CHALMERS
UNIVERSITY OF TECHNOLOGY

Computer Vision and Machine Learning based Navigation

Indoor navigation method of humanoid robot using RGB images

Master's thesis in Systems, Control and Mechatronics and Biomedical engineering

ANOOP SUBRAMANIAN, NATÁLIE ZUBKOVÁ

Department of Electrical Engineering, Division of Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Computer Vision and Machine Learning Based Navigation

Indoor Ńavigation of Humanoid Robot using Monocular Images

ANOOP SUBRAMANIAN, NATÁLIE ZUBKOVÁ

EXAMINER : Irene Y.H. Gu (Chalmers)
SUPERVISORS : Erik Sjödin, Åke Johansson (Ericsson)
Irene Y.H. Gu (Chalmers)

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Computer Vision and Machine Learning Based Navigation
Indoor navigation method of humanoid robot using RGB images
Anoop Subramanian, Natálie Zubková

Examiner: Irene Yu-Hua Gu, Chalmers University of Technology
Supervisors: Erik Sjödin & Åke Johansson, Ericsson
Irene Yu-Hua Gu, Chalmers University of Technology

Department of Electrical Engineering
Division of Automation and Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Telephone +46 31 772 1000

Cover: Humanoid robot Pepper, made by Aldebran robotics [1].

Typeset in LaTeX
Gothenburg, Sweden 2019

Computer Vision and Machine Learning Based Navigation
Indoor navigation method of humanoid robot using RGB images
ANOOP SUBRAMANIAN, NATÁLIE ZUBKOVÁ
Department of Electrical engineering, Department of Automation and Mechatronics
Chalmers University of Technology

Abstract

Nowadays robots and other artificial assistants became popular assets to the households, making the procedures like vacuum cleaning or grass cutting automatized procedure. Those kind of assistants typically get orientated in its surroundings by using the integrated sensors or by creating the map of the current environment, making the sensors crucial part. This project aims to avoid using sensors and maps and instead create a navigation method based only on the RGB images of environments the robot captures.

This thesis focuses on using deep machine learning for classification of the direction where robot can move, using only RGB images, computer vision and neural networks to decide the movement. The thesis tested directional image classification using monocular images, depth maps and semantically segmented images as inputs, and the performance of each method has been compared. In the final stage of thesis the best performing method using semantic segmentation has been chosen and implemented to robot for the practical testing.

The classifier network achieved test accuracy of 93.97% on dataset within Ericsson premises. The classification network produced 82.05% on unseen and drastically different test images obtained through web scrapping, making it a very attractive candidate for the further research and improvement.

Keywords: robotics, deep machine learning, convolutional neural networks, object detection, depth estimation

Acknowledgements

First of all we would like to thank our academic supervisor Irene Yu-Hua Gu, for her feedback on the approach we chose, especially the computer vision parts of the project, and continuous help with problems we were facing on the way.
Secondly, our thanks comes to to our supervisors Erik Sjödin and Åke Johansson, for integrating us to the Ericsson offices, helping solving the problems we faced, weekly meetings with feedback on our project and mainly their encouragement and support they provided on the way, to always keep us motivated.
Lastly, we thank Ericsson for providing such opportunity and all the tools we needed for developing such project.

Anoop Subramanian, Natálie Zubková, Gothenburg, August 2019

Acronyms

CNN Convolutional Neural Network
DNN Deep Neural Network
VSLAM Vision Simultaneous Localization and Mapping
FC Fully Connected Layer
MSE Mean-squared Error
SSD Sum of Squared Differences
SAD Sum of Absolute Differences
SGD Stochastic Gradient Descent
YOLO You Only Look Once
YOLO You Only Look Once
API Application Programming Interface
PSPNet Pyramid Scene Parsing Network
ReLu Rectified Linear Unit
PReLu Parametric Rectified Linear Unit
BN Batch Normalization

Contents

List of Figures

List of Tables

1
Introduction

Robotics is becoming more and more a daily part of our lives, in many cases making our lives easier. In the last decade the robotics industry has created millions of additional jobs led by consumer electronics and the electric vehicle industry, and by 2020, robotics will be a $100 billion worth industry, as big as the tourism industry [8]. Due to this reasons the robotics is important industry which is improving on daily basis and this project is also attempting to add improvement to already existing methods.

1.1 Background

Interest in robots and venture funding for robotics has exploded by more than 10x over the last six years and shows no signs of stopping.There is especially one branch of robotics that is about to become one of the fastest growing in the coming years : Autonomous indoor robots [9]. Safe and robust navigation is a critical function for all indoor robotics. Indoor navigation includes the challenges of obstacle detection, path navigation and self-localization. This is traditionally achieved using a vision-based agent supplemented by sensors such as lasers and sonars for obstacle detection and self-localization and a pre-set path-planning algorithm to assist the navigation of the robot [10]. However, there is an increase in the use of machine learning approaches using Mono- or Binocular vision for indoor navigation. Machine learning, if implemented well, could eliminate the need for expensive sensors and could work with single or twin cameras, hence making it a useful approach to look into.

Robotics history

Despite the fact that first use of the word "robot" occurred in a play R.U.R. (Rossum's Universal Robots) of Czech play writer Karel Čapek in year 1921, the first instances of robots date many more years back. Mechanical devices built to carry out a physical task occurred already around 3000 B.C.: Egyptian water clocks used human figurines to strike the hour bells. In the next centuries such mechanical devices expanded with invention of the pulley and the screw, wooden pigeon that could fly or hydraulically-operated statues that could speak, gesture and prophecy. Robotics inventions reached a relative peak (before the 20th century) in the 1700s; countless automata (i.e. robots) were created during this period. 19th century was also filled with the new robotic creations, such as talking doll by Edison and steam-powered robot by Canadians [11].

Although these inventions may have provided first inspirations to nowadays modern robots, the scientific progress made in 20th century in the field of robotics surpass previous advancements a thousandfold [11]. The first modern robots as we know them were created in the early 1950s by George C. Devol, who transformed modern manufacturing when he invented and patterned a reprogrammable robotic arm. In late 1960s, Joseph Engleberger acquired Devol's robot patent and was able to modify it into an industrial robot and form a company called Unimation to produce and market the robots. For his efforts and successes, Engleberger is known in the industry as "the Father of Robotics." Since that time the research and development of robots expanded enormously [12].

Pepper robot

Pepper is the world's first social humanoid robot able to recognize faces and basic human emotions. Pepper was optimized for human interaction and is able to engage with people through conversation and touch screen [1]. Pepper is available today for businesses and schools. Over 2,000 companies around the world have adopted Pepper as an assistant to welcome, inform and guide visitors in an innovative way. Another commercial use is Pepper serving as receptionist at several offices in the UK, where it is able to identify visitors with the use of facial recognition, send alerts for meetings and chat to prospective clients. Another branch where is Pepper used are banks and medical facilities in Japan, and also at the four hundred branches of Hamazushi restaurants in Japan [13].

Existing robot navigation architectures

Interest in robots and venture funding for robotics has exploded by more than 10x over the last six years and shows no signs of stopping. There is especially one branch of robotics that is about to become one of the fastest growing in the coming years : Autonomous indoor robots [9].
Safe and robust navigation is a critical function for all indoor robotics. Indoor navigation includes the challenges of obstacle detection, path navigation and self-localization. This is traditionally achieved using a vision-based agent supplemented by sensors such as lasers and sonars for obstacle detection and self-localization and a pre-set path-planning algorithm to assist the navigation of the robot [10]. However, there is an increase in the use of machine learning approaches using Mono- or Binocular vision for indoor navigation [14, 15]. Machine learning, if implemented well, could eliminate the need for expensive sensors and could work with single or twin cameras, hence making it a useful approach to look into.

1.2 Limitations

Throughout many investigations it can be concluded that techniques and methods to detect objects depend mostly on the application environment. There are various environments where the navigation is adjusted to the purposes and generalizing the

navigation onto more application environments lead to worse performance [16]. For this reason this work will be focused on indoor navigation environment only.

Time limitation constraints the training of the final network. In order to have a higher chance of better accuracy of the network, the number of categories for classification is reduced as much as possible. The movement of the robot will be limited to -30°to 30°with increments of 10°.

1.3 Ethical issues

Robotics ethics is growing interdisciplinary research with the aim of understanding the ethical implications and consequences of using robotic technology. As developers responsible for robots abilities the focus should be on the ethics and possible risk analysis of abilities we implement. Many areas of robotics are impacted, but in our case the focus will be on areas where robots interact with humans - ranging from elder care to medical robots, as the possible use of this robot could be in those areas [17].

Firstly, the aspect of social situations will be explored. With the social situations the question that arises is how robots should react in not protocol prescribed situation, while they should be allowed to override rules, be capable of employing moral emotions and be capable of some general ethical understanding that can guide their reasoning, decision-making and also justifications of their decisions and actions. Regarding to our work, if the ultimate goal would be to use our implemented robot as an assistance to elderly people, the important question would be if the robot should act according to given protocol at all the occasions or should be able to also perform actions - event based, or behave differently in critical situations.

As there is currently no official list of principles published, there possible solution to ethical considerations rising with this work could be solved using already existing proposals for such a set of principles. Apart of the already existing technical committee (TC) for robot ethics whose aim is to develop set of rules in understanding the ethical implications and consequences of robotic technology and in particular autonomous robots and provide IEEE-RAS with a framework for raising and addressing the urgent ethical questions connected to robotics research. Ever since its inception almost a decade ago in 2004, the TC (in its third generation now) has been involved in organizing various types of meetings (from satellite workshops at main conference, to standalone venues) to call attention to the increasingly urgent ethical issues raised by the rapidly advancing robotics technology [18].

Although TC is putting a lot of focus on ethics, there is not set of directly defined rules predefined, and thus the principles when developing robots abilities are derived from the article [6] that proposes five ethical principles as a basis for responsible robotics. This article with the present principles have influenced and continue to influence a number of initiatives in robot ethics but have not been formally published just yet [6].

Rule	Semi-legal explanation
1	Robots are multi-use tools. Robots should not be designed solely or primarily to kill or harm humans, except in the interests of national security.
2	Humans, not robots, are responsible agents. Robots should be designed; operated as far as is practicable to comply with existing laws fundamental rights freedoms, including privacy.
3	Robots are products. They should be designed using processes which assure their safety and security.
4	Robots are manufactured artifacts. They should not be designed in a deceptive way to exploit vulnerable users; instead their machine nature should be transparent
5	The person with legal responsibility for a robot should be attributed

Table 1.1: Principles for designers, builders and users of robots [6]

1.4 Aim

The aim of this project is to develop a new indoor navigation method in which the robot does not make map of surroundings (which is case of most indoor navigation methods) but decide at the moment and pick the direction of movement at each moment. Such approach will allow the environment not be just static, but when new obstacles like persons come to picture, robot will be able to register those and react accordingly. The other advantage that this solution provides is its price, as for the purpose of navigation only two HD cameras are needed, with no additional use of sensors.

1.5 Purpose and research question

There are many existing architectures on robot navigation in indoor/outdoor environments. However, most of them are just sensor based, or work on the principle of mapping the surrounding and then moving accordingly. One of such robots are also robot vacuum cleaners, which mostly work on the "wall following principle" (where it moves around the walls of your room, using its side-mounted, flailing brush to clean right into corners) and "random bounce" (where it cleans until it hits an obstacle, then moves off again in a random direction). There is also more intelligent approach called Vision Simultaneous Localization and Mapping (VSLAM), where onboard infrared cameras are used to take snapshots of room, gradually building up a picture so it is known where the robot is going and where it has already been.
The approach that this thesis is aiming to develop is a method which provides a cheaper solution, using only two HD cameras, no additional sensors and does not create any map of the surroundings but rather decide in a time of movement.
Therefore, the question that this work strives to answer is this:

Is it possible to make a robot decide on movement according to actual position and actual environment and adapt to case if the environment changes?

2

Theory

This section briefly reviews the theory behind the methods, and theoretical background of the methods which are considered to be useful for implementation.

The solution to indoor robot navigation problem from monocular images can be broken down into three main sub-components: 1) using monocular images, as is 2) generating a depth map from the RGB images which encapsulates localization features and 3) generating semantically segmented images which encapsulates obstacle features. Finally, these three are fed to a classifier neural network for navigation.

The following section details the approach used in the project to achieve each of the sub-problems. For each of the sub-problems multiple approaches have been tested and looked to in order to get the best solution for this specific application of Pepper robot.

2.1 Machine Learning

Machine learning is a scientific study of algorithms which depends on patterns and inferences and not explicit instructional algorithms. This subset of Artificial Intelligence uses a data-driven approach, creating a mathematical model of the sample data set. Machine learning is particularly powerful in areas like computer vision where feasibility of instructional algorithms is limited. The neural network framework [19] is one of the more popular approaches in machine learning.

Neural Networks are loosely based on the neuron-structure in the human brain, consisting of a collection of nodes. These nodes are fed with real numbers and a sum of a non-linear mapping of the nodes produces an output. Each node has a weight and a bias which are adjusted or "trained" to adapt to any given problem. A simple Neural Network is illustrated in Figure 2.1.

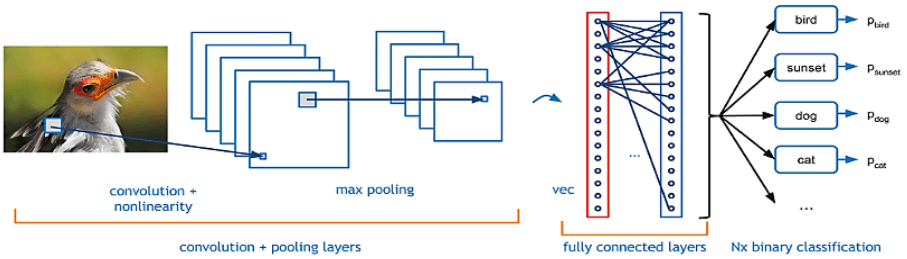


Figure 2.2: A typical example of CNN used for image classification

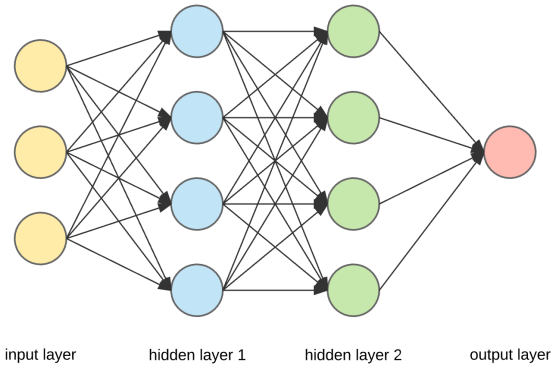


Figure 2.1: Model of a simple two-layer Neural Network

Between the input and the output layers, are the collection of nodes called "layers". The number of layers needed is not governed by any rule or law, it is in the full control of the designer of the neural network. A sub-category of neural networks, the deep neural network (DNN), extends this to include multiple hidden layers. DNN works by extracting simple features from the input, and using combinations of the simple features to generate more complex features. This results in the ability to fit linear and non-linear problems of any complexity.

2.2 Convolutional Neural Networks: Theoretical Framework

The Convolutional Neural Network (CNN) is a class of deep neural network (DNN), mostly used in image analysis. The core block of CNNs are the convolutional layers. The convolutional layers operates using learn-able filters which slide across the input and "learn" to capture the feature map for the particular hidden layer.

A single stage of CNN feature extraction is illustrated in figure 2.3. The hyperparameters available for tuning CNNs are the number of filters used in the convolutional layers, filter shape - typically chosen depending on the problem set and size of pooling - to reduce the spatial dimension.

Forward Propagation

Forward propagation is calculation of all intermediate values and the output for the network model. This forms the basis of training and further more prediction. For any hidden layer l, the output of the hidden layer $z_j^{[l]}$ is calculated by (2.1).

$$z^{[l]} = \sum_{k \in \mathbf{N}} w_k^{[l]} \cdot a_k^{[l-1]} + b^{[l]} \tag{2.1}$$

where $\mathbf{N}$ is the number of nodes in the hidden layer, $w_k^{[l]}$ is the weight of the filter at layer l, $a_k^{[l-1]}$ is the value of the corresponding spatial location from the previous layer $l-1$ and $b^{[l]}$ is the bias at layer l. The scalar $z^{[l]}$ is passed through an activation function to add non-linearity to the system.

$$a^{[l+1]} = h(z^{[l]}) \tag{2.2}$$

Activation functions are necessary to add non-linearity to the system[20]. The non-linearity is required to allow the system to learn more complex input signals, a linear mapping between the input and output signals limits the complexity of the system. Convolution operation stacks the forward propagation by the kernel volume, and slides it across the input space to produce the feature map. The feature map is generated through the three operations of weighting, biasing and activation. The feature map generated from the kernels can be visualized in the top right corner of Figure 2.3 for the example of image face recognition.

For image classification, the final convolution stage is proceeded by fully connected layers (FC). The fully connected layers are a vector of perceptrons. The perceptron performs the sum-of-products operation on the inputs $x_1, x_2 ... x_n$ and outputs the following $z = \sum_n^{i=1} w_i x_i$. The vector of perceptrons is called a fully connected layer since all such perceptrons are connected to all inputs to the layer, as illustrated in Figure 2.2.

Running the image through convolution stages and FC layers and performing the sum-of-products $y = h(\sum_n^{i=1} w_i x_i + b_i)$, where $h()$ is the activation function, w_i are the weights, and b_i are the biases constitutes the forward propagation. This is performed on the data to obtain the predictions.

Backward Propagation

To the finalize the weights and biases, the network has to be trained on annotated data, i.e, data with ground truths. To train the neural network, backward propagation must be performed.

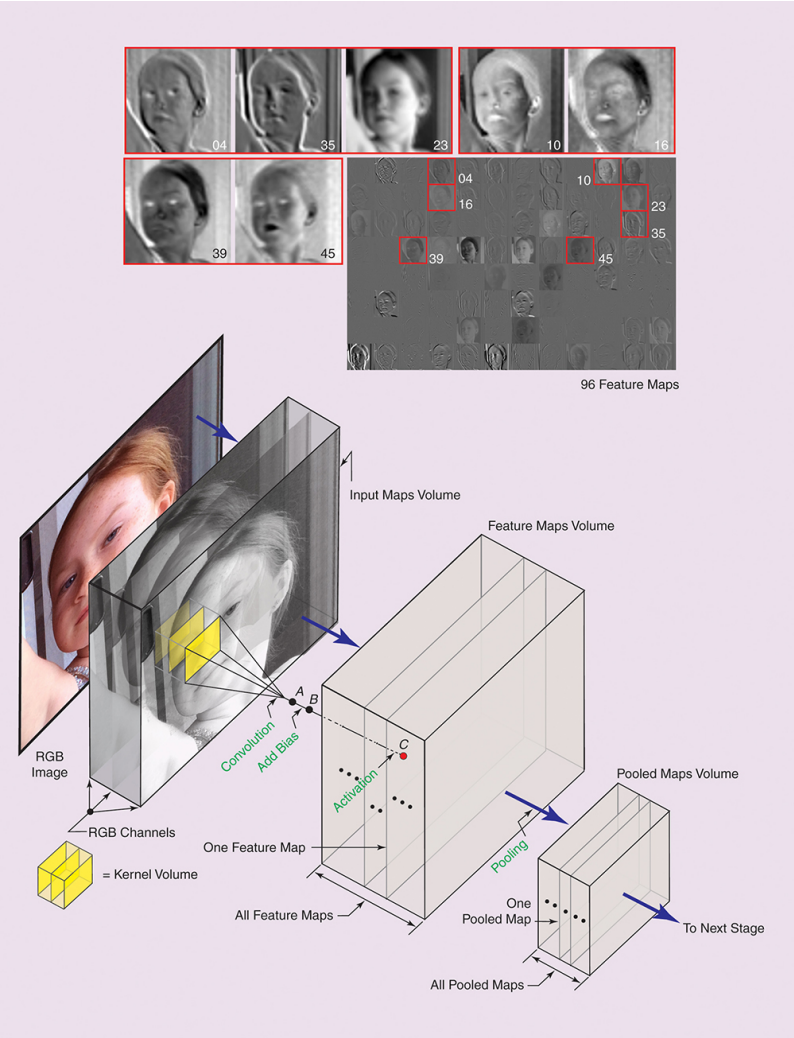


Figure 2.3: Single stage of a CNN [2]

After the calculation of the loss or error E, the training of the neural network becomes an optimization problem for tuning the weights and biases. The optimization problem can be rephrased as the minimization of error by adjusting the weights and biases. And this is achieved, most generally, by gradient descent.

$$w_{ij}(\ell) = w_{ij}(\ell) - \alpha \frac{\delta E}{\delta w_{ij}(\ell)} \tag{2.3}$$

$$b_{ij}(\ell) = b_{ij}(\ell) - \alpha \frac{\delta E}{\delta b_{ij}(\ell)} \tag{2.4}$$

Where, α controls the rate of gradient descent, known as learning rate. The method propagates the error backwards from the final FC layer through all the previous layers, hence termed as back-propagation. To get the differential of error with respect to the weights, $w_{ij}(\ell)$, and biases $b_{ij}(\ell)$, the following calculation is required:

$$\frac{\delta E}{\delta w_{ij}(\ell)} = h_j(\ell - 1)\Delta_i(\ell) \tag{2.5}$$

$$\frac{\delta E}{\delta b_{ij}(\ell)} = h_j(\ell - 1)\Delta_i(\ell) \tag{2.6}$$

where,

$$\Delta_j(\ell) = h'(z_j(\ell)) \sum_i w_{ij}(\ell + 1)\Delta(\ell + 1) \tag{2.7}$$

$$\Delta_j(L) = h'(z_j(L))[h_j(L) - t_i] \tag{2.8}$$

The quantities for 2.8 is readily available during the training phase as $h_j(L)$ is calculated from the forward propagation and t_i is available as the ground truth, and working backwards from that 2.7 can be sequentially calculated. These iterations are performed for a given number of epochs or until the error falls within in a predetermined range.

2.3 CNN: Practical Guide

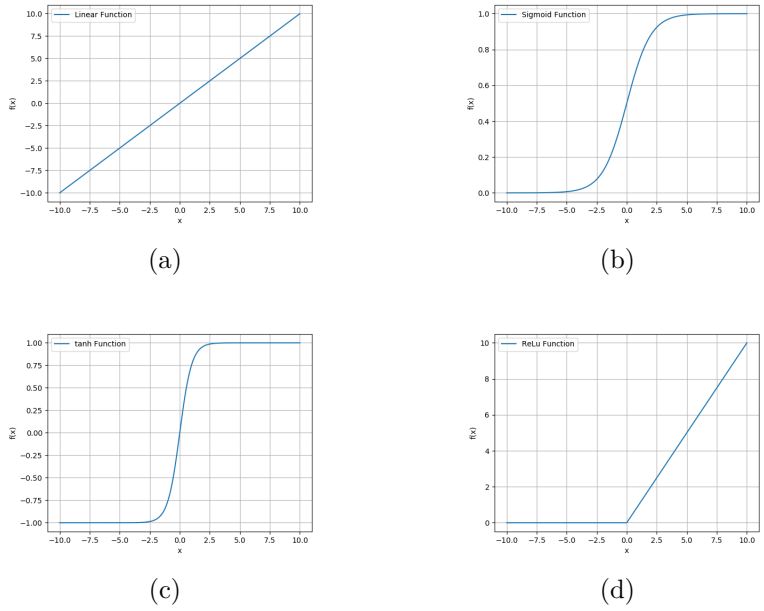
This section discusses the features and options available for designing the optimal CNN for the application. The section dives into the various activations, loss calculations, regularizers, and other options which help fine tune the model. All available frameworks and libraries for deep learning, such as Theano, Caffe, Tensorflow, Torch and MXNet, out of which Keras remains the most popular library, built on frameworks such as Theano and Tensorflow. The project develops the architectures on Keras, running Tensorflow backend.

2.3.1 Activations

Activation functions used in practice includes, typically, sigmoids $h(z) = 1/(1 + exp(-z))$, rectified linear units (ReLu) $h(z) = max(0, z)$, and hyperbolic tangents $h(z) = tanh(z)$. These functions are illustrated in figure 2.4.
Linear functions do not add any non-linearity to the system and are, therefore, almost never used. Sigmoid outputs a range from 0 to 1, making it ideal for binary classification problems, and is usually used in the final layer for classification. However, Sigmoids have a low convergence rate and not being zero-centered, the optimization can lead it in different directions. An alternative is to use tanh, which being zero-centered, makes the optimization much faster. ReLu provides the simplest mathematical form and gives the best convergence rate.
Sigmoid is ideal for binary classification, extending which to Softmax $\sigma(z) = \frac{e^{z_i}}{\sum_{j=0}^{K} e^{z_j}}$ where z_i is an element from the input vector z and $[0..K]$ are the classes, makes it ideal for multi-class classification. ReLu is ideally used in the hidden layers.

2.3.2 Pooling Layer

A pooling operation, or sub-sampling, is performed to downsize the image. This is performed to reduce the resolution of the image for the next stage, which allows for



(a) (b)

(c) (d)

Figure 2.4: Activation functions: (a)Linear (b)Sigmoid (c) tanh (d) ReLu, the commonly used functions

faster processing. However, this increase in performance requires an accuracy trade-off, hence performed only sporadically in the model architecture. The commonly used pooling operations are Max Pooling and Average Pooling [21]. Average Pooling computes the average of the the filter values $a_j = mean\{s_1, s_2, s_3....s_n\}$ where s_i is the values in the filter matrix. Max Pooling generates the maximum of the filter values, $a_j = max\{s_1, s_2, s_3....s_n\}$. Max pooling has proven to perform better.

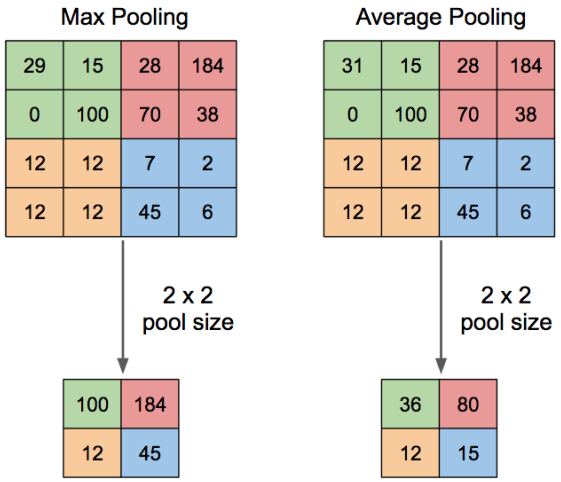


Figure 2.5: Demonstration of the two types of pooling operations

2.3.3 Batch Normalization

Batch Normalization increases the speed of training and the robustness of the model but normalizing the inputs in the hidden layers. Normalization of the hidden layers, reduces the internal co-variance shift, which allows for a more generalized learning of the problem.

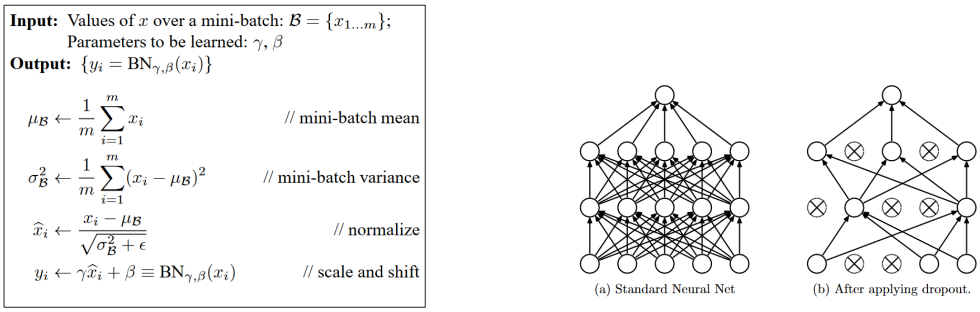


Figure 2.6: Batch Normalization (a) and Dropout (b) as methods to improve performance and robustness of the neural network

2.3.4 Dropout Layer

Dropout is one of the simplest and most elegant method to reduce over-fitting of the neural network model [22]. Dropout means temporarily dropping a few hidden units and it's input and output connections. The dropout layers takes a probability as input, and the hidden units are dropped in that probability.

2.3.5 Loss Functions

Loss functions are the most critical parameters for the well functioning neural network. The loss functions generate the error, which trains the neural network through backward propagation. The most commonly used loss functions are mean squared error (MSE), Hinge loss and cross-entropy loss.

$$E_1 = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{2.9}$$

$$E_2 = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1) \tag{2.10}$$

$$E_3 = -(y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)) \tag{2.11}$$

MSE (2.9), the average of sum of squared error, is most commonly used for logistic regression. It is a simplistic model and works well with gradient descent. Hinge loss(2.10) tries to ensure that the score of the correct category is higher than the sum of scores of wrong categories. It is particularly powerful for binary classification, however it doesn't perform well in multi-class classification. The cross-entropy

loss(2.11) increases as the probability of the wrong class increases. It heavily penalizes the highest probability but wrong class prediction. Cross-entropy loss is most commonly used for multi-class classification.

2.3.6 Optimization

With the error calculated, the objective now is to estimate the weights and biases. This is achieved using optimization techniques. While there are many optimization techniques available, the most commonly used and best performing optimizers for image-classification are Stochastic Gradient Descent (SGD), RMSProp and Adam.

$$SGD : \theta = \theta - \alpha \Delta_\theta J(\theta; x_i, y_i) \tag{2.12}$$

$$RMSProp : v = \eta \cdot v + (1 - \eta) d\theta^2 \tag{2.13}$$

$$\theta^{k+1} = \theta^k - \mu \frac{d\theta}{\sqrt{v} + \delta} \tag{2.14}$$

$$Adam : m = \beta_1 m + (1 - \beta_1) d\theta \tag{2.15}$$

$$v = \beta_2 \cdot v + (1 - \beta_2) d\theta^2 \tag{2.16}$$

$$\theta^{k+1} = \theta^k - \mu \frac{m}{\sqrt{v} + \delta} \tag{2.17}$$

$$AdaGrad : \theta^{k+1} = \theta^k - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{T} g_{\tau,i}^2}} \cdot g_i \tag{2.18}$$

where θ is the parameter to be updated, α is the learning rate, $J()$ is the loss, v is the sum of all squared gradients, η is the decay rate, δ is a small value to avoid division by zero, m is the momentum and β_1, β_2 are the decay weights for Adam and $g_i = \nabla \mathbf{Q}_i(w)$ is the gradient at iteration τ.
The choice of optimizer depends on the architecture and application, and while all three are equally good for image-classification, Adam and AdaGrad are generally preferred for deep learning.

2.4 CNN Architectures

CNN are extremely powerful when dealing with images, for both object detection and image classification. As the project uses CNN for object detection and for path planning (reformulated to image classification), this section discusses the available CNN architectures.

2.4.1 Vanilla CNN

A vanilla-CNN model is a simple CNN architecture with typically 3-6 convolutional blocks, and a total of 4-8 hidden layers with dropouts, pooling and normalization operations added to suit the application. This type of architecture performs very well for a small number of classes and does not overfit the system.

This architecture is usually the best starting point for any architecture, and can be used for most simple image classification problems. However, to get good performance and accuracy the architecture relies on a large dataset which good distribution among classes and good quality images.

2.4.2 Deep CNN for Object Detection

For more complex problems which require a more rigorous feature extraction from the input space, CNNs with more than 10 layers are typically used. Datasets such as PASCAL VOC [23] with 5304 images and 9507 annotated objects in the VOC2006 dataset or COCO [24] with 328k images and 2.5million labelled object instances cannot be used well with shallow CNN. With more layers, the architecture is able to draw and learn high-dimensional features separately, making it ideal for object detection and image classifications.

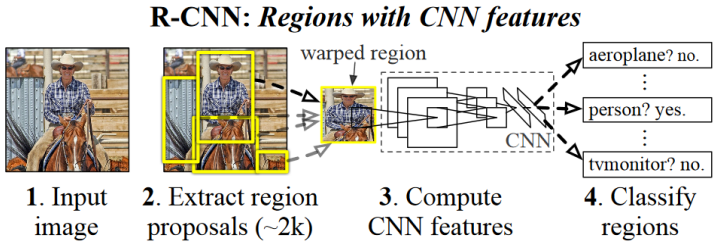
There are many deep CNN architectures designed specifically for image classification, the most popular ones being VGG16 and VGG19[25] with 21 and 24 hidden layers, GoogleNet or InceptionV1[26] with 22 hidden layers with inception modules within them, ResNet[27] with 18, 34, 50, 101 and 152 layer variants and MobileNet[28] with 30 hidden layers. These networks have achieved good results such as top-5 error rate of 3.57% with ResNet, 6.67% for GoogleNet/InceptionV1 and 7.5% for VGG16. The next subsection discusses how these networks can be used to build a powerful custom network for any required application.

The most popular deep CNN for object detection are R-CNN [29], Fast-RCNN[30], Faster R-CNN[31] and YOLO[3]. These networks rely on deep layers in the CNN and trained on huge datasets to be able to perform object detection. RCNN, Fast R-CNN and Faster R-CNN rely on a region of interest (RoI) propsal method which then feeds into a CNN to detect objects. YOLO on the other hand uses a single convolutional network to predict the bounding boxes and the object confidence. The image is split into a SxS grid and in each grid, m boundary boxes are taken. For each of the boundary box, YOLO outputs a class probability and offset value. The boundary boxes with higher probability than a predetermined threshold value are used to detect objects.

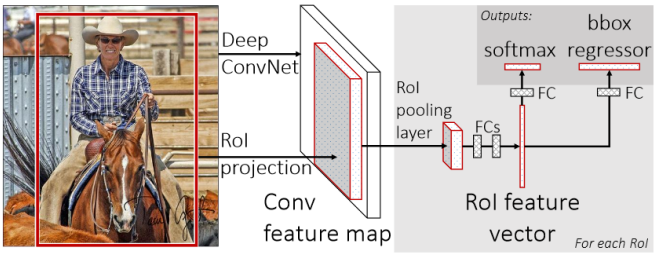
2.4.3 Transfer Learning

Most real world problems do not have access to good quality and larger datasets or the computational power to train large networks, which limits the performance of any standalone CNN developed from scratch. Transfer learning is the simplest and most popular method that provides a time-saving solution to image classification problems [32].

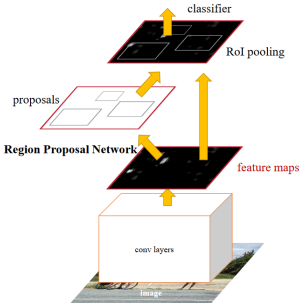
Transfer learning depends on using a trained model as the base for feature extraction from images and add a few trainable layers on top of the base, which adapts the system to the custom application as demonstrated in Figure 2.8.



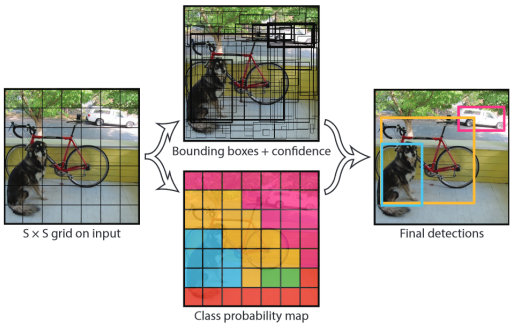
(a) RCNN



(b) Fast RCNN



(c) Faster RCNN



(d) YOLO

Figure 2.7: Overview of Deep CNN architectures for object detection [3]

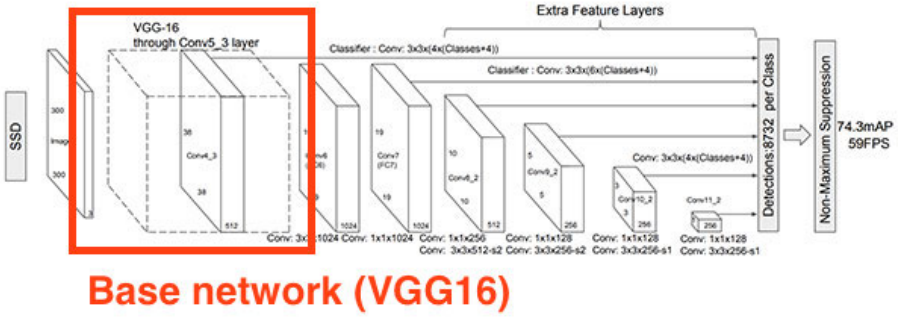


Figure 2.8: An example of transfer learning architecture [4]

2.5 Depth Estimation

Computer vision is a field of science that deals with enabling computers to see, identify and process images, trying to mimic the way that human vision works, and then provide appropriate output. As the computer must interpret what it sees and perform the appropriate analysis or act accordingly, computer vision is closely linked with artificial intelligence [33]. There are many applications where computer vision is used, the main clusters can be listed as retail and security, automotive, healthcare, agriculture, banking and industrial branch [34]. Many various methods and algorithms are used in computer vision, however, the method that will the project focus on is depth estimation, which is used to obtain idea about the robot's visual field. Depth estimation or extraction refers to the set of techniques and algorithms aiming to obtain a representation of the spatial structure of a scene [35]. It is fundamental task to estimate the positions of objects in the robot's visual field and depth estimation is a method that will be used to achieve that. There are several approaches how to get depth information from the images, and thus several approaches have been investigated and tested as explained in the following subsections.

Triangulation

Firstly, the method that has already been used and tested in robot of the same manufacturer, similar to Pepper, called NAO has been looked into. This approach has been used in soccer game using NAO robots, where the robots were playing with a red ball and at each video captured frame the distance to ball is calculated [36]. The method of triangulation uses the knowledge about height of the camera and the bearing to the point where the object meets the ground. The bearing is given by the position of the object in the image and the known orientation of the camera relative to the ground. For this purpose the dimensions and angles of cameras integrated in robot which are known are extracted to calculate the distance. The height of the camera h is fixed, and α is the camera view to horizontal line and varies with respect to the distance of the object to camera.

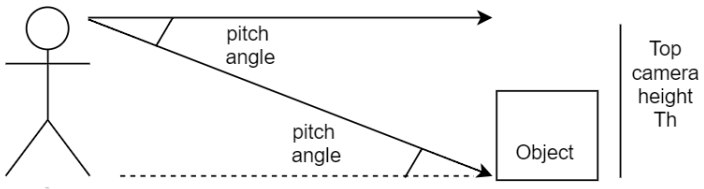


Figure 2.9: Demonstration of trigonometric properties to determine distance to object

As illustrated in the figure 2.9 the distance to the object can be easily calculated

using basic trigonometric formulas. There are two possible ways how to obtain the distance - using top or bottom camera, for which the equations are listed below.

$$D_{ctop} = \frac{H_{top}}{tan(\alpha)} \tag{2.19}$$

$$D_{cbottom} = \frac{H_{bottom}}{tan(\alpha)} \tag{2.20}$$

where H_{top} is height of the top camera, H_{bottom} is the height of bottom camera, pitch angle is a value obtained from robot's head, α = pitch angle + rotation angle, where rotation angle is in the case of Pepper robot 0° when the top camera is used, and 40° when the bottom camera is used. The accuracy achieved by this method has been tested and listed at the NAO soccer game article [36], and low absolute errors achieved lead to the conclusion that this method is sufficient enough for navigation. The analysis has been performed by testing how formulas perform for different distances, starting from the furthest point at 1.4m and slowly reducing to 0.1m away from robot, with step size of 0.1m. For each distance the resulting NAO's head pitch angle was obtained and the distances calculated. The results showed that percentage error gets under 10 % for distances larger or equal to 0.4m. The fact that under 0.4m this methods performs worse is not a limitation, as Pepper robot explicitly includes a move security principles [1] that prevent a collision in the case that distance under 0.4m would be wrongly calculated and for distances larger than that the collision should be avoided due to the low percentage error [36].
After investigating application of this method using Pepper robot, it has been realized that this approach uses already build in function the robot has - tracking the red ball, and every time the video frame is captured, pitch angle is extracted from Robot's head and used for calculation. Although Pepper robot also provides this possibility, the aim of this project is to provide solution which does not uses any build-in functions and thus another methods have been examined.

Depth Map Using 3D Camera

Another possible solution to create a depth map is to directly use 3D sensor/camera. When exploring this option two types of camera came into the focus. Firstly, Range camera, which is a device a that produces a 2D image showing the distance to points in a scene from a specific point. The resulting image, the range image, has pixel values that correspond to the distance.
Secondly, stereo camera, which is a type of camera with two or more lenses with separate image sensors or film frame for each lens, which allows the camera to simulate human binocular vision, and therefore capture three-dimensional images.
This sensors would provide easy solution to depth estimation, but after comparing the prices of such cameras and simple 2D cameras, it was concluded that this method would be very expensive.

Neural Networks

The next possible approach to this solution is using neural networks to generate depth map. Using such setup could lead into final solution consisting just of sin-

gle neural network, which would provide an elegant solution. Not many options are present which provide satisfactory results for monocular depth estimation, even fewer for indoor scenes. However, one of the viable options is the DenseDepth [37] network. DenseDepth, as shown in 2.10, is a CNN based encoder-decoder model, which takes a pre-trained and truncated DenseNet-169 as the encoder. The DenseDepth architecture has been trained on the NYUDepthV2 [38] dataset, which consists of a large number of indoor scenes, making this very relevant to the project work.

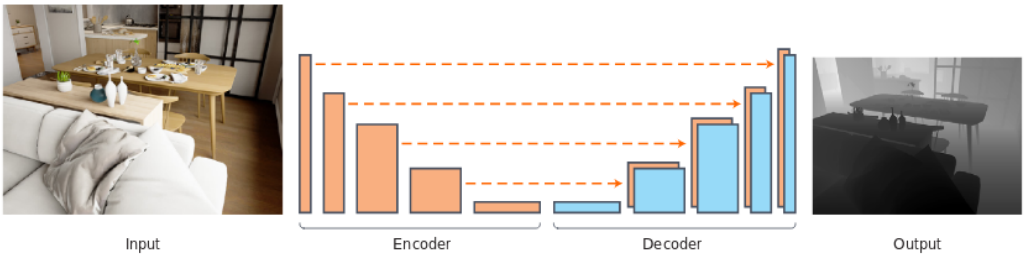


Figure 2.10: DenseDepth architecture

Multiple view geometry

Lastly, the multiple view geometry methods have been investigated and chosen to be a method that will be used for generating the depth map for further application on final neural network, as over the past few years the development of algorithms using multiple view geometry recorded a rapid expenditure, providing an approach which leads to excellent results [39].
There are many tasks and algorithms that can be done using multiple view geometry, but this project will only focus on two horizontally aligned camera view geometry.As mentioned previously, two cameras will be used. The cameras will be aligned horizontally and the optimal distance between them will be experimentally determined. The cameras will provide two views - left view and right view, as demonstrated in figure 2.11. The focal length of cameras is known, and thus the distance **D** to target **T** can be calculated using setup demonstrated in figure 2.11.
Distance **D** can be calculated

$$\frac{b_1}{D} = \frac{-x_1}{f} \tag{2.21}$$

$$\frac{b_2}{D} = \frac{x_2}{f} \tag{2.22}$$

and since $b = b_1 + b_2$:

$$b = \frac{D}{f}(x_1 - x_2) \implies D = \frac{bf}{x_1 - x_2} \tag{2.23}$$

It is possible to use this equation, because $x_1 - x_2$,the difference of two corresponding image points, so called disparity between the two images in pixels, can be calculated

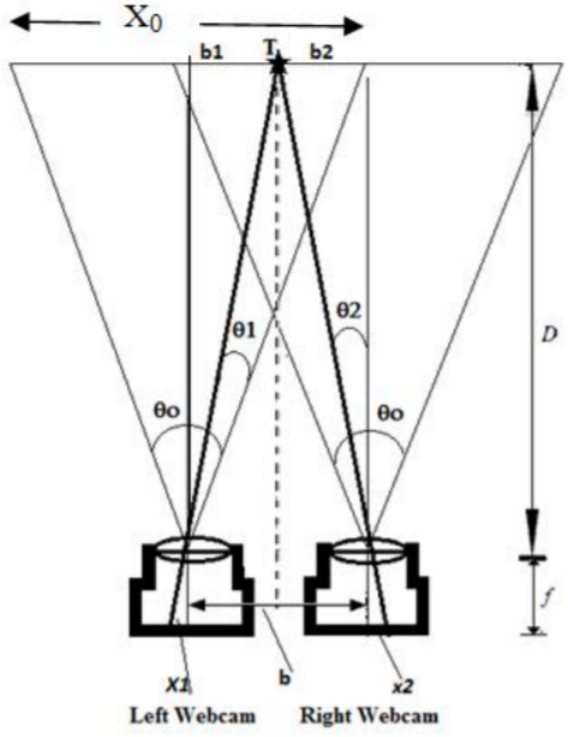


Figure 2.11: Two horizontally aligned cameras setup for distance calculation

using openCV or using basic image analysis formulas.
Another way how to calculate the distance can be using a view angle Θ

$$\tan(\frac{\Theta}{2}) = \frac{x_0}{D} = \frac{x_1}{f} \implies f = \frac{x_0}{2\tan(\frac{\Theta}{2})} \tag{2.24}$$

and from there the distance can be computed as :

$$D = \frac{bx_0}{2\tan(\frac{\Theta}{2})(x_2 - x_1)} \tag{2.25}$$

where x_0 is width of the image in pixels.

2.5.1 Stereo Vision

Stereo vision can be defined as extraction of 3D information from digital images. In traditional stereo vision, two cameras are displaced horizontally from one another to mimic a manner similar to a human binocular vision [40]. Stereo vision tries to imitate the human mechanism by capturing the scene from two horizontally displaced cameras, which will create two slightly different projections of the scenes. If this two images are then compared, the additional information is extracted - depth of the scene. This process of extracting 3D structure from pair of stereo images is called computational stereo, and the result of that is so called disparity

map, which will be used in this project to estimate depth of the scene. This method is widely used in different applications such as vehicle tracking, aircraft estimation and positioning and many other applications [40].

Disparity map

Concluding the information about multiple view geometry and stereo vision the disparity map can be defined and calculated. Disparity map refers to the difference in location of an object in corresponding two images as demonstrated in Figure 2.12.

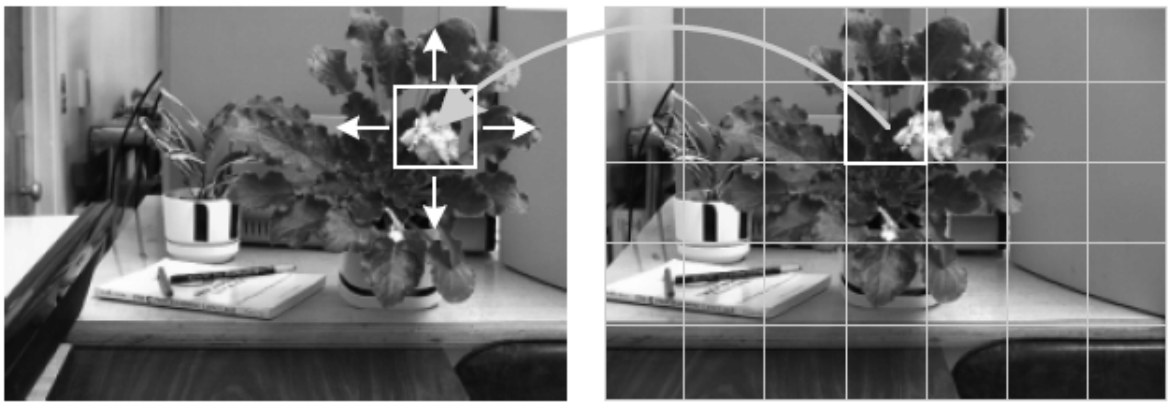


Figure 2.12: Matching images by correlation [5]

One of the simplest methods to calculate a similarity measure is calculated by subtracting pixels in a square area between the reference image I_1 (taken by left camera) and target image I_2 (taken by right camera). The pixel disparity is calculated using equation (2.26). This calculation is the followed by the aggregation of the absolute differences within the square window. If the images coincide exactly, the result will be zero [41].

$$\sum_{i,j \in W} |I_1(i,j) - I_2(x+i, y+j)| \tag{2.26}$$

To sum it up the procedure is thus following : the inputs to the formula are two images of the same scene taken by two horizontally aligned cameras. The images - pixel arrays are then parsed to the formula where the sum of absolute differences is calculated and the resulting output is again an image.

Implementing this algorithm to such taken images and then filtering the noise present, results in the gradient disparity map as presented in Figure 2.16. As can be seen from the Figure 2.16 the resulting image provides an information about depth of the objects in the image.

Figure 2.13: Original image taken by left camera

Figure 2.14: Original image taken by right camera

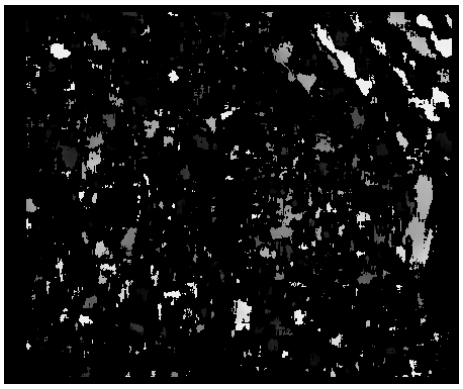


Figure 2.15: Generated disparity map before filtering

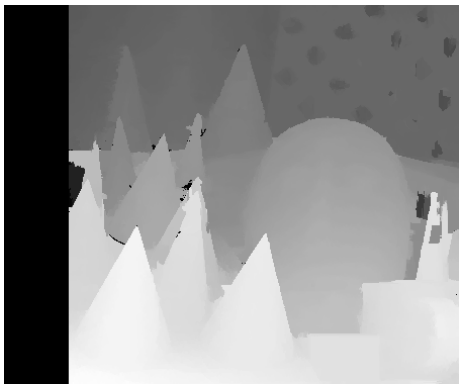


Figure 2.16: Generated disparity map after filtering

2.6 Path Planning

Path planning is crucial in a navigation systems. Good example, similar to this project problem is a vacuum robot, which uses fuzzy control to plan a trajectory to clean an area. The algorithm is designed to avoid collisions with static obstacles or falling down the staircases. Another example would be drones that can map mountainous regions and navigate along the shortest path to the target destination. Many algorithms have been implemented for path planning, with popular approach rising using neural networks [42], which is theoretically considered to be the approach for later implementation.

Spatial Memory

By using a memory based method, Robot creates its own world representation which is called a spatial memory. There are two forms of spatial memory creation methods - based on landmarks (topological) as illustrated in the Figure 2.17 or based on regular maps (metric) as depicted in Figure 2.18. In the topological based map the map expresses space in terms of connections between landmarks, robot than identifies

the landmarks and moves accordingly. The same way as in the topology, the map with nodes and edges is created. Nodes represent the possible goals or gateways and edges navigable paths between nodes. In contrast, in metric maps, the space is expressed in terms of metric distances, where path is determined from one point to a goal assuming the prior map of relevant aspects. The advantage of metric maps is that they can be also used to create topological maps, while the other way around it is not possible [43].

Figure 2.17: Topological spatial memory

Figure 2.18: Metric spatial memory

The downside of this method is the fact that a crucial part for creating a map of surroundings is the ability for the robot to sense objects around itself, the environmental conditions or its relative position. Robot then reports back this information and uses it for creating previously mentioned map. There are multiple sensors that can be used for this method such as optical sensors, GPS based sensors, ultrasonic range sensors or inertial measurement sensors. This method is disregarded due to the price of such sensors and the fact that the robot would have to adapt to every new environment it is placed to, and thus it would not provide a robust solution.

Convolutional Neural Networks

Another way to approach path planning is using image classification which is done by neural networks. Neural network, as fundamental classification algorithm is already widely used in many image classification problems. Due to its high performance and rapid development it could provide a good solution to real time image classification which is used in path planning problem [2].
In the case of path planning, the network is trained on the images of the environments which are individually labeled with the direction where robot can freely move without interfering with the obstacle. After the training, each input to the network is classified as the most probable direction where the robot can move.
This approach provides real-time solution, and due to high performance of neural networks is also robust to the dynamic obstacles which may appear in the environment, and thus provide a very attractive approach to explore more when it comes to path planning.

3

Methods Used in This Thesis

This chapter describes the methods from the theoretical framework applied to the custom data and tested in order to evaluate their performances for the final implementation to the robot. The essence of the system is to use RGB images and use CNN image classification to predict, at every given frame, the direction where to move next. The directions are classified as Straight, Left and Right. Through the chapter, the data collection and annotation, the architectural models of the CNNs for object detection and image classification and the multi-view geometry for the depth map estimation is discussed.

3.1 Structure of the System

This project proposes three different possible models that could be used to detect a free space where robot is able to move without interfering with the obstacle. All architectures will be tested in order to pick the one with the best performance for this specific application.
Firstly, two images taken by camera attached to the robot will serve as an input to algorithm for generating disparity map. The generated disparity map provides an information about depth in the image and gives a robot idea of its surroundings and thus it is used as an input to the pretrained CNN, which will predict the most probable free direction to move. The direction with the highest probability is then sent to the robot as a command to move. Whole approach is demonstrated in Figure3.1.

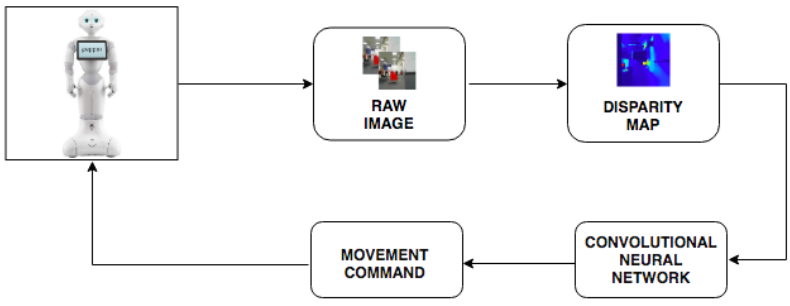


Figure 3.1: Basic idea model using disparity map

Secondly, another proposed model is presented in Figure 3.2. This approach uses

only one camera to capture an image that serves as an input to the object detection method. The output of the object detection method is an image with detected obstacles, which is then classified by the CNN and probability of the most probable free direction to move is predicted. This direction is sent to the robot as a command to move.

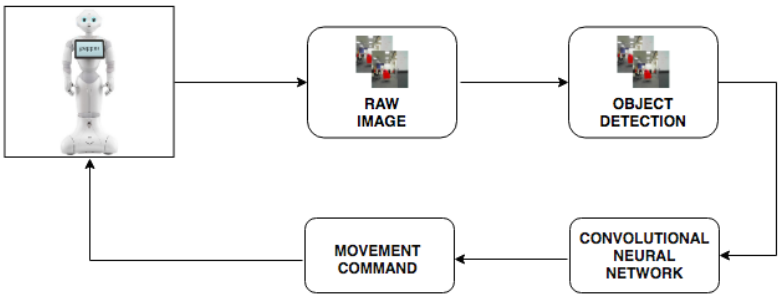


Figure 3.2: Basic idea model using object detection

Lastly, both of the previously mentioned models will be combined together to test if any improvement is achieved with addition of information. In this settings both disparity map and image with detected objects are combined together, to produce an image which is an input to the CNN and again the most probable free direction to move is predicted as demonstrated in Figure 3.3.

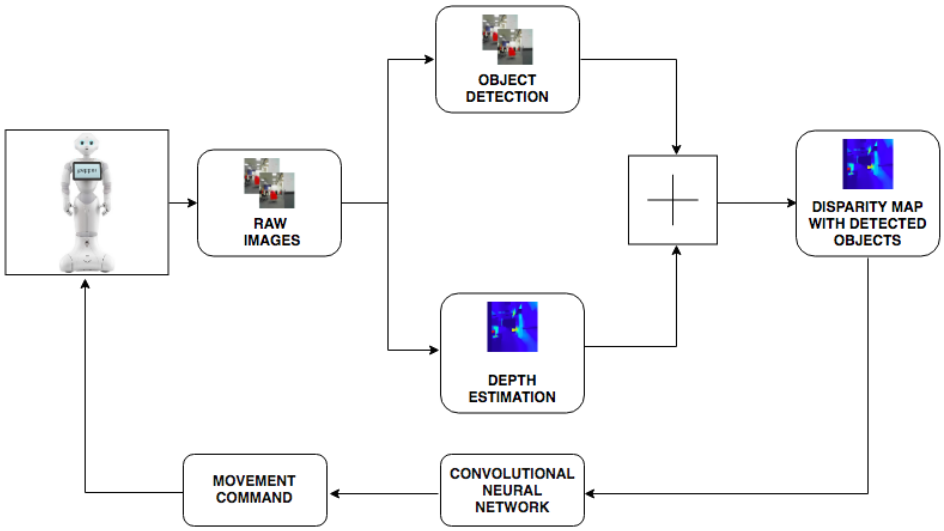


Figure 3.3: Basic idea model using disparity map and object detection

3.2 Implementation Requirements

Using the initial research and the results of experiments that have been performed before actual implementation of final method, the requirements for the overall solution of this project are listed.

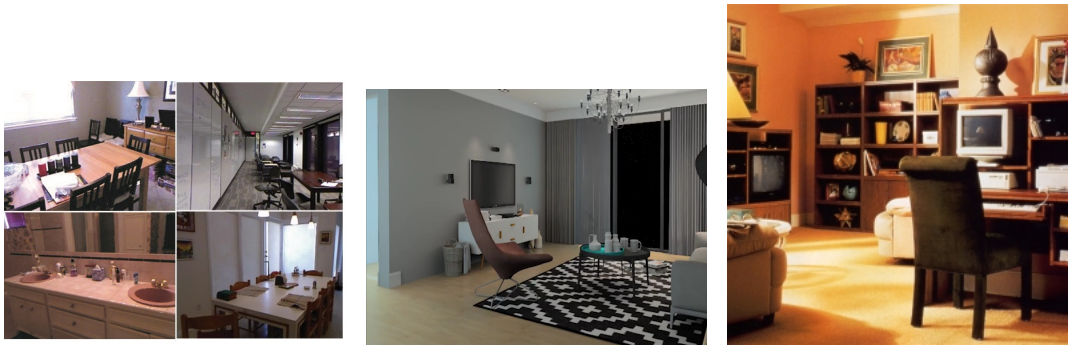
- Quantity of Data To train and validate neural network it is needed to have sufficient amount of data.
- Quality of Data Due to the specific problem of indoor navigation this project tries to solve, it is needed for data to be specific for this application, and thus the data are obtained manually.
- Computational Power Due to the fact that final product should be a robot navigating itself to the target, the fast computation is one of the major requirements, so a smooth locomotion of the robot is achieved.
- Software Structure As the final neural network takes an image which is combination of object detection and depth estimation, those two methods need to be performed at the same time to combine their results as an input to neural network
- Python version compatibility Due to the fact that Pepper robot API is build in Python2.7 and will not execute using Python3 and higher, while at the same time working with CNNs requires Python3, it needs to be made sure that the final solution counts on this problem and interpreters for each of three main tasks are set correctly

3.3 Data Collection and Annotation

As mentioned in the implementation requirements the adequate datasets with good quality images and equal distribution among all classes is essential for the functioning of the CNNS and therefore data gathering, augmentation and existing datasets exploration is a crucial part for the CNNs to perform well.

3.3.1 Existing Datasets

Using existing datasets is the first step in development of any Machine Learning application. A readily available large dataset is usually developed by a team of experts and provides a good spread of images across all categories, and a reasonable variance in the quality of the images.
In this project, three such datasets were discovered: SUN Database, NYU Depth Dataset and InteriorNet dataset. Scene UNderstanding (SUN) database contains 899categories and 130,519 images. This is a huge dataset with a spread of environment, category objects and lighting setup. NYU Depth Dataset contains 1449 RGBDimages, capturing 464 diverse indoor scenes, with detailed annotation. This is close to what's required for the project. And finally the InteriorNet dataset, a collection of interior scenes generated through rendered 3D models. A sample from the datasets are illustrated in figure 3.4.



(a) NYU Depth Dataset (b) InteriorNet Dataset (c) SUN Database

Figure 3.4: Sample of available datasets for indoor scenes

Another option was to a web-scrapper script to download indoor scenes from websites such as Flikr or Google images. A simple Python script can download images based on "keywords" or "tags", allowing the user to build a dataset automatically.
However, the existing datasets, upon further inspection, was not suitable for the project. The NYU Depth Dataset, though provides the depth map as well, has too much variation in the camera angles making it difficult to annotate which direction to move in. The same problem persists with the InteriorNet Dataset and the SUN Database, apart from not having corresponding images for depth estimation. The problem with web-scrapping is that there is no guarantee on what kind of images are downloaded, as illustrated in figure 3.5.

Figure 3.5: A sample of the downloaded images from a web-scrapper script for keyword "Office"

3.3.2 Custom Dataset

It is clear that for the use case in the project, the dataset has to be made on our own. The setup included the Pepper robot and an auxiliary binocular camera setup, designed specifically to be attached to Pepper and suited to the project requirements. The two in-built cameras in Pepper do not provide the necessary images to compute depth map of the indoor scene, hence an auxiliary attachment had to be designed. Furthermore, the auxiliary camera attachment allows us to control the position of the camera with respect to ground reference, and allows us to control the distance between the cameras, which affects the depth map created.

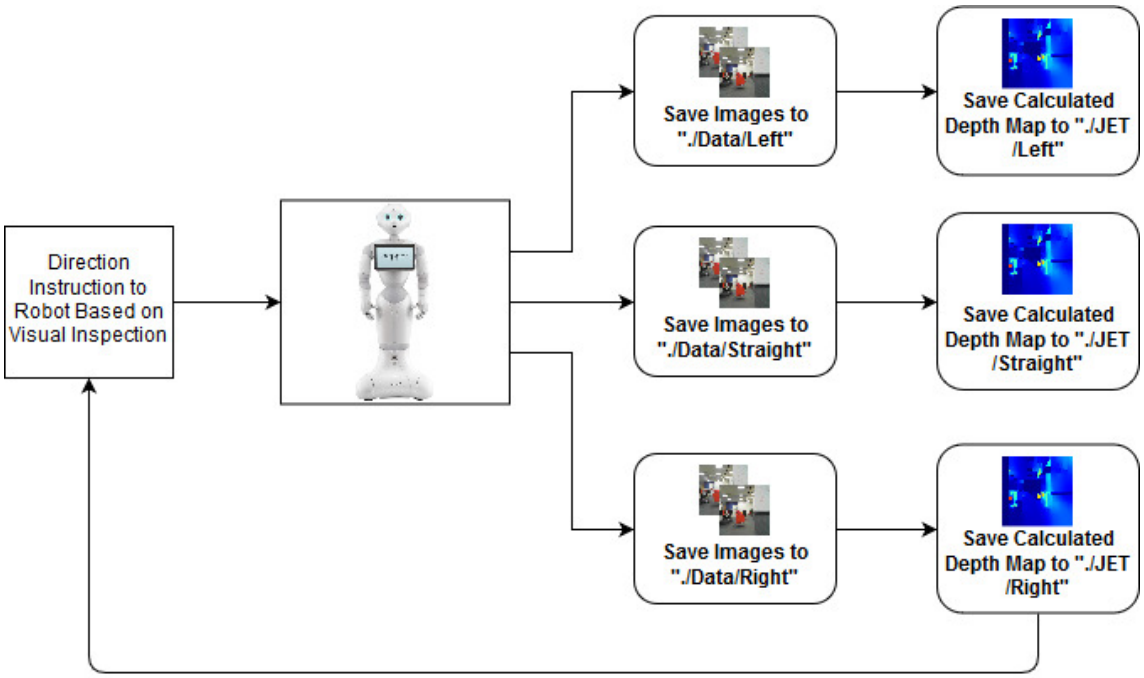


Figure 3.6: Overview of data collection system

The data was collected in the Ericsson office under natural and artificial lighting. The obstacles were arranged to ensure equal distribution between classes, as much as possible.

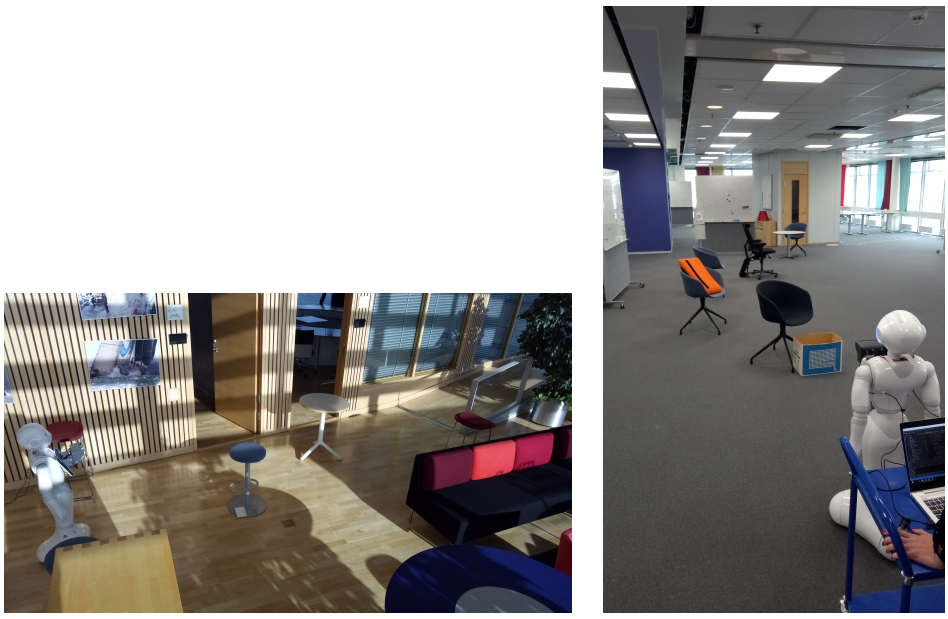


Figure 3.7: Examples of the environment where data was collected

By discussion with students working with Pepper robot in 2018 and from their thesis on Human-Machine Interaction, Communication Initiation Probability Estimation [44] important warning has been brought up. The same as in this project, their

project used own gathered data, and when first trained and evaluated it has performed much worse than expected, due to the problem of uneven data. The amounts of images for training in individual classes were highly unbalanced, leading to bad validation accuracy. After the warning the amounts of images in individual classes were compared and it has been realized that amounts are uneven as well. Naturally, the folder containing images with direction straight contains twice as data as folder left and right, as it is the most common direction to take. This problem lead to disregarding of many straight samples to keep the number of images in each folder even. From that time on the surrounding have been always arranged in order to obtain even amounts of directions.

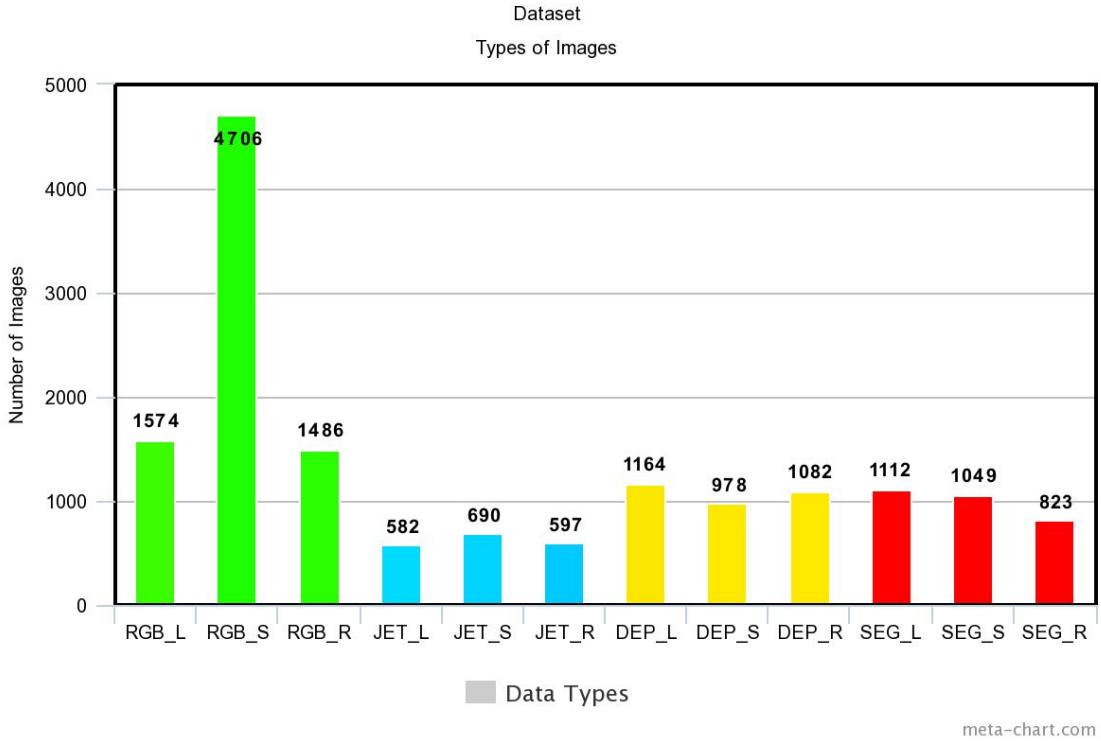


Figure 3.8: Distribution of data for the necessary classes, a)RGB - images captured from the camera, b) JET - images produced from multi-view geometry and JET filtering in OpenCV, c) DEP - images produced from the DenseDepth CNN network and d) SEG - images from the PSPNet segmentation network. The suffices L, R and S refer to the classes Left, Right and Straight respectively

The total number of raw RGB images captured were 7766. However, it can be observed from figure 3.8 that the data is heavily biased toward the "Straight" category. Hence, for the purposes of the project we limit the number of images used per class to that of the lowest class. Since the depth map in Black & White or JET color map require a pair of images, the number of images generated are roughly half of the raw images which after cleaning produces 1869 images, however there is a fairly equal distribution of images across classes. Similarly, semantic segmentation produces about 2984 images in total after cleaning the dataset, with equal split between left and straight categories, with fewer images in the right category.

3.3.3 Data Augmentation

As previously explained, neural networks require large amounts of data. To obtain such data is often expensive and laborious. Data augmentation is a great tool to overcome this issue by artificially inflating the training set with label preserving transformations. There are many possible data augmentation schemes to choose from in order to make data the most appropriate for the training data sets.
Generally, there are two types of transformations that can be applied - geometric transformations and photometric transformations. Geometric transformations alter the geometry of the image with the aim of making neural network invariant to change in position and orientation. Examples of geometric transformations include flipping, cropping, scaling and rotating. On the other hand, the photometric transformations amend the color channels with objective of making neural network invariant to change in lighting and color. Such changes just adjust image lighting and color while leaving the geometry unchanged. Examples of such techniques are jittering, edge enhancement and PCA photometric methods [45].
Considering the method used to estimate the depth and direction, few different data augmentation methods have been explored in order to make sure that the transformations applied to images will not affect the performance of the methods. Original idea was to use rotation, shift, shear and zoom. After testing the zoom has been disregarded, as often by zooming the image and skipping the side pixels, some objects presented on pictures have been deleted - especially the crucial close ones, and so this transformation is disregarded. The transformations being used are explained and demonstrated as follows.

Rotation

Firstly, the rotation is used. The angle of rotation can be set manually, and the empty pixels are set to be filled with the value of the nearest pixel. After testing, it has been decided not to rotate images more than 20°. The data are then randomly rotated in the range of 0-20° and used as new data for further training. As visible in Figure 3.9 rotation does not make any significant or method affecting changes, and it can be used for training together with original images.

(a) Original image (b) Left rotated image (c) Right rotated image

Figure 3.9: Demonstrations of data augmentation - rotation

Shift

Another augmentation method used in this project is shift. There are multiple types of shifts that can be used. Good examples are width shift and height shift, which are also being used in this project. Width shift and height shift are ranges - as a fraction

of total width or height, within which to randomly translate pictures vertically or horizontally as demonstrated in 3.10. In this project the values for fraction of total dimension 0.2 is used.

(a) Original image (b) Height shifted image (c) Width shifted image

Figure 3.10: Demonstrations of data augmentation - shift

Shear
Shearing is randomly applying shearing transformations. In plane geometry, a shear mapping is a linear map that displaces each point in fixed direction, by an amount proportional to its signed distance from the line that is parallel to that direction and goes through the origin [46]. For this type of transformation a coefficient was again set to 0.2 and the sheared images can be observed in Figure 3.11.

(a) Original image (b) Sheared image (c) Shared image

Figure 3.11: Demonstrations of data augmentation - shear

Although data augmentation adds more images, the images are correlated and it does not add the same value as if whole new images would be added, but it still promises an increased accuracy [47]. The augmented data will be used just for training, while original data will stay the same for validation.

3.4 Depth Estimation with Binocular Images

When deciding on the best method to create a depth map for further usage few obstacles needed to be faced through. First of all, the realization that integrated robot's cameras, can not be used complicated the approach. When capturing the images it has been realized that cameras provide different views and the scenes being

captured have nothing in common, and thus no multiple view method can be used. Integrated 3D camera has been disregarded due to the unsatisfactory resolution. For this reason, and also reason of making the solution of indoor navigation more general and possible to perform even without robot, two external web cameras are used.
Disparity map is then created by capturing the images as robot moves, and creating disparity map from each pair of images.
To create a disparity map a Python library OpenCV is being used. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code [48]. The library has more than 2500 optimized algorithms, including the help function to create a disparity map. The input to the function are two images - one taken by left camera and one by right camera. After that the disparity map is created using stereo matching algorithms. Unfortunately, those kind of algorithms tend to make quite a few errors on challenging sequences. These errors are usually concentrated in uniform texture-less areas. The possible way how to deal with these errors is to detect potentially inaccurate disparity values and invalidate them. Another way would be application of some filtering procedure to align the disparity map edges with those of the original image and propagate disparity values from high to low confidence regions like half occlusions. This can be done using so called weighted least squares filtering (WLS), which will be due to its simplicity and good tested performance used in this project. [49].

3.5 Object detection - Semantic Segmentation

In order to choose a method for object detection, the specific application of this project needs to be taken in account. As the project solves the problem of finding a free space for a robot to move, the task is not to categorize the objects but to obtain an idea about the general space in the image. For this purpose the method of semantic segmentation has been chosen, as it provides a good understanding of the scene, and one of the possible categories that is classified is a floor, which for this project provides an area where robot can move.
Semantic segmentation classifies every pixel of the image to the known or trained categories. It is one of the most popular methods for scene parsing, providing details of location, label and shape of the known categories. The general architecture of semantic segmentation can be divided into two parts, decoder and encoder. Encoder section performs the feature extraction and is usually a pre-trained image classification network like VGG16 or ResNet. The encoder is usually followed up with a decoder network, which semantically projects the low resolution features extracted by the encoder onto the high resolution pixel space [50].
Semantic segmentation architectures typically follow one of three approaches - region based segmentation, fully convolutional network based segmentation and weakly supervised semantic segmentation. Region-based segmentation work by feature extraction of the free form image and classifying the object detection regions. During

prediction, the method works by transforming the region based predictions into pixel based prediction by assigning the class of the highest score in the region to the pixel. This approach is memory intensive and is not adept to making precise pixel-wise segmentation. Mask-RCNN is the most popular architecture based on this approach.
The fully convolutional network (FCN) approach maps the input pixel to output pixel. The FCN is an extension of CNN and consists of purely convolutional and pooling layers. The FCN approach can perform the semantic segmentation task on arbitrary input size, and is significantly faster and consumes less memory than the region-based approach. The only downside is that due to the pooling layers, the resolution of the output map suffers. PSPNet and SegNet are two popular architectures using this approach and PSPNet is the one chosen for the thesis work.

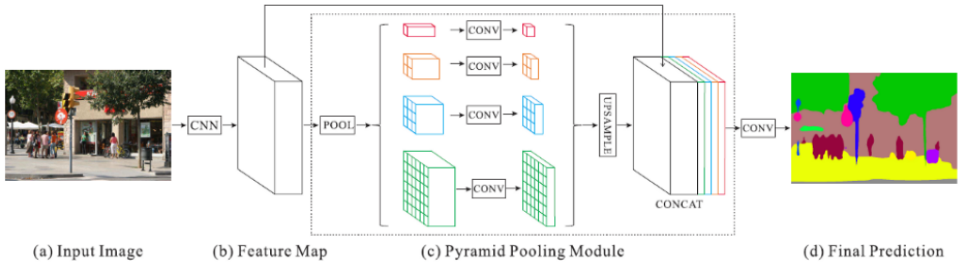


Figure 3.12: PSPNet architecture

Pyramid Scene Parsing Network (PSPNet)[51] is a FCN based semantic segmentation architecture. The key feature of the architecture is the Pyramid Pooling Module as described in 3.12. The Pyramid Pooling Module, extracts the feature map on different levels before concatenating the output of the module to be fed into the final classification layers. This hierarchical structure preserve the complex spatial information from the complex feature set in ADE20K dataset on which PSPNet is trained on. The coarsest features are extracted by the convolutional layer (indicated as red in 3.12) and moving down till the finest features (indicated as green in 3.12). PSPNet achieves pixel-wise accuracy of 80.88% on the ADE20K [52] dataset. The PSPNet surpasses SegNet, DilatedNet and FCN on the ImageNet[53] challenge with a score of 57.2% compared to 40.79%, 45.67% & 44.80% respectively.

3.6 Classification Networks

Last method being developed in this project is path planning using neural networks. For this purpose new architectures have been created and several already existing architectures have been modified and tested, in order to pick the best approach for this problem with respect to validation and test accuracy. There are many possible networks that could be used for image classification, but considering the limitations as lack of data and taking in account the fact that what is needed is not a classification of the objects but classification of free areas to move, it can be concluded that the networks that a focus will be put on, are those that do not

extract too precise and too much of features and do not require large datasets. Due to these reasons the chosen networks are chosen accordingly and presented in this chapter.

Vanilla CNN

As a baseline for the performance analysis, a simple shallow CNN architectures were tested with raw images, depth map with JET color map and depth map with black & white map. The approach with Vanilla CNN is to start with as few layers as possible. The reasoning behind that is the scarcity of data. A shallow network will not extract as many features from the image and will help generalize the problem better for a small dataset. The hyperparameters to tuned were number of Conv2D blocks, Conv2D filter numbers, Conv2D filter sizes, FC Layer sizes and optimizers.

Transfer Learning

Another approach that has been chosen to be tested, due to the fact that it could provide a solution to lack of the data and limited number of samples available for training is transfer learning. In transfer learning methods the CNNs are pretrained on the the existing datasets such as VGG16, InceptionV3 ResNet and MobileNetV2. In contrast to the Vanilla CNN approach, with Transfer Learning, the aim is to extract the features of the image as best as possible, using appropriate chosen datasets such VGG16, InceptionV3 MobileNet and ResNet50 base models.

New Concept Architecture

Lastly, by using the understanding of features extracted from depth map images and RGB images, a few branched CNN architectures were proposed and tested. The aim of these architectures is to supplement the feature extraction from the RGB images with the feature extraction from the depth map.

4

Test results on individual module methods

This section discusses the experiments performed using previously explained methods. Each of the methods is tested separately using the custom dataset, to make sure the performance is satisfactory using the captured images of Ericsson building. The results of the experiments together with the limitations that were faced during the testing of separate methods are described and evaluated in the following chapter.

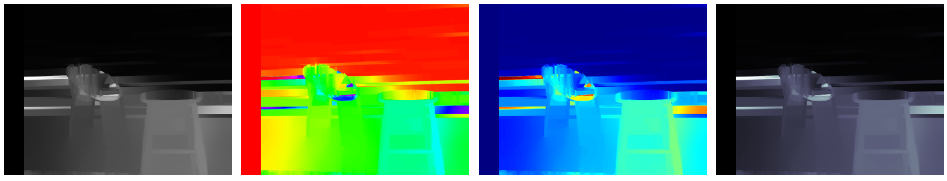
4.1 Depth Estimation by Disparity Map

Experimenting with disparity maps was done through changing distance between cameras, choosing different colour maps and also filtering the produced solution in order to avoid noise presented. The depth map was generated at 30, 50, 80 and 100 mm distance between cameras and with Grayscale, HSV, JET and BONE color maps in OpenCV. The goal is to identify the best parameters which provides the clear distinction between near- and far- objects. The parametric experiments were tested on 20 images from left and right cameras. A sample on one set of left-right images is illustrated in Figure 4.1.
The setup for calibrating the camera setup includes a stool at 1 meter, a chair at 2 meters and wall at 3 meters. Ideally, a good depth map will provide a clear distinction in the colors for near- and far-objects. It was observed that higher baseline gave a better result for depth map, providing a good segmentation of nearby and far-off objects. However, to assist learning, it is advantageous to have a clear segmentation of the nearby objects which is generated well in the 50mm baseline and JET color map.

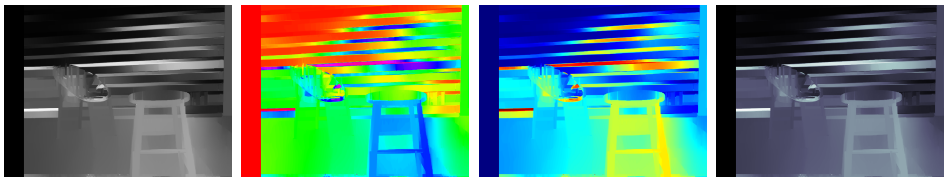
4.1.1 Limitations

The problem that this project is facing with creating a good disparity map is alignment of cameras. Although the printed 3D structure provides a solid holder for cameras, the creation of disparity map requires as precise horizontal alignment as possible, which is sometimes challenging. Another limitation is the robot motion which adds some disturbance to captured images. Those factors will be taken in account when evaluating the overall performance.

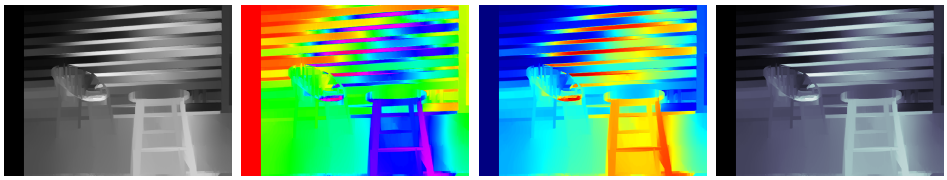
(a) Base for depth map generation



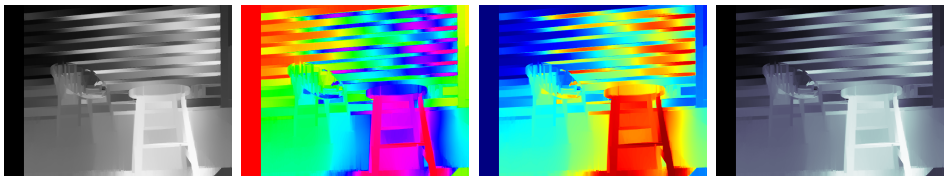
(b) Depth maps at 30 mm baseline



(c) Depth maps at 50 mm baseline



(d) Depth maps at 80 mm baseline



(e) Depth maps at 100 mm baseline

Figure 4.1: Sample of depth map experiment (left) Grayscale (center left) HSV (center right) JET (right) BONE

4.2 Path Planning

For the purpose of building the neural network for path planning the Keras library available in Python is chosen. Keras is a neural networks API, which is written in Python. It was developed to enable fast experiments and building neural networks quickly. It allows easy and fast prototyping and supports both convolutional networks and recurrent networks, as well as combinations of this two. Another advantage Keras has is that it runs seamlessly on CPU and GPU [54]. There are several reasons why Keras was picked for this project but main advantage that Keras provides for this project is its modularity which allows to pick modules that can be combined together but also stand alone to create new models.
Before deciding on architecture or building the new neural network few architectures will be explored. Not only architectures, but also different datasets are used, in order to prove the idea that disparity map and object detection provide improvement when it comes to the classification.

4.2.1 Limitations

Important limitation that needs to be mentioned, and that could lead to lower validation accuracy is the problem of classification of direction, whether it is by neural network or human eye. When looking at the collected data and also while initial testing, it has been realized that some of the images can be classified as multiple directions. The same way as in the the real world, in some cases when looking at the scene and target, it can not be clearly said which direction is the "correct" one if there are multiple free ways to go. Possible solution to avoid this problem would be to arrange surroundings in a more clear setup that would strictly determine which direction is the right one. This solution could provide a possible improvement but it would not mimic the real world and thus it has been decided to keep the data gathered and be aware of the limitation that this problem brings.

4.2.2 Dataset and Setup

Multiple already existing architectures have been modified and used as a base for testing solutions. Some of the architectures have been made from scratch, in some cases layers have been added or removed, activation functions changed. Other approach was to use transfer learning - keep the base of pre-trained architecture and modify just classification layers. Lastly, building own architecture has been done, in order to fit a solution to the specific problem this project faces.
One of the most important requirements and also obstacles that our approach faces, is the low amount of data/images as they are obtained manually. The chosen neural network will have to be able to avoid over fitting and preferably do not require large amount of data as most of the neural networks do.
First of all the training will be performed on images without bounding boxes and without the depth information in order to confirm the idea that proposed method will provide more information for neural network and thus achieve better results.

Naturally after that, the experiments will be run on the custom dataset to confirm this idea and compare the accuracy.
The test set contains very general images as there is no specific information about depth or objects, just raw images. It was expected that without having more information it will not perform well, exactly as the demonstrated values confirmed. The following table shows the amounts of data that have been used for the training and validation. When organizing the data, it has been sen that 80 % of all data will be used for training purposes and 20 % of all data for validation purposes. As discussed previously, in order to avoid imbalance data in different classes sets, the classes were aimed to contain similar numbers of samples, varying maximum by 10%.

Table 4.1: Amounts of images used for training

Direction	Training Images	Augmented Training Images	Validation Images
Left	400	2400	150
Straight	450	2700	200
Right	400	2400	150

The second set created was for the depth maps generated from the raw images. This operation requires a pair of images, generates at most half the size of the dataset of the raw images.

Table 4.2: Amounts of JET and Grayscale depth maps used for training

Direction	Training Images	Augmented training Images	Validation Images
Left	200	1200	100
Straight	250	1500	150
Right	200	1200	100

4.2.3 Classification using RGB and Depth Map Images

This section discusses the training results from all the CNN architectures that have been put into the test. In order for an architecture to be considered for further application, the low limit on the validation accuracy has been set to 80 %, and thus architectures not fulfilling this requirement will be disregarded and on the other hand, architectures with validation accuracy above 80% will be tested on robot, and final results presented in the results chapter.
The training have been performed on Google Colab platform, which provides 2vCPU @ 2.2GHz, 13GB of RAM and Tesla K10 GPU with 12GB of graphics memory. It should be noted that none of the trainings performed feature the "early stopping" callback, therefore the validation set is equivalent to the test set. Therefore, performance decisions can be based on the validation accuracy.

Vanilla CNN

Firstly, it has been decided to build new neural network from scratch. When building this network previously mentioned limitation of low number of data and quality

has been taken in account, and the created architecture has been inspired by the networks that are tested to perform well on small datasets. When creating this architecture the goal was to distinguish where is a free space to move on the image, not to differentiate between objects or categorize the objects. Due to that reason the architecture contains only 3 convolutional layers, as there are not objects that need to be classified, and thus no precise features extraction is needed. The resulting Vanilla CNN is depicted in Figure 4.2.

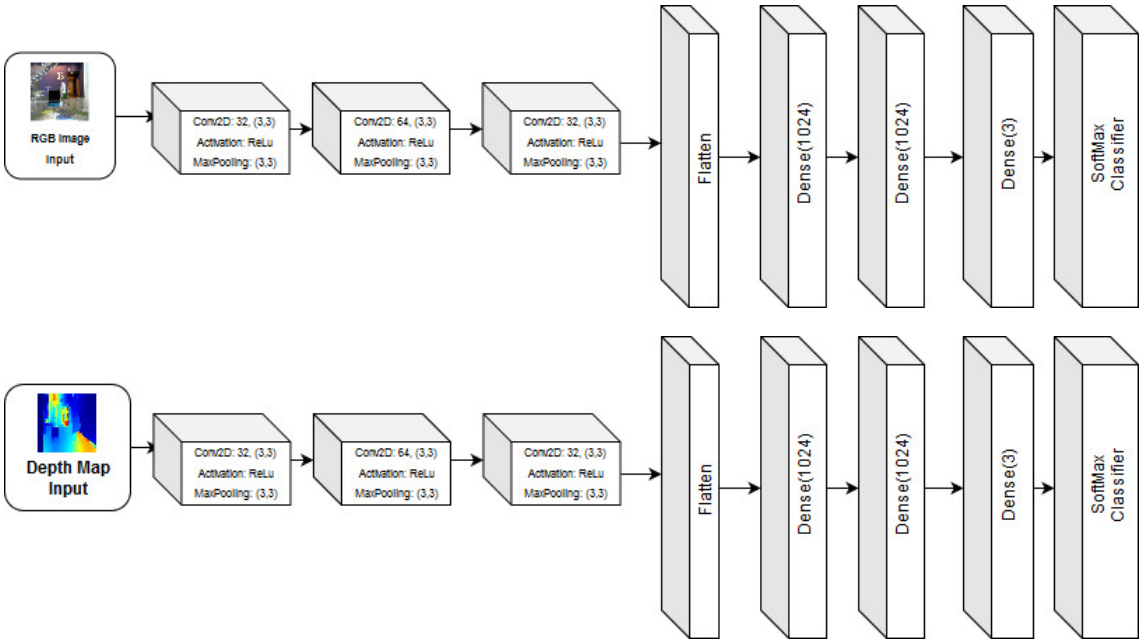


Figure 4.2: Vanilla CNN architectures for (a)RGB Images and (b)Depth Map images

First of all, the network has been evaluated on RGB images, to provide a baseline accuracy. The resulting accuracy and losses are presented in Figure 4.3.

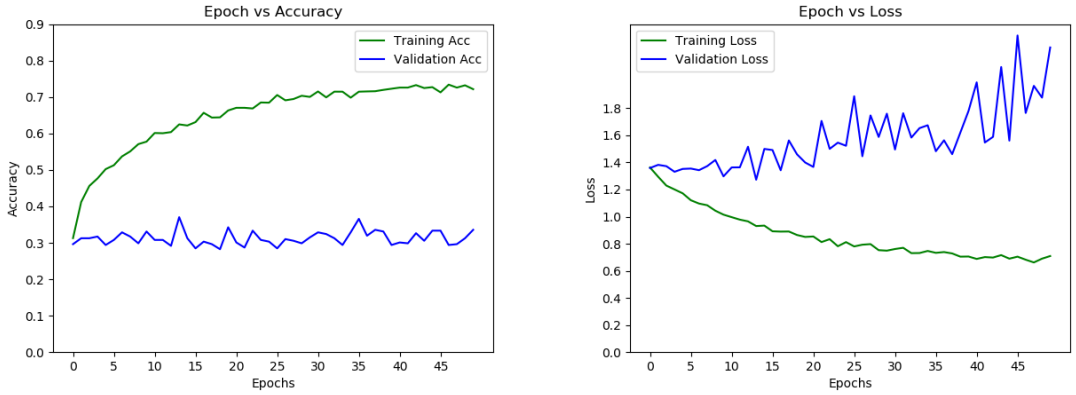


Figure 4.3: Accuracy and Loss with respect to epochs

It can be observed the model is over-fitting the data. Analyzing the feature map

generated after the convolution we observe that the plain images have too many features to unpack, as seen in Figure 4.4.

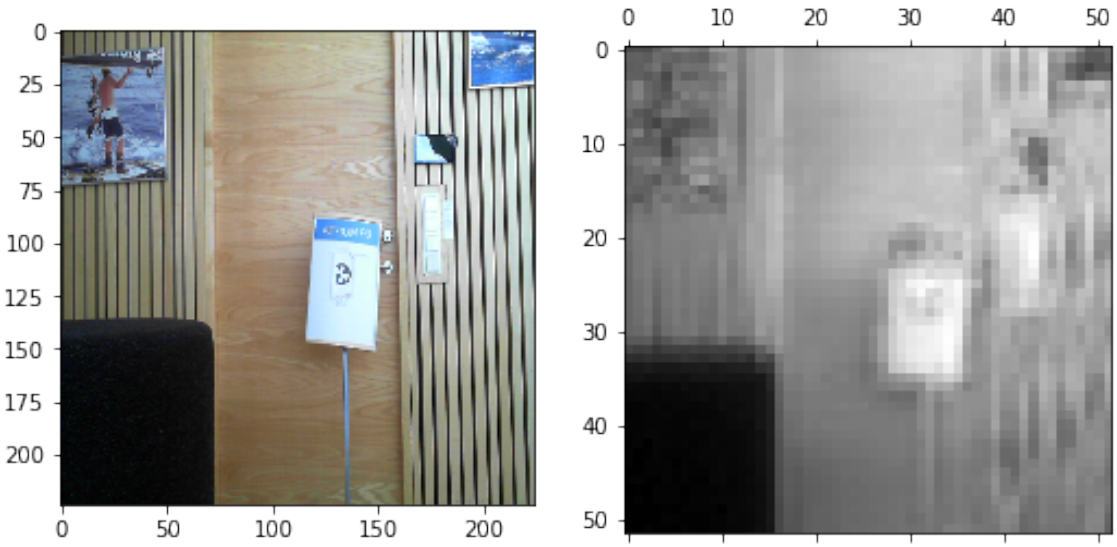
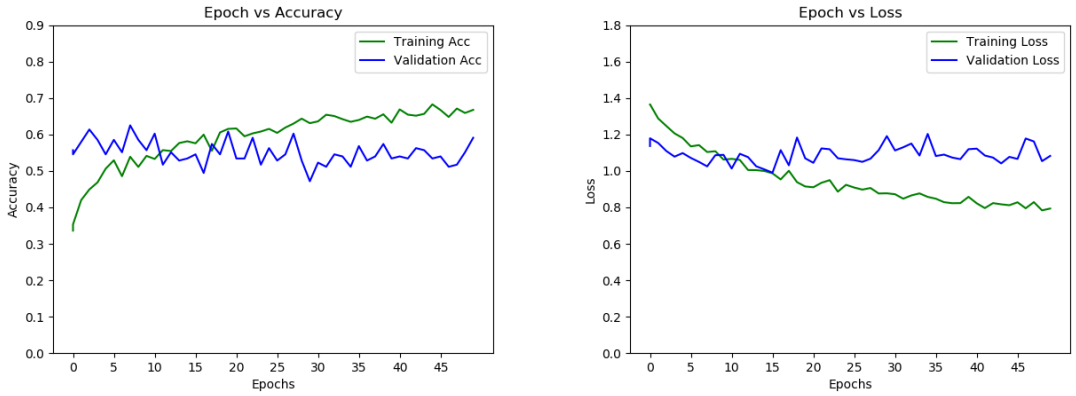


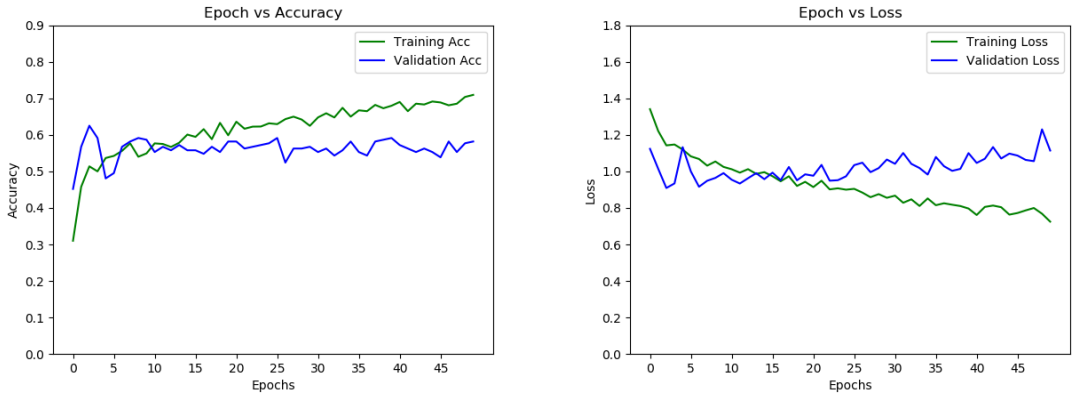
Figure 4.4: Visualization of output from the convolutional block

As can be observed in Figure 4.3 The validation accuracy was only 34 % and it is clearly visible it is over-fitting, and thus this architecture using only raw images has been disregarded, but put in test with different set of images to see if there could be an improvement. To counter the problem of too many features to extract, the vanilla CNN was trained on depth maps - depth maps with JET color map and Grayscale color map. As can be seen in Figures 4.5, 4.6 accuracy improved but



(a) Training on JET depth map

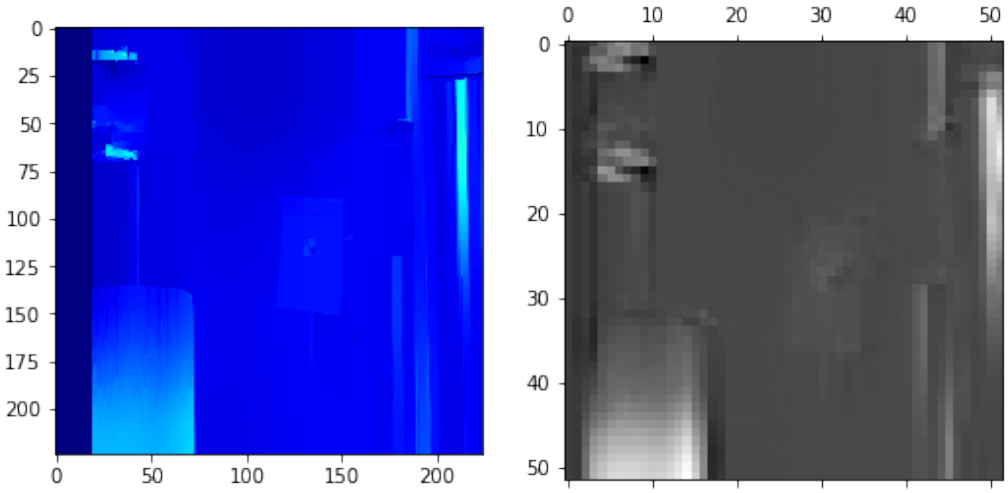
Figure 4.5: Result of training on JET colour map images



(a) Training on Grayscale depth map

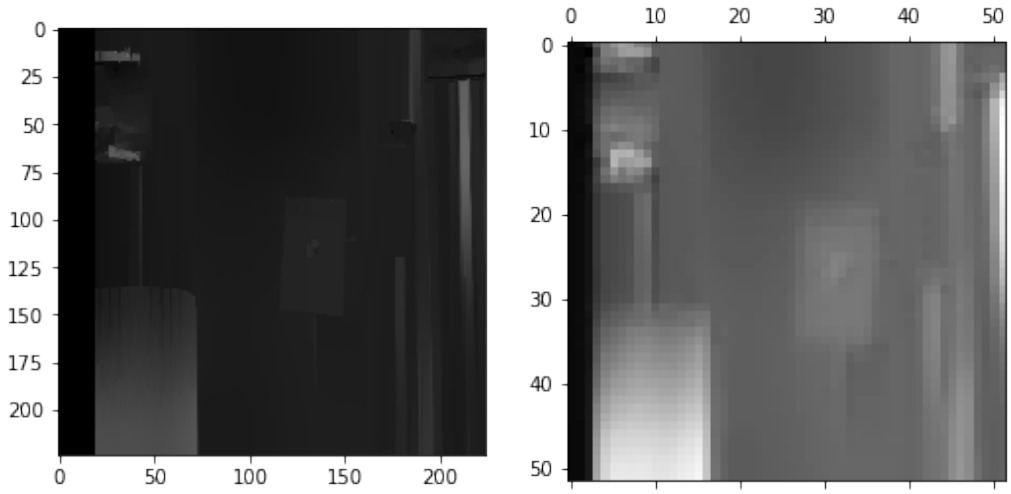
Figure 4.6: Result of training on Grayscale colour map images

As expected, the validation accuracy has improved significantly. This is substantiated by the visualization of the output from the convolutional block for both types of input images in Figure 4.8. It can also be observed that the network is not overfitting the data as the training accuracy and validation accuracy are quite similar, but the problem of low accuracy (below 80% requirement) still remains the same, and so also this improved version is disregarded for the final application.



(a) Input (left) JET Depth map and visualization of extracted feature map (right)

Figure 4.7: Visualization of feature extractions



(a) Input (left) Grayscale Depth map and visualization of extracted feature map (right)

Figure 4.8: Visualization of feature extractions

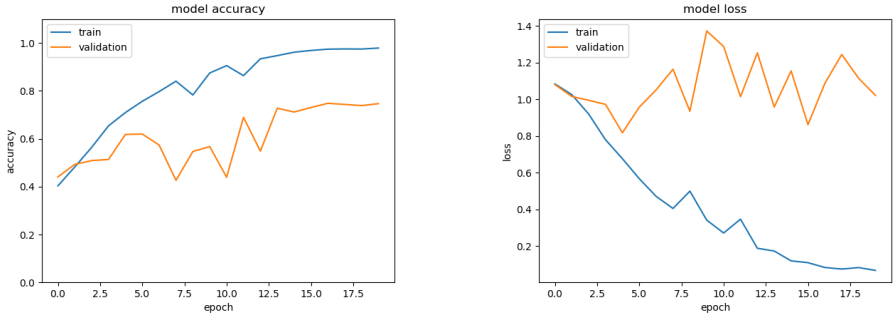
Transfer Learning Networks

Secondly, the transfer learning methods were put to test. The bottle neck neural network used was pre-trained neural network using VGG16, InceptionV3, ResNet and MobileNet. Table 4.3 shows the network and training parameters which produced the best validation accuracy after tuning.

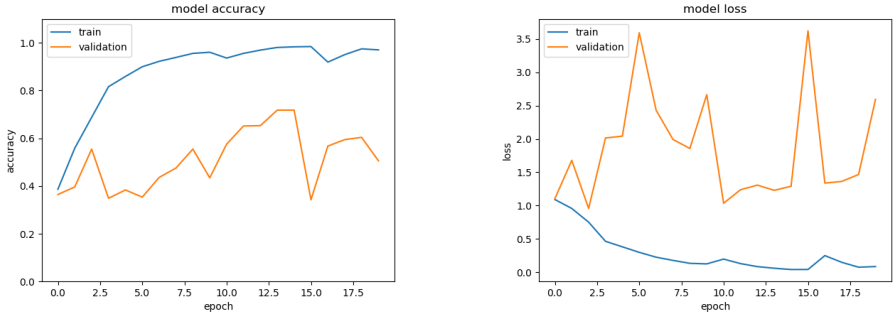
Base	Conv2D	Pooling	FC	Dropout	Optimizer	Batch
VGG16	16, (2x2)	Max: 2x2	1024, 512, 3	-	RMSProp	32
Inception	16, (3x3)	Max: 2x2	1024, 512, 3	0.3	SGD	16
ResNet50	32, (2x2)	Max: 2x2	200, 3	0.5	SGD	8
MobileNet	16, (2x2)	Max: 2x2	512, 512, 3	0.4	Adam	16

Table 4.3: The architectures for transfer learning approach. The input image size, 224 x 224 x 3, remains the same for all models

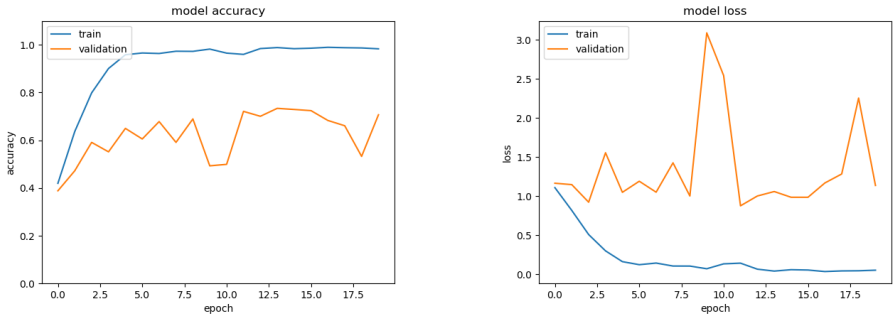
As it is clearly visible from the Figure 4.9, this architecture is definitely not suitable for this project, and it will not be considered for the further improvement due to the low accuracy starting point and extreme overfitting after the best tuning efforts. The reason is believed to be the overly complex nature of the base network which is extracting too many features for the classifier to work effectively. However, with this understanding, the next approach taken is to simplify the convolutional layers and running the classifier on low to medium level features.



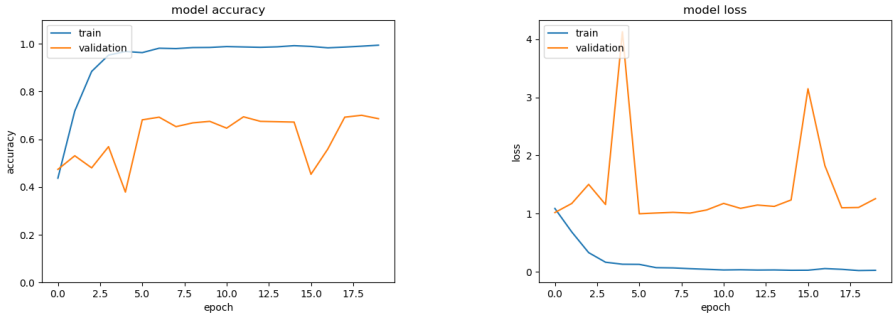
(a) VGG16



(b) InceptionV3



(c) MobileNet



(d) ResNet50

Figure 4.9: Training results for transfer learning approach

Branched CNN

As seen from the results in the previous experiments, the CNNs perform better with depth map images, as they reduce the number of features without losing the essential information required for making the path planning decisions. However, a lot of features are not captured in the depth map with JET color map, depth map with grayscale coloring and the plain images. This inspired us to develop a new architecture type, which can learn to extract features from more than one type of input. The figure 4.10 shows the three new types of architectures proposed for the solution of the navigation problem. The idea is to run feature extraction for each type of input with the best performing vanilla CNN or transfer learning and combining the output before the fully connected classification layers.
and the accuracy and losses are presented.

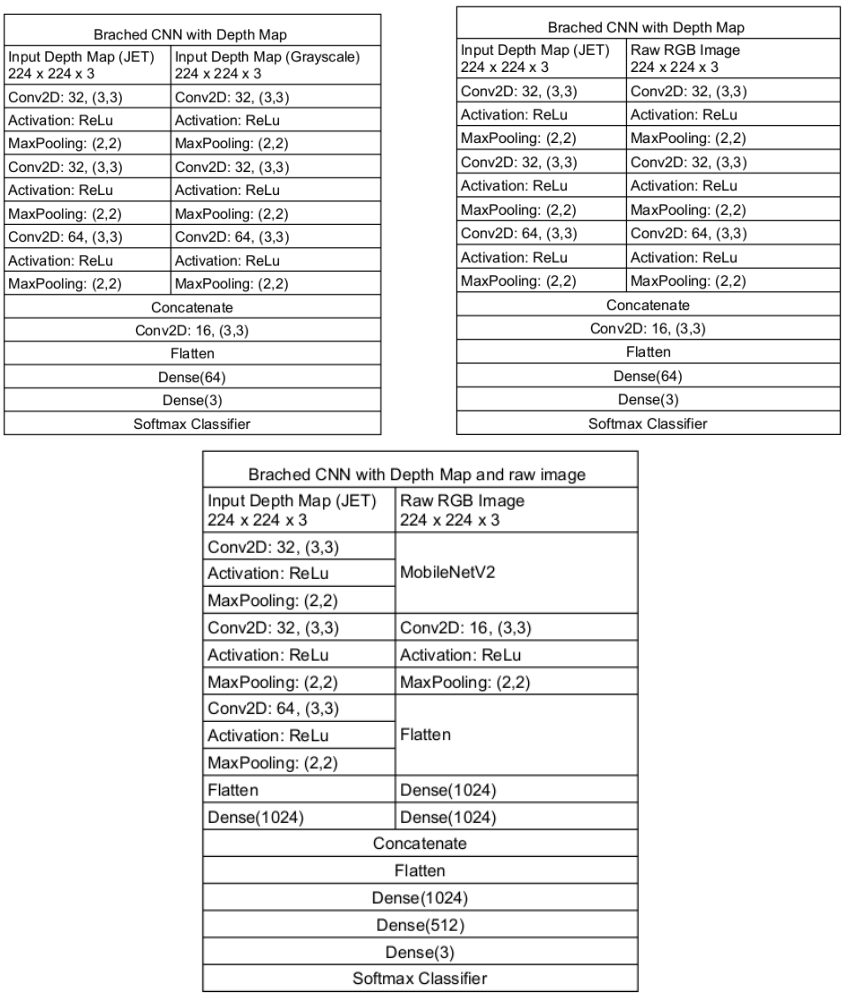
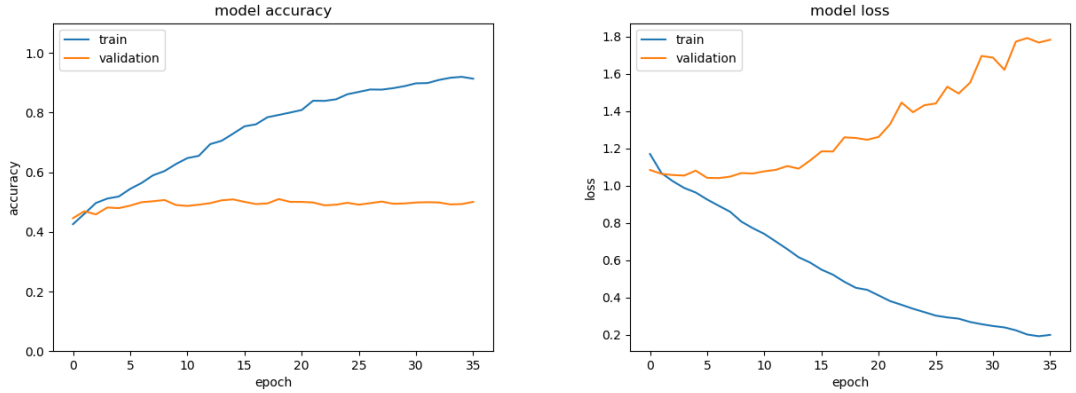


Figure 4.10: Branched Architectures tested

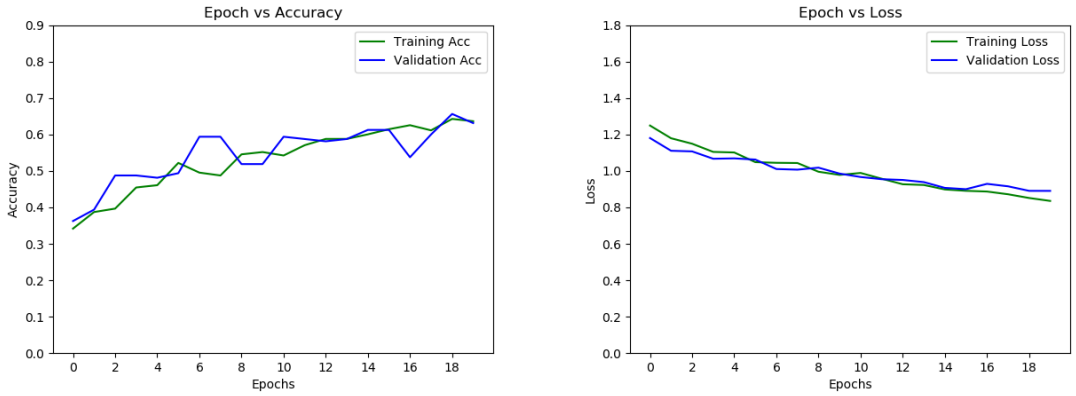
The input to the new type of CNN architectures is the same as the ones used in the previous training instances. The smaller size of the depth map dataset limits the number of plain images used in the training, hence becoming the bottleneck in the training operation. Figure 4.12 shows the results of the training on the branched architecture and it is apparent that the training accuracy is significantly better than

any previous attempts and the validation accuracy is at par with the vanilla CNN with a much smaller training and validation loss. Even though this architecture provides an improvement it still does not exceed the requirement of 80% validation accuracy, and thus again this is disregarded as the final architecture to be used. The overfitting comes from adversarial nature of the input images for the branches and the noisy nature of the input images, which often produces inaccurate images.



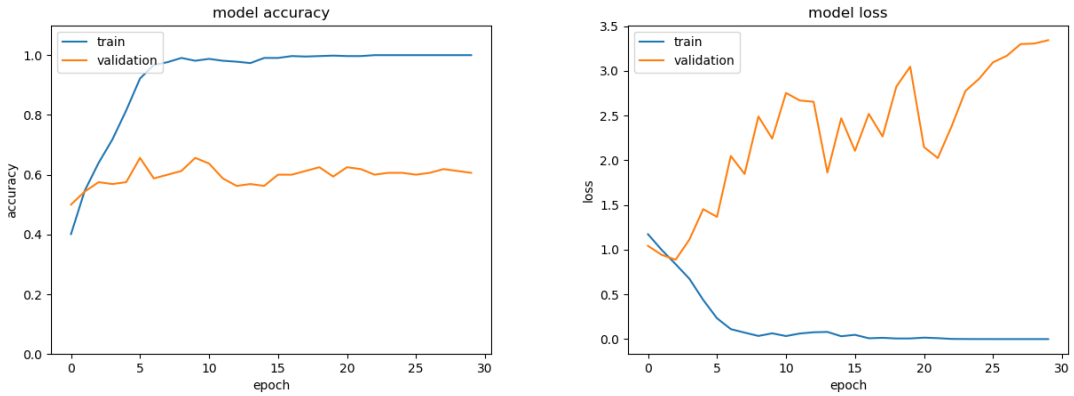
(a) Depth map input (JET + grayscale)

Figure 4.11: Training results with the proposed new architectures



(a) Depth map (JET) and plain image input

Figure 4.12: Training results with the proposed new architectures

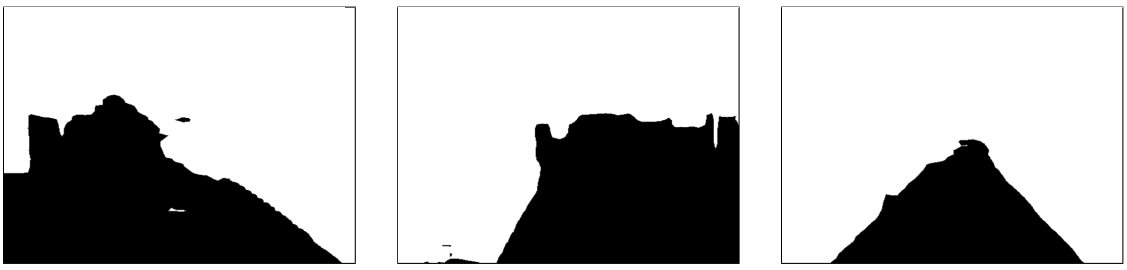


(a) Depth map (JET) and plain image input with MobileNetV2 base

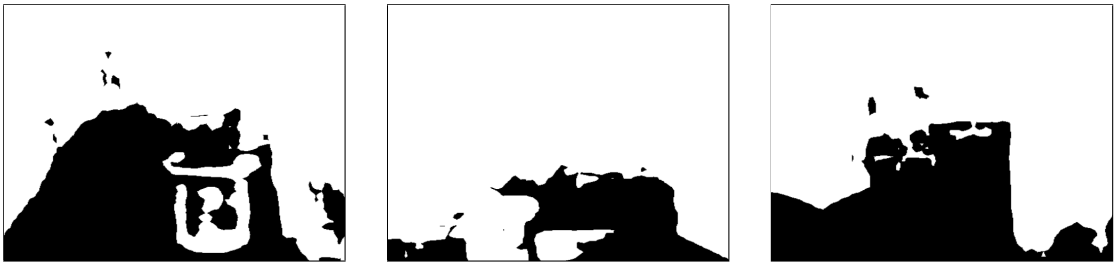
Figure 4.13: Training results with the proposed new architectures

4.3 Classification with Semantically Segmented Images

The final approach tested was using semantically segmented images. For the purposes of testing this approach and training the network, the predicted images from the original RGB images using the pre-trained PSPNet. The segmented images contain the floor place segmentation in black and the other categories in white. This allows the network to understand only the necessary spatial information. Figure 4.14 shows the type of images produced for the left, right and straight classes, which will be the input for the classification network.



(a) Good quality segmented images for left, right and straight classes



(b) Worst quality segmented images for left, right and straight classes

Figure 4.14: Semantically segmented images

It can be observed in the segmented images that even the noisiest images provide sufficient information for human classifier to distinguish between the directional classes, hence this approach could be promising to provide better results.
The classifier network based on these segmented images is using the PanoNavi architecture [55], which uses spherical images for directional classification. The architecture is described in table 4.4.

Layer number	Features	Model
0	101x101x3	Inputs
1	98x98x3	Conv
2	49x49x32	Pool
3	-	BN+PReLu
4	46x46x32	Conv
5	23x23x32	Pool
6	-	BN+PReLu
7	20x20x32	Conv
8	10x10x32	Pool
9	-	BN+PReLu
10	7x7x32	Conv
11	3x3x32	Pool
12	-	BN+PReLu
13	288 ->200	FC1
14	-	Dropout
15	-	PReLu
16	200->3	FC2
17	-	Softmax

Table 4.4: PanoNavi based classifier network architecture

The network was trained on 2013 training images and 672 validation images over the three classes for 50 epochs and 128 images per batch. The Ädagradöptimizer was found to perform the best among all others.
As can be seen from Figures 4.15, this model provides the best accuracy so far with the highest validation accuracy of 95.83%. It also offers the least over-fitting or variation between the training epochs. This is chosen as the preferred network and approach for testing on the real-life problem.

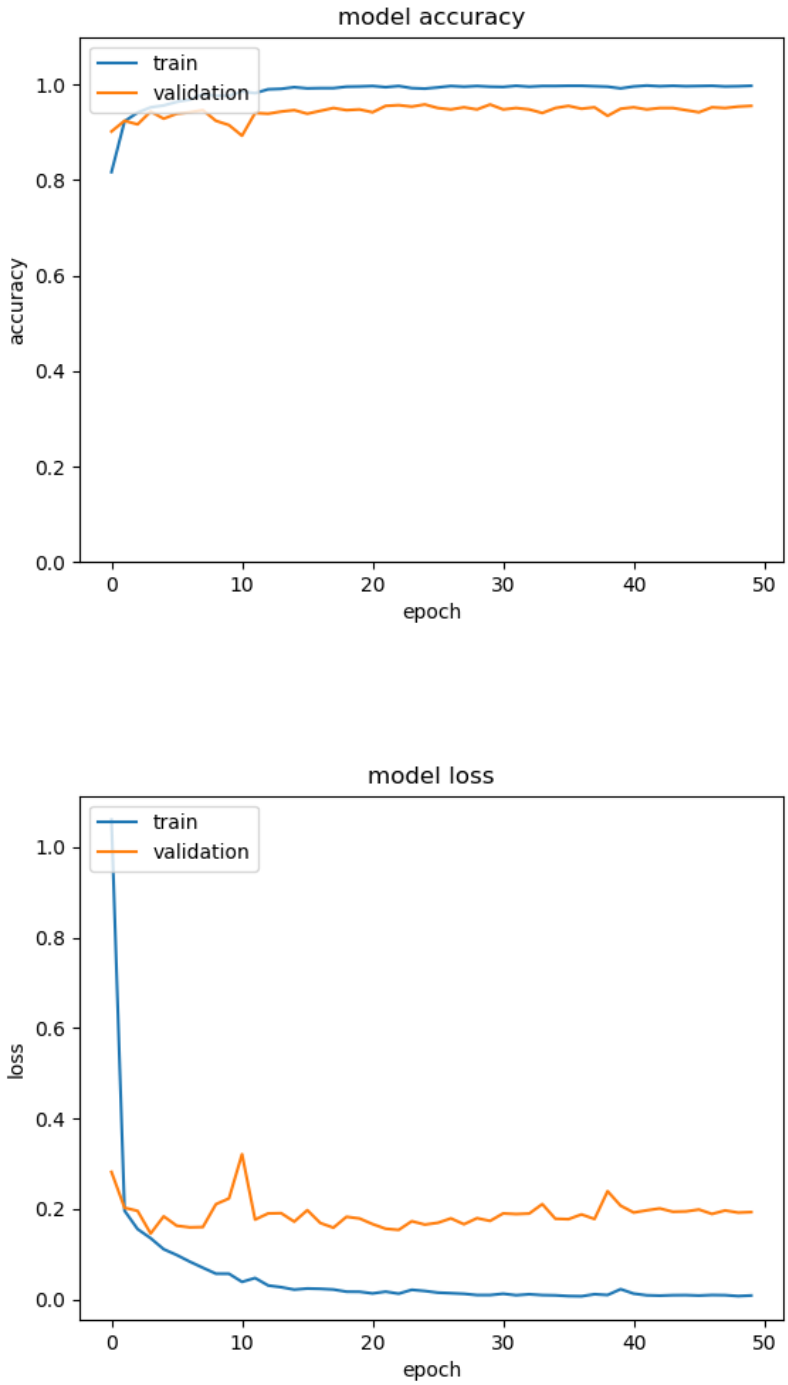


Figure 4.15: Training logs for the classification based on semantically segmented images

5

Real World Performance

This chapter covers the testing of trained network on real-life and unseen dataset. The saved model from the classifier network based on the semantically segmented network preceeded by the PSPNet is chosen as the prediction architecture for the implementation. The navigation setup is implemented using the Pepper robot as the navigator and the environment is the Ericsson office premises.

The structure of the implementation is shown in figure 5.1. The software is divided into two sections. The server side performs the tasks related to the image capture, image processing to segmentation and finally running prediction on the input image. The prediction instruction is sent over web sockets to the client side. The client side is responsible for the interface with the Pepper robot. The client side receives the prediction instructions from the server and sends commands to the robot, and sends instruction back to the server upon completion of the robot motion.

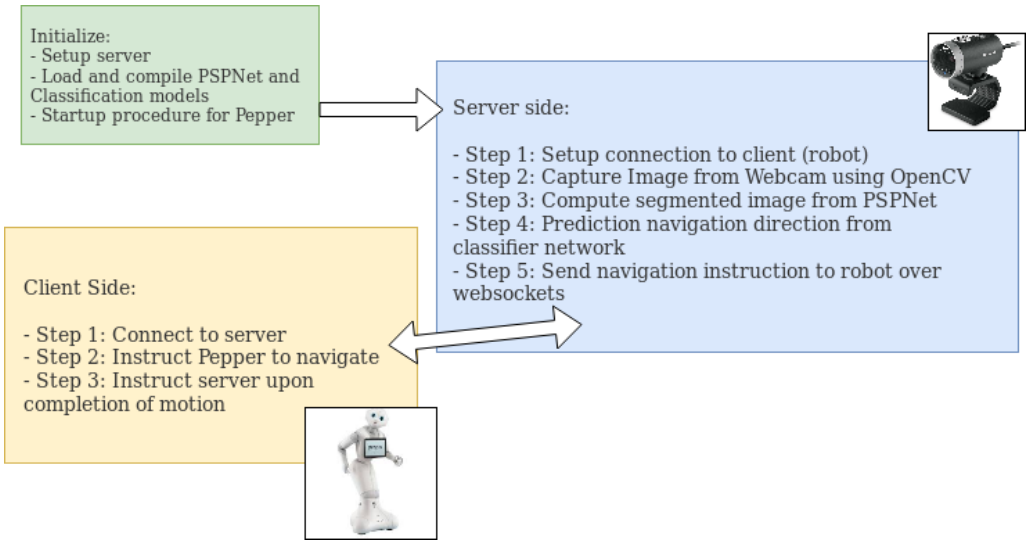


Figure 5.1: Software Architecture of the final implementation on Pepper robot

The server side runs on a HP Elitebook x360 1030 G3 Notebook PC, with Intel® Core™ i5-8250U with Intel® UHD Graphics 620 (1.6 GHz base frequency, up to 3.4 GHz with Intel® Turbo Boost Technology, 6 MB cache, 4 cores). The specifications are sufficient to run the required deep learning operations, although the execution time is quite slow. The hardware specifications have been the key issues in terms of performance times for all test cases.

Setup

To test the performance using Pepper robot, two test cases have been designed. The first implementation tests the performance of the classification network in conjunction with the robot, in the Ericsson premises. This would be the first checkpoint for the proof of concept, as it tests the robot's ability to navigate through obstacles using data which co-relates with the training images. The second test is designed to check the performance of the classification network on unseen data which does not co-relate to the training images at all. This provides a generalized idea of the performance of the classification network. The model that has been tested, in both test cases, follows the basic flow as illustrated in Figure 5.2.

Figure 5.2: Setup for case tests

Test case 1

The first set of tests were performed in the Ericsson office campus with the Pepper robot. Care was taken to test the performance in different lighting, obstacles and floor patterns to exhaustively test the method. The sample of environments tested in, are shown in figure 5.3.
Multiple runs were performed in the office area using the predicted directions and compared with the directional assessment by human classifier. The test runs produced 299 movements, 98 in left direction, 94 in right and 107 in straight direction. The overall accuracy of the movements made was 93.97%. Figure 5.4 illustrates the confusion matrix, which gives insight into the classification capabilities of the method.
The network classification is seen to perform very well, with all the classes being predicted accurately. It indicates that the testing accuracy is close to the validation accuracy, making this a good network for the application of indoor navigation using rgb images.

Test Case 2

The second test case is chosen to test the network on completely unseen data. This is to study the performance of the network on environments different from the ones trained on.
This test is not performed on the Pepper but the predictions are compared with the human classifier. The test images are chosen with different lighting, rooms with more unknown objects and different camera angles. This set is extremely adversarial,

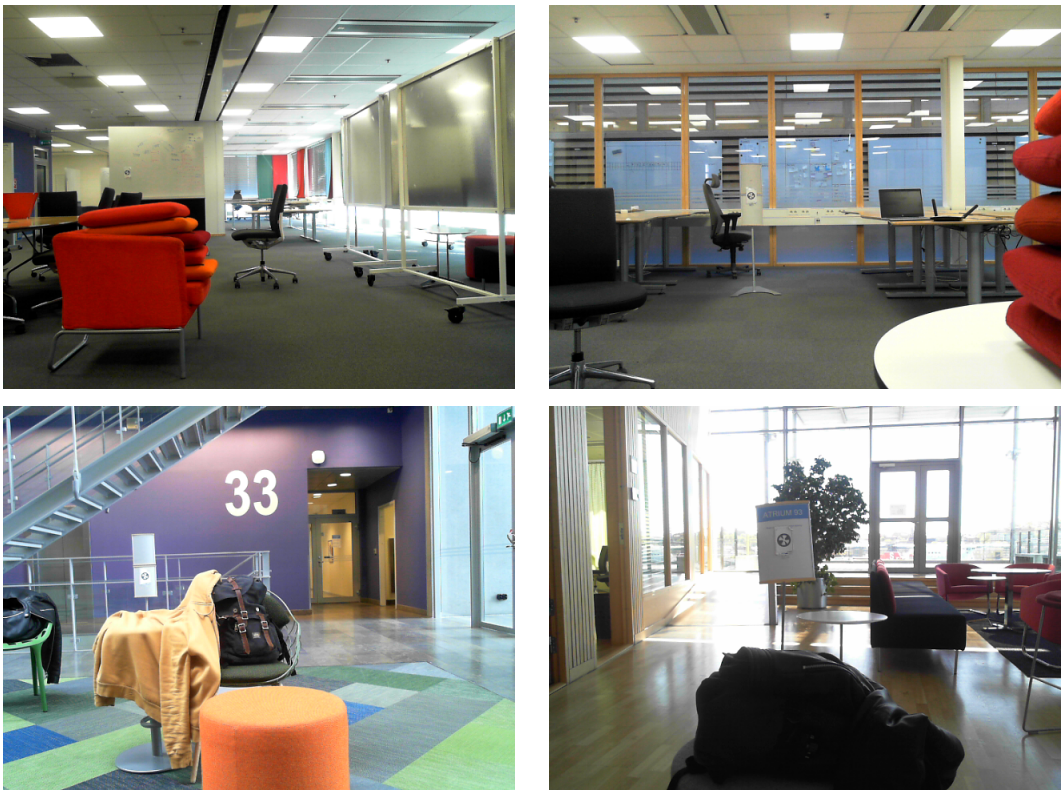


Figure 5.3: Sample of environments in which the method was tested on

with ambiguous classes for even human classifiers and including rendered images. A sample of the images are shown in figure 5.5.
The network was tested on a total of 117 images split among 48 left, 36 right and 33 straight classes. The images were scrapped off the internet using a Python web-scrapper. A cautionary warning must be provided as the webscrapper can produce a large number of irrelevant data.
The network predicted with an accuracy of 82.05% on this test set. The detailed split of the predicted classes is illustrated in the confusion matrix, figure 5.6.
While the accuracy is still quite good, it fails to classify many more classes in this set. The classification of left class is extremely accurate, the right class with medium accuracy and straight class with the lowest accuracy among the three classes. However, observing the confusion matrix, it is observed that the low accuracy comes from the classification of straights as lefts and lefts as straights. This prompts the inspection of segmented images from the test set.
The segmented image by themselves don't offer any clues for the wrong classification. However, the network resizes the image to 101x101. This change in resolution, changes the spatial information of the image, resulting in incorrect classification. Figure 5.8 shows the input to the classification network from the new test case. This shows the clear shift in the most floor space area from near the center of the image to the left of the image. Therefore it's recommended that capture rgb images for the network maintain a squared aspect ratio as opposed to rectangular.

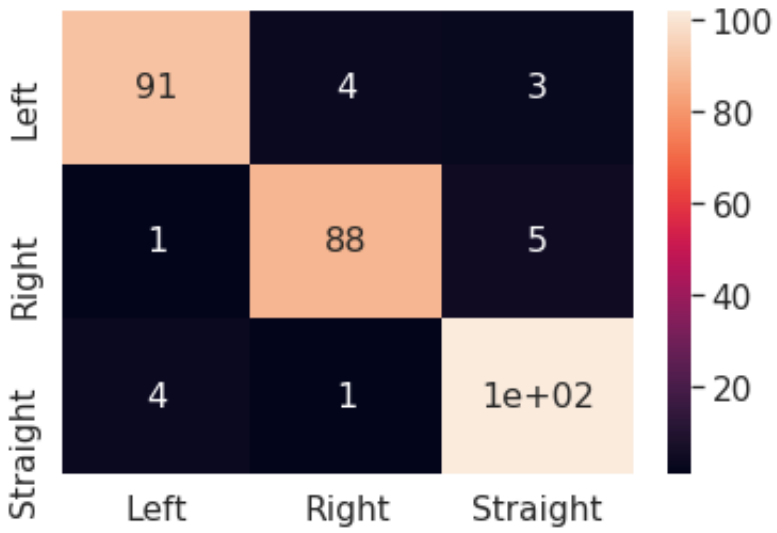


Figure 5.4: Confusion Matrix for test case 1

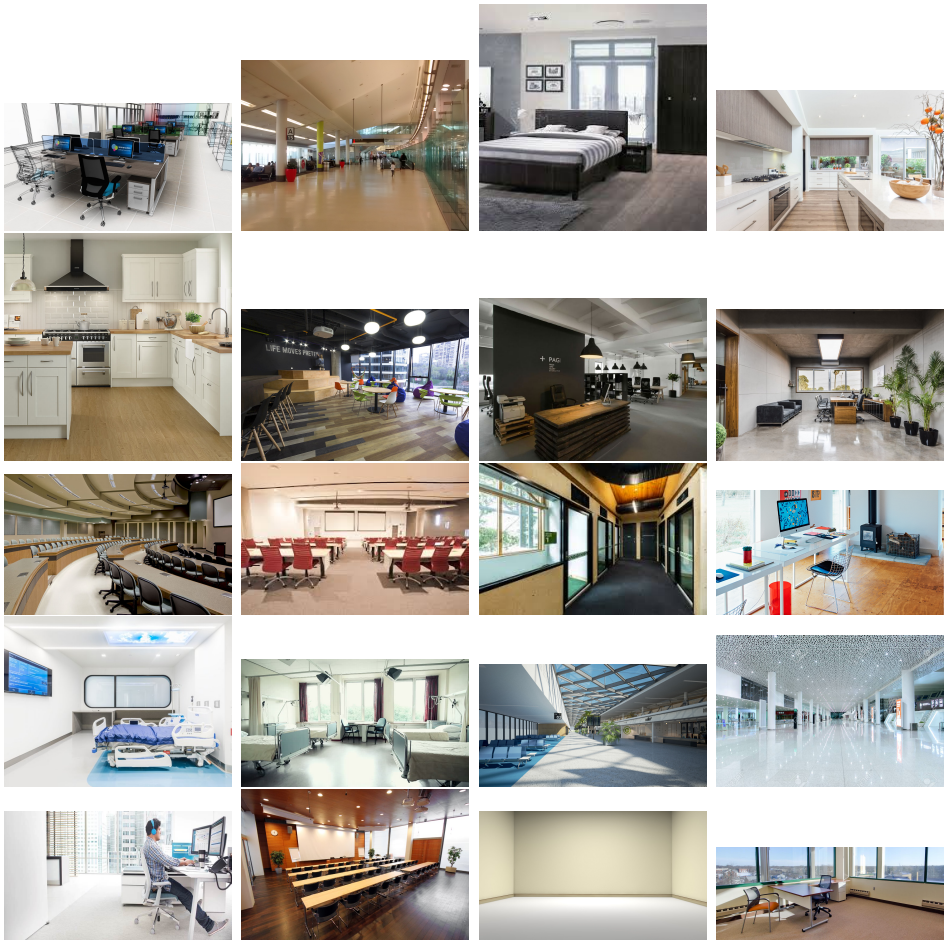


Figure 5.5: Sample of environments for the second test case

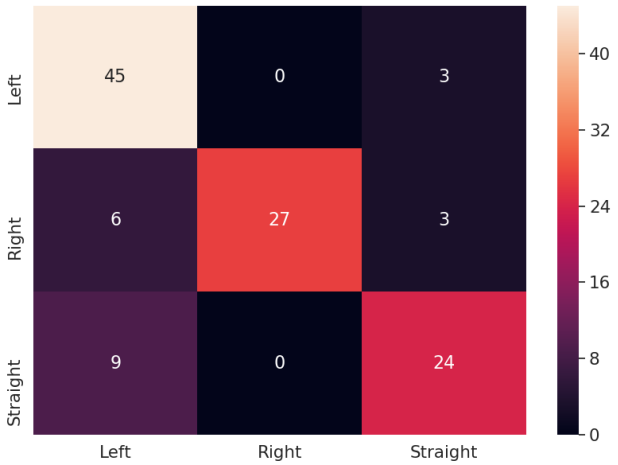


Figure 5.6: Confusion matrix for test case 2

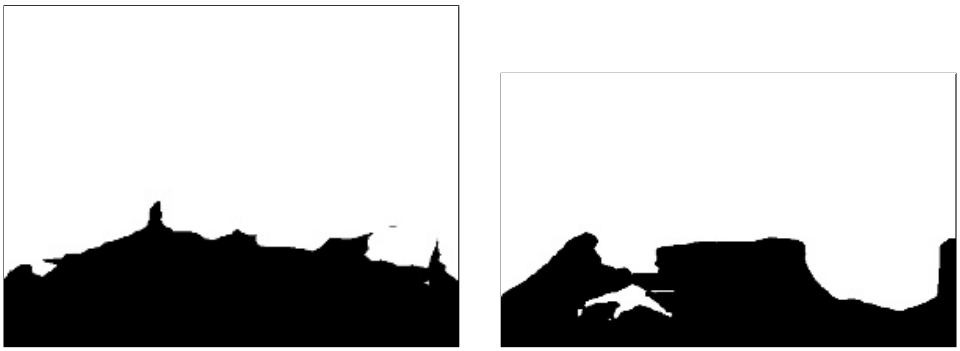


Figure 5.7: Segmented images of the second test set

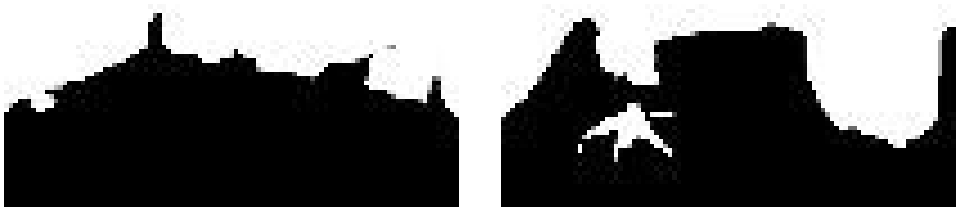


Figure 5.8: Resized images as input to the classifier network demonstrates the distortion of spatial features

6
Conclusion and Future Work

Conculsion

To sum up the presented results, the project shows that the explored method for indoor navigation using just RGB images could be in a future replacement of the sensor based methods as its cheaper version. The project lays down a basic guideline for data collection and investigates the minimal requirements from an image to successfully perform the navigation task. The project also proves the generalization of PanoNavi network to segmented images, providing an accuracy of 93.97% on the test dataset.
For the implementation of robot, the presented result of testing accuracy of 93.97% still leaves on average unsure 6% and thus for the safety of the robot there would still need to be a sensor backup solution to make sure the robot does not crash into the surroundings.
The segmentation network combined with the classification network also performs well in completely untrained environments, with an accuracy of 82% despite the test set being poor in quality. Overall, the results in this study indicate that it would be worth continuing research on this topic using machine learning approaches.

Future Work

The main limitation that this project was facing was lack of the relevant dataset, and thus the main improvement that could be achieved in the future work would be by creating better and wider dataset or explore existing datasets to find suitable ones for this application.
The current solution focuses on real time frame-by-frame decision making using CNN. An alternative approach could be to attempt the problem using sequence of images and a recurrent neural network (RNN) or reinforcement learning on video sequences.

Bibliography

[1] Aldebaran Software Documentation. "Pepper Documentation," 2016. [Online]. Available: http://doc.aldebaran.com/2-4/home_pepper.html. Accessed on: January 18, 2019.

[2] R. C. Gonzalez, "Deep convolutional neural networks [lecture notes]," *IEEE Signal Processing Magazine*, vol. 35, no. 6, pp. 79–87, Nov 2018.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[4] A. Rosebrock. A gentle guide to deep learning object detection. [Online]. Available: https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/

[5] C. Pardo-Beainy, E. Gutierrez Caceres, F. Jiménez L., and L. Quintero, "Disparity map generation, from the use of rectified images," 09 2013, pp. 1–6.

[6] M. Boden, J. Bryson, D. Caldwell, K. Dautenhahn, L. Edwards, S. Kember, P. Newman, V. Parry, G. Pegman, T. Rodden, T. Sorrell, M. Wallis, B. Whitby, and A. Winfield, "Principles of robotics: regulating robots in the real world," *Connection Science*, vol. 29, no. 2, pp. 124–129, 2017. [Online]. Available: https://doi.org/10.1080/09540091.2016.1271400

[7] Microsoft®. (2018) Lifecam studio™. [Online]. Available: https://www.microsoft.com/accessories/en-us/products/webcams/lifecam-studio/q2f-00013

[8] L. Pagliarini and H. Hautop Lund, "The future of robotics technology," *Journal of Robotics, Networking and Artificial Life*, vol. 3, p. 270, 02 2017.

[9] Travis Deyle. "Why Indoor Robots for Commercial Spaces Are the Next Big Thing in Robotics," 2018. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/robotics-hardware/indoor-robots-for-commercial-spaces. Accessed on: January 28, 2019.

[10] O. Wijk and H. Christensen, "Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data," *Robotics and Autonomous Systems*, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889099000858

[11] Eric Roberts. "Robotics : A brief history," 2018. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html. Accessed on: January 24, 2019.

[12] The Editors of Encyclopaedia Britannica. "George Charles Devol, Jr.," 2019. [Online]. Available: https://www.britannica.com/biography/George-C-Devol. Accessed on: January 25, 2019.

[13] Parmy Olson. "Softbank's Robotics Business Prepares To Scale Up," 2019. [Online]. Available: https://www.forbes.com/sites/parmyolson/2018/05/

30/softbank-robotics-business-pepper-boston-dynamics/#461b68224b7f. Accessed on: February 1, 2019.

[14] L. Z. Li Wang and G. Huo, "Visual semantic navigation based on deep learning for indoor mobile robots," *Complexitys*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889099000858

[15] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," *CoRR*, vol. abs/1609.05143, 2016. [Online]. Available: http://arxiv.org/abs/1609.05143

[16] A. C. e. a. Hernández, "Object detection applied to indoor environments for mobile robot navigation," *Sensors (Basel, Switzerland)*, vol. 16,8 1180, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889099000858

[17] M. Scheutz, "What is robot ethics? [tc spotlight]," *IEEE Robotics Automation Magazine*, vol. 20, no. 4, pp. 20–165, Dec 2013.

[18] IEEE robotics and automation society. "Robot ethics," 2018. [Online]. Available: https://www.ieee-ras.org/robot-ethics. Accessed on: January 24, 2019.

[19] DeepAI. "What is a Neural Network?," 2018. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/neural-network. Accessed on: April 8, 2019.

[20] A. S. Walia. (2017) Activation functions and it's types-which is better? [Online]. Available: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

[21] V. Suárez-Paniagua and I. Segura-Bedmar, "Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction," *BMC Bioinformatics. 2018; 19(Suppl 8): 209*, 2018 Jun 13.

[22] 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, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[23] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," *Lecture Notes in Computer Science*, p. 740–755, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10602-1_48

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298594

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR), Jun 2016. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.90

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.81

[30] R. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.169

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, p. 1137–1149, Jun 2017. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2577031

[32] Pedro Marcelino. "Transfer learning from pre-trained models," 2018. [Online]. Available: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751. Accessed on: April 11, 2019.

[33] Techopedia. "Computer vision," 2018. [Online]. Available: https://www.techopedia.com/definition/32309/computer-vision. Accessed on: March 17, 2019.

[34] Ayn de Jesus. "Computer Vision Applications – Shopping, Driving and More," 2016. [Online]. Available: https://emerj.com/ai-sector-overviews/computer-vision-applications-shopping-driving-and-more/. Accessed on: March 17, 2019.

[35] Pablo Revuelta Sanz, Belén Ruiz Mezcua and José M. Sánchez Pena. "Depth Estimation – An Introduction," 2015. [Online]. Available: https://www.intechopen.com/books/current-advancements-in-stereo-vision/depth-estimation-an-introduction. Accessed on: April 12, 2019.

[36] P. S. Hock, S. Parasuraman, M. A. khan, and I. Elamvazuthi, "Humanoid robot: Behaviour synchronization and depth estimation," *Procedia Computer Science*, vol. 76, pp. 276 – 282, 2015, 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915037916

[37] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *CoRR*, vol. abs/1812.11941, 2018. [Online]. Available: http://arxiv.org/abs/1812.11941

[38] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[39] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[40] M. M. N. E. R. A. P. I. Mendoza Guzman Vıctor Manuel, Meja Munoz Jose Manuel and S. R. Everardo, "Disparity map estimation with deep learning in stereo vision," *Universidad Autnoma de Ciudad Jurez*, 2017.

[41] J. Rambhia. (2013) Disparity map. [Online]. Available: https://jayrambhia.com/blog/disparity-mpas

[42] Y. Chen and W. Chiu, “Optimal robot path planning system by using a neural network-based approach,” in *2015 International Automatic Control Conference (CACS)*, Nov 2015, pp. 85–90.

[43] U. U. Thomas Hellström. "path planning". [Online]. Available: https://www8.cs.umu.se/kurser/5DV053/VT11/utdelat/Lectures/ch9-10.pdfl

[44] J. S. Carl-Henrik Hult, “Human-machine interaction, communication initiation probability estimation,” 2018.

[45] L. Taylor and G. Nitschke, “Improving deep learning using generic data augmentation,” *CoRR*, vol. abs/1708.06020, 2017. [Online]. Available: http://arxiv.org/abs/1708.06020

[46] E. W. Weisstein. "shear.". [Online]. Available: http://mathworld.wolfram.com/Shear.html

[47] L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning,” 12 2017.

[48] F. Chollet. About opencv. [Online]. Available: https://opencv.org/about/

[49] OpenCV. Disparity map post-filtering. [Online]. Available: https://docs.opencv.org/3.1.0/d3/d14/tutorial_ximgproc_disparity_filtering.html

[50] A. Ouaknine. Review of deep learning algorithms for image semantic segmentation. [Online]. Available: https://medium.com/@arthur_ouaknine/review-of-deep-learning-algorithms-for-image-semantic-segmentation-509a600f7b57

[51] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” *CoRR*, vol. abs/1612.01105, 2016. [Online]. Available: http://arxiv.org/abs/1612.01105

[52] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[54] F. Chollet. (2016) Building powerful image classification models using very little data. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[55] L. Ran, Y. Zhang, Q. Zhang, and T. Yang, “Convolutional neural network-based robot navigation using uncalibrated spherical images,” *Sensors*, vol. 17, no. 6, p. 1341, 2017.

A

Appendix 1: Auxiliary Cameras

For the entirety of the project, a binocular vision system has been the requirement. Although the robot, Pepper, has three in-built cameras, the positioning of the cameras were not suitable for the use-case. To assist the robot in the navigation, an auxiliary binocular system was fabricated.
The two auxiliary cameras added are Microsoft® LifeCam Studio™. The specification of the cameras are listed in table A.1.

Device Type	Webcam
Connection Type	Wired (4-pin USB Type A)
Manufacturer	Microsoft
Compliant Standards	UL, VCCI, ISO 9001, GOST, cUL, ICES-003, ISO 14001, WHQL, CB, MIC, FCC, WEEE, UkrSEPRO
Image Sensor	CMOS
Focus	Automatic
Focal length	Minimum 10 cms
Image Resolution	2560 x 2048

Table A.1: Specifications of Microsoft® LifeCam Studio™ [7]

The auxiliary system provides more freedom in the position of the system and controlling the distance between the cameras for optimal performance for the project. The cameras were housed in 3D printed assembly using ABS Plastic. The camera mounting had to have the flexibility in terms of placement of cameras, alignment of the assembly with respect to the robot and provide provisions for addition of Raspberry Pi. The schematics of the assembly and the sub-components are illustrated in figure A.1.

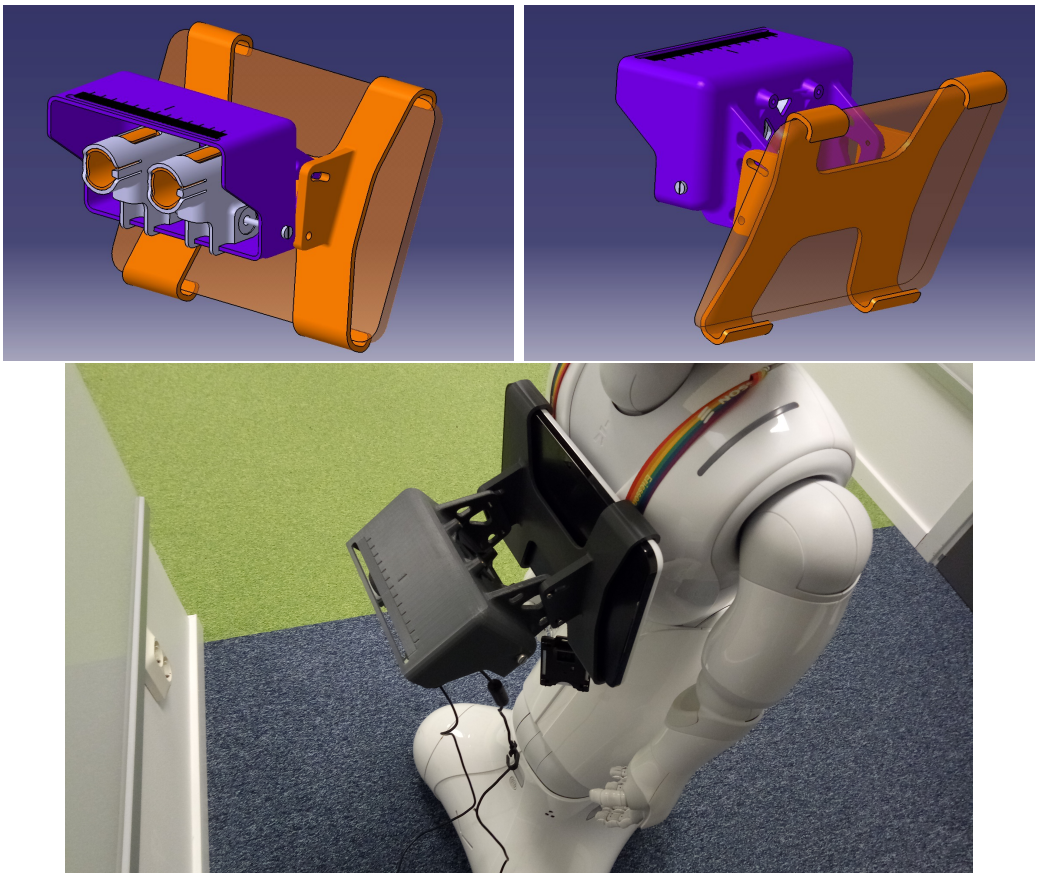


Figure A.1: Auxiliary camera mounted onto the robot touch-screen for capturing the left and right images for depth map generation

B

Appendix 2: Data Collection

Collection and annotation of data for the training of the neural networks is the most critical part of the project. The data needs to be relevant to the problem, provide clear images with a good distribution and equal distribution of the classes, and a variety of image features. The data collection was done at the Ericsson office space - the indoor space with a good collection of obstacles, yet not too many to require retraining of the object detection network.
The robot and auxiliary camera setup was controlled by a script run on a laptop and run in a controlled environment. The captured image is saved after before next movement of the robot, for every step of the robot. The saved images were stored in the following directory structure:

Data
|— Left
|— Right
|— Straight

Figure B.1: Examples of the environment where data was collected

Figure B.1 illustrates examples of the environment where the data was collected. The conditions controlled under for the data collection were the obstacle types, obstacle positions, target placement and target distance. It was also ensured that the collected data had a good distribution of images under different lighting and different backgrounds to help generalize the problem and to help avoid overfitting in the neural network.