 Traditional Computer Vision methods

In last decade, researchers had developed several computer vision models such as the Bag-of-Words (BoW), Spatial pyramid, Histogram of Oriented Gradients (HOG) for visual object/scene recognition problems [54]. The general block diagram of traditional computer vision based image classifier is depicted in Figure 3.3. The bag-of-words model represents an image as an unordered collection of local patches. These *patches* are treated as *words* in a document. The bag-of-words model is based on a codebook of visual words

which is a fixed set of visual features. The process generates a histogram of visual word (feature) occurrences out of an image. The object is classified using probabilistic models based on the occurrence of these features/patches of an image [53]. These histograms are used to train a classifier using machine learning algorithm such as SVM. One of the disadvantages of the bag-of-words model is that it disregards the spatial relationships among the image patches.

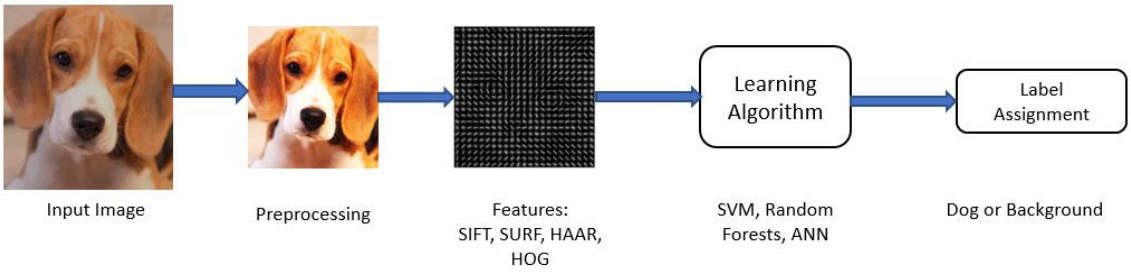


Figure 3.3 Traditional Image Classifier

Feature detection is a low-level image processing operation that is used to solve many computer vision tasks. The most popular feature descriptor is the Histogram of Oriented Gradients (HOG) feature descriptor [54]. In the first step, HOG algorithm extracts the HOG descriptors from the samples of positive image set of the object(s). In the second step, the algorithm extracts the HOG descriptors from the samples of negative image set of the object that does not contain the object to be detected. In practice, the size of the negative image set must be large compared to positive image set. In the third step, a machine learning based classifier such as the linear support vector machine algorithm is trained using both positive and negative image sets to detect the object. The support vector machine algorithm is a popular machine learning algorithm published by Vapnik et al. in 1995 [12]. The HOG can be seen as an extension of the scale-invariant feature

transform (SIFT) descriptor [55] from a circular interest region to a rectangular detection window. For a larger object detection dataset, HOG method needs to be performed for every object class in order to extract various features from each class [50].

After many years of intense research in this direction, researchers have developed various feature extraction and description algorithms. All the above methods use feature description algorithm such as SURF, SIFT, and HOG to extract unique features [50]. It is very challenging to hand-code these features for the problem of object classification, scene understanding, and activity prediction which use large image dataset [2]. Furthermore, if there is diversity within object or scene category, then it is almost impossible to work with hand-coded feature detection algorithms.

The computer vision algorithms such as HOG and SIFT have been used by many researchers for very complex computer vision tasks like ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [60]. In the ImageNet 2012 challenge, Alex et al. [4] came up with the machine learning algorithm that is known as a convolutional neural network (CNN) to classify 1.2 million images. Using high computing resources and the convolutional neural network algorithm, they were able to improve the accuracy of image recognition by over 10% in comparison with traditional computer vision techniques existing at that time. Soon then after, artificial neural network based methods found widespread use in complex computer vision tasks. The difference between the traditional algorithms and the CNN algorithm is the absence of hand-coded feature detection and descriptor block of Figure 3.3. Rather CNN learns the characteristics of the objects during the training (learning) phase of the classifier. AlexNet convolutional neural network architecture was based on LeCun et al. architecture which was published in 1998 as

Gradient-Based Learning Applied to Document Recognition [9] and is depicted in Figure 3.4. This neural network structure was used to recognize handwritten digits initially and was not widely used due to its dependence on high-speed computing power necessary for the algorithm to execute in real time. The input to the neural network is a handwritten digit image and the output is the predicted digit. One of the advantages of this architecture is that it considers the spatial relationships among the image patches that was a major issue in the BOW method. The intermediate layers are known as convolutional and pooling layers that are explained later in this Chapter 3. Due to developments of GPUs in last decade, the high computing power of modern day GPUs is now sufficient to train this convolutional neural network model. The convolutional neural network algorithm for scene recognition problems like buildings classification, floor classification, and corridor classification is considered in this thesis. To recognize the scene and location information using a trained convolutional neural network, images were collected at different time and locations which is covered in the subsequent section. A brief introduction on an artificial neural network is covered in Section 3.3.

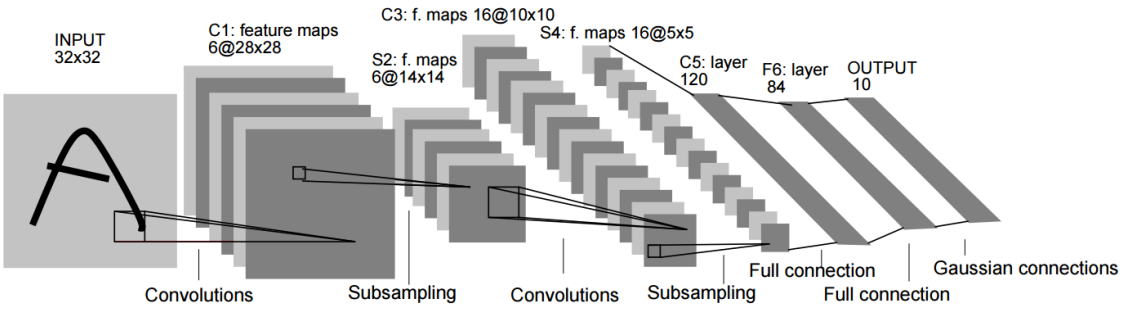


Figure 3.4 LeNet-5 Convolutional Neural Network Architecture [9]

3.2 Data Collection

Among the buildings on the University of Oklahoma (OU) main campus are Carson Engineering Center (CEC), Felgar Hall (FH), Devon Energy Hall (DH), Engineering Laboratory (EL), Sarkeys Energy Center (SEC), and Rawls Engineering Practice Facility (REPF). Figure 3.5 shows the map of the main buildings which we used in our experiments. Based on the maps of OU's buildings and corridors, we considered capturing images of corridors of DEH, FH, and CEC for our 1st experiment. In our second experiment, we considered capturing indoor pictures of five floors' of DEH. DEH is shown in Figure 3.6. In third experiment, we considered capturing images of corridors of fourth floor in DEH.

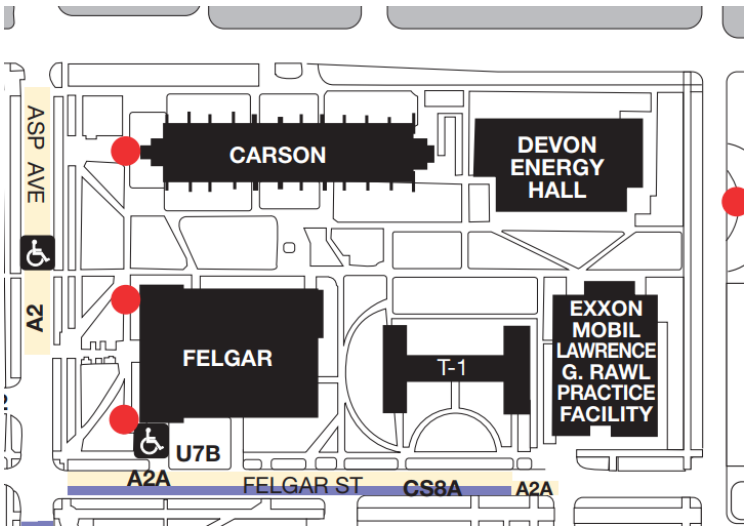


Figure 3.5 OU Engineering Main Campus Map

Figure 3.6 Devon Energy Hall, University of Oklahoma, Norman

We took videos using a commercial SONY digital still camera which has 30mm wide-angle lens and generates more stable, clear and high definition video data. Considering the intensity variance of ambient light inside the buildings, we captured several sets of data at different times of the day. The camera was hand held at about 1 meter above the ground to keep the camera's shooting angle at the mobile robot level. We also gave sufficient spatial variance to the viewing angle of the camera, to produce a detailed dataset regarding mobile robot's perspective. As shown in Figure 3.7, each row contains sample training images of each floor taken at a different time in Devon Energy Hall (DEH).

(a) First floor of DEH

(b) Second floor of DEH

(c) Third floor of DEH

(d) Fourth floor of DEH

(e) Fifth floor of DEH

Figure 3.7 Sample images of five floors of DEH

For the building-level data collection, the experimenter made several trips walking around each segment while shooting an indoor scene using a monocular camera. These multiple trips were made at different times to include all possible lighting conditions in the dataset. 32 videos are recorded for the first experiment. For the floor-level data collection, the experimenter walked in the centric position of each corridor, shooting the corridors as if the mobile robot would capture. 50 videos were recorded for the floor-level image classification problem. The images required for training a machine learning algorithm were extracted from those videos. The whole procedure includes four steps:

- Extract images from all frames of the video and resize to 240×240 pixels
- Remove low quality blurred images
- Split the dataset into two parts: 20% images from each category as validation dataset, and use the rest as training dataset
- One more walk at the same location for the test dataset batch.

3.3 Deep Convolutional Neural Network (DCNN)

3.3.1 Artificial Neural Networks

The human brain can be described as one complex neural network. Artificial neural networks are computing systems inspired by biological neural networks that represent a human brain. An important application of this artificial neural network is pattern recognition. A convolutional neural network (CNN) is type of artificial neural networks and has been used widely in classification tasks.

An *artificial neural network (ANN)* consists of simple processing nodes, the neurons, and weighted connections between those neurons as shown in Figure 3.8. The network can be further divided into multiple layers such as input layer, hidden layer(s), and output layer.

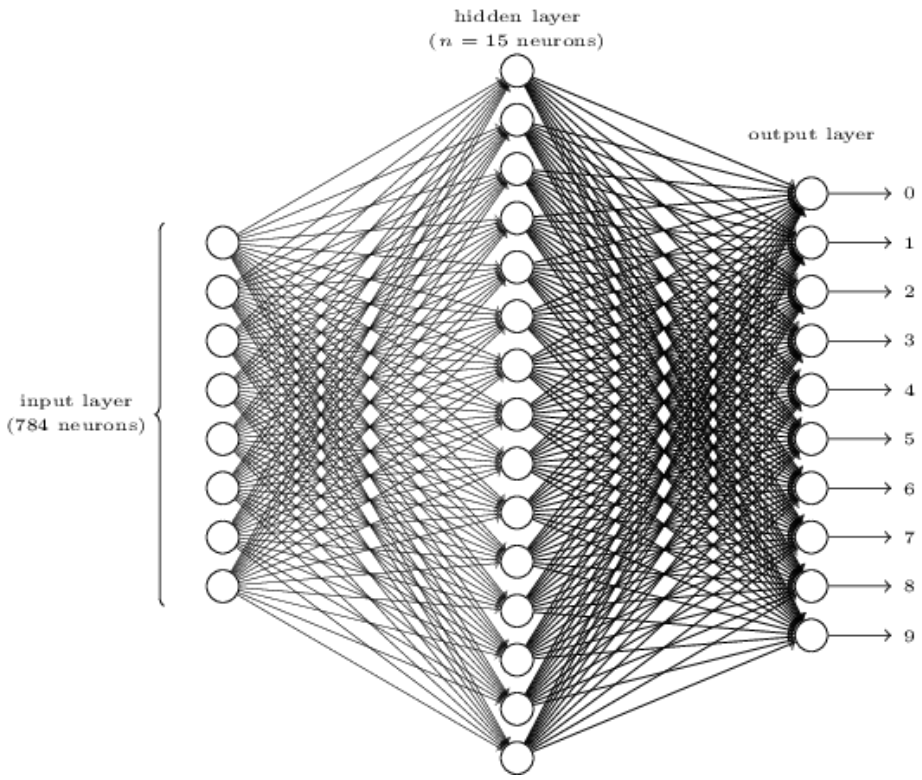


Figure 3.8 Feedforward Neural Network [49]

In the example given in Figure 3.8, there are 784 neurons in the input layer which hold the incoming data value. Each of the neurons is connected to every neuron in the hidden layer, and every hidden layer's neuron is connected to the output layer neurons through weights. All these neurons are activated by passing through a function known as an activation function. This neural network can be used for classification such as handwritten digits classification. It can classify after proper weights are assigned to each connection in the network. These weights are not randomly assigned, but learned by backpropagation

algorithm [11]. This algorithm backpropagates the errors at the output layer through all the intermediate weights to compute the gradients. Later, these weights are updated to decrease the error. These steps are repeated until the classifier exhibits minimal error.

The feedforward neural network is useful for one dimensional data. For the input data, such as videos and images, feedforward neural network is not very efficient as it does not take into account of spatial structure and relations of the data [49]. In the next section, we describe convolutional neural network that uses a special architecture that is well-adapted to classify two-dimensional data such as images.

3.3.2 DCNN Architecture

A *convolutional neural network (CNN)* is a special type of neural network for 3D input that has an ability to learn unique features among various 3D data like images. A convolutional neural network leverages the ideas of local connectivity, parameter sharing, and pooling of hidden units. The idea behind the local connectivity is that each hidden unit is only connected to a subregion of the input. Without local connectivity, fully connected layers would have an unmanageable number of parameters as every neuron of a layer would be connected to all neurons of the next layer in two-dimensional. Parameter sharing is about sharing matrix parameters across other neighbor hidden units of the same layer. The units organized into the same ‘feature map’ share parameters. Thus, it extracts the same features at every position regardless of the object location within the image. There are three major types of layer in DCNN: convolutional layer, pooling layer, and fully connected layer as shown in Figure 3.9.

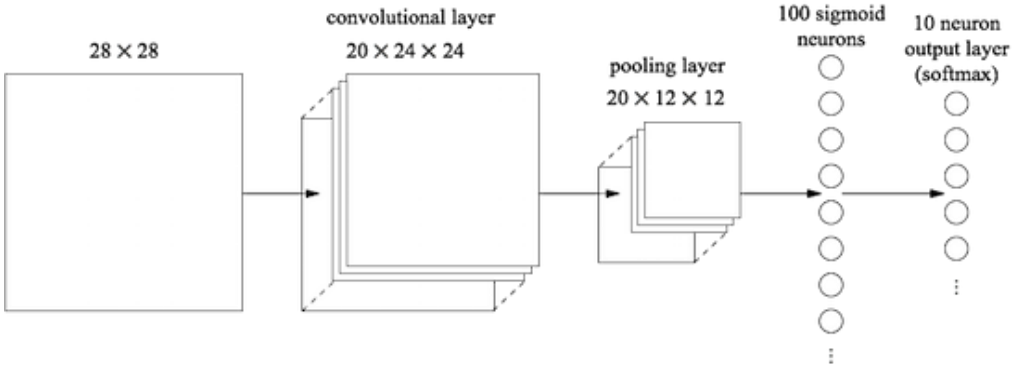


Figure 3.9 Simple convolutional neural network for 28x28 image size input and 10 output classes with one convolutional layer and one pooling layer. [49]

3.3.3 Convolutional Layers

Convolutional layer operation is shown in Figure 3.10. There two different filter sets W0 and W1 (in red) are convolving with the input size of 7×7×3 pixels and taking stride of 2 in both the dimensions (in blue). The output volume of size 3×3×2 is shown in the green. The Convolutional layer parameters are as shown below:

- Kernels = 2 (W0, W1)
- Number of spatial filters per kernel = 3 (W0[:, :, i], i=0,1,2)
- Stride = 2
- Zero padding = 1

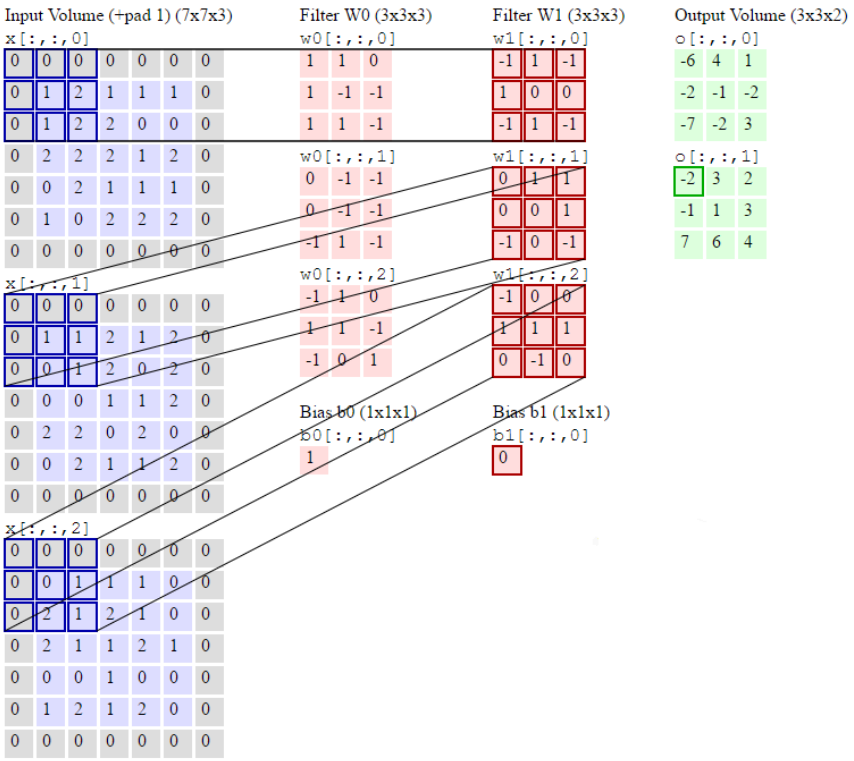


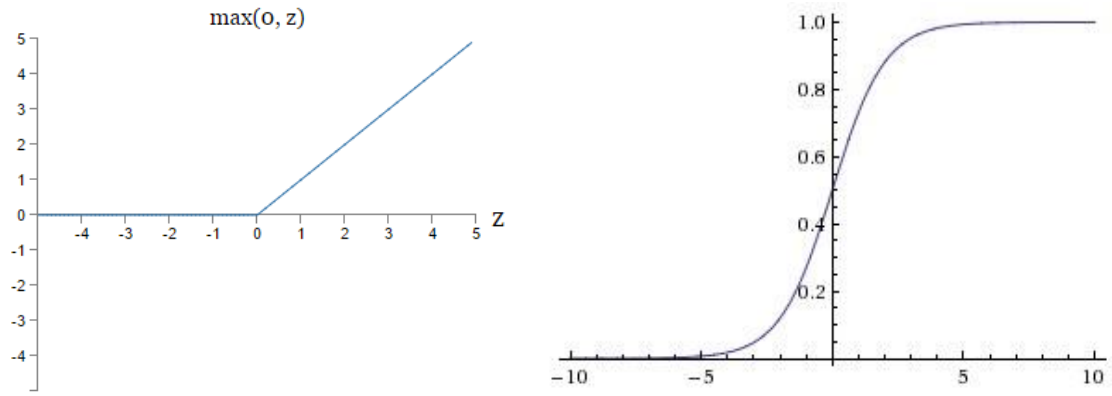
Figure 3.10 Sample Convolutional layer [10]

That is, we have two filters of size 3 × 3, and they are applied with a stride of 2. Moreover, a zero padding of size one is being implemented to the input data. We do that so that we preserve as much information as possible in first layers of the network. Therefore, the output layer size has spatial size (5 - 3 + 2)/2 + 1 = 3. In Figure, the 3 × 3 O[:,:, 0] matrix is the output after convolving W0 filter with the input matrix. Each feature map forms a grid of features/kernels. It can be computed with a discrete convolution:

$$y_j = ReLU((\sum k_{ij} * x_i) + b)$$

Here, k_{ij} is the weight kernel, and x_i is the ith channel of input. All the convolutional layers use the Rectified Linear Unit (ReLU) activation function. Often, the convolutional layer and ReLU layer are considered as one layer. It introduces invariance to the unit in

the previous layer. ReLU applies an elementwise activation function to the input image, such as the $max(0, z)$ thresholding at zero as shown in Figure 3.11.



(a) (b)

Figure 3.11 (a) Rectified Linear Unit and (b) Sigmoid activation function [10]

Rectified Linear Units (ReLU) and Sigmoid functions are used as activation functions of the hidden layers' and output layer's neurons respectively. All the convolutional layers' outputs were fed into Rectified Linear Unit nonlinearity function; while sigmoid activation function is used only for the fully-connected output layers. Later, the cross-entropy error function is applied to the output of sigmoid activations to get the classification outputs where one means the correct predicted class and zero means the incorrect predicted class.

3.3.4 Sub-sampling Layers

There are two different ways to subsample the output of the convolutional layer: Max-Pooling and Mean Pooling. The pooling/subsampling introduces invariance to local translation. In this research, we use a max pooling layer between two convolutional

layers. Its purpose is to progressively reduce the spatial size of the representation to reduce the number of parameters and computations in the network.

We use a pooling layer with filters of size 2 × 2 applied with a stride of 2 and down sampled every channel in the input by two, with the result of discarding 75% of the activations as shown in Figure 3.12 below. Max pooling can be used with different strides and sizes. The stride of p is standard pooling stride for the filter of size p × p. Therefore, we get the dimension reduction by the factor of p at the output of max pooling layer.

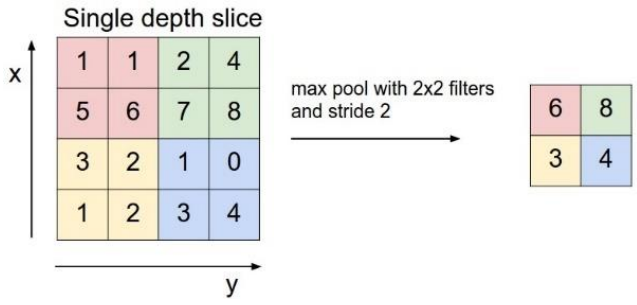
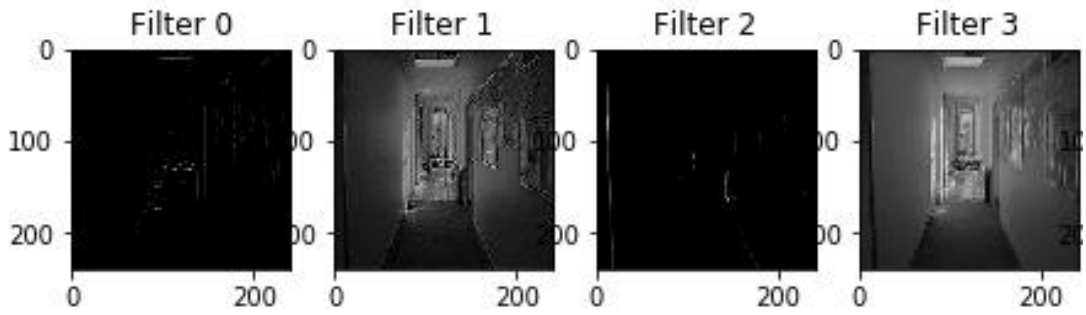
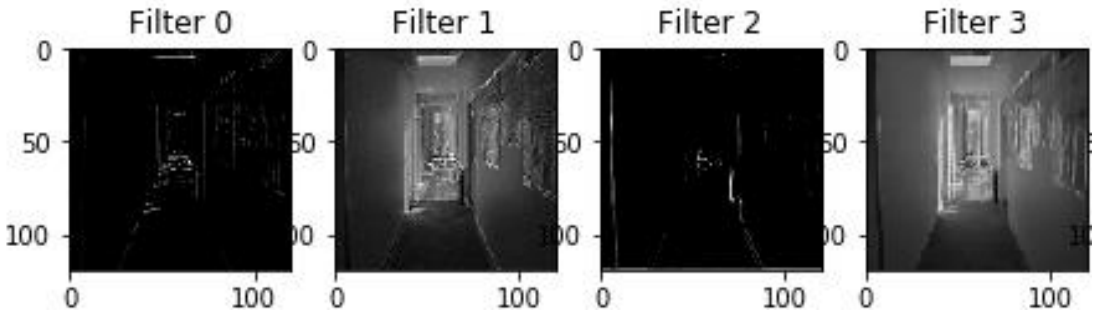


Figure 3.12 Max pooling layer with 2 × 2 filter [10]

The sample pooling operation on one of the corridor images is shown in Figure 3.13. The 240 × 240 output from second convolutional layer was down sampled to 120 × 120 at the output of the max pooling layer. Only four activation maps are shown in the image to demonstrate the sub-sampling operation. Every ‘max’ operation would, in this case, be taking a max over four numbers. Pooling operation does not cost any extra parameters, and therefore, the depth dimension remains unchanged.



(a) Input to Max Pooling Layer



(a) Output of Max Pooling Layer

Figure 3.13 Sample images (a) Input to Max Pooling Layer: 240x240 image size (b) Output of Max Pooling Layer: 120x120 image size

The second type of sub-sampling layers is known as a mean pooling layer. As the name suggests, the mean/average pooling layer operates independently on every depth slice of the input and down-sample it spatially, using the 'mean' operation. Every 'mean' operation would, in this case, be taking an average of all the elements in 2 × 2 matrices (m=2):

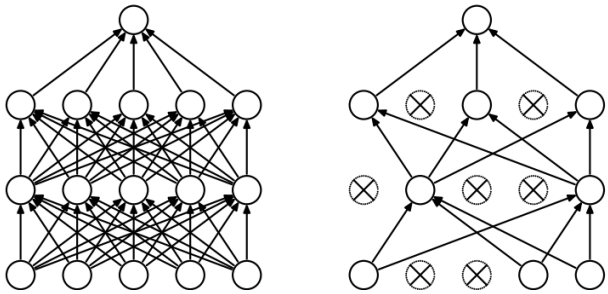
$$y_{ijk} = (1/m^2) \sum_{p,q} x_{i,j+p,k+q}$$

One of the first successful convolutional neural network architecture is shown in Figure 3.4. Here, the input is 2D gray-scale image. Layer C1 is a convolutional layer with six feature maps, and each feature map is generated by convolving 5 × 5 filter kernel to

the input image of size 32 × 32. The next layer is called a sub-sampling layer with similar six feature maps and of size 14 × 14. These maps generated by applying max operation on 2 X 2 patch on C1 feature maps. The last three layers are fully connected layers which make a feed-forward neural network. In our experiments, we use max pooling layers with filter of size 2 × 2. The performance of the DCNN highly depends on weight initialization. In our experiments, all the convolutional kernels were initialized normally distributed with zero mean and 0.1 standard deviations. When initializing all the neurons with zero mean and 0.5 (higher) standard deviation normally distributed, the network can converge to poor local minima, or it may not train very well. Keeping the small standard deviation is often preferred by neural network researchers to reduce the invariance shift [4]. The stride sizes for convolutional and pooling layer were one and two respectively.

3.3.5 Regularization

One of the important issues while training a neural network is overfitting. One of the regularization methods to avoid overfitting is known as the dropout [10]. The dropout technique was published by Srivastava et al. [10] and is shown in Figure 3.14 for a feed-forward neural network.

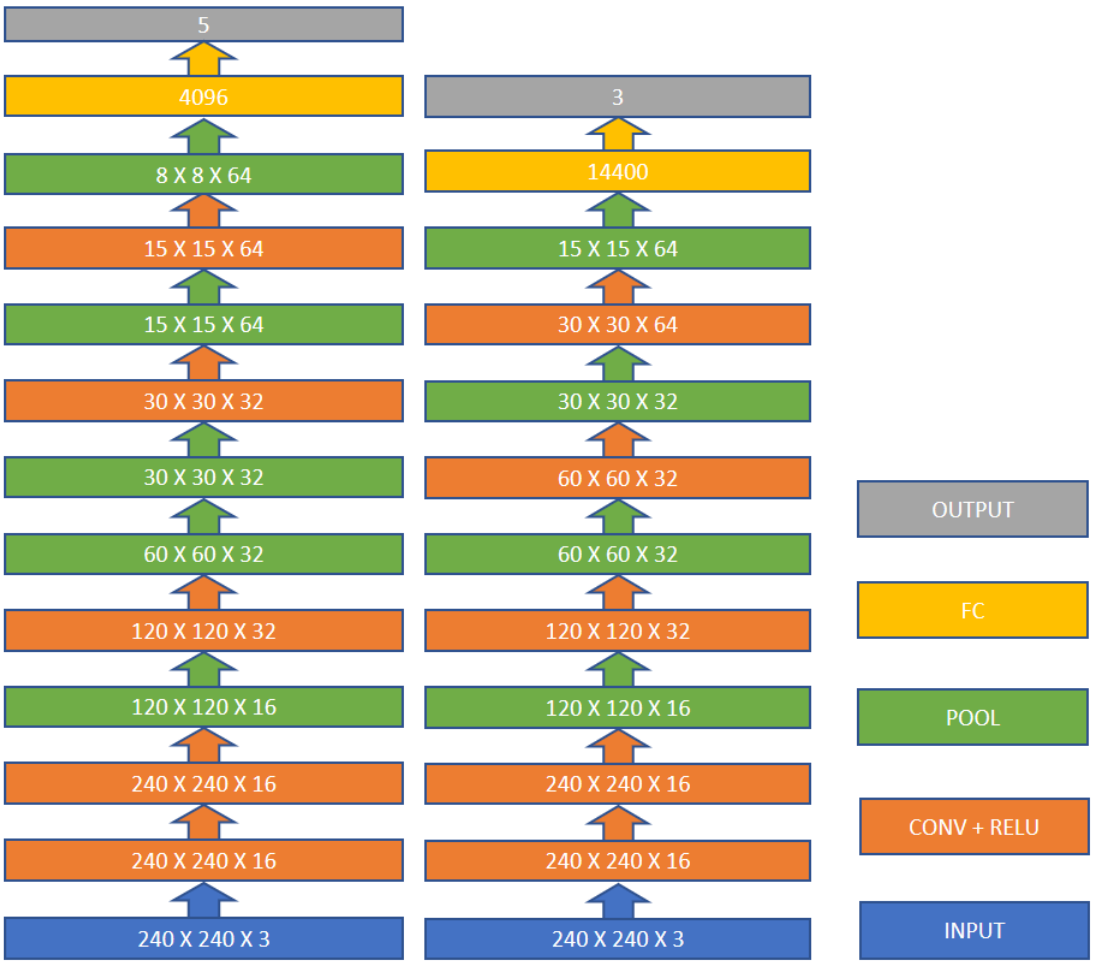


(a) (b)

Figure 3.14 (a) Standard Neural Network (b) After applying Dropout [10]

As shown in Figure 3.14, the major difference between the standard neural network and dropout neural network is that some neurons among the layers have been randomly set to zero in the dropout neural network type during the training phase. However, all the neurons remain in the network during validation and testing phase which causes the neural network to avoid overfitting in the test phase [10]. The dropout layers can be applied after pooling layers in the convolution neural networks. To train the dropout efficiently, we use a mini-batch learning method that divides the training images into small batches, and with each batch, we train the samples with Adaptive Moment Estimation (Adam) optimization algorithm [11]. Each time we take mini batches as a training input we randomly drop 30% of the neurons by applying the binary mask to all the hidden layers of CNN. The Adaptive Moment Estimation (Adam) optimization algorithm is one of the most popular methods that compute adaptive learning rates for each parameter. The algorithm takes the first moment (the mean) and the second moment (the uncentered variance) of the gradients into consideration while updating the weights. The moment coefficients $\beta 1$ and $\beta 2$ were set to 0.9 and 0.999 respectively as suggested in the paper [11]. The convolutional neural networks used for the scene recognition problem have more number of layers than the AlexNet [4]. We use multi-layer convolution neural network for buildings, floors, and corridor classification problems. Figure 3.15 shows the convolutional neural network architectures of the indoor scene recognition problems. Figure 3.15 (c) shows the color coding of all layer types - convolutional, pooling, fully connected (FC), and output layers. The DCNN architectures are made up of these layers as depicted in the figure. As depicted in the figure, the image is fed at the input block and the output of the predicted class is presented at the output

layer. The interconnections between the two blocks represent convolutional kernels or max-pooling matrices. In this work, we use 3x3 convolutional kernels for every convolutional layer. The third dimension (depth) of output of convolutional layer depends on the number of weight matrices used between the layers. For example, the output of the first convolutional layer is 240 x 240 x 16 which suggests that the sixteen convolutional kernels were used between input layer and first convolutional layer.



(a) (b) (c)

Figure 3.15 (a) DCNN Architecture for Floor classification (b) DCNN Architecture for Building/Corridor classification (c) Color coding for layers

Training of the deep convolutional neural network was done on Amazon Web Services (AWS) and FloydHub using g2.2xlarge GPU instance which is parallel computing environment and FloydHub NVIDIA GPU respectively. The FloudHub server includes virtual GPU computing node with 1536 CUDA cores and 12GB RAM for video memory. Training takes approximately 3-6 hours to train each network. The discussions of the experimental results are given in the next section.

3.4 DCNN based localization – Experimental Results

3.4.1 Recognition of Buildings

In this section, the results from training a building classifier is explained. We used the various indoor images of DEH, CEC, and FH to train a classification model. There were approximately 1795 training images, 504 validation images, and 329 testing images. The DCNN architecture is shown in Figure 3.15 (b) and it took about 3.5 hours to train on a 12 GB FloydHub GPU. The first and most significant quantity to track while training a neural network is the loss function. As shown in Figure 3.16, this learning curve can give valuable insights into the amount of overfitting in the neural network model. The difference between the training loss value and validation loss value is known as the generalization error. This generalization error should decay over epochs for a perfect neural network model. The increase in the generalization error after certain epochs can be seen in the plot which represents the overfitting.

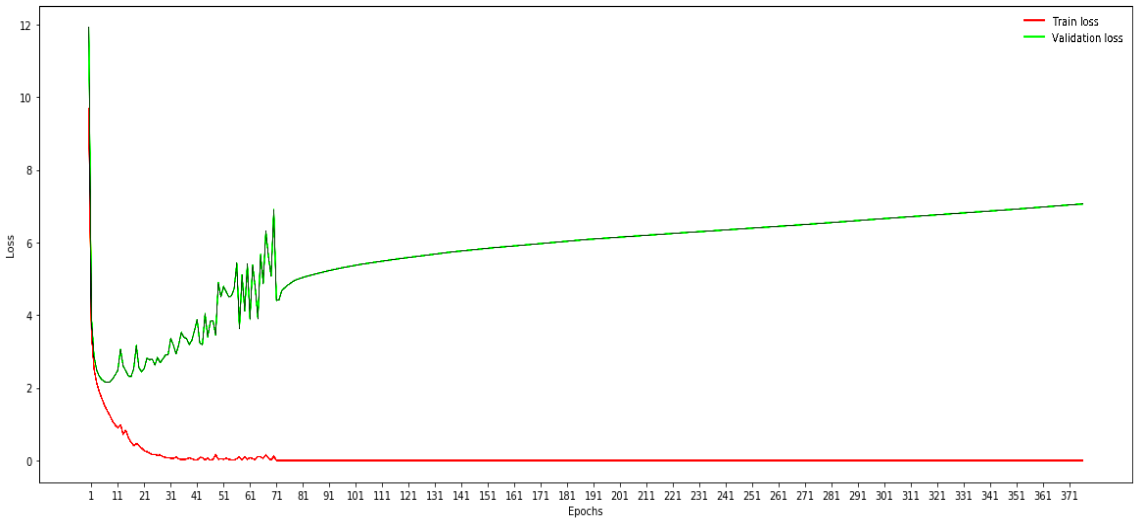


Figure 3.16 Building classifier learning curve (without Dropout)

After applying dropout, the generalization error seems to be constant over epochs as shown in Figure 3.17.

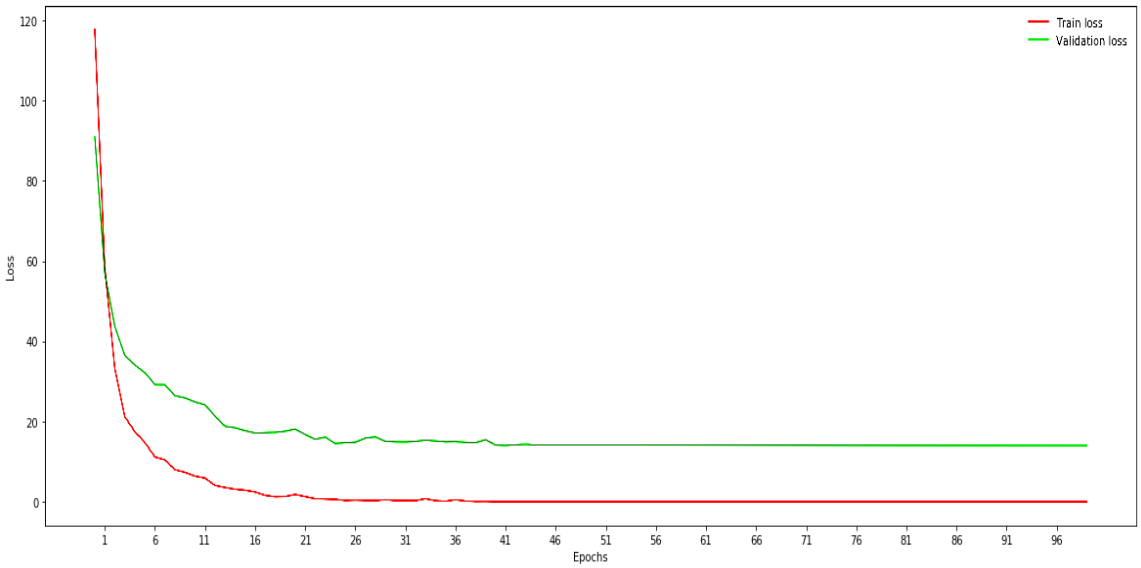


Figure 3.17 Building classifier learning curve (with Dropout = 0.2)

The second significant quantity to track while training a classifier is the validation/training accuracy. Based on the validation accuracy, the parameters and depth

of the network were changed to achieve highest possible validation accuracy. The trained model achieved an 82% accuracy. As shown in Figure 3.18, the normalized confusion matrix of test images represents the network performance for classifying FH, DEH, and CEC buildings. The darker diagonal of the confusion matrix represents the better accuracy of the network model. As it can be seen, Felgar Hall and CEC building images are most easily distinguishable. This model can be used to find a location of a robot on the campus. This model is useful only when a robot is starting its operation without prior information about the location and is useful when the kidnapped robot problem occurs.

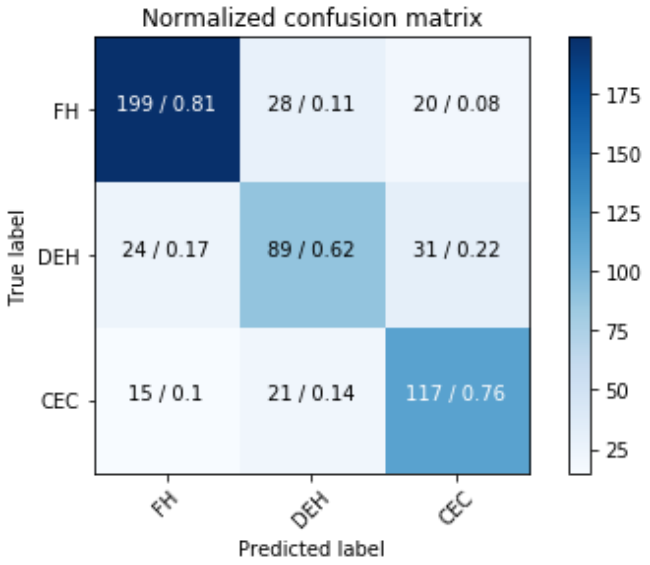


Figure 3.18 Normalized confusion matrix of building classifier

3.4.2 Recognition of Levels of a Building

Floor/level classification neural network architecture is very identical to the building classification network except an extra pooling layer number four as shown in Figure 3.19. This makes the network smaller than the building classification in term of some layers and parameters. We used the various indoor images of five floors of DEH building to train a classification model. There were approximately 2245 training images, 720

validation images, and 540 testing images. The training took approximately five hours. The learning curve of the floor classifier is shown in Figure 3.19.

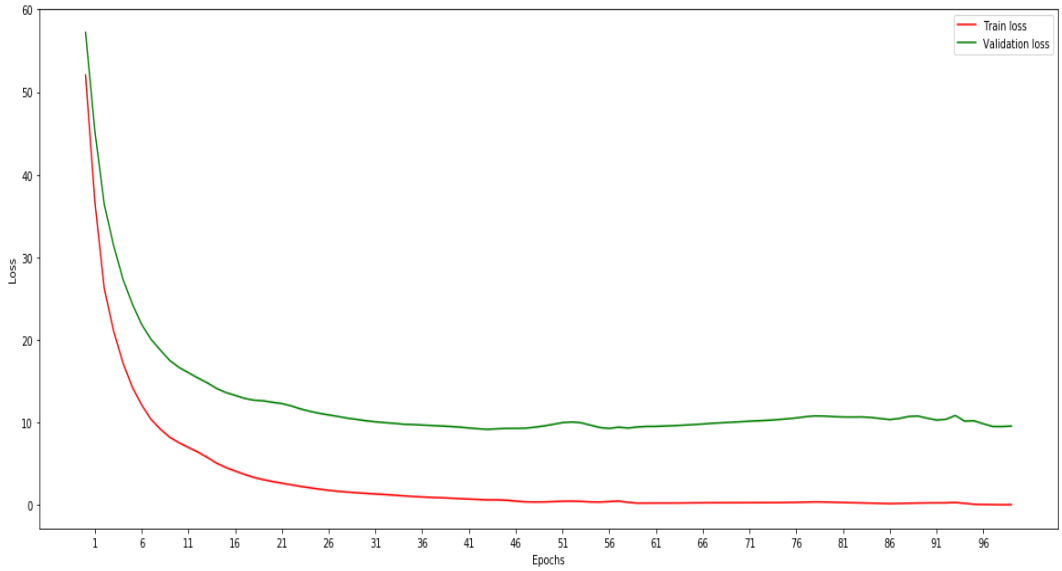


Figure 3.19 Floor classifier learning curve (with Dropout = 0.2)

The validation set of five floor-levels of DEH classifier achieved an approximately 60.7% accuracy. As shown in Figure 3.20, the normalized confusion matrix of test images represents the network performance for classifying five floors of DEH. X axis represents predicted labels and Y axis represents true labels. Each block contains two values: (1) number of images and (2) accuracy. For example, from total 78 images of level-one, the image classifier recognized 62 images correctly and 17 images incorrectly. This result exhibits an accuracy of approximately 79% for the level one of the DEH building. Moreover, there are 35 and 16 images of the third floor-level recognized as the fourth floor and fifth floor respectively. The network performance can be visualized with this confusion matrix as darker shade in diagonal represents the better accuracy of the model.

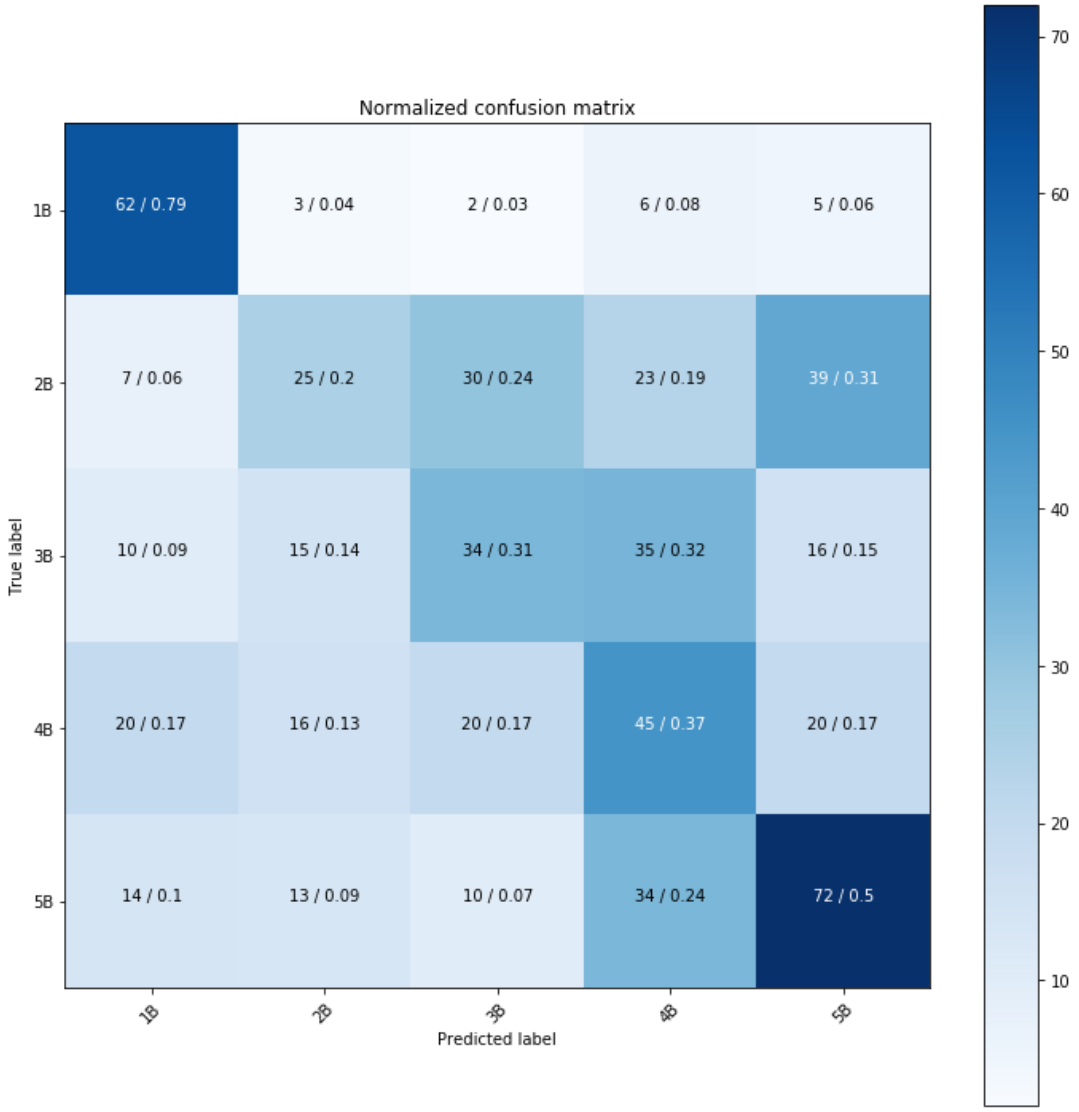


Figure 3.20 Normalized confusion matrix of floor classifier. X axis represents predicted floor/level and Y axis represents true floor/level.

A sample of false detected images is shown in Figure 3.21 below. As it can be observed, there is no unique feature in the image that can be used to classify the floor efficiently. Moreover, being very close to the end of the corridor makes it a much more challenging problem to recognize the level and the location within the hallway. This issue is known as a fine-grained classification problem [7]. This issue arises when there are many feature similarities among the classes and also when there are many various features within the

class. This result suggests that the proposed DCNN architecture does not work well for some recognition tasks where visual features are very difficult to distinguish among classes. The network was trained multiple times to evaluate the consistency of the performance of the model. As expected, the results were the same or worst in almost every experiment.

Figure 3.21 True: 2nd Floor. Predicted 3rd Floor

As it can be seen from the confusion matrix, floor-level number two and three are very identical and difficult to classify with higher accuracies. On the contrary, floor-level number one and five can be easily recognized. Further improvement in the architecture can be evaluated such as increase in depth of the network or width of the network to evaluate the classifier accuracy. Furthermore, the training images can be taken uniformly and by a constantly moving robot to have a deep convolutional neural network to work better in detecting and finding unique features. To solve this issue, one could add a unique feature or landmark in the scene to make the floor-level recognition robust. In this scenario, passive markers can be placed in the scene to recognize the different floors and to localize robot's position using monocular images. These artificial markers add unique

landmarks to the floors which can be used to recognize a floor-level using computer vision algorithms.

3.4.3 Recognition of Corridors

In this section, DCNN is trained to recognize the corridors of the floor which is one step closer to the finer localization of the indoor spaces for a mobile robot. This approach can help an autonomous robot to localize in the map using 240 x 240 RGB images. Images of the 4th floor of the Devon Energy Hall are captured to evaluate the performance of the network. The 4th floor can be divided into three corridors, and they are connected in C shape. In this section, we train the deep convolutional neural network to classify each corridor. The architecture of the DCNN remains unchanged, and so there are five convolutional layers followed by the pooling layers as depicted in Figure 3.8 (b). The only change is the number of neurons in the output layer. Here, the dataset is divided into six classes. Walking from East to West and vice versa are considered different categories in each corridor. Therefore, there two classes for each corridor of the 4th floor of the Devon Energy Hall. Figure 3.22 shows the confusion matrix of the experiment. The labels are as following 1A, 1B, 1C, 2A, 2B, 2C. Here, '1' in the labels represents the walking direction from West to East; and '2' accounts for the walking direction from East to West. A, B, and C represent West exit, Main, East exit corridors.

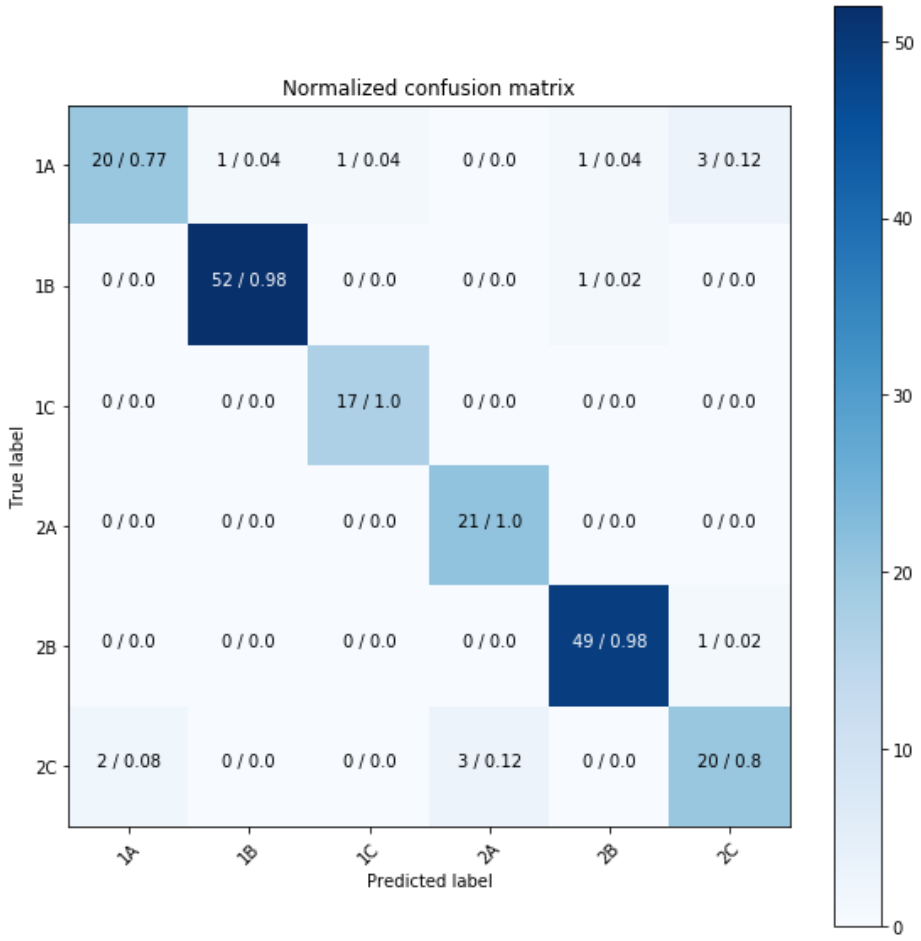


Figure 3.22 Normalized confusion matrix for corridor classifier

Figure 3.23 shows the learning curve for this experiment. The best model based on the validation loss and validation accuracy from the experiment is later used to test on the test dataset.

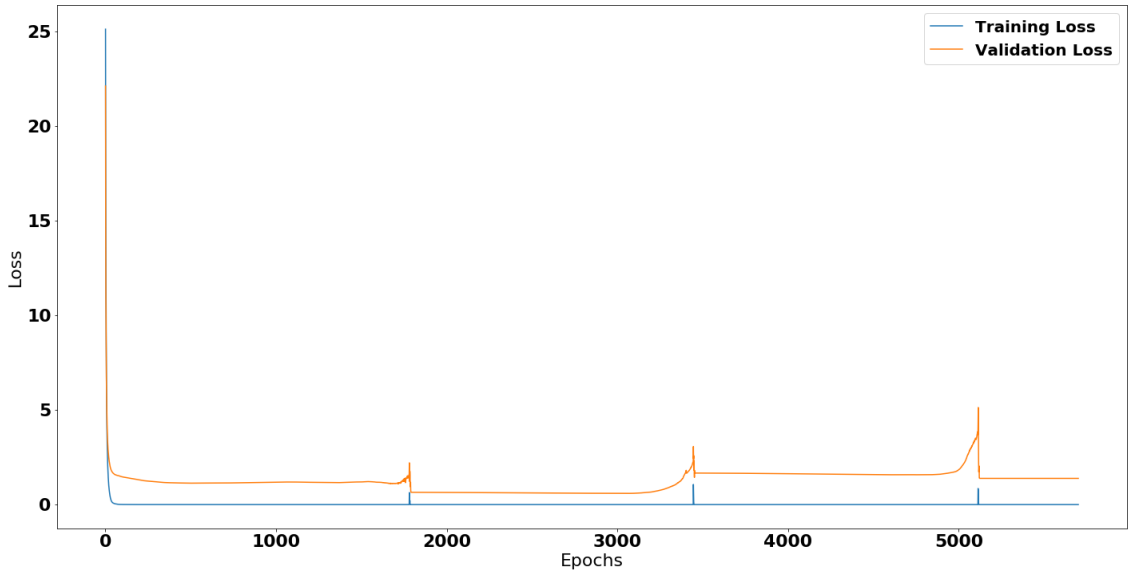


Figure 3.23 Learning curve for corridor classifier

The achieved test set classification accuracy is 88.7%. An experiment to classify all the corridors of multiple buildings can be performed to evaluate the neural network performance.

3.4.2 Visual Representation of DCNN Layers

The visualization of deep convolutional neural network layers can help to find better features and to improve and make changes in the architecture if there is a need. In this visualization, an image was fed into a network and the outputs at every layer were recorded. The visualizations of CNN layers are shown in Figure 3.24 to Figure 3.33 for one CEC building image.

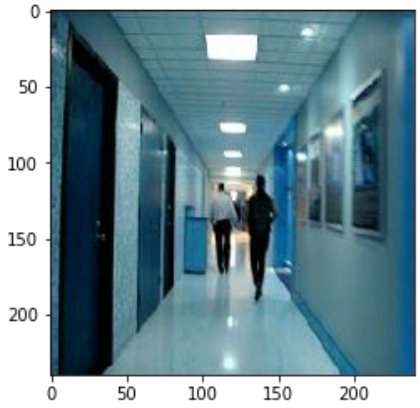


Figure 3.24 Input image

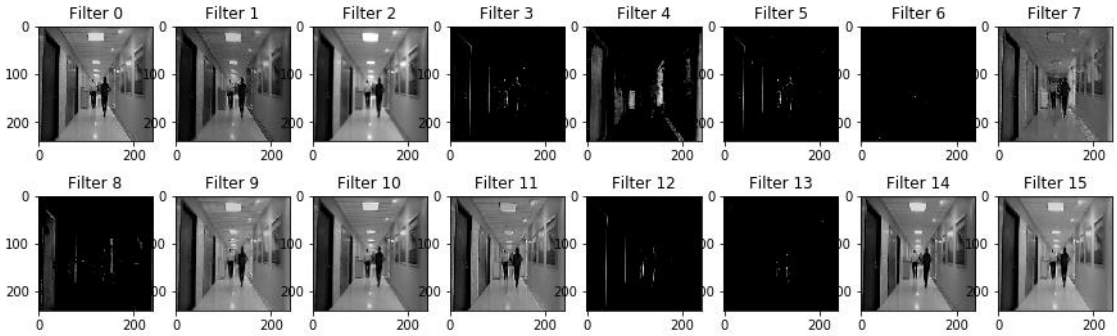


Figure 3.25 Output of Convolutional Layer 1

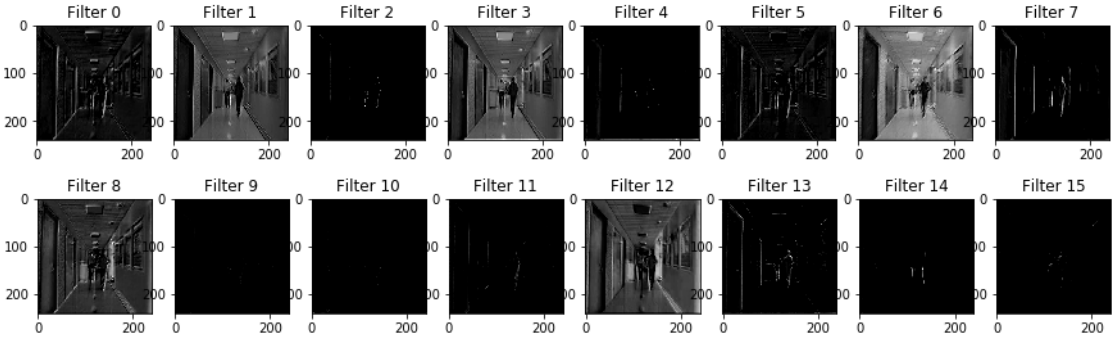


Figure 3.26 Output of Convolutional Layer 2

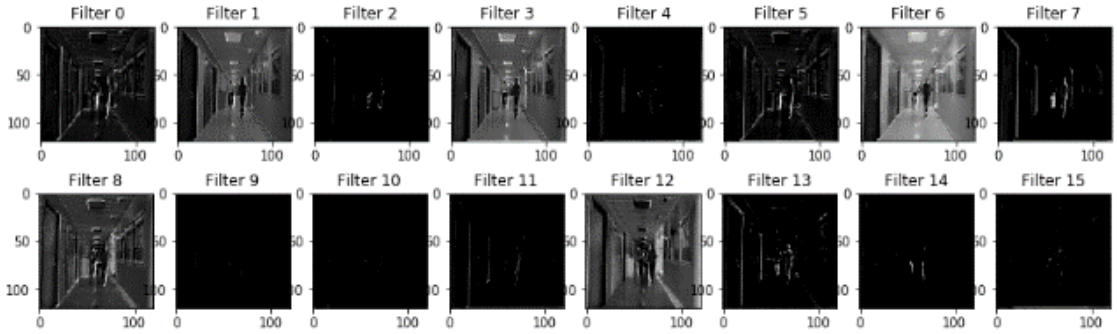


Figure 3.27 Output of Pooling Layer 2

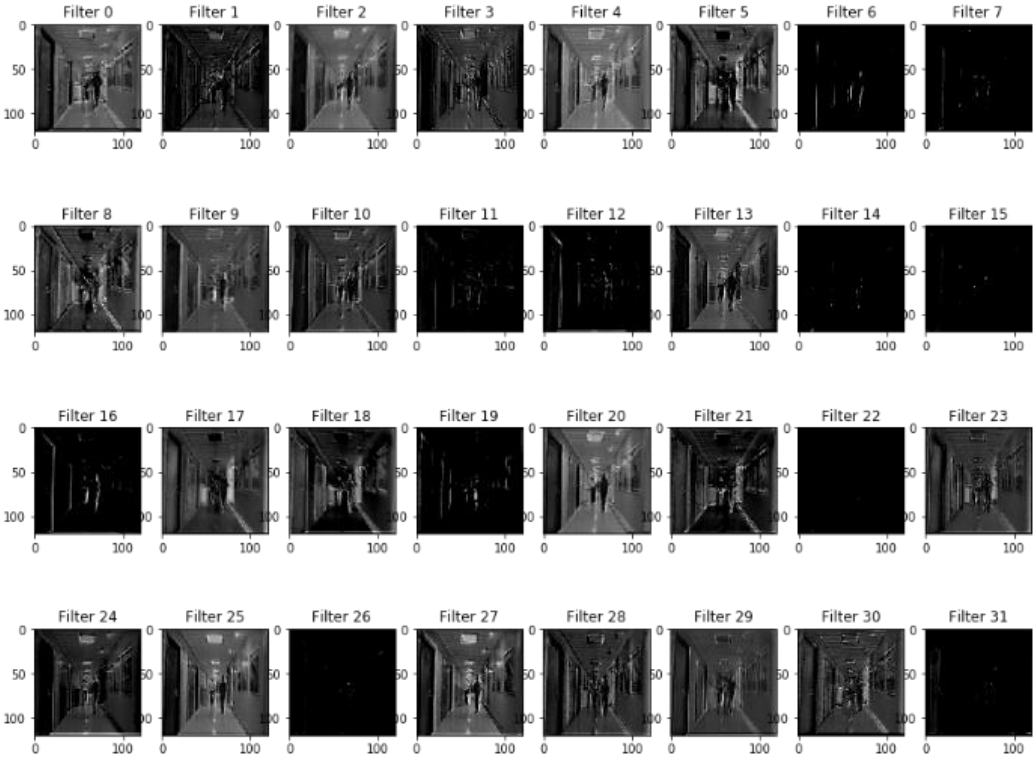


Figure 3.28 Output of Convolutional Layer 3

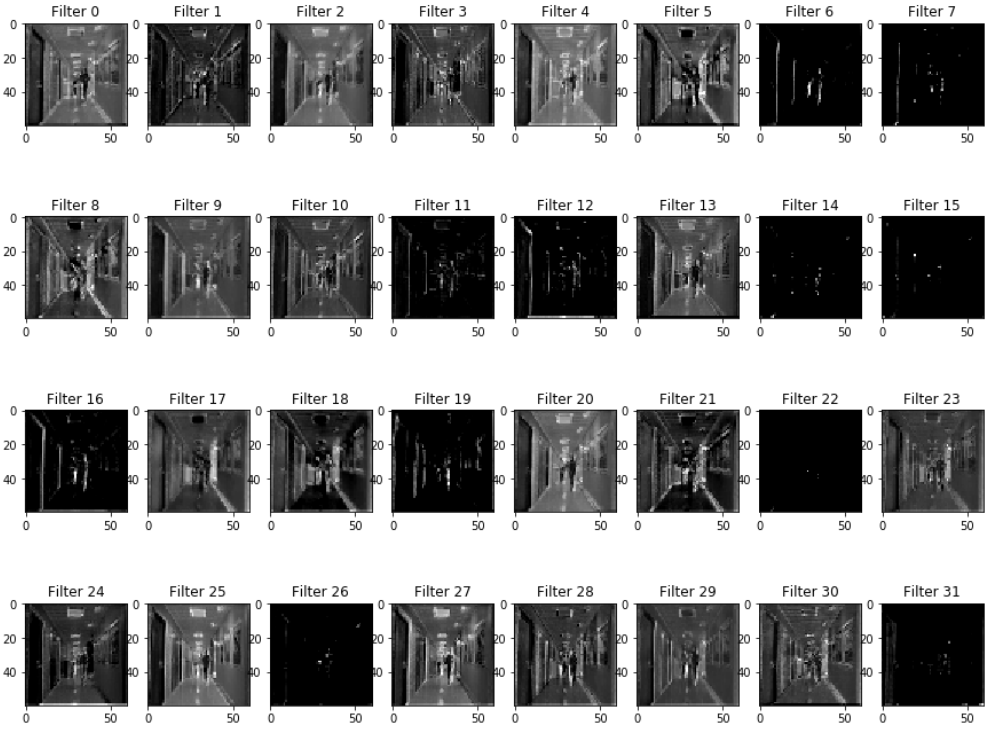


Figure 3.29 Output of Pooling Layer 3

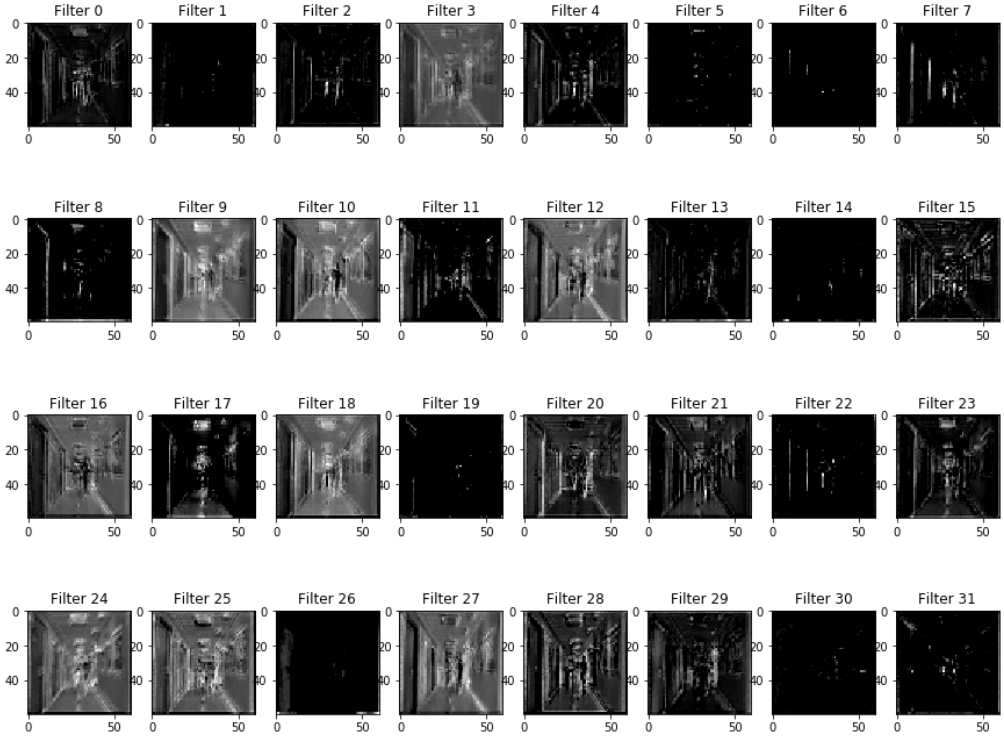


Figure 3.30 Output of Convolutional Layer 4

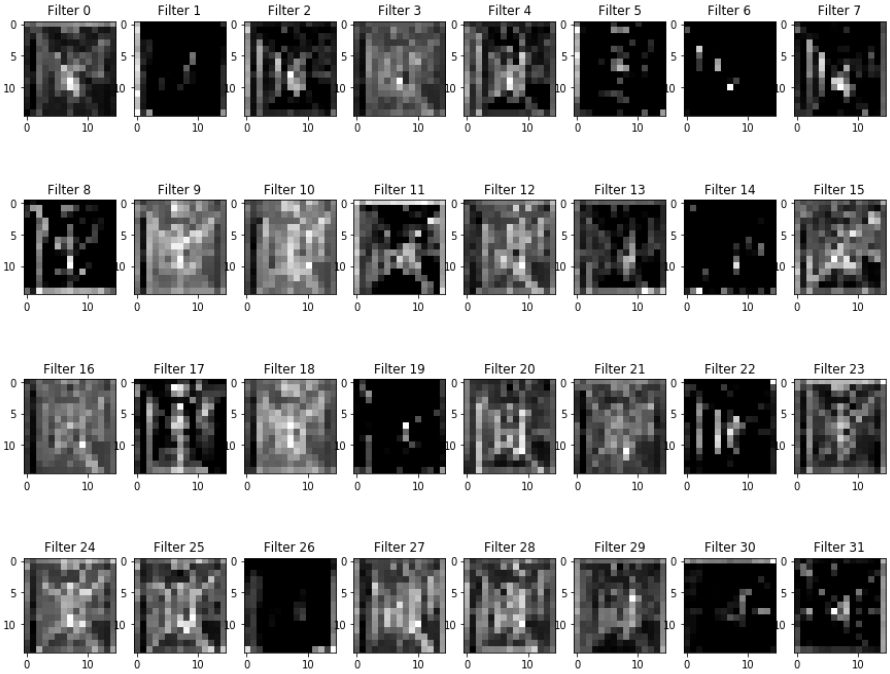


Figure 3.31 Output of Pooling Layer 4

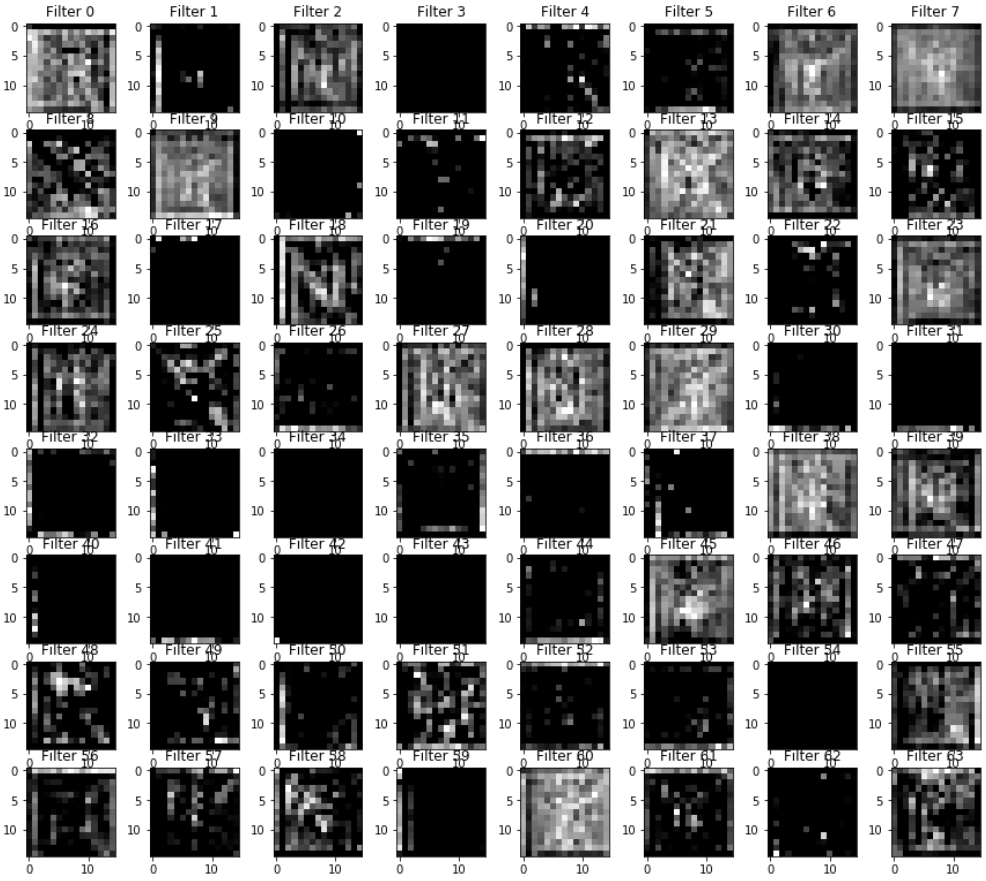


Figure 3.32 Output of Convolutional Layer 5

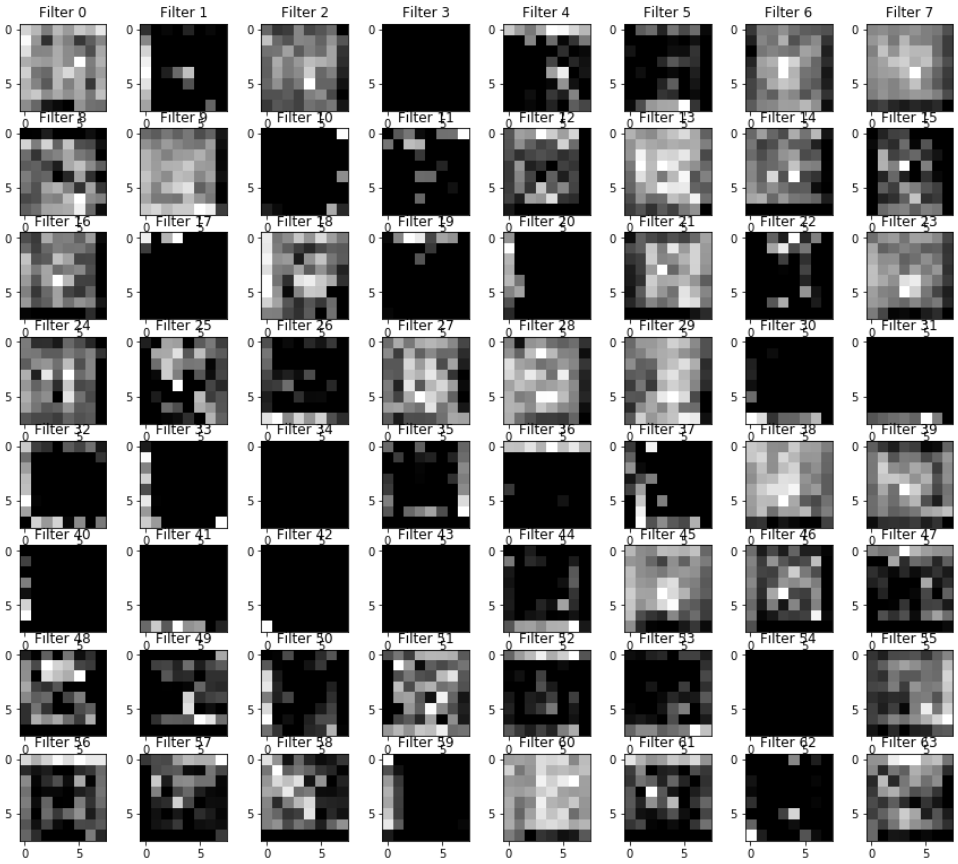


Figure 3.33 Output of Pooling Layer 5

3.5 Discussion and Conclusions

Map generation and localization are important phases for the robot to autonomously navigate and accomplish the tasks. Generating localization maps using on-board monocular cameras is not an easy task. The purpose of this experiment was to recognize the scenes within the building or even within the corridor by using monocular image or sequence of video frames to have fast and coarse localization. The experiment presented in this chapter can be useful to recognize approximate location of the robot by feeding an image to the Deep Convolutional Neural Network, and the algorithm can return the location information within milliseconds. Furthermore, the model size of the DCNN is within the range of 1 megabytes – 6 megabytes (MB). It requires sufficient

images to train the neural network model which can be generated using the SLAM technique explained in Chapter 4 or by manually labeling the images. For this experiment, the images were labeled and preprocessed manually. The drawback of this architecture is that it can only be used for coarse localization.

The architecture performance can be further evaluated by training more number of images and using deeper and high-level network models like ResNet [23]. This network is only useful when there is a plenty of training images and computing time and resources. More importantly, weight initialization plays very significant role in the training. In all the experiments, the weights of the neural network were initialized as normally distributed around zero mean. The popular weight initialization technique for ImageNet challenges like Xavier [24] can be evaluated in future for the problem of indoor scene recognition work. In this proposed technique, the weights are initialized based on the number of input and output neurons rather than normally distributed weights. Furthermore, the above all results were produced by dynamic tests, i.e., images were taken by walking in the hallways and holding the camera. This data collection method generates few blurred images in training and validation sets. For future work, a mobile robot in steady motion can be used to collect data for further performance evaluation of the model which would capture steady and clearer images. This method can only be used for very coarse localization. To get a finer localization and mapping, there is need of a method that can help a robot to determine its precise location. One of the methods is to add artificial landmarks or features in the scene to recognize the relative pose of the robot in the corridor or floor-level. For example, planar markers can be added as a unique

feature of the environment to use the metric localization information of a robot in the environment to improve the efficiency, which is the topic of the next Chapter 4.

Chapter 4: Localization using Monocular Images

4.1 Introduction

Mapping and Localization using monocular images are challenging problems due to the absence of sufficient features that can represent a map of an environment. Furthermore, the monocular image may not suffice to extract all the information we need to create a map of an environment. In past decades, researchers have used several algorithms to localize and create maps using high-tech sensors like LiDAR, depth camera, stereo camera, Radar, etc. These systems can be used to generate high-resolution 2D and 3D maps. However, they are computationally costly and sometimes very slow to operate in real time. In this chapter, we propose a new method to approximately localize a robot using monocular images and passive planar markers. This method can be used to rapidly explore an unknown environment and later it can also be used to augment the high-resolution maps generated in a known environment.

In the past decade, Augmented Reality (AR) and Virtual Reality (VR) have been highest interests for the gaming industry. In the past, researchers have explored and found multiple methods like marker-based and markerless methods for augmented reality (AR) applications. The marker-based method is the very popular method for the localization and object detection algorithms among AR engineers due to low cost and easily implementation. The marker-based method uses external markers to augment the reality by showing objects and replacing those markers in an image or video. However, use of these AR markers for mapping an environment for the indoor mobile robot is unheard of. Our methodology involves placing planar augmented reality marker patterns on the walls of an unexplored environment and detecting them using an onboard mobile robot camera

hardware. A single monocular camera can be used to identify and estimate the relative pose of the marker in real-time using computer vision algorithm. Such markers are widely used in Augmented Reality (AR) applications for object localization and augmentation. One of the AR examples is shown in Figure 4.1. The chairs and lamp are not real and are augmented in the original image.

Figure 4.1 Augmented Reality example with virtual chairs and a virtual lamp [14]

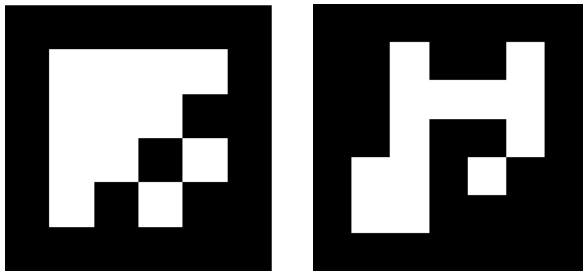
There are different types of AR marker being used by researchers, e.g. ARToolkit [16], ARTag [17], Intersense [18], and ArUco [15], etc. In this thesis, we present the system design and localization technique by using relatively newer ArUco markers. Moreover, ArUco dictionary can be custom generated based on the number of markers required for the application. This dictionary contains markers with unique identification numbers as shown in Figure 4.2. These generated markers have error correction built-in, and the marker patterns are mathematically generated to maximize the distinction among the different marker codes. The algorithm to make the ArUco markers is explained by S. Garrido-Jurado et al. [15]. The test results of the thesis indicate that it is possible to detect displacement of an indoor mobile robot relative to the fixed marker position.

Furthermore, the results of experimentation with the system focused on mobile robot localization are presented in next section.

4.2 Marker based Localization - Experimental Results

4.2.1 Methods and Experimental Setup

The planar markers in the ArUco system are defined by p × p array. For this experiment, we consider evaluating the performance of markers with size p = 6 and 7. Each of squares is binary coded, and so they are either white or black but the outer two rows and columns define the border of the marker, and these boundaries are color coded black as shown in Figure 4.2. The internal binary code of these markers, as explained by S. Garrido-Jurado et al., is defined by four and five words respectively which is equivalent to removing the black padded rows and columns. Each word contains the same number of bits as a word, i.e. five bits per word for five words system and four bits per word for four words system.

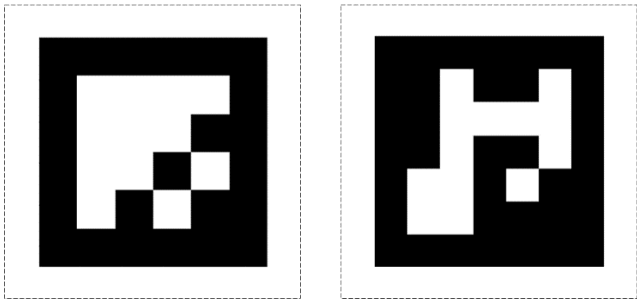


(a) (b)

Figure 4.2 (a) 4 x 4 ArUco marker (b) 5 x 5 ArUco marker

In total, both markers contain 16 and 25 bits respectively. These markers are encoded with a slight modification of Hamming code explained by S. Garrido-Jurado et al. [15]. This coding technique is very efficient such that the system mostly detects the marker in the environment. Based on the number of bits in the marker, ArUco algorithm

can generate many different markers. For example, for 4 x 4 (p = 6) marker system, ArUco produces 1024 different markers in the dictionary. After detecting the marker in the incoming image or video frame, inbuilt functions of ArUco library provide translational and rotational vector information of the marker referenced to the origin of the optical axis of the camera. However, when placing the marker on the dark painted walls, ArUco built-in algorithm fails, and successful marker detection happens once every 2-3 seconds. Moreover, in our experiment, the detection rate decreases when placed on glass walls due to varying sun lighting conditions in the background, which make it difficult for the algorithm to filter and recognize the location of the marker. To resolve this issue, we add two white padding rows and columns around the ArUco marker during our experiment. This change in the marker is very efficient and the algorithm recognizes it most of the time on dark painted walls or glass walls. The modified ArUco markers are shown in Figure 4.3 below.



(a) (b)

Figure 4.3 Padded markers (a) 4 x 4 ArUco marker (b) 5 x 5 ArUco marker

A Logitech C615 USB camera was used to capture images and record real time video input for high-resolution mode and low-resolution mode experiment respectively. Both experiments, high and low resolution, was performed by operating the camera in 8

MP and 2 MP resolutions respectively. The autofocus mode was turned off for all experiments. Before proceeding to the experiments, intrinsic camera parameters required for the marker detection were estimated using the camera calibration functions in OpenCV. To accomplish this, we took 32 images of the 9 x 6 black & white checkerboard from various angles. This is a very decisive step as the image correction is necessary to reduce the error rate. During the experiment, the resolutions were set to 3264 x 2448 and 1600 x 1200 respectively. OpenCV3.2.0 and its companion version of the ArUco library were used to perform all the experiments in this chapter. A sample image of the marker and corridor is shown in Figure 4.4.

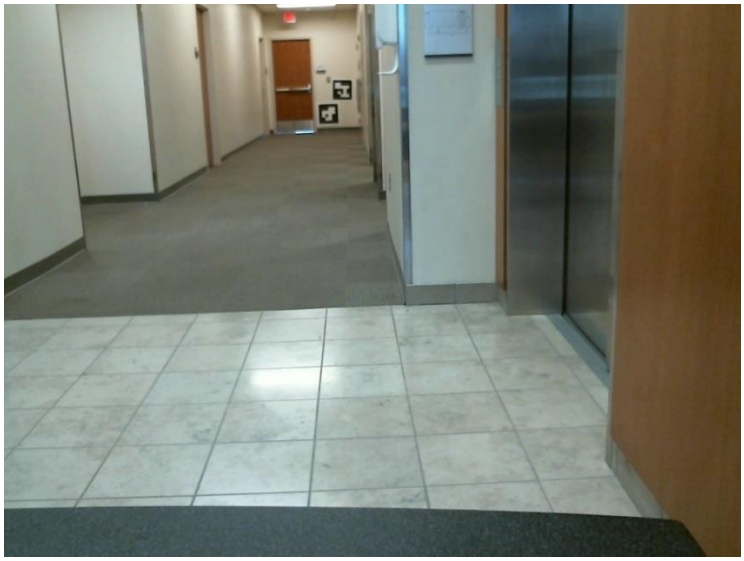


Figure 4.4 Markers placed at the end of the corridor (at about 60 feet)

4.2.2 Performance Evaluation and Experimental Results

In this section, we place 4x4 and 5x5 markers at the end of the corridor to perform two experiments – low-resolution mode and high-resolution mode. In the first experiment, i.e. low-resolution mode, we investigated the tracking accuracy for both

padded markers regarding distance from the camera in 1600 x 1200 resolution. With the padded border included the marker size was x = y = 2 feet. For this experiment, we observe that the tracking error gradually increases over the distance for both the markers in Figure 4.5. The markers were placed within 30 degrees of field of view of the camera. The Root Mean Squared Error (RMSE) in feet vs. distance in feet is plotted to evaluate the performance of the markers. For every 10 feet and up to 60 feet distance, 30-40 sample images were collected in the corridor, and the markers' distances were estimated in the camera coordinate system. The error bars indicate two standard deviations from the mean. The decrement in accuracy over the distance means that the smaller the area of pixels on the image, the less accurate the system is.

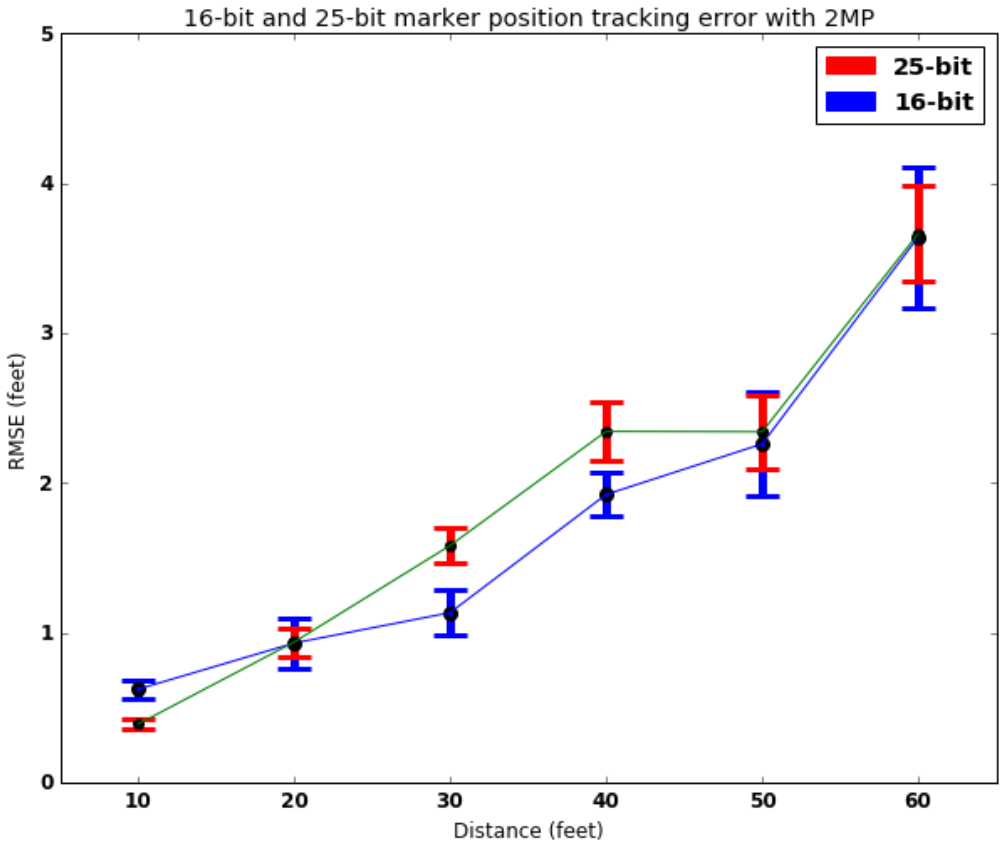


Figure 4.5 16-bit and 25-bit marker position tracking error for 2MP camera resolution

As it can be seen from the plots above, Root Mean Squared Error (RMSE) is very small and steady for 4 x 4 marker for up to 30 feet. Both, 4 x 4 and 5 x 5 markers yield 3.2-4.0 feet RMS error at 60 feet of distance. And it is very perceptible that 5 x 5 marker has comparatively small variance even at 60 feet. During the experiment, we also observed that 4 x 4 marker (p = 6) has low detection rate at longer distances like 60 feet. As shown in Figure 4.6, 5 x 5 marker was effortlessly detected, and system predicted ~63.84 feet whereas 4 x 4 marker problematic to identify in this image. This observation is expected as low bit marker patterns can be easily camouflaged at longer distances.

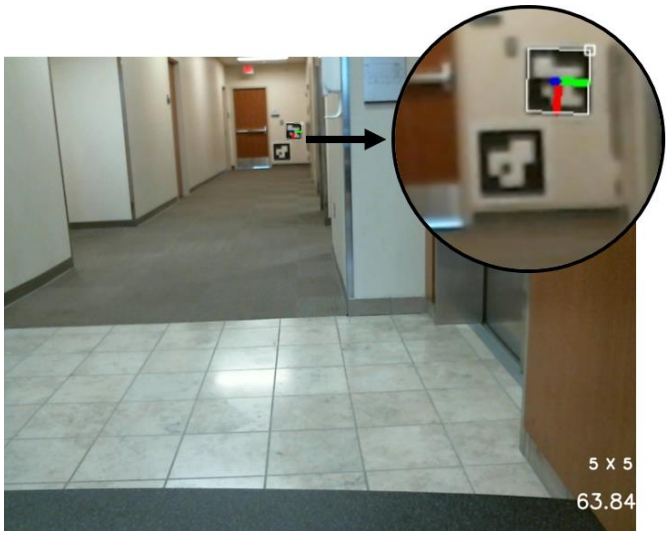


Figure 4.6 Detection rate at about 60 feet.

In practical application, one can use both types of marker together to obtain localization and to build better accuracy maps. In the second experiment, we evaluate the system performance to obtain a high-resolution approximate location information using same markers. First, we take image samples of the corridor at 8 MP resolution. Then, we run our algorithm to detect the markers in the high-resolution images. The Root Mean Squared Error (RMSE) in feet vs. distance in feet is plotted to evaluate the performance

of the markers in Figure 4.7. From the test results, we see that the tracking error gradually increases over the distance for both the markers. The error bars show two standard deviations in the plots. The RMS error is about 1 feet at 60 feet for both marker types which is much better than the low-resolution mode. The curves seem to be constant up to 40 feet and are at below 0.5 feet RMS error.

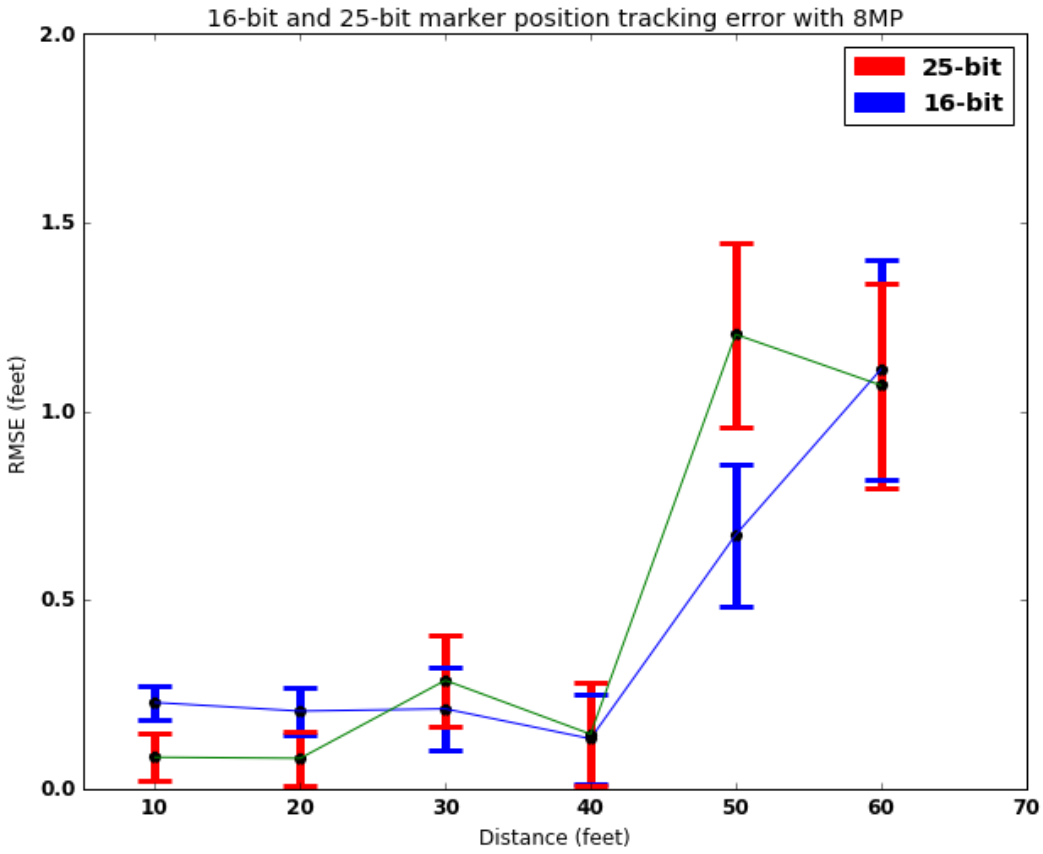


Figure 4.7 16-bit and 25-bit marker position tracking error for 8MP camera resolution

However, the downside of this mode is the amount of time to capture the image and detect and estimate the pose. The Logitech USB camera does not support real-time video recording at this resolution and only supports image capture mode. The total time to capture and run the algorithm may take up to 1200 ms. This may not be used for fast

moving mobile robots to localize and map its environment. However, different localization and mapping policies can be established to enforce a mobile robot to take advantage of the high-resolution mode.

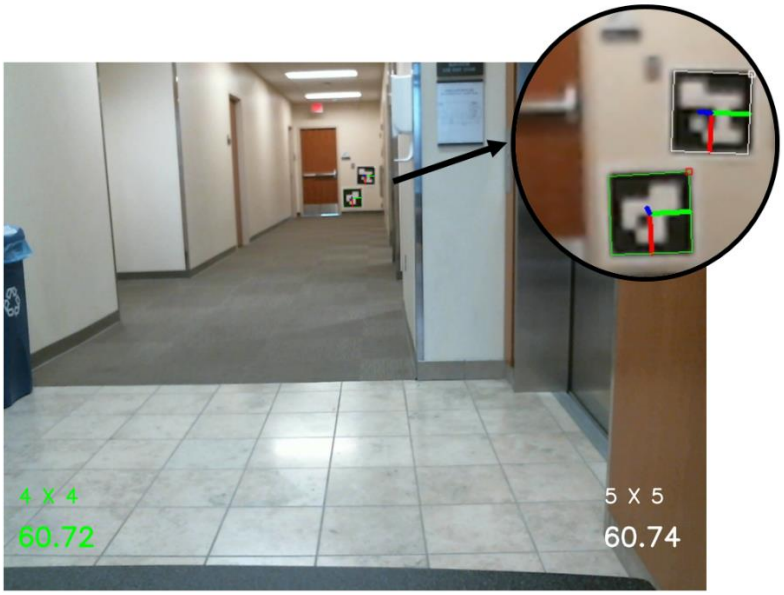


Figure 4.8 Marker detection in high-resolution mode at 60 feet

The sample marker detection in high-resolution mode is shown in the above Figure 4.8. The estimated distance for both markers is shown in the bottom of the image. Green fonts represent the 4 x 4 marker estimation, and white fonts represent the 5 x 5 maker estimation. Using one marker and its coordinates information, a robot can approximately localize in an environment.

4.2.3 Marker based Localization Experimental Results

This part of the chapter focuses on answering two major questions of any autonomous robot should have: “Where am I on the map?” and “What is around me?”. Once the autonomous robot has those answers, navigation becomes very well-organized, and the system can plan effectively to accomplish various tasks autonomously. In this

subsection, we experiment of using multiple markers in the multi-resolution modes to estimate the location and distance to surrounding objects very precisely. To imitate the robot localization, we took several videos and images of a corridor at the height of approximately 1.2 meters. For a long-term autonomy, the markers can be placed at the height of the mobile robot. Few different resolution and size markers were placed on the walls of the corridor as shown in Figure 4.9. Localization of a robot in high-resolution mode is shown in Figure 4.10. The ArUco library pose estimation function returns the pose of the marker in terms of translational and rotational vectors around X, Y, and Z axis. Rotation matrix was obtained by using Rodrigues rotation transformation. The converted rotation matrix was used to obtain Euler's and quaternions angle representations. With the use of quaternion angles, one can transform the co-ordinate reference frame from camera system to ArUco marker's co-ordinate system.

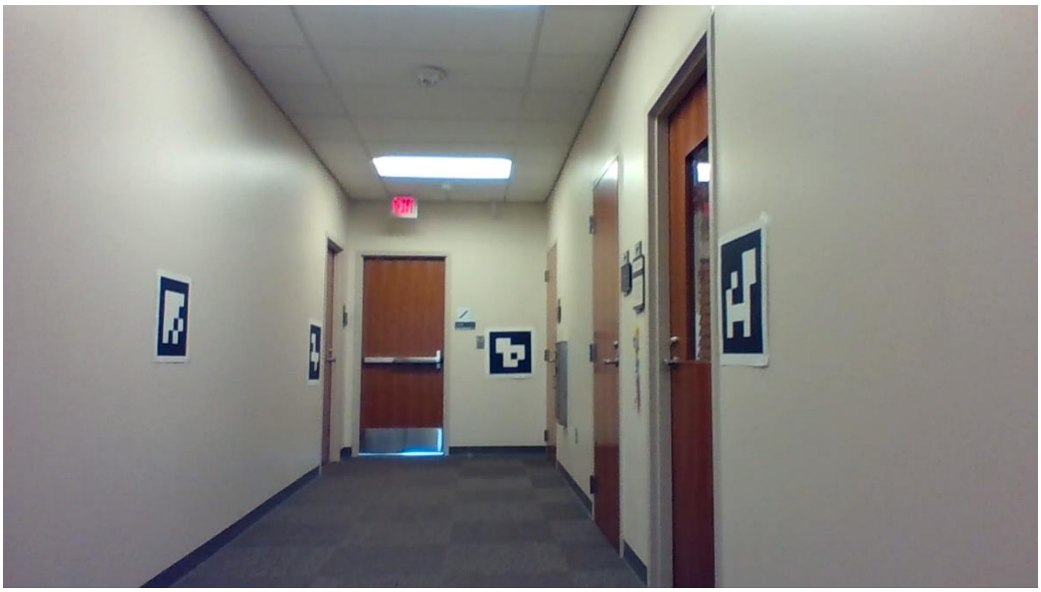


Figure 4.9 ArUco markers' placement in the corridor

Figure 4.10 Localization in the corridor using ArUco markers. Locations of all the markers in camera coordinate system and estimated robot locations are displayed below the marker IDs.

Table 1. Mean estimated robot location from various markers

Robot Location (X,Y,Z) (feet)	**Mean estimated robot location from various markers (feet)**			
	Marker ID: 0	**Marker ID: 1**	**Marker ID: 5**	**Marker ID: 6**
(4,3.3,10)	(3.9,3.16,9.97)	(3.97,3.12,9.7)	(3.95,3.20,9.8)	(4.01,3.1,10.03)
(3.2,3.3,14)	(3.19,3.03,14.1)	(3.39,3.27,14)	(3.1,3.09,14.03)	(3.29,3.0,13.95)
(5,3.3,20)	(5.13,3.2,19.87)	(4.87,3.4,19.8)	(4.70,3.49,20.1)	(5.1,3.16,20.21)

The captured information was then processed to detect the ArUco markers and to estimate their pose relative to the camera coordinate system. The values displayed at the bottom of Figure 4.10 denote the robot location, i.e. (3.26, 2.97, 13.9) feet represents the (X, Y, Z) in the reference coordinate system. Here, the origin is considered at left bottom corner of the front wall. Direction from the wall to the robot is considered Z axis. Other X and Y axis considered in left to right and bottom to top directions respectively. The experiment was performed to evaluate the performance of the system. Results from the experiment is shown in Table 1. This result is very promising and validates the working system of approximate localization using monocular images and ArUco markers.

Using ArUco library, system can generate 1024 unique markers of size 4 x 4. This can also help to augment the localization information by using predefined marker IDs for important aspects of the environment like dead-end, path turn right, path turn left, T-shaped end, glass doors etc. For example, marker IDs 13 and 201 can be used at all dead-ends and T-shaped ends in the environment respectively. Thus, markers can also be used for storing architectural information about the environment. This can be explored in future work.

4.3 Discussion and Conclusions

Map generation and localization are critical functions required in a robot that has autonomous navigation function for the system to autonomously navigate and accomplish the tasks. Obtaining approximate localization information using on-board monocular cameras is not an easy task. While, depth cameras, stereo cameras, and LiDAR can be used to localize and map the environment very precisely, they are computationally expensive and require a significant amount of space to store the high dimension maps.

The purpose of this experiment was to augment traditional localization techniques by using monocular images to have fast and coarse localization solutions in an unknown environment. The experiments presented in this chapter can be useful to explore an unfamiliar environment rapidly.

The accuracy and system performance regarding the angle with the optical axis can be performed in future work. However, the best accuracy is achieved at an optical axis as further away the marker is rotated it gets squeezed nonuniformly. In this regard, multiple markers can be used in the environment which can help to introduce the angular invariance in the pose estimation. Furthermore, the above all results were produced by static tests, i.e., images were taken at every ten feet and by keeping the camera steady. Later, a moving mobile robot can be used to collect data for further performance evaluation of these markers with respect to the speed of the robot.

Chapter 5: Conclusions of this Thesis and Scope of Future Work

5.1 Conclusions

The design of rapid and low-cost localization techniques for autonomous robots to navigate in an unknown or partially known environment was addressed in this thesis. Commonly used SLAM techniques using LiDAR and depth cameras are computationally demanding and can be slow depending on the complexity of the environment. Advanced sensors like LiDAR do not work well in the presence of the glass walls in the environment, whereas sensors like depth cameras have limitations when used outdoors. Further, these techniques ae inadequate for time-constrained applications such as search and rescue. Traditional computer vision algorithms also do not efficiently recognize a scene when required to operate under varying lighting conditions, object orientation, object occlusion, etc. These issues can hinder execution of localization and navigation tasks.

In this thesis, a strategy using the DCNNs coupled with ArUco planar markers was developed and demonstrated to address the problem of rapid and efficient localization technique for an autonomous robot to traverse through an unknown environment. The implementation carried out in this thesis had the following components:

(a) Various types of indoor scene images were collected and preprocessed for the problem of scene recognition (Chapter 3).

(b) The architectures of DCNN were developed for image classifiers to recognize indoor spaces like buildings, floors, and corridors.

(c) DCNN models were trained using computing resources such as GPUs in order to recognize buildings, floors, and corridors spaces using a single monocular image (Chapter 3).

(d) The approximate location information of a robot in the environment was generated using state-of-the-art computer vision algorithm and ArUco planar markers (Chapter 4).

The experimental results show the following.

(i) The trained DCNN model for buildings classification achieved more than 75% accuracy.

(ii) The floor level classifier achieved approximately 49% accuracy.

(iii) The corridors classifier achieved very high 88.7% accuracy.

(iv) In low-resolution mode, the algorithm detects the ArUco marker with an RMS error of 3.5 feet when at a distance of 60 feet. In the high-resolution mode, the algorithm detects the ArUco markers with an RMS error of 1 foot at about 60 feet.

(v) The 4x4 ArUco marker showed better accuracy compare to the 5x5 marker. However, the detection rate of the 4x4 marker was not reliable beyond a distance of 55 feet.

(vi) Multiple markers can be placed on walls and/or important objects to obtain approximate localization information and to detect objects in a structured environment.

The results in this thesis demonstrate that DCNN algorithm coupled with ArUco marker algorithm can be used to localize in the unknown environment without resorting

to high-end computational resources and cost. The ArUco marker algorithm is useful for rapid localization in a partially known or unknown structured environment. While the results show the feasibility of the proposed approach, further studies are to be conducted to improve on the classification accuracy. One of the issues that affected the accuracy of the classification algorithms developed in this research is the similarity in the appearance of several buildings and their interiors.

The results of the research reported in this thesis demonstrate that low cost, low resolution techniques can be used in conjunction with deep learning techniques to obtain approximate localization of an autonomous robot. It is anticipated that this approach can be subsequently extended to develop low-resolution maps of the environment.

5.2 Limitations and Scope of Future Work

This study was limited to the generation of coarse localization information of the robot using monocular images. Similarity in the interiors of the buildings caused lower than expected classification accuracies. Further study is required to determine ways to improve the accuracy of the classifier. Stereo images provide one such tool to improve the representation of the environment and will be considered in future work. The performance of deep neural network model depends on the size of the dataset used in the development and training of the model. The size of the dataset can be increased to achieve better performance. Semantic segmentation is one of the many ways to identify and find the better static features and has been successfully demonstrated using a deep convolutional neural network for outdoor scenes [30]. The use of small markers and high-resolution cameras can be utilized in the experiment for low-level representation of the

environment. Later, this method can be coupled with multiple convolutional neural network architectures to construct the low-level representation of surroundings.

References

1) E. Zamora and W. Yu. Recent Advances on Simultaneous Localization and Mapping for Mobile Robots. IETE Technical Review Vol. 30, Issue 6, 2013.
2) S. Lowry, N. Sunderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford. Visual Place Recognition: A Survey. IEEE. Transactions on Robotics (TRO), Vol. 32, pp. 1–19, 2016.
3) W. Chen, T. Qu, Y. Zhou, K. Weng, G. Wang, and G. Fu. Door Recognition and Deep Learning Algorithm for Visual Based Robot Navigation. IEEE International Conference on Robotics and Biomimetics IEEE ROBIO 2014, pp. 1793-1798, 2014.
4) A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Network. In Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.
5) J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun Database: Large-Scale Scene Recognition from Abbey to Zoo. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2010.
6) A. Quattoni and A. Torralba. Recognizing Indoor Scenes. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009.
7) S. H. Khan, M. Hayat, M. Bennamoun, R. Togneri, F. A. Sohel. A Discriminative Representation of Convolutional Features for Indoor Scene Recognition. IEEE Trans. Image Process., vol. 25, no. 7, pp. 3372-3383, Jul. 2016.
8) R. Jain, R. Kasturi, and B. G. Schunck. Machin Vision Published by McGraw-Hill, Inc., ISBN 0-07-032018-7, 1995.

9) Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, Vol. 86, pp. 2278–2324, 1998.

10) Online Stanford Course: CS231n Convolutional Neural Networks for Visual Recognition. URL http://cs231n.stanford.edu/

11) N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research Vol. 15, p1929-1958, 2014.

12) Diederik P. Kingma, and Jimmy Ba. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 1–13. 2015.

13) C. Cortes, V. Vapnik. Support-vector Network. Machine Learning, Vol. 20, pp. 273-297, 1995.

14) R. Silva, J. C. Oliveira, G. A. Giraldi. Introduction to Augmented Reality. http://virtual.lncc.br/~rodrigo/links/AR/node19.html. 2003.

15) S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marın-Jimenez. Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion. Pattern Recognition, Vol. 47, Issue 6, pp. 2280–2292, 2014.

16) H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99), USA, 1999.

17) M. Fiala. ARTag, A Fiducial Marker System Using Digital Techniques. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 2, pp. 590–596, 2005.

18) L. Naimark and E. Foxlin. InterSense Incorporated. IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2002.

19) G. Dudek and M. Jenkin. Computation Principles of Mobile Robotics. 2nd Edition. Cambridge University Press, 2010.

20) K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. arXiv:1703.06870, 2017.

21) J. Biswas and M. Veloso. Localization and Navigation of the CoBots Over Long-term Deployments. The International Journal of Robotics Research. Vol 32, Issue 14, pp. 1679 – 1694, 2013.

22) A. Wilson. Choosing a 3D Vision System for Automated Robotics Applications. Vis. Syst. Des. Vol. 19, 2014. URL: http://www.vision-systems.com/articles/print/volume-19/issue-11/features/choosing-a-3d-vision-system-for-automated-robotics-applications.html

23) K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2015.

24) Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp 249-256, 2010.

25) T Lozano-Perez. Spatial Planning: A Configuration Space Approach. IEEE Transactions on Computers, 108-120, 1983.

26) J. Borenstein and Y. Koren. Real-time Obstacle Avoidance for Fast Mobile Robots. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, Issue 5, pp. 1179-1187, 1989.

27) O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots.33 International Journal of Robotics Research, Vol. 5, Issue 1, pp 90-98, 1986.

28) I. Horswill. Polly: A Vision-Based Artificial Agent. Proceedings of the Eleventh National Conference on Artificial Intelligence (AAA1'93), The MIT Press, 1993.

29) H. F. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I. IEEE Robotics and Automation Magazine, Vol. 13, Issue 2. Pp 99–110, 2006.

30) T. Bailey and H. F. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II. Robotics and Autonomous Systems (RAS), Vol. 13, Issue 3. Pp 108–117, 2006.

31) C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. IEEE Transactions on Robotics Vol. 32, Issue 6, pp 1309-1332, 2016.

32) A. Kundu, V. Vineet, and V. Koltun. Feature Space Optimization for Semantic Video Segmentation. Computer Vision and Pattern Recognition (CVPR), 2016.

33) S. Thrun W. Burgard D. Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005.

34) YouTube Video - New CoBots are 'help on wheels' - Science Nation. Channel: National Science Foundation. URL: www.youtube.com/watch?v=fJwBU51Rri8

35) J. A. Castellanos, J. M. Martnez, J. Neira, and J. D. Tardós. Experiments in Multisensor Mobile Robot Localization and Map Building. Proceedings of 3rd IFAC Symposium Intelligent Autonomous Vehicles, pp. 173–178, 1998.

36) K. S. Chong and L. Kleeman. Feature-based mapping in real, large scale environments using an ultrasonic array. Int. J. Robot. Res., Vol. 18, Issue 1, pp. 3–19, 1999.

37) D. Crisan and A. Doucet. A Survey of Convergence Results On Particle Filtering Methods For Practitioners. IEEE Trans. Signal Processing, Vol. 50, Issue 3, pp. 736–746, 2002.

38) A. J. Davison, Y. G. Cid, and N. Kita. Real-time 3D SLAM with Wide-angle Vision. Proc. IFAC/EURON Symposium Intelligent Autonomous Vehicles, 2004.

39) G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csobra. A Solution to the Simultaneous Localisation and Mapping (SLAM) Problem. IEEE Trans. Robot. Automat., Vol. 17, Issue 3, pp. 229–241, 2001.

40) M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous Localization and Map Building in Large-scale Cyclic Environments using the Atlas Framework. Int. J. Robot. Res., Vol. 23, Issue 12, pp. 1113–1140, 2004.

41) J. Folkesson and H. I. Christensen. Graphical SLAM—a Self-correcting Map. Proceedings of IEEE International Conference on Robotics and Automation, pp. 791–798, 2004.

42) K. Murphy. Bayesian Map Learning in Dynamic Environments. Advances in Neural Information Processing Systems. 1999.

43) S. Thrun, W. Bugard, and D. Fox. A Real-Time Algorithm for Mobile Robot Mapping with Applications to Multi-Robot And 3D Mapping. Proceedings of IEEE International Conference on Robotics and Automation, pp. 321–328, 2000.

44) S. Thrun, Y. Liu, D. Koller, A. Ng, and H. Durrant-Whyte. Simultaneous Localisation And Mapping with Sparse Extended Information Filters. Int. J. Robot. Res., Vol. 23, Issue 7–8, pp. 693–716, 2004.

45) J. Borenstein and Y. Koren. Real-time Obstacle Avoidance for Fast Mobile Robots. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, Issue 5, pp. 1179-1187, 1989.

46) R. B. Tilove. Local Obstacle Avoidance for Mobile Robots Based on the Method of Artificial Potentials. General Motors Research Laboratories, Research Publication GMR-6650, 1989.

47) B. H. Krogh and C. E. Thorpe. Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles. Proceedings of the 1986 IEEE International Conference on Robotics and Automation. pp. 1664-1669, 1986.

48) A. Kelly, Mobile Robotics: Mathematics Models and Methods, Cambridge, U.K.: Cambridge Univ. Press, 2013.

49) M. A. Nielsen. Neural Networks and Deep Learning. Determination Press, 2015.

50) R. Szeliski, Computer Vision: Algorithms and Applications. Springer, New York. 2011.

51) J. Canny. A Computational Approach to Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol 8, pages 679–714, 1986.

52) B. Siciliano, O. Khatib. Handbook of Robotics. 2nd ed. Berlin: Springer Science & Business Media, 2016.

53) N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

54) D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, Vol. 60, pp. 91-110, 2004.

55) I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016.

56) R. Mur-Artal and J. D. Tardos. Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM. Robotics: Science and Systems (RSS), 2015.

57) R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-Time Dense Surface Mapping and Tracking. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11, pages 127–136, 2011.

58) A. Angeli, S. Doncieux, J. Meyer, and D. Filliat. Visual Topological SLAM and Global Localization. In Proceedings of the IEEE International Conference on Robotics And Automation, pp 4300–4305, 2009.

59) A. Howard. Multi-Robot Simultaneous Localization and Mapping Using Particle Filters. In Robotics: Science and Systems, pages 201–208, 2005.

60) J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012, 2012. URL http://www.image-net.org/challenges/LSVRC/2012/

61) J. Biswas, M. Veloso. Depth Camera Based Indoor Mobile Robot Localization and Navigation. International Conference on Robotics and Automation, 2012.

62) K. Ho, P. Newman. Loop closure detection in SLAM by combining visual and spatial appearance. Robotics and Autonomous Systems 54 (2006), pages 740–749, 2006.