Tampere University

Palash Halder

A ROBOTIC ENGINE ASSEMBLY PICK-PLACE SYSTEM BASED ON MACHINE LEARNING

Automation Engineering
Master of Science
February 2020

ABSTRACT

Palash Halder: A robotic engine assembly Pick-place system based on machine learning
Master of Science
Tampere University
Automation and Robotics
February 2020

Industrial revolution brought humans and machines together in building a better future. Where in one hand there is need to replace the repetitive jobs with machines to increase efficiency and volume of production, on the other hand intelligent and autonomous machines have still a long way to go to achieve dexterity of a human. The current scenario requires a system which can utilise best of both the human and the machine. This thesis studies a industrial use case scenario where human-machine combine their skills to build an autonomous pick place system.

This study takes a small step towards the human-robot consortium primarily focusing on developing a vision based system for object detection followed by a manipulator pick place operation. This thesis can be divided into two parts : 1. Scene analysis, where a Convolutional Neural Network (CNN) is used for object detection followed by generation of grasping points using object edge image and an algorithm developed during this thesis. 2. Implementation, it focuses on motion generation while taking care of external disturbances to perform successful pick-place operation. In addition human involvement is required which includes teaching trajectory points for the robot to follow. This trajectory is used to generate image data-set for a new object type and thereafter generating new object detection model. The author primarily focuses on building a system framework where the complexities related to robot programming such as generating trajectory points and informing grasping position is not required. The system automatically detects object and performs a pick place operation, resulting in relieving user from robot programming. The system is composed of a depth camera and a manipulator. Camera is the only sensor available for scene analysis and the action is performed using a Franka manipulator. The two components work in request-response mode over ROS.

This thesis introduces a newer approaches such as, dividing an workspace image into its constituent object images and performing object detection, creating training data, generating grasp points based on object shape along length of an object. The thesis also presents a case study where three different objects are chosen as test objects. The experiments are a demonstration of the methods applied and efficiency attained. The case study also provides a glimpse of the future research and development areas.

Keywords: CNN, ANN, ROI, ROS

PREFACE

The past few years has been a long journey from settling in a foreign country to making new friends and gaining knowledge to develop as a human being. With this thesis I mark the beginning of my career in the field of automation and robotics. To finish this work I needed patience, hard-work, work-life balance to name a few. Also, the support of my thesis supervisors and people around me helped me to get going and complete this thesis.

The work presented in the thesis implemented machine learning methods in robotic assembly application. This thesis was carried out at the Faculty of Engineering and Natural Sciences at Tampere University. I owe my deep gratitude to Assistant Professor Roel Pieters for providing me with this opportunity, his ideas and being patient with me through this journey. I would also like to thank Doctoral researcher Alexandre Angleraud for his feedback and ideas during the thesis work.

Here, I would like to express my gratitude and thank my parents, family members and friends back home for their support during my studies. I would like to express my gratitude towards my employer Wartsila Finland Oy and Visual Components Oy to provide me ambient environment to continue with my thesis while working.

And lastly, but by no means least, I would like to thank the Department of Automation Technology and Mechanical Engineering (ATME) for providing me with the Robotics lab and state-of-art robot and equipment to complete my thesis.

Tampere, 20th February 2020

Palash Halder

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

1.1 Overview

Object manipulation and grasping is a capability that humans have developed over a long period of time with evolution. With greater need of automation enabling a robot to autonomously detect and perform grasp in cluttered or uncluttered scene is a major issue for which we are still trying to find a feasible solution. Currently robots are either carefully hand-programmed or scripted or they are directly controlled by a human operator. In the current scenario grasping different objects in a factory environment would require a complex program which will have limitation in flexibility and scalability. However, thanks to recent development in image recognition technology and affordable computing there is greater possibility of combining robotics and machine learning to create a system which can detect and grasp object with minimal human involvement.

Open-source deep learning libraries like *tensorflow, Pytorch & Torch, Caffe* etc. have made it possible to build neural networks as simple as joining *LEGO* blocks. Generally, there are ready made data-sets with which the network can be trained and tested. In cases where there are no ready-made data-set to train a network, user has to create its own data-set. In addition there are image generators which can automatically create a set of images with different characteristics from a single image to generate a larger training data-set. In short in order to use the power of Neural network one needs an existing training data-set and a network which can learn the features to differentiate between different objects.

State-of-the art grasping technology such as developed by *covariant.ai* uses reinforcement learning and special suction type grippers to perform grasping operation (Ackerman 2020). This system also uses *sim2real* transfer where the network is trained in simulation in addition to real-life training. Similarly, another state-of-art grasping method invented by *open.ai* utilizes reinforcement learning as the brain for grasping (Simon 2020). This system also relies on training during simulation to generate a model which can have 100 years of experience in grasping different objects. All these technologies require higher computation power and rely on learning from the mistakes similar to a human, as such it is possible to generate the best model suited for a specific task. In contrast to the current trends, the previous grasping methods relied on shape based grasp model, these type of models were inefficient in case of an unknown object in uncluttered or cluttered scene (Klingbeil et al. 2011), also in cases of known objects it was difficult to obtain a full

3-d model of object and apply approaches such as friction cones (Mason and Salisbury 1985), form- and force-closure (Bicchi and Kumar 2000), pre-stored primitives (Miller et al. 2003).

In this thesis the emphasis is to build an autonomous robotic system which can pick and place different parts of an engine. This starts with creating a training data-set of the engine parts followed by creating a Convolutional Neural Network (CNN) to learn features of different image parts. Together with work done for object detection the thesis also presents an algorithm to identify the finger positions and wrist orientation for a two-fingered robot gripper.

1.2 Objective and Approach

The overall objective of this thesis is to fabricate an autonomous robotic pick and place system with the following requirements:

1. Efficient object detection.
2. Generate grasp positions.
3. Scalability.

Here, item 1 suggests that the developed classifier should be able to identify and locate the object in 2D workspace. Using an intel real-sense camera the object location is determined w.r.t the robot base-frame. The next item 2 states grasping position should be such that the robot can grip and pick the object up, the object should not fall during motion between pick up location to drop location. The last requirement 3 means user can add newer object to the system, in order to add newer object user has to create a image data-set and add it to the existing data-set. The network should be trained again to perform efficient object detection. In this thesis three different component of an engine were chosen as test case as shown in figure 5.1.

The approach implemented in this thesis can be explained by figure 1.1.

In **stage 1** the robot sets itself in start position overlooking the work-space. The Intel real-sense camera mounted on gripper takes an image of work-space and finds different objects present in work-space.

If the object specified by user is found in the work-space then the grasping algorithm tries to find an optimum grasping position in **stage 2**.

On request from robot, the grasping position in camera coordinates is sent to robot which is converted to robot base frame coordinates and the robot controller initiates the grasp thread in **stage 3**.

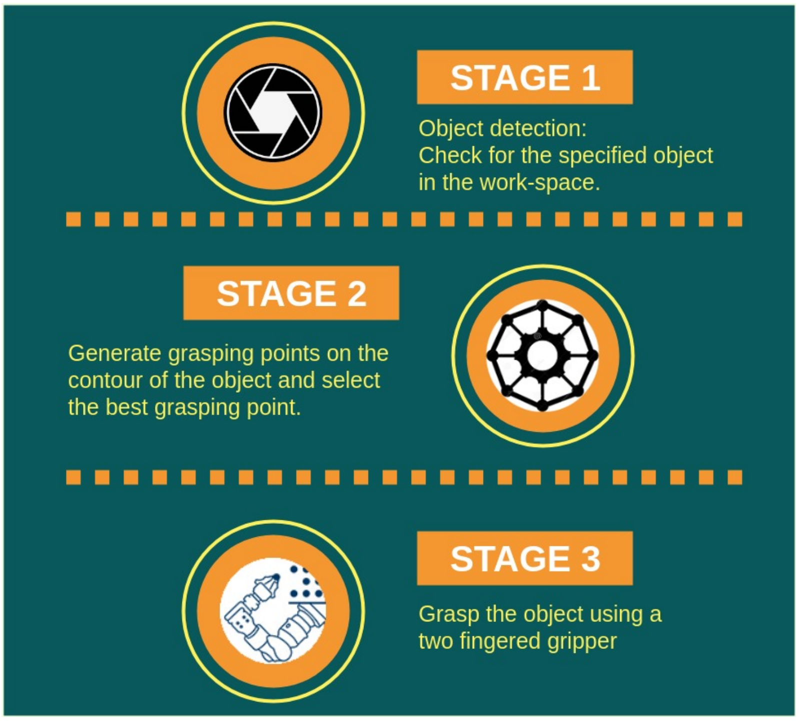

STAGE 1
Object detection:
Check for the specified object in the work-space.
STAGE 2
Generate grasping points on the contour of the object and select the best grasping point.
STAGE 3
Grasp the object using a two fingered gripper

Figure 1.1. *Research Methodology*

1.3 Research questions

All the three requirements mentioned in section 1.2 of the system will be discussed in terms of the techniques of fabrication, methodology, designs, and the results will be discussed in this thesis. However, the specific objective of the thesis is to create a framework which is self reliable and scalable enough to perform robotic pick and place in a cluttered environment with minimal human interference. As such the thesis aims at answering the following research questions:

1. What features a machine learning framework to have for an industrial pick-place application?
2. How to select grasp positions?
3. What kind of robot controller to use?

1.4 Limitations and Challenges

The following limitations and challenges were met during the development of this thesis.
Limitations:

- The thesis is based on using a two fingered gripper - although these grippers are easy to manufacture and use, three fingered grippers are suitable to pick delicate objects with precision and strength (ennomotive 2018).
- It is assumed that the object color and the background color is different such that object shape can be detected from the background - edge detection is successful if

there exists a gradient in the pixel value on both side of the edge. In cases where there is no such gradient we may get false edges or no edges at all.

- The gripper is perpendicular to the work-space plane at the time of grasping - grasp is performed with a top-down motion, the grasp algorithm checks for the points along the side of the object. The force exerted by the gripper is along the plane of surface. Other gripping positions are neglected in this thesis.
- Grasp are performed at the outer edges to the object - with the current algorithm if we had to grasp a tea cup, it will grip the cup at the edges rather than gripping the handle of the cup.
- With increase in number of object types modification in CNN will be required - the simple CNN developed is not adequate for greater number of object types.

Challenges:

- No ready made image set present - since the engine components are unique and there are no ready-made image data-set available from the manufacturer, the data-set needed to be generated. The image data-set generation is explained in section 3.1.
- Object types are part of an engine, thus uncommon - there is not one solution to pick objects of these types, e.g. a cup should be grasped at the handle. Here from a set of grasp position, the one placed centrally were chosen.
- Effect of shadows on detecting object shape - shadows created false edges, the effect can be reduced by changing the parameters of the edge detector.
- Ready made packages for path planning and grasping were unsuitable in this case - 3rd-party motion planner caused limit violation for the robot on collision with surface. A motion planner based on trapezoidal velocity profile was developed to compensate for collision during pick and place of the objects as explained in algorithm 1.
- Separate computers for vision and robot controllers - Real-time kernel is required in robot controlling PC. With real-time kernel it is not possible to use Nvidia Card. Hence, two separate computers were used one for object detection and the other for robot control.

1.5 Outline

This section provided an overall structure of thesis. The thesis is structured as follows. The first chapter introduced with the issues this thesis tries to solve. This is followed by the second Chapter introducing the underlying concepts and theories which formulated this thesis. Chapter 3 introduces the methodologies developed and the research during the course of this thesis followed by Chapter 4 with the implementation of methods and the result of this thesis. Chapter 5 discusses how the research questions were solved and concluded this thesis with conclusion and future scope.

2 THEORETICAL BACKGROUND

In this chapter, an overview of the theoretical background relevant to the scope of this thesis is provided. Section 2.1 introduces the concept of image segmentation and classification both classical and latest advancement in image segmentation are discussed. This is followed by section 2.2 which describes the robot grasping techniques and its theories. Section 2.4 describes robot control and discusses the various types of robot control available and provides a comparison between the methods.

2.1 Image Segmentation

In this section image segmentation techniques are discussed. Image Analysis and Pattern Recognition are the first stage for Image Segmentation. Clustering image areas into different regions based on homogeneity or discontinuity of certain characteristics like color, intensity or texture provides information about the objects presence in an image.

The process of Image Segmentation can be divided into three stages (Ku. Vasundhara 2014):

1. **Image processing** - Cleaning the image, which can be removal of outlier pixels to removing of noise.
2. **Initial object discrimination** - The image is divided into groups of objects having similar attributes.
3. **Object boundary clean up** - The edges of objects are reduced to single pixel width.

Discontinuity and **Regularity** at a pixel position are utilised to separate different regions in an image. A good segmentation techniques preserves the area, edges and degree of curvature for the objects in the image. In the consequent sections a brief description of the traditional methods is followed by discussing the state of art technology in image segmentation the **Deep Neural Network**.

2.1.1 Traditional Image Segmentation Method

Computer vision and image analysis requires efficient feature extraction, this features is similar to the ones used in machine learning and pattern recognition. A feature can be defined as an information which can be understood by the computer and is relevant to classify different image areas. Features focusing on different characters of an image are used for segmentation, such as Pixel color, Histogram of oriented gradients (HOG) (Dalal and Triggs 2005), Scale-invariant feature transform (SIFT) (Lowe 2004), SURF (Bay, Tuytelaars and Gool 2006), Harris Corners (Bruce and Kornprobst 2009), Features from Accelerated Segment Test (FAST) (Rosten and Drummond 2005) just to name a few. Both unsupervised and supervised learning can be used in image segmentation.

Binary Thresholding

Thresholding is a simple method for segmentation of gray scaled image which is widely used in biomedical areas e.g X-ray CT scanner or MRI (Magnetic Resonance Imaging) equipment which produces gray scaled images. As the name suggests popular method of binary thresholding converts a grey-level image into a binary image, at first a threshold value is selected, pixel values are classified as black (0) if it's value is below the threshold, while white(256) implies the pixel value is above the threshold. It is used to distinguish foreground from the background (M. Jogendra Kumar 2014). On the other hand Multilevel thresholding separate the image based to multiple threshold values and as such works very well for images consisting of different colored objects. figure 2.1 is an example of Binary Thresholding on grey image.

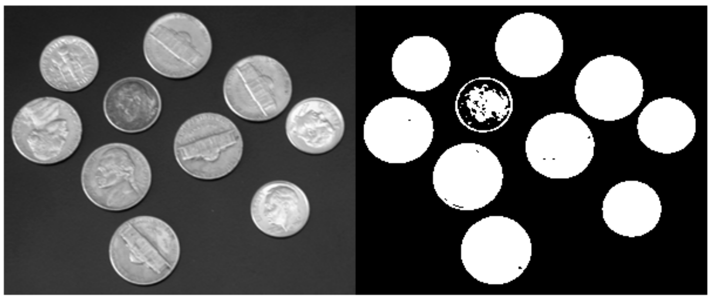


Figure 2.1. *Grey Image and Result of Binary Thresholding*
Source: `https://www.mathworks.com/help/images/ref/imbinarize.html`

K-means

K-means clustering is a popular algorithm for unsupervised clustering. As an initial requirement the desired number of clusters are provided beforehand to perform k-means clustering on an image. This number of clusters is analogous to the number of classes in classification. At first, the feature space is populated with randomly placed k number of centroids. In the next step, each data point is assigned to the nearest centroid, in successive steps the centroid moves to the center of the cluster, and continues the process of clustering until the stopping criterion is reached (Hartigan 1975). As such k-means algorithm is one of the easiest algorithm of understand and implement. It can be used in both binary and multilevel classification.

figure 2.2 shows an example of image segmentation using clustering. From the image it can be observed k-means was successful in segmenting out the dog from the background. In the example number of clusters was chosen as 3, which resulted in classifying the shades of the floor also.

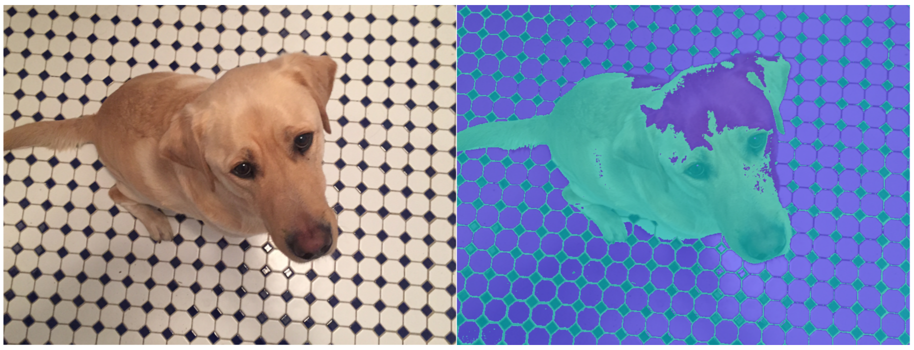


Figure 2.2. *Segmentation with k means*
Source: `https://www.mathworks.com/help/images/ref/imsegkmeans.html`

The algorithm is not suitable for large data-sets, since the time taken to look at all the images and perform k-means clustering is very high. Also classification result vary immensely with number of clusters being chosen. With increase in computation time the hardware requirements also become more expensive to implement.

Support Vector Machine (SVM)

Support vector machine (SVMs): SVMs are one of the popular binary classifiers which have successfully performed well on numerous tasks (Liu, Deng and Yang 2018). training data can be represented as (x_i , y_i) here x_i represents the feature vector and $y_i \in$ {-1, 1} represents the binary label for training example $i \in$ {1, . . . , m}. Where w is a weight vector and b is the bias factor (Liu, Deng and Yang 2018) as shown in 2.3

The binary classification equations are:

$$\vec{w} \cdot \vec{x} - b = 1 \text{(data points on or above the boundary is classified as 1)} \tag{2.1}$$

and

$$\vec{w} \cdot \vec{x} - b = -1 \text{(aata points on or above the boundary is classified as -1)} \tag{2.2}$$

The above equations can be rewritten as

$$\vec{y_i}(\vec{w} \cdot \vec{x_i} - b) \geq 1 \tag{2.3}$$

Geometrically distance between the two hyperplanes defined in Eq.2.1 and Eq.2.2 is $\frac{2}{\|\vec{w}\|}$. SVM tries to find an optimal solution which tries to find the minimum $\|\vec{w}\|$ which maximises the distance between the hyperplanes.

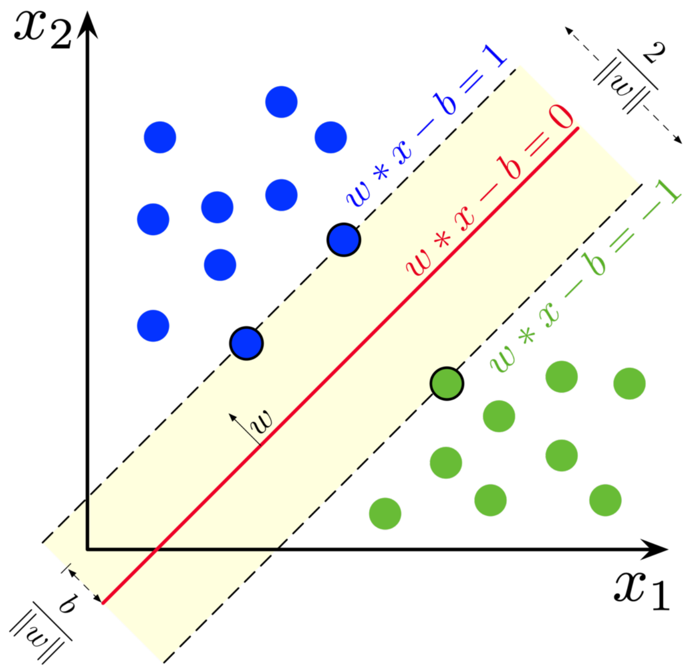


Figure 2.3. *SVM binary classifier*
Source: `https://en.wikipedia.org/wiki/Support-vector_machine`

Edge based segmentation

An edge can be defined as a series of connected pixels that lie on the border of two regions having different grey values. Pixel positions that are nearer to the boundaries of objects in the image are selected by edge-based segmentation. **Filtering**, **Differentiation** and **Localization** constitute the three main steps in the process of edge based segmentation (Chaturvedi et al. 2017). The most commonly used edge detection algorithms are Sobel, Canny, Prewitt, Robert, Marr and Hilldreth (Aravindh and Manikandababu 2015). Canny edge detection is more complex than other edge detection algorithms, inspite of additional complexity it is one of the best and popular edge detection technique (Ilkin et al. 2017). Mathematical expression of the three main goals for edge detection (Ilkin et al. 2017) are given below:

- Low error rate - the detected and actual edges are as close as possible to each other.
- The edge points can be well defined - the distance between actual edge center to detected edge point should be minimum.
- Single edge point answer - the detector selects only one point as the correct edge, the smallest local maximum around the edge.

In this thesis the edge based segmentation was applied. Edge detection provided a easy way to generate contour surrounding the objects. This contours were then classified as region of interest (RoI) by comparing the enclosed contour area with an empirically calculated threshold value.

2.1.2 Recent Deep Neural Network for Image Segmentation

Artificial Neural Network (ANN) got its inspiration from biological neurons. An artificial neuron forms the basic building block of an ANN. Each single artificial neuron takes some inputs which are then weighted and summed up. The weighted output is then passed through a transfer function or activation function, this is the score value from a single neuron (Liu, Deng and Yang 2018). An example of neural model is illustrated in figure 2.4. Different stacking of the neurons leads to formation of Restricted Boltzmann Machine (RBM) (Larochelle and Y. Bengio 2008), Recurrent Neural Network or Recursive Neural Network (RNN), Convolutional Neural Network (CNN) (LeCun and Bengio 1995), Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) and other types of ANNs. The basic architecture is illustrated in figure 2.4

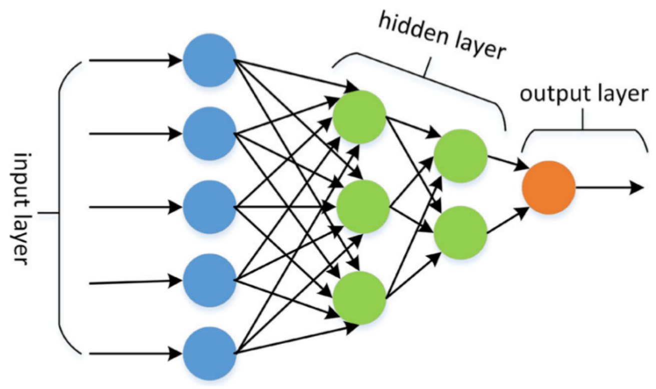


Figure 2.4. *Simple model of Artificial neural network*
Source: (Liu, Deng and Yang 2018)

Convolutional Neural Network (CNN) (LeCun and Bengio 1995) uses shared-weight architecture, which is inspired by biological processes. The inter-neuron connectivity pattern are made to mimic the organization of the animal visual cortex. Another important concept is receptive field, which makes selective response of individual cortical neurons to stimuli such that it is restricted to a region of the visual field. Also, CNN possess the property of shift invariant or space invariant, which is based on their shared-weight architecture and translation invariance characteristics. Due to this excellent structure of CNN, it has obtained remarkable results on image classification, segmentation, and detection. The following part will present the recent progresses which is based on CNN and its modification for better image semantic segmentation.

In the following sections some of the segmentation technique which formed the core idea for image segmentation in this thesis are described. There has been major development in the field of image segmentation but implementing the techniques as a whole required creating marked data-set such as used in *COCO* challenges. Since during this thesis an automated image data-set generation was implemented as explained in section 3.1, and the generated data-set consisted of single object type during training, a simpler CNN based on LeNet was sufficient for classification.

Fast Region-based Convolutional Network (Fast R-CNN)

Fast Region-based Convolutional Network (Fast R-CNN) (Girshick 2015) is based on region proposals i.e. Region of Interest (RoI). It reduces the time necessary to analyse all region proposals by analysing only the interest regions.

It starts with a main Convolutional Neural Network (CNN) which takes the entire image as input. This is followed by application of selective search methods on the produced feature maps from the previous layer to detect Region of Interests (RoIs) so that it can be fed into a fully connected layer. Also, RoI pooling layer is used to reduce the feature maps size to get valid RoI which have fixed height and width so that it can be fed into a fully connected layer. A softmax classifier is then used on the generated vector to predict the object and generate the localizations boxes.

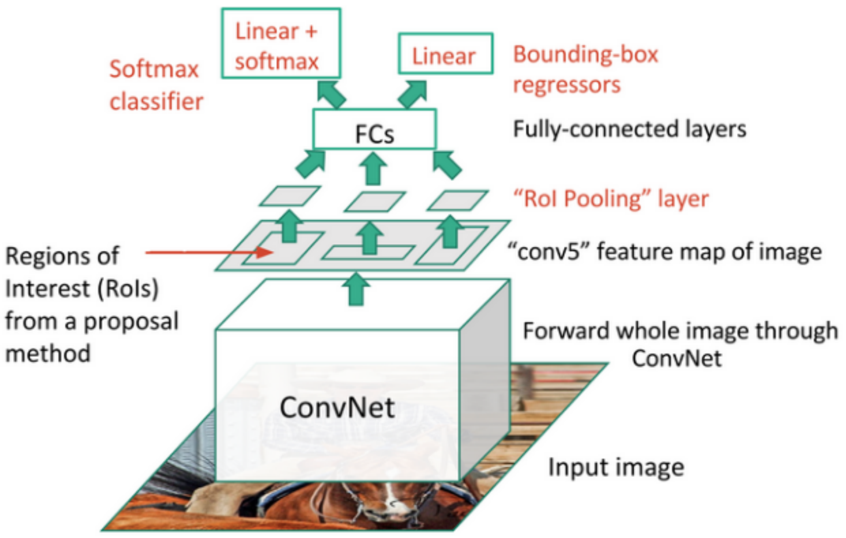


Figure 2.5. *Fast-RCNN architecture*
Source: (Xu 2017)

You Only Look Once (YOLO)

The YOLO model (Redmon et al. 2016) is targeted for real-time applications. It starts with dividing the entire image into grids which have multiple boundaries associated. Each grid cell can predict only one object.

The entire image is taken as input, this is then divided into $S \times S$ grid. Each grid cell is associated with one object and then the grid cells predict bounding boxes around the object. Every boundary box is defined by 5 elements x, y, w, h and a box confidence score. The confidence score is a parameter which indicates likelihood of an object (objectness) inside the box and the accuracy of the the boundary box. This is followed by normalization of the bounding box width w and height h by the image width and height. Hence, x, y, w and h are all between 0 and 1 (Hui 2018).

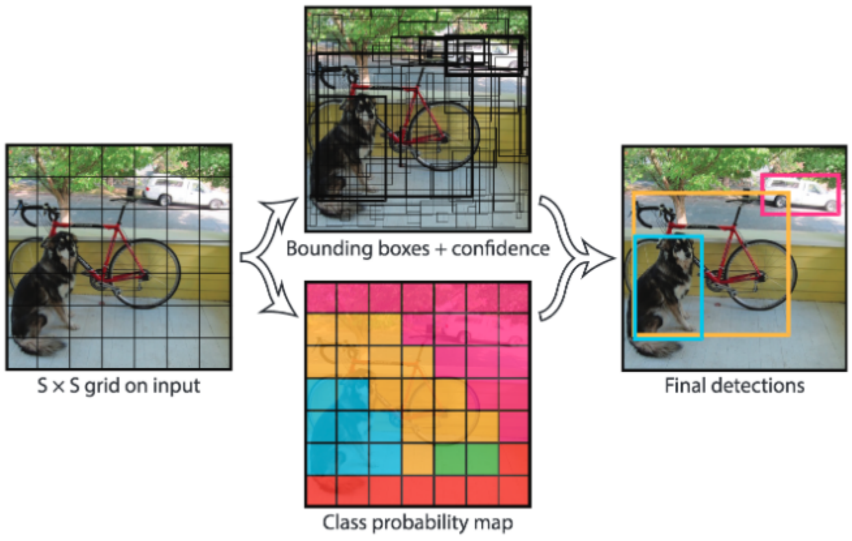


Figure 2.6. *YOLO example application. $S \times S$ Grid-size divides the entire input image, using regression B bounding boxes are predicted. Each bounding box is associated with a class having the highest confidence score.*
Source: (Redmon et al. 2016)

YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). If C is the number of estimated probabilities for each class. B is the fixed number of anchor boxes per cell, and each of these boxes are defined by the elements defined above namely x, y, w, h and confidence values, then the final layer provides an output $S \times S \times (C + B \times 5)$ tensor corresponding to the predictions for each cell of the grid.

Single-Shot Detector (SSD)

Single-Shot Detector (SSD) (Liu1 et al. 2016) is simlar to the YOLO model. In SSD all the bounding boxes and their respective class probabilities are predicted all at once with an end-to-end CNN architecture. The model uses different size of filters (10×10, 5×5 and 3×3) to construct the convolutional layers. The bounding boxes are predicted using the Feature maps from convolutional layers at different position of the network. As the final step the yield is at that point handled by a particular convolutional layers having 3×3 filters called extra feature layers which produces a set of bounding boxes which are comparative to the anchor boxes of the Fast R-CNN.

Each box is characterised by 4 parameters namely, center coordinates, width and height. In addition a vector representing the probabilities corresponding to the confidence for each object class of object is produced.

To keep only the most relevant bounding boxes a Non-Maximum Suppression method is also used at the end of the SSD model. To avoid lot of negative boxes being predicted a Hard Negative Mining (HNM) is also used, where only a a sub-part of these boxes is selected during the training. The boxes are ordered by confidence and the top is selected depending on the ratio between the negative and the positive which is at most 1/3.

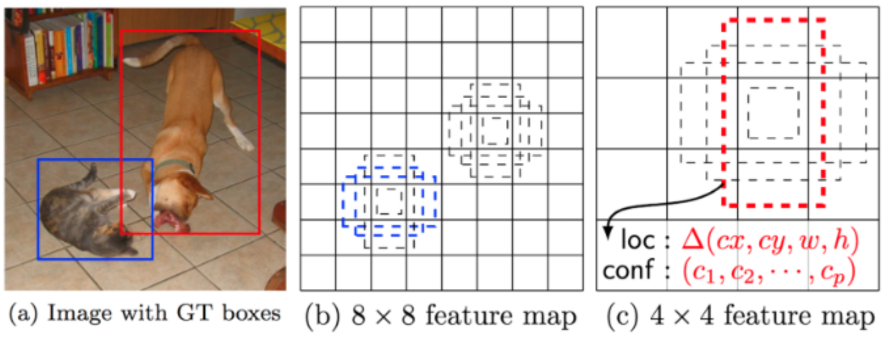

loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$
(a) Image with GT boxes
(b) 8×8 feature map
(c) 4×4 feature map

Figure 2.7. *SSD example application. Input to the model is the image and the ground truth bounding boxes. Boxes with different aspect ratio are selected by the feature map. The boxes localisation and aspect ratio are updated during training to match the ground truth*

Source: (Liu1 et al. 2016)

2.2 Robot grasp

This section will review some of the base concepts for generating a stable robot grasp. Here the theoretical basis for generating a two-fingered grasp is more relevant to the thesis and as such will be the focus. Although some concepts remain the same irrespective of the gripper type.

In the recent review of (J. Bohg and Kragic 2014a), the two main methods for grasp synthesis algorithms are Analytical and Data-Driven methods (Jabalameli, Ettehadi and Behal 2019). Analytical approaches tries to find grasp solutions through kinematics and dynamics formulations (A. Sahbani 2012). Similar research in grasping such as (Bicchi and Kumar 2000), (A. M. Okamura and Cutkosky 2000), (A. M. Okamura and Cutkosky 2000), (Park and Starr 1992) and (Howard and Kumar 1996) generate the grasping criteria using robot and or object models. The grasp in equilibrium should be force-closure, stable, dexterous. There are some difficulties with analytical approach as in the real world scenario it is difficult to model a task, and it is too ideal to assume readily available physical and geometrical model (A. Sahbani 2012) . Despite the efficiency in simulation environment it is difficult to perform efficient grasp in a real world scenario based on the the classical metrics mentioned earlier (Diankov 2010),(J.Weisz and Allen 2012).

With Data-Driven methods on the other hand, the grasps are based on experience and knowledge base. As such these methods depend on human experience, accumulated knowledge of objects and acquired data (Jabalameli, Ettehadi and Behal 2019). In research such as (J. Bohg and Kragic 2014a) classified Data-Driven approaches depending on the encountered object, where object were classified as one of the three types unknown, known or familiar to the method. In case of unknown objects the first step involves modelling object shape with primitives such as boxes, cylinders and cones and define grasping strategy based on that. Ultimately, match the 3D mesh of the object in obtained data with their grasp database during this online phase. Some research based on probabilistic framework (R. Detry and Piater 2009) estimate the pose of known object in an unknown scene. Whereas others used empirically calculated robot hand kinematics and planning control loop for grasp included a human operator (Ciocarlie and Allenc 2009). Other methods employed encountered object 2D and/or 3D features to extract features such as texture or shape etc which can be used as a similarity parameter (J. Bohg and Kragic 2014b). Whereas, in (A. Saxena and Ng 2008) a logistic regression model based on the labeled data sets. This generated model applied on the 2D image .

For any object manipulation task selecting a stable grasp decides the success or failure. (A. M. Okamura and Cutkosky 2000) defined a stable grasp as a grasp which can perform force closure on the object. In a pick and place type manipulation the grasp needs to be disturbance resistant, meaning the contact forces are enough to resist any possible motion of the object (A. M. Okamura and Cutkosky 2000). For a successful planar grasps such as in (Nguyen 1986) force closure can be a determining condition. For this type of grasps the applied forces are along the plane and the only input is in the form of shape of the object. Also the convex sum including the three contacts describe contact between object and robot fingertips.

Definition 1: A wrench convex represents the range of force directions that can be exerted on the object.

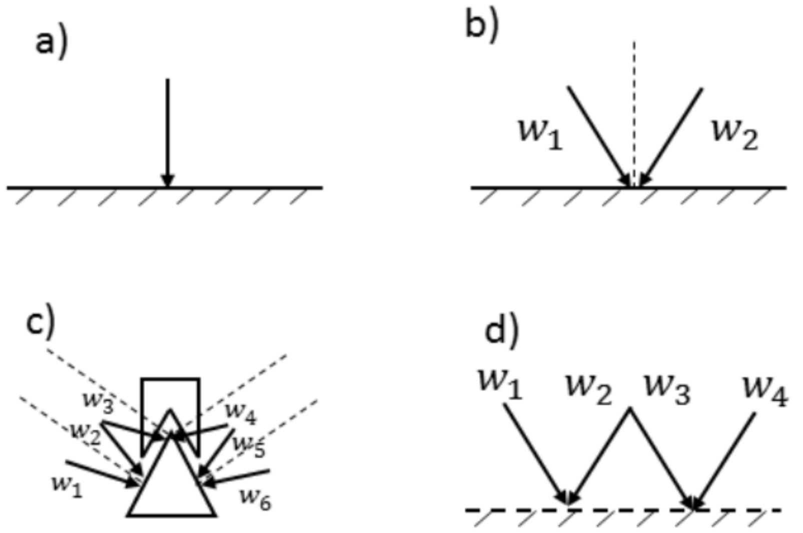


Figure 2.8. *Planar contacts examples: a)Point contact without friction b) Friction Point contact c) Contact with Soft finger d) Contact on edge*
Source: (Jabalameli, Ettehadi and Behal 2019)

The primitive contacts and their wrench convexes in 2D in figure 2.8. The two wrenches forming the angular sector illustrates the Wrench convexes. The finger must apply force in normal direction in a frictionless contact. However, force direction into the wrench convex can be used with friction contact.

There are two theorem based on (Nguyen 1986) which deal with the set of planar wrenches.
Theorem 1: (Nguyen I) A set of planar wrenches W can generate force in any direction if

and as it were in the event that there exists a set of three wrenches (w_1; w_2 ; w_3) whose respective force directions f_1 ; f_2 ; f_3 satisfy:

1. two of the three directions f_1 ; f_2 ; f_3 are independent.
2. a strictly positive combinations of the three directions are zero i.e.

$$\sum_{i=1}^{i=3} \alpha_i f_i = 0$$

Theorem 2: (Nguyen II) A set of planar forces W can generate clockwise and counter-clockwise torques if and only if there exists a set of four forces (w_1; w_2 ; w_3; w_4) such that:

1. three of four forces have lines of action that do not intersect at a common point or at infinity.
2. let p_{12} (resp. p_{34}) be the points where the lines of action of w_1 and w_2 (resp. w_3 and w_4) intersect. There exist positive values of α_i such that

$$p_{34} - p_{12} = \pm(\alpha_1 f_1 + \alpha_2 f_2) = \mp(\alpha_3 f_3 + \alpha_4 f_4)$$

During force-direction closure it is checked if the friction cones formed by the contact forces span every directions in the plane. Whereas during Torque closure tests it is checked if the sum of all applied forces generates pure torques. Also from the above Theorem I and II, among the four wrenches at-least three independent wrenches is a necessary condition for a force closure grasp in a plane. Assuming frictional contact, we can assume two wrenches for each point contact. Thus, for a planar force closure grasp a minimum of two contacts with friction is required. As stated by (Park and Starr 1992) and (Nguyen 1986), conditions for creating a planar force closure grasp is shown in figure 2.9:

- **Two opposing fingers**: For grasp with two contact points, p_1 and p_2. With friction between the contacts a force closure is possible if the segment made by $p_1 - p_2$ points out of and into two friction cones located at p_1 and p_2 . Mathematically speaking, let ϕ_1 and ϕ_2 be the angular sectors associated with the friction cones at e_1 and e_2 , then satisfying $arg(p_1 - p_2) \in \pm(\phi_1 \cap -\phi_2)$ is the necessary and sufficient condition for successful grasp with two point contacts with friction.
- **Triangular grasp**: A three point contacts grasp with friction, located at p_1;p_2 and p_3 is force closure, if and only if there is a point, p_f (force focus point) such that: i) For friction cone at the ith contact is such that at p_i , the segment p_f - p_i points out of friction cone. ii) With k_i as the associated unit vector pointing out of the edge for segment $p_f - p_i$. A a strictly positive combinations of the three directions is requried to be zero as

$$\sum_{i=1}^{i=3} \alpha_i f_i = 0$$

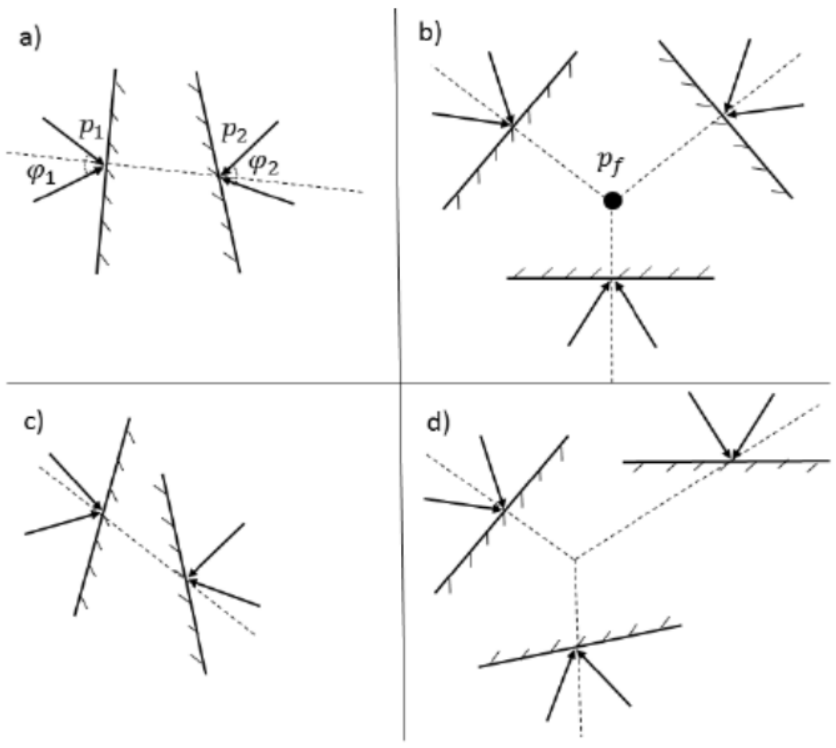


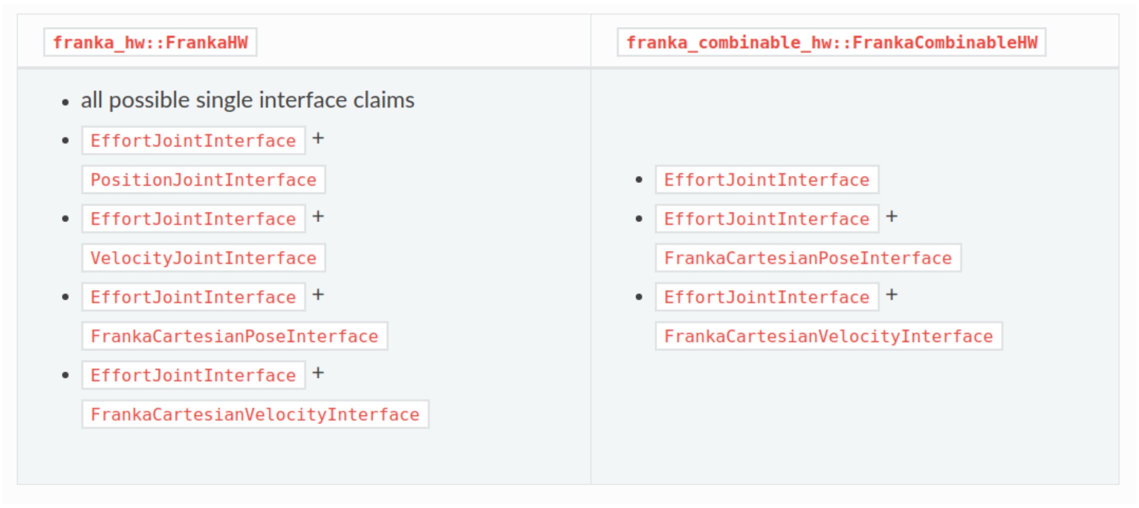
Figure 2.9. *Geometric interpretation of Force closure using two finger gripper (in opposition) and triangular type end effector. (a) Successful force closures grasps for two-finger (b) Successful force closures grasps for triangular end-effector, while (c) impossible force closure grasps for two-finger and (d) impossible force closure grasps for triangular end-effector.*

Source: (Jabalameli, Ettehadi and Behal 2019)

2.3 Robot Control

In this thesis Panda robot from Franka Emika (GmbH 2017a) robot was used. The **Franka_hw** package which is based on **libfranka** API. The **libfranka** API is the hardware abstraction of the robot and is utilised by the ROS control framework (GmbH 2017b). Only some specific claimable combinations of commanding interfaces are allowed by the controller interface (GmbH 2017b) for the robot, since it does not make sense for e.g. command joint positions and Cartesian poses simultaneously. The possible claims to command interfaces are shown in figure 2.10

In this thesis combination of **EffortJointInterface** and **Franka CartesianVelocityInterface** was choosen to create a controller having **VelocityControl** and **impedanceControl**. In this chapter the author analyses the controllers present and provides a overview for choosing the above mentioned controller for this thesis. In the next sections the author provides a brief overview of the different control mechanism available and makes a comparison between the methods.



`franka_hw::FrankaHW`	`franka_combinable_hw::FrankaCombinableHW`
• all possible single interface claims • `EffortJointInterface` + `PositionJointInterface` • `EffortJointInterface` + `VelocityJointInterface` • `EffortJointInterface` + `FrankaCartesianPoseInterface` • `EffortJointInterface` + `FrankaCartesianVelocityInterface`	• `EffortJointInterface` • `EffortJointInterface` + `FrankaCartesianPoseInterface` • `EffortJointInterface` + `FrankaCartesianVelocityInterface`

Figure 2.10. *Combination of command interface*
Source: `https://frankaemika.github.io/docs/franka_ros.html`

2.3.1 Position/Force control algorithms

Stiffness control by only position feedback or by force feedback corrections is a method involving the relation between position and applied force (Zeng and Hemami 1997). The stiffness of the joints are the deciding factor in calculating stiffness of end-effector. Desired stiffness of the end-effector can be acheived by adjusting the stiffness of the servos in the robot joints, thus enabling us to follow a desired trajectory as well as apply a desired force.

Exact position control necessitates that the controller end-effector servos are capable to reject all the aggravations, for example, dynamic powers of the robot itself, outside powers, etc .that may follow up on the end-effector with the high stiffness of the controllers. To fulfill this need of disturbance elimination one can use highly stiff servo controllers controlling the joints of the manipulator. The feedback gain is the parameter that decides on the stiffness of the servo. On the other hand for force control, the system stiffness is made minimum as allowed by the system. Thus for force control it can be concluded that the lower servo stiffness is used.

In end so as to actualize stiffness control, the stiffness ought to rely upon the specific task, as such one should attempt to control the stiffness of end-effector in various directions to accomplish the ideal motion control. There can be two ways to adjust the stiffness of manipulator end-effector. In the first control gains of end-effector servos is adjusted whereas the second way could be to adjust/change the stiffness of joint servos. Accordingly the powers applied on the environment by the end-effector is controlled by the stiffness of the end-effector. The underlining principle of active stiffness control is shown in figure 2.11.

In the figure 2.11 the terms are defined as follows. $\mathbf{J}$ - Jacobian matrix of the robot, $\mathbf{X_D}$ - task space desired position vector, $\mathbf{X}$ and $\dot{\mathbf{X}}$ - task space position and velocity vectors, $\boldsymbol{\Delta}\mathbf{X}$ - error in position vector, $\boldsymbol{\Delta\theta}$ - the joint angle and displacement vector, τ_p - stiffness

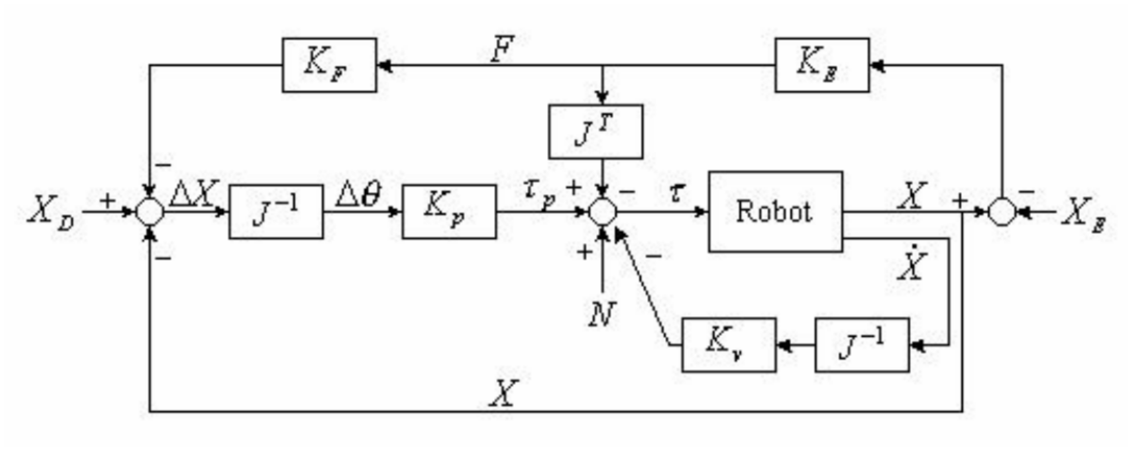


Figure 2.11. *Active stiffness control loop*
Source: (Dede 2003)

control joint command input vector, $\mathbf{N}$ - a vector of nonlinear feed-forward compensation for Coriolis and centrifugal and gravity forces, $\boldsymbol{\tau}$ - vector of gross joint torque/force input, $\mathbf{X_E}$ - the contacted environment position vector, $\boldsymbol{K_E}$ - the total stiffness of the environment and sensors, $\mathbf{F}$ - calculated world space contact force (or torque) vector. $\boldsymbol{K_p}$ and $\boldsymbol{K_v}$ - the diagonal matrices control gains, $\boldsymbol{K_F}$ - position command modifying compliance matrix

The fundamental system comprises of a task space containing the robot, the control system comprises of nonlinear compensation and velocity feedback for linearizing the robot dynamic system. A mix of proportional feedback of force and position is controlled by stiffness control loop which characterizes joint torque $\boldsymbol{\tau_p}$. $\boldsymbol{\tau_p}$ is thus defined by the following equation

$$\boldsymbol{\tau_p} = \boldsymbol{K_p \Delta\theta} \tag{2.4}$$

The unit of $\boldsymbol{K_F}$ in figure 2.11 is displacement/force, which is stiffness. Thus manipulator mechanical stiffness can be controlled by tuning the matrix $\boldsymbol{K_F}$. With changing environmental parameters the matrix $\boldsymbol{K_F}$ should also be updated to exert the same force on different environments.

2.3.2 Impedance control

Impedance control is another well known robot control technique which can be of different forms relying upon the measured signals, for example, velocity, position and force and their different mix. In basic impedance control a gain matrix is used to multiply with the forces sensed, this gain matrix is determined by the mechanical impedance present in the system. The blend of this offers the changes to be made for the position and velocity (Zeng and Hemami 1997).

The target of impedance control is to build up an ideal dynamical connection between the end-effector applied force and position. The mechanical impedance $\boldsymbol{Z_m}$ is the con-

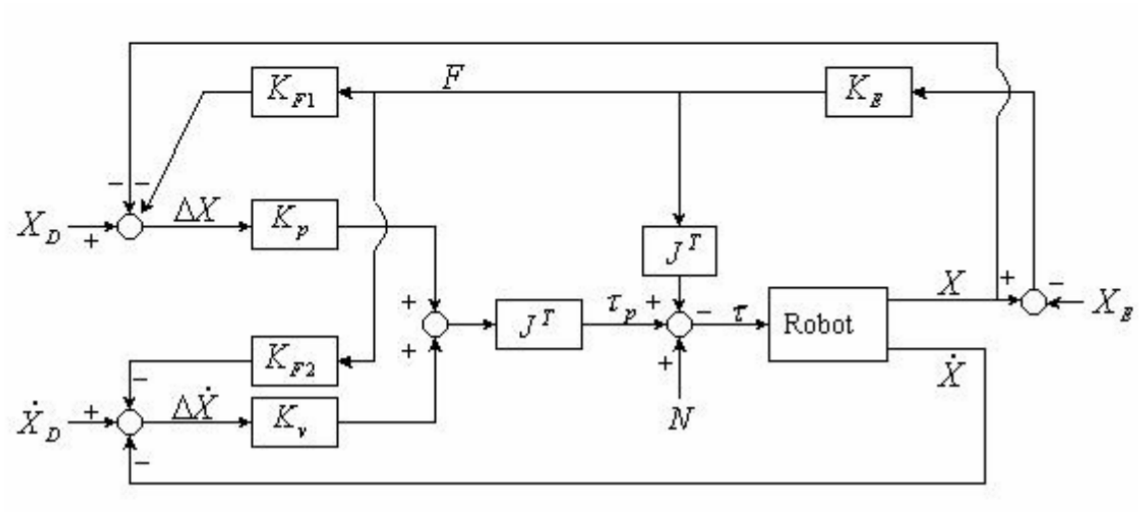


Figure 2.12. *Impedance control loop*
Source: (Dede 2003)

nection between the applied force F and the velocity $\dot{X}$. In the frequency domain, this is represented by

$$F(s) = Z_m(s)\dot{X}(s) \tag{2.5}$$

in terms of position $X(s)$ it is

$$F(s) = Z_m(s)\dot{X}(s) \tag{2.6}$$

Desired impedance can be indicated as,

$$sZ_m(s) = Ms^2 + Ds + K \tag{2.7}$$

The constant matrices M , D and K represents the ideal inertia, damping and stiffness values, respectively. Generally, as in this case, the target impedance is chosen as a direct second-order system to imitate dynamics of mass-spring-damper (Ha et al. 1998). It is assignment of impedance control to ensure the behavior of the controlled system to be as managed by equation (2.7). Impedance control has been implemented in different forms, depending on how the measured signals, i.e. speed, position or force are utilized. figure 2.12 shows the structure of a essential impedance control loop, which decides an suitable value for $Z_m(s)$.

2.3.3 Hybrid Position/Force Control

In non-deterministic environments, force and position information are combined into unified control strategy for movement of the end-effector, this type of control is known as the hybrid/position/force control. This type of control has the intrinsic advantage of analyzing the position and force component freely to take advantage of best available control mechanism among each and then combining the end-result from each at the final stage to compute the joint torques (Fisher and Mutjaba 1991). As such in this sort of control there exists a relationship between the position and the applied force of the end-effector, these relationship varies on case to case premise. A hybrid position/force control scheme is shown in figure 3.5.

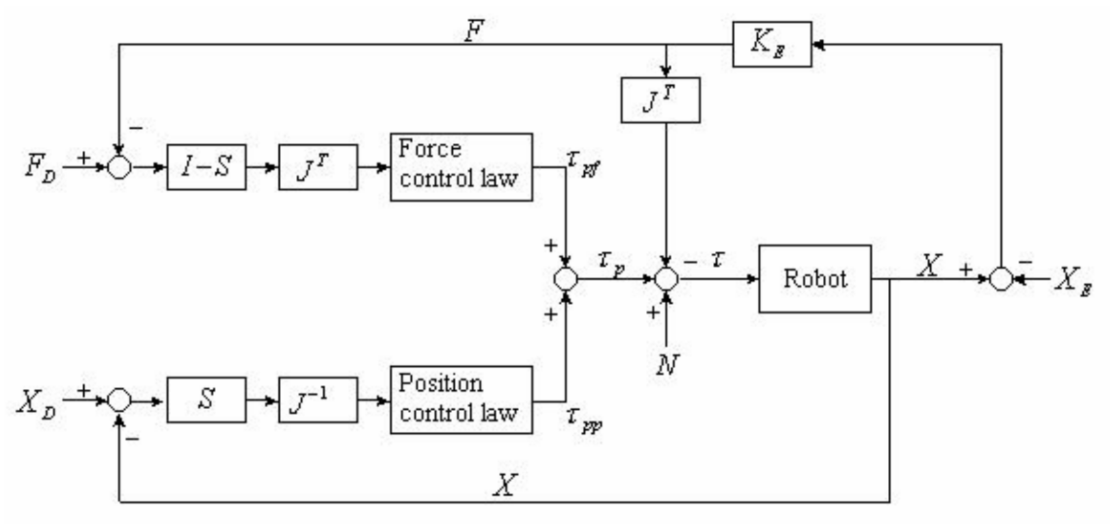


Figure 2.13. *Hybrid control loop*
Source: (Dede 2003)

In figure 2.13, the notations are as follows: $S = diag(s_j)(j = 1...n)$ - compliance selection matrix, n - degree of freedom. The matrix S decides the sub-spaces for which position or force are to be controlled, and s_j - as 1 or 0. When $s_j = 0$, the jth DOF is force controlled, otherwise it is position controlled. The value of S matrix can be constant, can alter with the configuration or can continuously change in time (Fodor and Tevesz 1999).

For each setup, a generalized surface can be characterized in a constraint space with position restrictions along the normals to the surface and tangential force limitations, which suggests, the end-effector can not move along the normals into the surface and can not apply forces along the tangents of the surface. Utilizing this information, S matrix is formed.

The command torque is defined by

$$\tau_p = \tau_p p + \tau_p f \tag{2.8}$$

$\tau_p f$ and $\tau_p p$ and are the command torques acting in force and position sub-spaces, respectively.

In this way, force control and position control are decoupled. Hence designing a law for independent control with distinctive control execution prerequisites for wanted position and force trajectory tracking is conceivable with this sort of control. Normally, the position control law in figure 2.13 consists of a PD action, and the force control law consists of a PI action. This is because for the position control, a faster response is more desirable, and for the force control a smaller error is more preferable.

2.3.4 Comparisons among Different Control Schemes

In this thesis a robot controller which can execute motion through a free space to reach the target point and is able to compensate for the collision during motion was required to perform a pick operation from the robot ready state to the grasping state where the gripper collided with the surface. Here the author presents a comparative study between different control mechanisms as discussed above. At first we start with position control, which is suitable where the robot is free to move in the task space and the motion is not constrained by external factors like human etc. On the other hand Force control needs feedback data of contact force, thus the robot controller ought to have the right sensor data to read the contact force. Also, this requirement of Force control makes this mechanism suitable for areas where the robot is required to be in contact with environment to decide on the next steps. During free motion since there is no feedback component which can provide feedback on the motion being executed, the robot can go out of control. Hybrid position/force control separates assignment space into two subspaces, called the position-controlled and force-controlled subspaces. This situation complicates the control mechanism, in a few cases amid control, the control law should be switched to adjust with the necessities within the task-space, which may lead to unsteady reactions. On the other hand, impedance control may be a bound together control technique appropriate for all control stages, including free movement, obliged motion and the transients, without the requirement to switch control modes. Hence, the author chose impedance control for controlling the manipulator.

3 METHODOLOGIES

In this chapter, the methodologies relevant to the scope of the study in this thesis is presented. The chapter starts with explaining how the author created the image data-set followed by the image segmentation techniques being used. Then in the last section the desired the robot controller characteristics are discussed. A flow diagram figure 4.5 shows the interaction of the mentioned techniques.

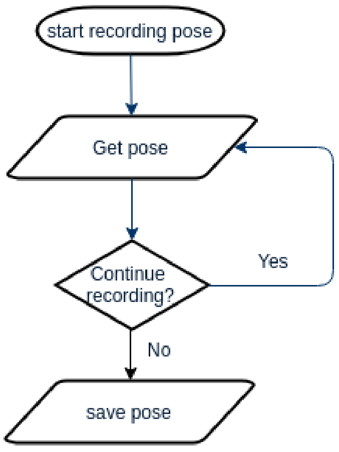
3.1 Creating Training Data

In conventional machine learning problem, we train our model based on objects which have readily available training data-set such as images of a cat, dog etc. A training data set for industrial purpose is not available in most cases. There are many components and they differ in size, color, weight etc. In this thesis parts of an engine were taken as the test objects. The thesis work started with creating an image data set of the engine parts. Utilising this image data-set we can train a neural network to build a detector to classify different engine components.

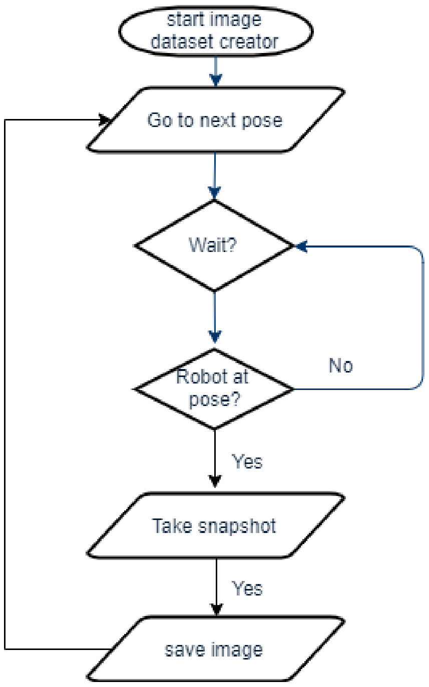
The work started with taking pictures manually and then training few basic neural networks. In image recognition the larger the data-set, the better is the learned model. To generate an image data-set an automated method for collection of images was required. This led in creation of a system in which the robot end-effector moves through different positions looking at the object from different angles towards the work-space.

The system uses the capability of Franka robot to record the joint values and save them to a local file. A single object whose image database is to be created is placed in the work-space. The process starts with a human instructor moving the end-effector through space and the system keeping a record of the joint values as shown by the flowchart in figure 3.1a. This results in the robot going through different positions and capturing the image of the object from different angles.

Next the system reads from the stored joint values and moves the robot through space as shown in figure 3.1b. On reaching the desired location robot sends a signal which makes the camera to capture image of work-space. The system then stores image in a local folder. In the current system each location creates one RGB, one Edged and one Depth image. This was done to keep provision for using the other image format for creating a better classifier. The author proceeded with using RGB image only in this thesis.



(a) Pose Generation



(b) Imageset Generation

Figure 3.1. *Training Data Creation*

As the joint values are recorded and saved the same values can be utilized to create image data-set for other objects. As such the author recorded the joint values once and used them to generate image database for all the test objects. This system allows to easily create image data-set for unknown objects and retrain the learning model

3.2 Object Detection

The background studies in image segmentation forms the basis of image segmentation technique used in this thesis. The issue with using existing technique was creating a image database which is labelled as used in COCO challenges (in 2017 and 2018). The COCO data-set for object segmentation is composed of more than 200k images with over 500k object instance segmented. Thus, finding a classifier which can use single object image to learn and then localise the object in work-space as is present in COCO images as shown in figure 3.2 where the areas where the persons exists are zoned out.

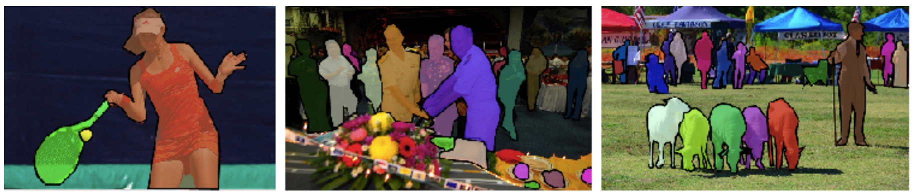


Figure 3.2. *Example COCO images with object instance segmented*
Source: `https://cocodataset.org/workshop/coco-mapillary-iccv-2019.html`

In this thesis the process of image segmentation is a two step process. First, the RoI is found from the work-space and then the identification is done using the classifier trained with the images generated in section 3.1. figure 3.3 shows the flowchart for identifying object in work-space.

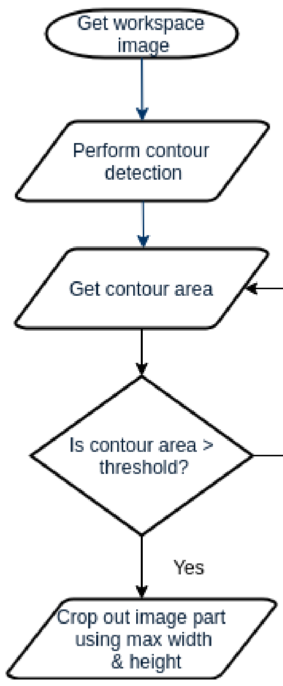
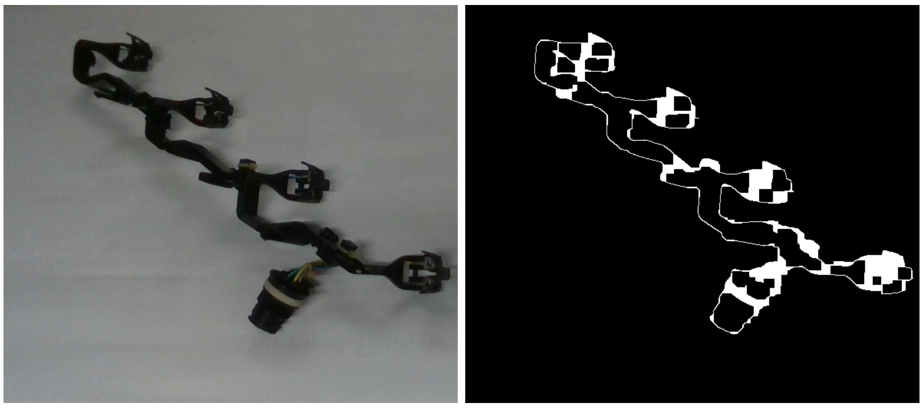

Get workspace image
Perform contour detection
Get contour area
Is contour area > threshold?
No
Yes
Crop out image part using max width & height

Figure 3.3. *Image Segmentation*

3.2.1 Finding Region of Interests (RoI)

Searching for RoI is based on using edge detection technique(Canny edge detector). After capturing image of work-space a contour around the objects are formed. This is followed by calculating the area enclosed by the contour as shwon in figure 3.4. This contour area is compared with a threshold value which indicates presence of object at that location. The threshold value is empirically calculated.



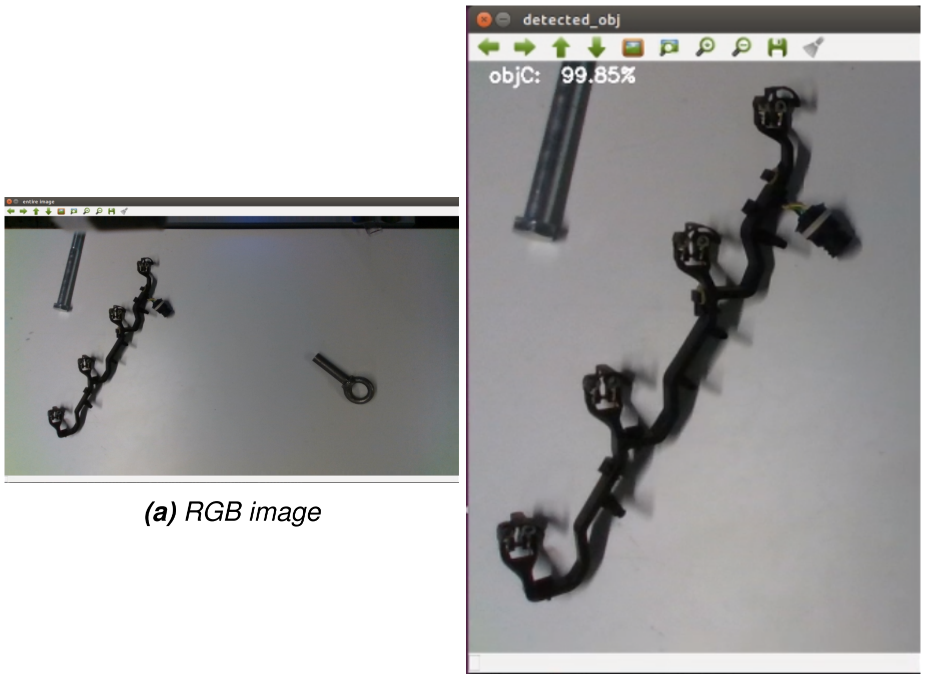
(a) *RGB image* ***(b)*** *Edged image*

Figure 3.4. *Threshold detection for ROI*

As seen in figure 3.5b the contour contains broken edges which can be connected by using a higher **canny threshold** value. Thus this method of contour area calculation from work-space image can be an indication of presence of some object, thus providing an RoI. The next step is to go through all the RoI's and identify them using the classifier developed which is explained next.

3.2.2 Object identification

As explained in previous section the RoI area provides an indication of object presence. if the $contourarea > threshold$, then these area is cropped having dimension equal to $maximumlength \times maximumwidth$ of object with some padding. Thus using this technique it is possible to crop out a part of image which contains only one object as shown below in figure 3.5



(a) *RGB image*

(b) *Edged image*

Figure 3.5. *Crop RoI*

After finding the RoI region and cropping out the image part the cropped image is used by the generated network model to identify the object. In this thesis a simple CNN was used as a classifier. The classifier uses The LeNet architecture which was first introduced by (Y. LeCun et al. 1998). As the name of the paper suggests, the authors implementation of LeNet was used primarily for Optical character recognition (OCR) and character recognition in documents.

3.3 Motion planning and controllers

In ROS ecosystem there exists ready made packages such as **MoveIt** which implement motion planning and can control the robot. This thesis started with utilizing ready made packages and sending goal coordinates to the end-effector. Although the ready-made controllers can be easily integrated and provide better visualisation, those packages were unable to compensate for the collision during motion or collision with work-space floor. The ready-made packages require accurate measurement so collision can be avoided. Since this thesis utilised only one depth camera as the sensor for checking the environment, there were inaccuracies in measurement. These issues required development of custom motion planner suited for this thesis.

In this thesis the custom motion planner developed takes the start and goal point as input in Cartesian space. It calculates a linear motion between the two points and calculates the way-points on the linear path. The way-points are series of small steps to reach goal points avoiding any violation in the robot limits. Other parameters such as the velocity profile is calculated based on the industry standard trapezium profile, also the acceleration period is mirrored to create the deceleration period. The middle section during robot motion is scaled such that the robot moves at a constant speed. The velocity profile is modelled around two main input parameters as shown in figure 3.6:

- The desired target speed of end-effector travel.
- The acceleration and deceleration of the end-effector.

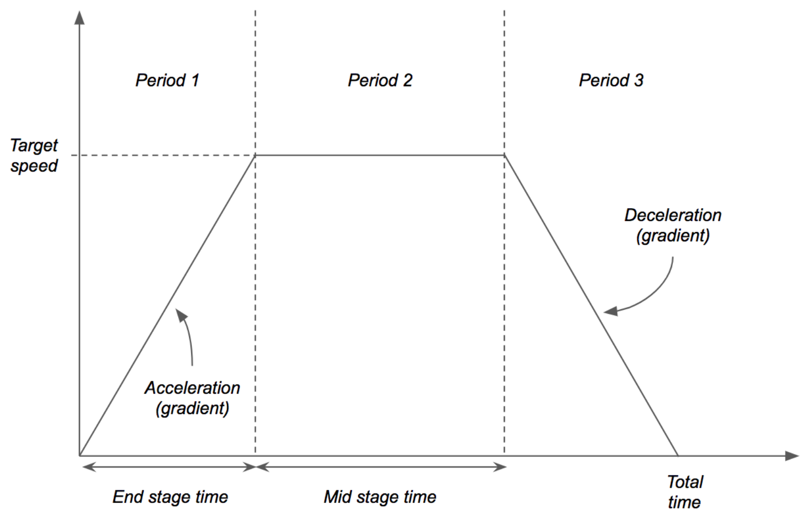


Figure 3.6. *Trapezoidal velocity profile*

The custom controller developed for this thesis utilised command from a combination impedance control method as explained in section 2.3.2 in combination with the velocity control explained above. This combination of controllers allowed for compensating with collision with surface, which was not possible with ready-made controllers.

4 IMPLEMENTATION

This chapter forms the core of the actions performed in achieving autonomous robot grasp. The author starts with explaining the devices used during, followed by the core architecture and concluding with software design implemented for this thesis.

4.1 Devices used

- **Robot:** Panda from Franka Emika is 7-DOF manipulator. It has torque sensors in all the seven joints and there exists extensive and useful libraries of controllers and hardware interfaces to control the robot.
- **Camera:** Intel RealSense D435 is a USB-powered depth camera and consists of a pair of depth sensors, RGB sensor, and infrared projector. It has software libraries implemented in *C++* and *python*.
- **Computer:** The setup required two separate computers one for implementing the vision part and the other for robot commands. Panda arm requires real-time kernel installation in the workstation PC to send real-time control values at 1 kHz. On the other hand vision requires a computer with GPU. A two system configuration keeping vision and robot control side separate was implemented to eliminate the limitation of GPU powered device not being able to use real time kernel.

4.2 Architecture

The architecture used in this thesis comprises of the work-space, the manipulator, a depth camera and two computers. A pictorial representation of system architecture is shown in figure 4.1:

- **FCI:** The Franka Control Interface (FCI) permits a quick and direct low-level bidirectional association to Panda. It gives the current status of the robot and empowers its direct control via an outside workstation PC associated by means of Ethernet. By utilizing open source *C++* interface *libfranka* real-time control values are send at 1 kHz.
- **PANDA:** The robot used in this thesis is a 7-DOF manipulator. There is separate control for the arm(robot links) and hand(gripper). It is possible to send control commands at 1 kHz. From the franka control interface documentation, the control commands are:

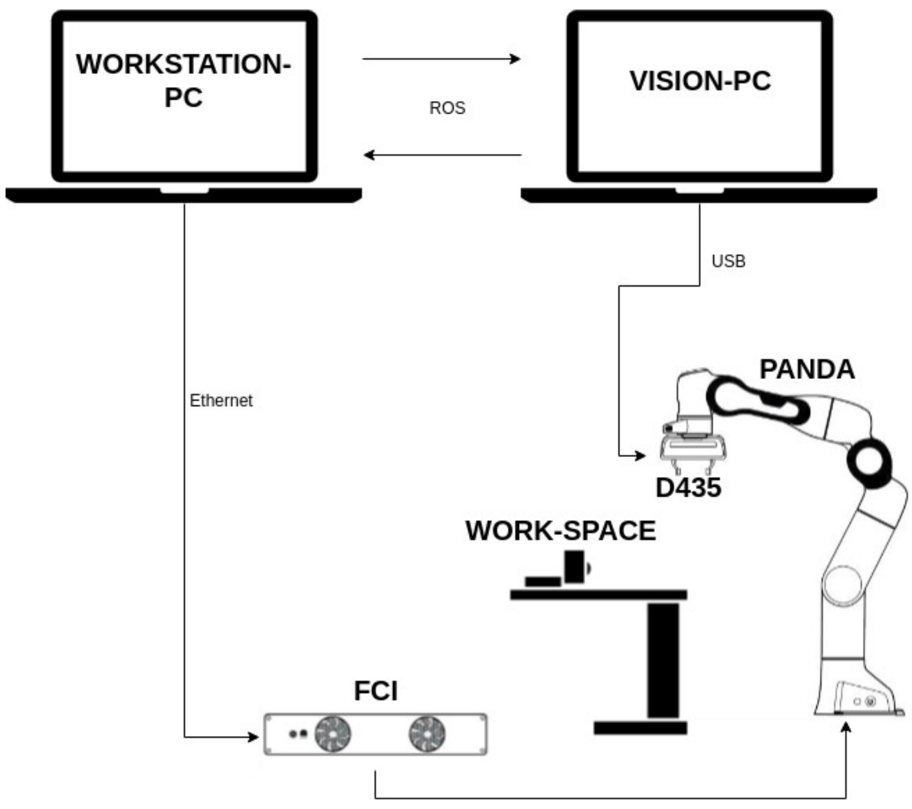


Figure 4.1. *System Architecture*

- Gravity and friction compensated joint level torque commands.
- Joint position or velocity commands.
- Cartesian pose or velocity commands.

At the same time, measurement data can be received at 1 kHz. Measurement data received are:

- Measured joint data, such as the position, velocity and link side torque sensor signals.
- Estimation of externally applied torques and wrenches.
- Various collision and contact information.

In addition Forward kinematics of all robot joints, Jacobian matrix of all robot joints and Dynamics: inertia matrix, Coriolis and centrifugal vector and gravity vector can be received from the robot via the FCI.

- **WORKSTATION-PC:** Controller PC connected with FCI via ethernet. The robot commands are send using this PC. Linux with PREEMPT_RT patched kernel is the operating system for this computer. Since the robot sends data at 1 kHz frequency, it is important that the workstation PC is configured to minimize latencies. This PC also communicates with **VISION-PC** via ROS. The **VISION-PC** and **WORKSTATION-PC** are in the same Wifi network and proper configuration is made so that they can communicate over ROS.

- **D435:** Intel realsense depth camera is mounted on the grippers to observe the work-space. As mentioned in wikipedia pages (Wikipedia 2018) Intel RealSenseDepth Camera D435 is perfect for capturing stereo depth in a assortment of applications that offer assistance to see the world in 3D. The camera incorporates the Intel RealSense VisionProcessor D4 highlighting high depth resolution - up to 1280x720 at 30 frames persecond (fps), long-range capabilities, global shutter technology and a wide field of view. With the global shutter technology and a wide field of view (91.2° x 65.5° x 100.6°),the Intel RealSense depth Camera D435 offers exact depth perception when the object is moving or the gadget is in movement, and it covers more field of view, minimising blind spots. It is designed for simple setup with USB 3.0 and is in a sleek form factor for portability.
- **VISION-PC:** This is responsible for capturing the work-space image. It is connected with the D435 camera. This PC is equipped with latest 8th Gen Intel® Core^{TM} i7 processor and overclockable NVIDIA® GeForce® GTX 1060 graphics. This PC is suitable for image processing and faster computation to generate grasping point on the objects in the work-space.

4.3 Software Design

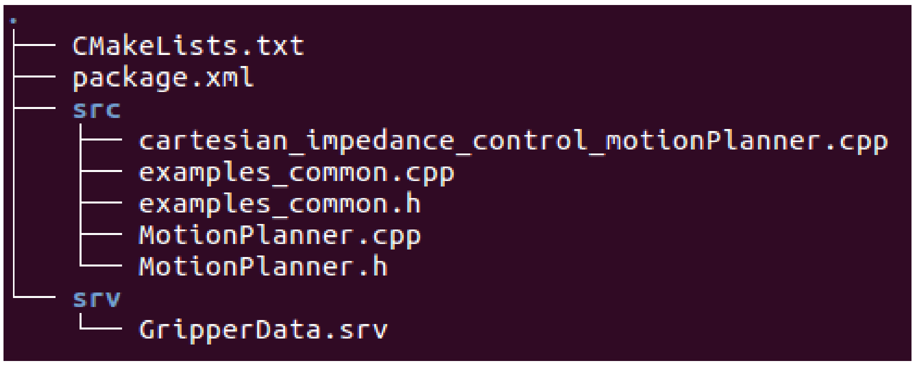
This sections explains the software packages developed during this thesis. Some of the important algorithms developed are explained with *pseudo code*. Broadly, there are two packages which are developed in this thesis:

- *panda_c++* - this package is responsible for robot control. It starts up robot motion and takes goal commands from the vision system.
- *panda_python* - this package is responsible vision part and the calculations associated with grasping.

Both these packages and there working are explained below which also serves in explaining the work of this thesis.

4.3.1 Robot Control

The package layout is as shown in figure 4.2. This package contains the necessary codes related to controlling the robot. This controller commands are via the workstation PC to the robot. During the thesis it was essential that the there exists a server which listens to grasp commands. Also after performing the current task the robot goes back to its ready position and waits for next grasp command. During robot motion the robot does not take any more commands. The components of this package are explained in the following sections.



```
.
├── CMakeLists.txt
├── package.xml
├── src
│   ├── cartesian_impedance_control_motionPlanner.cpp
│   ├── examples_common.cpp
│   ├── examples_common.h
│   ├── MotionPlanner.cpp
│   └── MotionPlanner.h
└── srv
    └── GripperData.srv
```

Figure 4.2. *Robot control folder structure*

Grasp Data

This is the data structure which was defined to communicate with the ROS server. The data structure is defined as follows:

```
# A name for this grasp
string id

# The position of the end-effector for the grasp.  This is the pose of
# the "parent_link" of the end-effector, not actually the pose of any
# link in the end-effector.  Typically this would be the pose of the
# most distal wrist link before the hand (end-effector) links began.
geometry\_msgs/PoseStamped grasp\_pose

# The position of the end-effector for the release.  This is the pose of
#the "parent_link" of the end-effector, not actually the pose of any
#link in the end-effector.  Typically this would be the pose of the
#most distal wrist link before the hand (end-effector) links began.
#geometry_msgs/PoseStamped release_pose
---
# send response of grasp result
bool grasp\_result
```

Each item is explained in the message comments. The **request** from vision system sends out *geometry_msgsPoseStamped* as grasp and release pose for the robot. The *geometry_msgsPoseStamped* **grasp_pose** consists of Cartesian position (x, y, z) and orientation in quaternion (w, x, y, z).

Robot Motion

The robot motion can be divided into two parts. The first part is the Motion planner which takes care of generating a plan to reach from current position to goal position. The plan generated should be within the velocity and acceleration limit of the robot. The second part is concerned with execution and it consists of robot controllers and executors. It takes the motion plan trajectory points and commands the robot to move through a set of steps to reach the goal point. The two parts are explained below:

- **Motion Planner**: The *trajectory* component is created after receiving way-points to goal. Motion between the robot current position and the grasping position is divided into two parts. The first motion plan is made in a way that the robot positions itself over the object grasping location. The second motion is then a vertical straight motion to grasp object. The user can define additional way-points. Adding additional way-points reduces the chances of robot gripper disturbing the object position by touching it during its motion.

  The velocity profile applied is the industry standard trapezium profile as shown in figure 3.6. The acceleration period is mirrored to create the deceleration period. The middle section is scaled accordingly; keeping the end-effector at a constant speed. Algorithm 1 represents the algorithm for motion planner. This method takes the path including the way-points defined by the user.

Algorithm 1 Motion planner algorithm

```
Require: path with waypoints
 1: procedure TRAJECTORY GENERATION                ▷ way-points between start to goal
 2:     $dx = acceleration * dt * *2$ ▷ path is discretised into small, equally sized units
 3:     $discritisedPath =$ path divided into small steps of dx
 4:     $targetSpeed = sqrt(sizeOf(discritisedPath) * acc)$
 5:     $EndStageTime = targetSpeed/acceleration$
 6:     $EndStageDisplacement = EndStageTime * targetSpeed/2$
 7:     $MidStageDisplacement = PathLength - 2 * EndStageDisplacement$
 8:     $MidStageTime = MidStageDisplacement/TargetSpeed$
 9:     $TotalTime = EndStageTime * 2 + MidStageTime$            ▷ refer figure 3.6
10:     $timeList = [0, dt, dt * 2, ..., TotalTime]$
11:     $speedValues = []$
12:     for t in timeList do
13:         if $t <= EndStageTime$ then
14:             $speedValues \leftarrow acc * t$                 ▷ insert acc*t to speedValues
15:         else if $t >= EndStageTime + MidStageTime$ then
16:             $speedValues \leftarrow -acc * t + c$ ▷ insert -acc*t + some constant velocity in
    mid region to speedValues
17:         else if $t > EndStageTime$ then
18:             $speedValues \leftarrow targetSpeed$       ▷ insert targetSpeed to speedValues
19:         end if
20:     end for
21:     $trajectory = []$
22:     for v in speedValues do
23:         $samples = speed_value * dt/dx$            ▷ number of samples in the path list
    corresponding to the speed value
24:         $trajectory \leftarrow samples$              ▷ add the sample set to trajectory
25:     end for
26:     return $trajectory$                        ▷ The trajectory between two points
27: end procedure
```

After receiving the goal position. The robot sends its current position and the goal position with one way-point as default as explained earlier, to the planner and it receives a list of trajectory points with velocity associated with it. This list of points is then send to the robot controller to perform actual motion on the robot.

- **Robot motion control**: Franka can implement combinations of commanding interfaces, in this thesis a combination of *EffortJointInterface + FrankaCartesianVelocityInterface*. In this thesis as discussed earlier impedance control with velocity was chosen as the controller to perform a smooth grasp operation while taking care of the collision to the work-space.

  In the velocity control part the controller checks its current location w.r.t the target location defined by the motion planner. After the motion plan is generated the target location is updated from a separate thread. The velocity controller algorithm 2 takes this updated target values and generates the required velocity values to control the robot.

Algorithm 2 Velocity control algorithm

```
Require: robotState
 1: procedure MOTION CONTROL ▷ Control robot motion impedance and velocity control
    ▷ Get state of robot
 2:     $currentX = robotState.X$
 3:     $currentY = robotState.Y$
 4:     $currentZ = robotState.Z$
                    ▷ Get the difference in position from planned State to current state
 5:     $vecX = targetX - currentX$
 6:     $vecY = targetY - currentY$
 7:     $vecZ = targetZ - currentZ$
 8:     $normaliseFactor = sqrt(vecX^2 + vecY^2 + vecZ^2)$
 9:     $velocityX = speedLimit * vecX/normaliseFactor$
10:     $velocityY = speedLimit * vecY/normaliseFactor$
11:     $velocityZ = speedLimit * vecZ/normaliseFactor$
12:     return $velocity$                                  ▷ velocity in cartesian
13: end procedure
```

In impedance control a set of value for damping and stiffness are defined by the user. The joint torque to be exerted is calculated and the command is sent to the robot joints. In joint space we can start with Lagrangian formulation in robot dynamics for robot control:

$$\tau = M(q)\ddot{q} + c(q, \dot{q}) + g(q) + h(q, \dot{q}) + \tau_{\text{ext}} \tag{4.1}$$

where q is the joint angular position, M denotes symmetric and positive-definite inertia matrix, c represents the Coriolis and centrifugal torque, g is the gravitational torque, h includes further torques for e.g. inherent stiffness, friction etc., and τ_{ext} is total environmental external forces. The actuation torque τ on the left side is the input variable to the robot.

One may also provide a proposal for control law, when actuating a robot from current position to goal in the following form:

$$\tau = K(q_{\mathrm{d}} - q) + D(\dot{q}_{\mathrm{d}} - \dot{q}) + \hat{M}(q)\ddot{q}_{\mathrm{d}} + \hat{c}(q, \dot{q}) + \hat{g}(q) + \hat{h}(q, \dot{q}) \tag{4.2}$$

where q_{d} is the desired joint angular position, K and D represent the control parameters, and $\hat{M}$, $\hat{c}$, $\hat{g}$, and $\hat{h}$ denotes the internal model of the corresponding mechanical terms.

Inserting equation 4.2 in equation 4.1 we get,

$$K(q_{\mathrm{d}} - q) + D(\dot{q}_{\mathrm{d}} - \dot{q}) + M(q)(\ddot{q}_{\mathrm{d}} - \ddot{q}) = \tau_{\mathrm{ext}}. \tag{4.3}$$

Eq. 4.3 forms the core of impedance control and is implemented in the algorithm as shown in the psuedo code 3

Algorithm 3 Impedance control algorithm

```
Require: robotModel, jointDamping, jointStiffness
1: procedure IMPEDANCE CONTROL    ▷ Control robot motion impedancecontrol    ▷ Get coriolis of robot
2:     $coriolis = robotModel.coriolis(robotState)$
3:     $jointTorque = []$
4:     for joint in robotJoints do                              ▷ 7 joints in this thesis
5:         $torque = coriolis + stiffness + damping$
6:         $jointTorque \leftarrow torque$              ▷ insert torques values for each joint
7:     end for
8:     return $jointTorque$
9: end procedure
```

The robot controller take the above two control methods as control functions and calls the Franka robot control method

$$robot- > control(impedanceControl, cartesianVelocityControl) \tag{4.4}$$

4.3.2 Vision

In this section the author describes code snippets and their working in this thesis. The vision part is divided into following parts:

- **Objector Detector**: A 4-layer CNN was used to generate a classifier for detecting the objects under consideration. The architecture of CNN is shown in figure 4.3.

  As mentioned in methodologies LeNet architecture was used to construct this classifier. The LeNet architecture is straightforward and small, (in terms of memory footprint), making it perfect for basics of CNNs — it can even run on the CPU (if

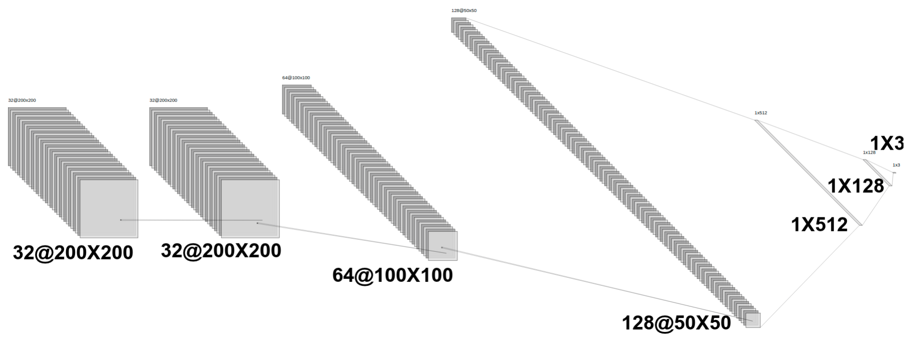


Figure 4.3. *CNN architecture*

the system does not have a suitable GPU). LeNet classifier being very simple in design can be easily upgraded by adding or removing layers from the network. The author tested the results of using different architecture and also tested changing the number of layers in the classifier. The author chose the 4-layer network design due to the following reasons:

- 4 layer architecture provided better classification result w.r.t 3-layer architecture on a data base consisting of RGB images of the objects.
- With increase in the number of layers beyond 4 layers, the training time increased without any actual increase in the accuracy of the classifier.
- Overfitting was an issue associated with increase in number of layers.

In short the 4-layer architecture was chosen in an empirical way after testing with other combination.

- **RealSenseCamera**: This part of the software package is responsible for vision. It includes generating image database, grasp detection etc. In this section the author discusses the image data-set generation algorithm and the reeb graph algorithm which forms the basis of generating grasp pose.

  Algorithm 4 employs openCV methods to find objects in an image and then crop out the RoI. This RoI represents the object in work-space.

  This function takes the image from the realsense camera and then find the contours using *Canny* edge detector. Then the area of the contours is calculated, if the contour area is greater than threshold defined by user it is sensed as an object (RoI). This RoI is then cropped out using the maximum dimension of the object. The cropped image can then be stored to create image data base or can be sent to the classifier for classification.

- **ReebGraph**: Reeb graph can be used for recovering hidden structure which can result in data skeletonization such as finding a core structure for an object. This class is responsible for calculating the grasping point for the object. The following code snippets explains the principles. Reeb graph tries to find a line which follows the shape of the object placed centrally. Algorithm 5 implements this logic by taking the contour image from *RealSenseCamera* and creating a masked image first. A

Algorithm 4 Image Set formation

```
1:  procedure IMAGESETGENRATOR          ▷ Create image Set for training and testing
                                        ▷ OpenCV package was used for image analysis
2:      $blurImage = GaussianBlur(grayImage)$
3:      $edgedImage = Canny(blurImage)$
4:      $edgedImage = dilate(edgedimage)$
5:      $edgedImage = erode(edgedimage)$
6:      $Contours = FindContours(edgedimage)$
7:      for $contourinContours$ do
8:          if $contourArea(contour) > threshold$ then
9:              $ImageList \leftarrow contour$        ▷ add as an image of the object
10:         end if
11:     end for
12: end procedure
```

Algorithm 5 ReebGraphGenerator

```
1:  procedure REEBGRAPHGENERATOR        ▷ Create masked image of the object in
    work-space with its miodpoint
2:      for points in objectImageContour do
3:          if point is inside the contour then
4:              $pixelValue = 1$
5:          else
6:              $pixelValue = 0$
7:          end if
8:      end for
9:      $findMidPointInMaskedImage$
10: end procedure
```

masked image of an object has all points inside the closed contour set as 1 while the others are set to 0. The resulting image can be as seen in figure 4.4.

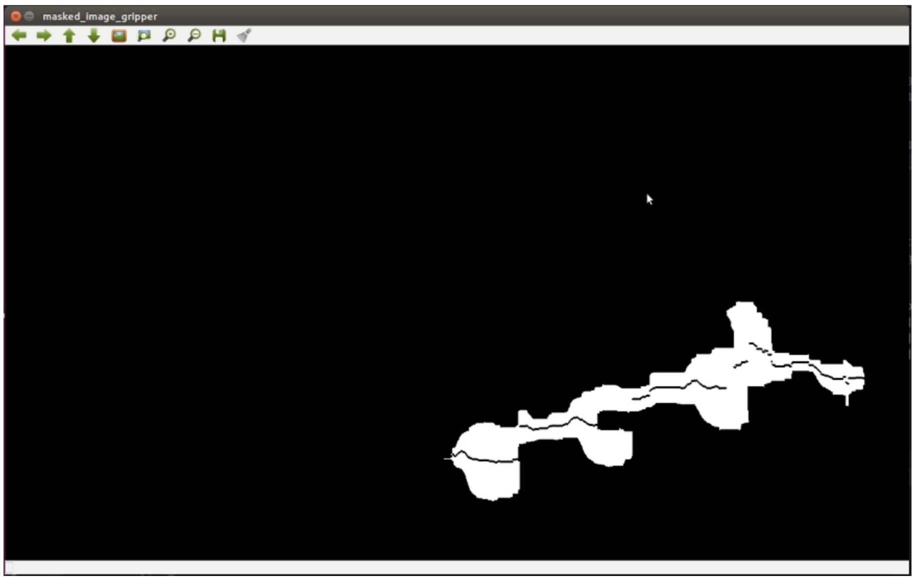


Figure 4.4. *Masked image and midpoint detection*

From the masked image approximate mid-points are located in the masked area. This is checked by calculating the gradient between 1 to 0 or 0 to 1. The gradient check is done both in horizontal and vertical direction. Pixel location where the gradient change occurs is saved. So for 1 to 0 we have one location and for 0 to 1 we get another pixel location. The areas between the change in gradient represent object part and the mid-point is the pixel position which is placed at the centre between the change is gradient 1 to 0 to 0 to 1. There may be areas where the gradient change occur multiple time. These type of areas point that the object shaped is curved in those area. These areas are neglected from grasping position as it is easier to grasp straight areas with a two-fingered gripper.

After finding the midpoints the points are joined by *polyfit* line which gives an approximate of the orientation of the object. Thus the grasping point and orientation component are calculated using reebGraph. This data is then packed into ROS message as mentioned in Grasp Data mentioned above and sent to the robot.

4.4 Work flow

The work flow is divided into two parts. The first part is creation of object image database this is followed by splitting the data-set into training and testing with the image data-set created. The network is then trained with the training data and tested with the test data available. This process is done once and the detector model is saved. The second part starts with using the detector to locate the object as desired by the user and performing the grasping operation.

4.4.1 Image database creation

This work starts with placing the object in the robot work-space and taking images using different robot pose. For this step ROS package **Moveit** is used. The user first collects the *joint_states* which contains the trajectory the robot will go through for capturing images of the object to create image data-set. After collecting the *joint_states*, a simple script is used to send the robot to all the collected *joint_states* and collect sample images. In the collection there are three types of images 1. **RGB image** 2. **Edged image** 3. **Depth image**. In this thesis different combination of images and image type was used, RGB image was considered as the best for this this since with additional edge and depth data the training time increased without any improvement in detection.

4.4.2 Grasp operation

The author presents a communication diagram to explain the communication and states the system goes through during a full grasp operation. In figure 4.5 the flow is as follows:

1. Send robot to ready position above the work-space. Here the entire work-space is

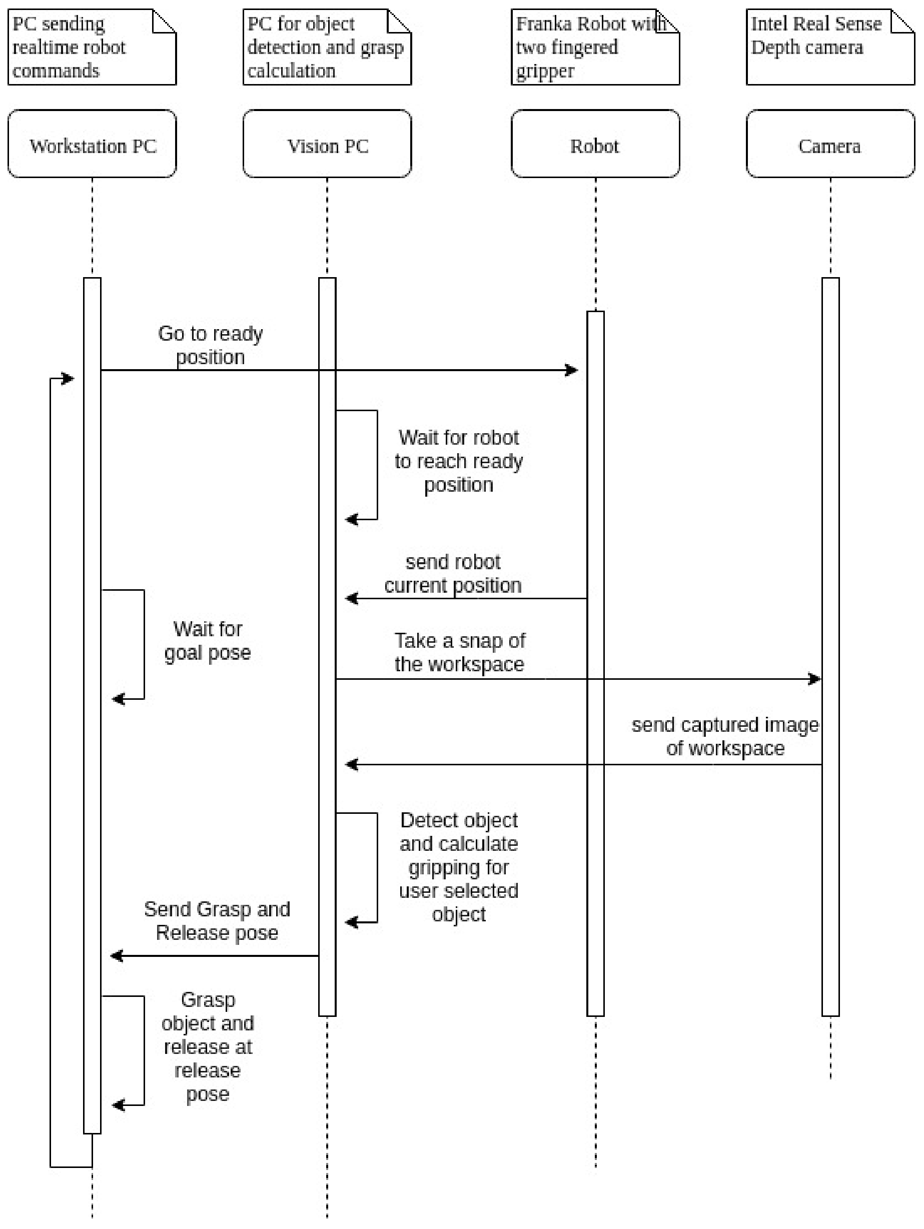

PC sending realtime robot commands
PC for object detection and grasp calculation
Franka Robot with two fingered gripper
Intel Real Sense Depth camera
Workstation PC
Vision PC
Robot
Camera
Go to ready position
Wait for robot to reach ready position
send robot current position
Wait for goal pose
Take a snap of the workspace
send captured image of workspace
Detect object and calculate gripping for user selected object
Send Grasp and Release pose
Grasp object and release at release pose

Figure 4.5. *Flow Diagram*

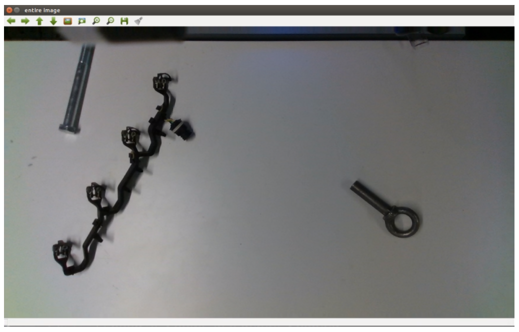
visible from the camera.

2. The camera node waits till the current robot position is received from the robot.
3. After receiving current robot position camera takes an image of work-space, it then asks the user to indicate which object to pick.
4. In the vision PC side a object detector searches for the object as selected by the user and if it is present further calculation is made to grasp the object. First a masked image with only the object under consideration is made as shown in figure 4.4. Mid-point calculation is made and then the grasping position is selected.
5. The robot waits for a goal to go. On receiving grasping pose and release pose from vision PC a motion plan is made to reach grasp pose.
6. The motion plan places the robot directly above the grasping point, from there the

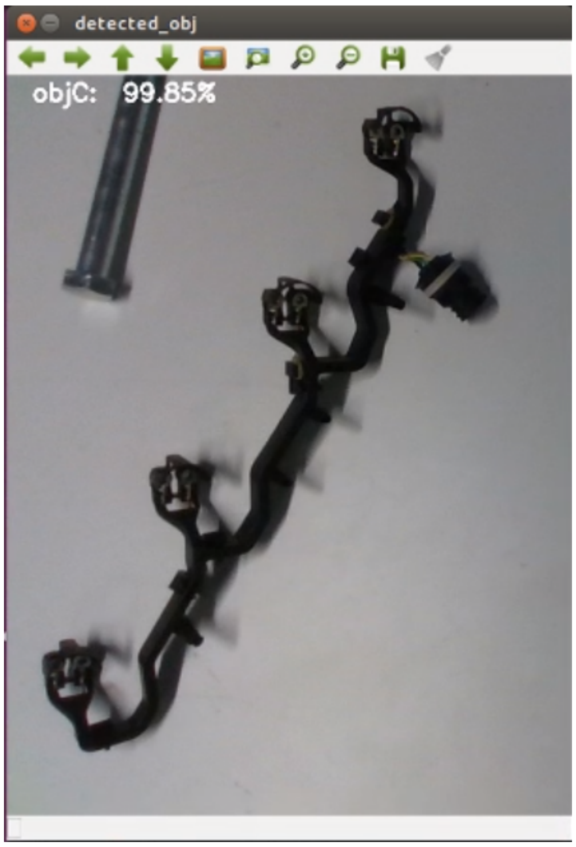
robot proceeds vertically down to grasp the object.

7. With object grasped next motion plan is made to reach the release pose and after releasing the object the robot returns to ready position and is ready to take next grasp command.

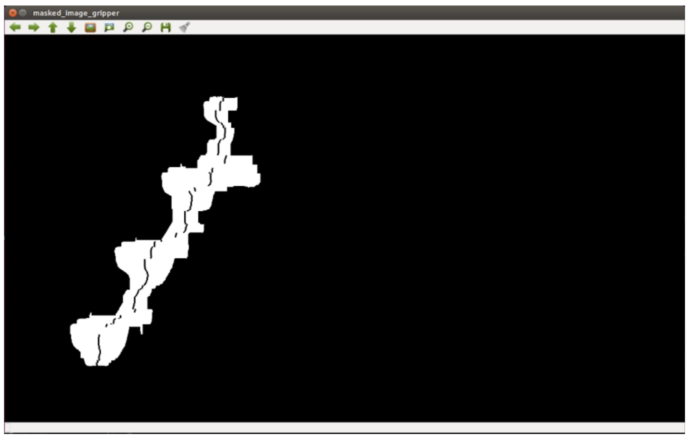
In figure 4.6 pictorial representation the work flow for object detection is shown, with the gripping point shown in figure 4.7.



(a) *Scene*



(b) *object detection*



(c) *Masked image*

Figure 4.6. *Vision workflow*

The workflow starts with taking an image of the entire work-space from **ready position**, this is shown in figure 4.6a, second stage starts with detecting zones with object, then

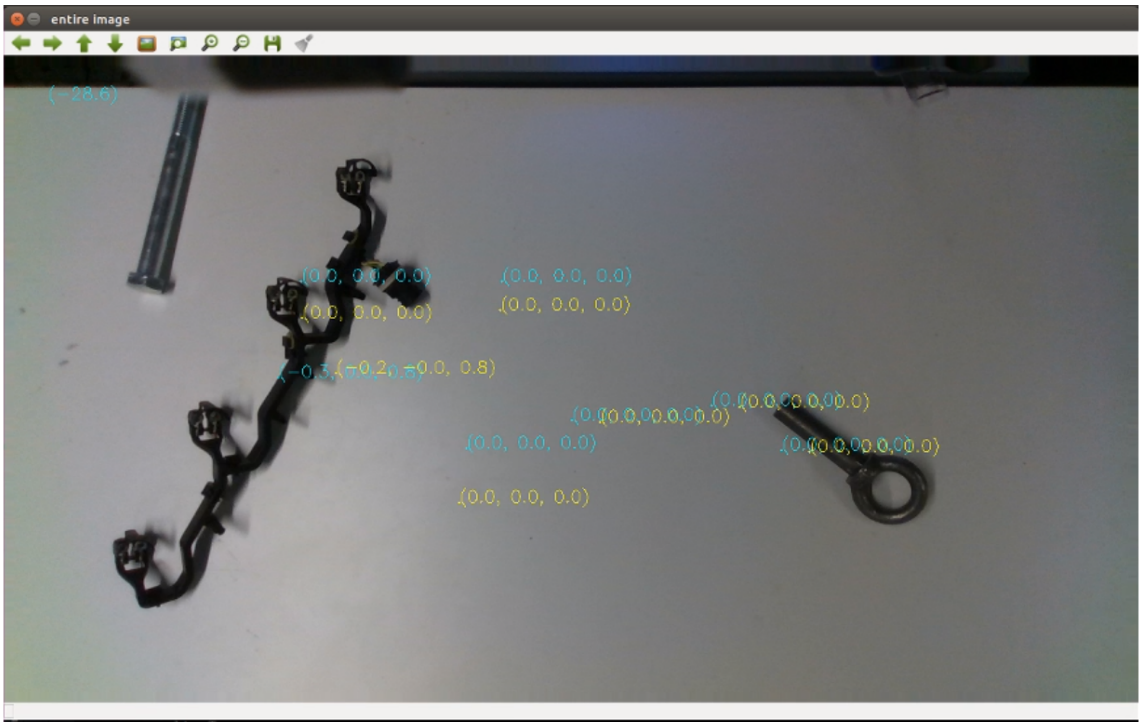


Figure 4.7. *grasping position*

the type of object in that zone is detected represented in figure 4.6b with the object name and confidence score. Third stage shown in figure 4.6c creates a masked image with the entire work-space masked as 0 and only the area with object is masked 1. After this mid-point calculation in the object area is performed to generate grasping position and orientation as shown in figure 4.7. In figure 4.7 previous grasping position are also shown and not to be confused. The system creates only one grasping position for each snap.

5 EXPERIMENTAL RESULTS

In this thesis a franka manipulator with seven-degree-of-freedom was chosen. The robot can withstand a maximum payload of 3kg and the it can grip objects with a grasping force of 70N of maximum width 80mm (ActiveRobots 2018).

5.1 Experimental Setup

The experimental setup used in this thesis composed of system as shown in figure 4.1. An actual working setup can be seen from this link: experimental Grasp Scene. During this thesis the following were the deciding factors:

1. **Surface color** : White colored surface was selected to contrast the object color from surface color. This condition is necessary for accurate edge detection.
2. **Lighting** : Overhead white lights were used. This type of lighting can cause shadows which caused error in grasping and grasping during this thesis.
3. **Object Location** : The test objects were placed on top of the surface with different orientation as shown in figure 5.2. It is assumed that objects are grasped using force along the plane of surface.
4. **Error/Noise** : Due to calibration and measurement error the grasping point were different from calculated points.

5.2 Evaluation of object recognition

An engine is composed of different type of component. Three components of different shape, size and color was chosen during this thesis as shown in figure 5.1.

The training set composed of 500 images of each type and the test set included 100 images of each type. As explained earlier RGB image was utilised for training the network. The total training time with a GTX 1060 graphics card was around 20 minutes.

Here the performance of the system developed can be calculated similar to **success rate** as done in (Delowar et al. 2017):

$$successrate = \frac{ErrorNumberofmisclassification}{Totalnumberoftestdata} \tag{5.1}$$

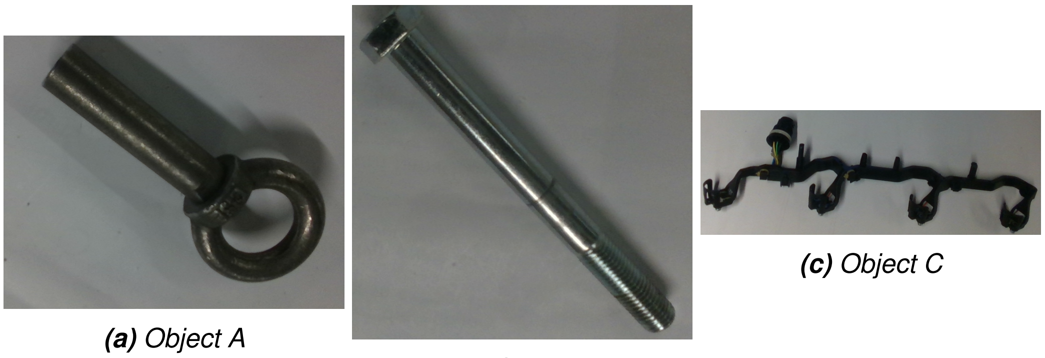

(a) Object A *(b) Object B* *(c) Object C*

Figure 5.1. *Experimental objects*

Referring to Table 5.1 there was a total of 12 scenes which were tested during this thesis. The detection results of different objects are different. From the table 5.1 it can be inferred that

- The designed detector was able to detect **objectB** and **objectC** accurately with a success rate of 100%.
- The detector falsely detected **objectA** as **objectB** in 50% of the test cases. This can be attributed to the fact that both objects were metallic and the detector was not able to classify between them.

Hence, the success rate as defined in equation 5.1 for this thesis method was 0.5. This accuracy is dependent on factors such as described in section 5.1

5.3 Evaluation of object grasp

The performance of the system can be divided into two parts evaluation of detection as mentioned above and then the ability of the robot to pick and place the detected object. In table 5.1 the grasping result is also presented the important points to note are:

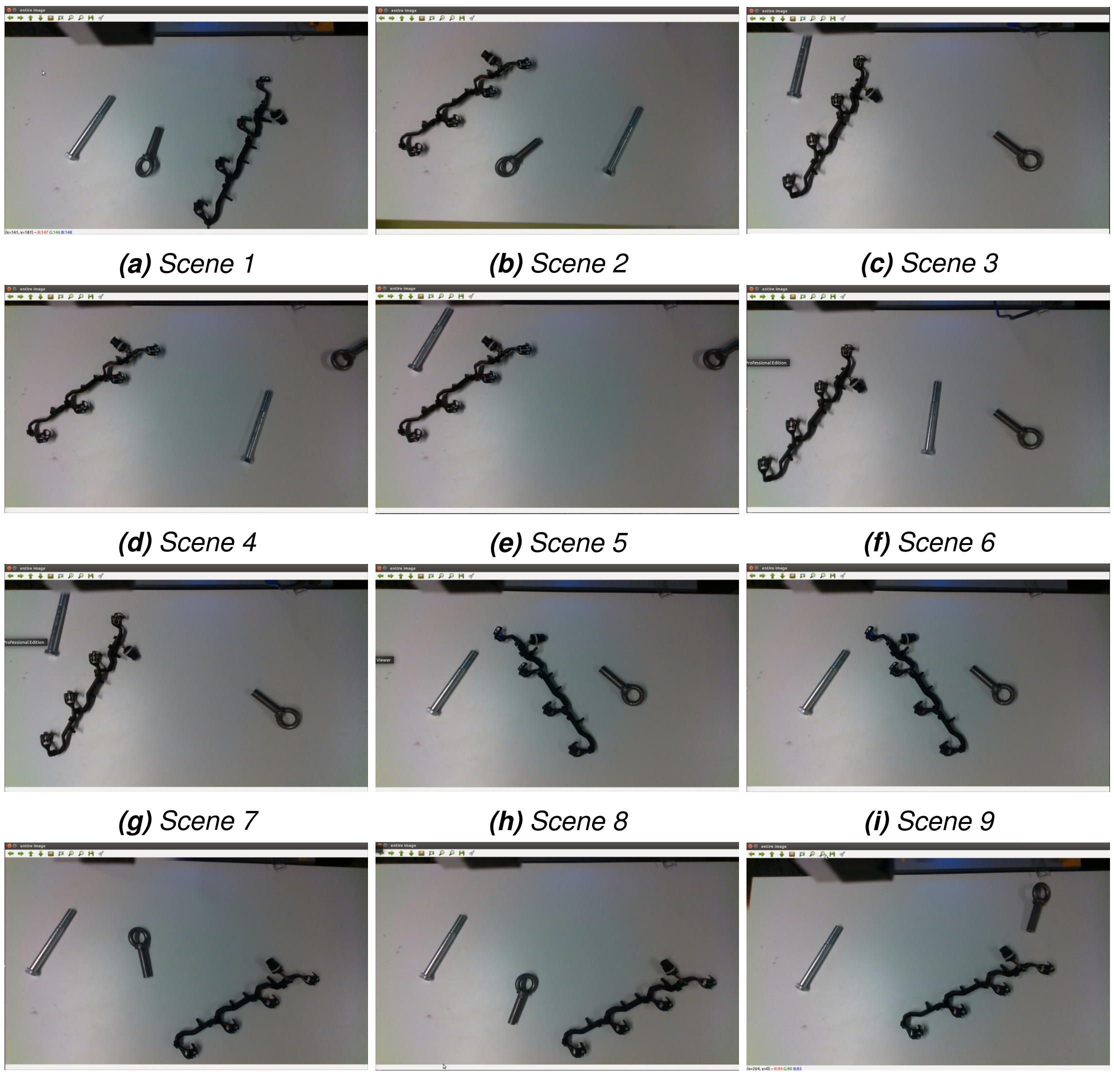
- In the scenes if the object detection is **false** then grasping is also considered as **false**. Hence grasping result for **objectA** is 50%.
- Grasping result for **objectB** is 66%. Whereas grasping result for **objectC** is 83%.

This result is based on the experimental results formed during checking with all the test objects being present in the work-space. The image of the work-space was captured from top ready position of the robot, such that entire work-space area is captured. Different object location were chosen as shown in figure 5.2.

5.4 Experimental Data

In this section the experimental scenes and their associated data for performance evaluation are presented. For evaluation a total of 12 different scenes comprising of the test objects in different location were tested. In figure 5.1 the test objects included in this thesis are shown.

In figure 5.2 the different scenes used for performance evaluation are shown. All objects are placed on top of a table in different location. The entire work-space is placed in such a way that the system while capturing a snapshot of the work-space is able to view the entire work-space.



(a) *Scene 1* ***(b)*** *Scene 2* ***(c)*** *Scene 3*

(d) *Scene 4* ***(e)*** *Scene 5* ***(f)*** *Scene 6*

(g) *Scene 7* ***(h)*** *Scene 8* ***(i)*** *Scene 9*

(j) *Scene 10* ***(k)*** *Scene 11* ***(l)*** *Scene 12*

Figure 5.2. *Experimental Scenes*

The figure 5.2 shows the state of the objects in the work-space. These snaps are taken by the camera fixed with robot gripper from its ready position. In this experimental setup 3 different object from a engine were selected as the sample objects. A CNN network first detect the the object as desired by the user and then the grasping position is found. In the following table 5.1 **Detection result** represents the $True$ and $False$ detection of the object. **Grasp result** represents whether the object was grasped properly, as such the robot was able to pick and place the object.

Scene	Object name	Detection result	Grasp result

Scene 1	object A	True	True
	object B	True	True
	object C	True	True
Scene 2	object A	False	False
	object B	True	False
	object C	True	True
Scene 3	object A	False	False
	object B	True	True
	object C	True	True
Scene 4	object A	False	False
	object B	True	True
	object C	True	False
Scene 5	object A	True	False
	object B	True	True
	object C	True	True
Scene 6	object A	False	False
	object B	True	True
	object C	True	True
Scene 7	object A	False	False
	object B	True	False
	object C	True	True
Scene 8	object A	False	False
	object B	True	False
	object C	True	True
Scene 9	object A	False	False
	object B	True	False
	object C	True	True
Scene 10	object A	True	True
	object B	True	True
	object C	True	False
Scene 11	object A	True	True
	object B	True	True
	object C	True	False

Scene 12	object A	True	True
	object B	True	True
	object C	True	True

Table 5.1. *Experimental data*

With this table 5.1 this section can be concluded and the discussions and conclusions are presented in the next chapter.

6 DISCUSSION AND CONCLUSIONS

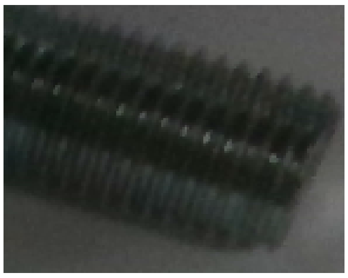
6.1 Discussion

This section discusses and reviews the work done in this thesis and how the research questions presented in chapter 1 have been addressed. The subsections in the next section holds discussions pertaining to each of the posed research questions and create a automated pick place system.

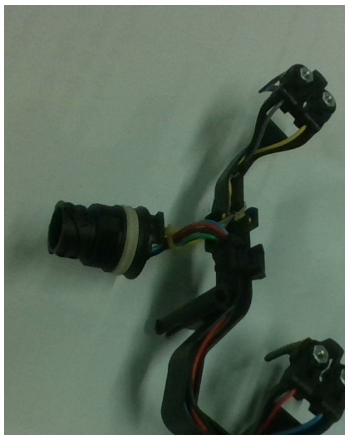
6.1.1 RQ1: What features a machine learning framework should have for an industrial pick-place application?

This question deals with creation of a framework which can be followed to generate an object detection model in an assembly environment using the manipulator. As mentioned in section 4.4.1 the robot goes through a set of location selected by a human operator and captures the image of the object to create a image data-set. Utilising the robot for data-set creation makes the process more efficient and faster, the user needs to define the trajectory once. And the same defined trajectory can be utilised again to create data-set for other objects. The human operator should go through the images captured to clear the noisy and malformed images which may not be good for training the network as shown in figure 6.1.

(a) *Object A*



(b) *Object B*



(c) *Object C*

Figure 6.1. *Noisy and malformed images*

The data-set created consisted of three types of images for each image captured with a depth sensor camera. In this thesis RGB image, depth image and edged image were saved in the data set. Manipulating the parameters during training different models utilising RGB, depth and edged data were tested. With additional information there was no significant improvement in detection results.

To train a detector using the generated image data-set, a simple 4 layer neural network based on LeNet architecture was used. During this thesis experiments were performed by increasing the number of layers, this addition did not produce any remarkable improvement in detection but the training time increased.

6.1.2 RQ2: How to select grasp positions?

As discussed earlier the camera is positioned on top of work-space and takes an image of the current work-space state, with this image, a RoI is generated which indicates presence of some object. Presence of the desired object is confirmed using the detector. Next a masked image is placed where the object is present. This masked image is just rough approximation of the object.

This approximate shape made it possible to generate a set of points placed centrally on this masked image as shown in figure 4.4. After finding the central points we can set the grasping point to be on both sides of the central point and hence were able to find grasping position to grasp object efficiently. From the set of central points the point which is placed closest to the centre of the entire object is selected. Also connecting the set of points by a line, it was possible to find the orientation of the object . This orientation is used such that the grip is performed perpendicular to the surface of object.

6.1.3 RQ3: What kind of robot controller to use?

During the start of this thesis open-source controllers such as **MoveIt** was tested to control the robot movement. Although much easier to use **MoveIt** required accurate grasping information and failed on collision with the surface. This thesis required to keep account of the inaccuracies in measurement from a depth camera and had to implement a robot controller which can cater to collision with the surface.

During this thesis impedance controller was chosen to control movement of the robot and perform a successful grasping operation. As explained in section 2.3.4 impedance controller not only provided ways to deal with collision but can also be used in applications where human interaction with the manipulator is necessary.

6.2 Conclusion and Future Scope

This section presents the summary and contribution of this thesis in the field of robotics and machine learning in a assembly line.

6.2.1 Thesis Summary

This research started with the idea of building a framework which can be developed and scaled up to utilise the existing technology in object detection for an assembly line as explained in the introduction chapter. This lead to a research journey involving a Panda robot from Franka emika and some simple neural networks to build a system fully capable of performing a basic pick and place operation in an assembly line.

With the theoretical background, the author provided the an abstract insight of the research in the field of object detection, robot grasping, robot controllers. This core technologies formed the basis of this thesis. This was followed by the methodologies developed during the course of this thesis to implement a working solution. The solution can detect test objects with a simple network and can perform pick and place operation as desired by the user.

The chapter on implementation provided an idea of how all the different methods from different fields of engineering came together to build a system for the completion of this thesis. The chapter on experimental results compared results obtained in this thesis with other research methods in the same area.

6.2.2 Future Research

As for directions of future research, the author proposes a few different directions taking into account the limitations mentioned in the first chapter. The future scope for this thesis will be:

1. Create a better detector - As explained in section 6.1.1, a simple detector which was chosen as a proof of concept. There has been advance in the field of object detection such as explained in section 2.1.2. A future work can be to utilise those methods in object detection with the image data-set created during this thesis.
2. Ways to negate effects of shadows - Lighting can be important factor regarding detection of edge of the object. The shadows can create false edge which can be reduced using better lighting or using a filter to reduce the effect of shadows.
3. Use depth data to generate better detector and compute grasp points - All the images collected consisted of RGB, Edged and depth. Although depth data was available the data could not be utilised to generate a better detector. Depth data can be utilised since each pixel position in an image will have different depth values and this will be unique for each object.
4. Motion control - A controller which can detect obstruction, stop on detection and proceed when the obstruction is removed should have been the ideal controller for this application.

REFERENCES

Ackerman, E. (2020). *Covariant Uses Simple Robot and Gigantic Neural Net to Automate Warehouse Picking*. [Online; accessed 08-March-2020]. URL: `https://spectrum.ieee.org/automaton/robotics/industrial-robots/covariant-ai-gigantic-neural-network-to-automate-warehouse-picking`.

Simon, M. (2020). *This Robot Hand Taught Itself How to Grab Stuff Like a Human*. [Online; accessed 08-March-2020]. URL: `https://www.wired.com/story/this-robot-hand-taught-itself-how-to-grab-stuff-like-a-human/`.

Klingbeil, E., Rao, D., Carpenter, B., Ganapathi, V., Ng, A. Y. and Khatib, O. (2011). Grasping with Application to an Autonomous Checkout Robot. *IEEE International Conference on Robotics and Automation*.

Mason and Salisbury (1985). *In Robot Hands and the Mechanics of Manipulation*. The MIT Press.

Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: a review. *IEEE International Conference on Robotics and Automation*.

Miller, A. T., Knoop, S., Allen, P. K. and Christensen, H. I. (2003). Automatic grasp planning using shape primitives. *In International Conference on Robotics and Automation (ICRA)*.

ennomotive (2018). *7 Types of Robot Grippers and their Industrial Applications*. [Online; accessed 20-March-2020]. URL: `https://www.ennomotive.com/robot-grippers-industrial-applications/`.

Ku. Vasundhara, H. L. (2014). Study of Region Base Segmentation Method. *International Journal of Advanced Research in Computer Science and Software Engineering*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision, 2004*.

Bay, H., Tuytelaars, T. and Gool, L. V. (2006). SURF: Speeded Up Robust Features. *European Conference on Computer Vision*.

Bruce, N. D. B. and Kornprobst, P. (2009). Harris Corners in the Real World: A Principled Selection Criterion for Interest Points Based on Ecological Statistics. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

M. Jogendra Kumar Dr. GVS Raj Kumar, R. V. K. R. (2014). Review on Image Segmentation Techniques. *International Journal of Scientific Research Engineering Technology*.

Hartigan, J. A. (1975). *Clustering Algorithm*. Wiley Publishing.

Liu, X., Deng, Z. and Yang, Y. (2018). *Recent progress in semantic image segmentation*. Springer Link.

Chaturvedi, R. N., Munot, K., Mehta, N. and Reddy, S. M. (2017). A Review on Image Segmentation Techniques with an Application Perspective. *IInternational Journal of Advanced Research in Computer Science*.

Aravindh, G. and Manikandababu, C. (2015). Algorithm and implementation of distributed canny edge detector on fpga. *ARPN Journal of Engineering and Applied Sciences*.

Ilkin, S., Hangisi, S., Tafrali, M. and Sahin, S. (2017). The Enhancement of Canny Edge Detection Algorithm Using Prewitt Robert And Sobel Kernels. *International Conference on Engineering Technologies*.

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. *International conference on machine learning*.

LeCun and Bengio (1995). *handbook of brain theory and neural networks*. MIT Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

Girshick, R. (2015). Fast R-CNN. *IEEE International Conference on Computer Vision*.

Xu, J. (2017). *Review of Deep Learning Algorithms for Image Semantic Segmentation*. [Online; accessed 24-August-2019]. URL: `https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9`.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Conference on Computer Vision and Pattern Recognition*.

Hui, J. (2018). *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*. [Online; accessed 24-August-2019]. URL: `https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088`.

Liu1, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision*.

J. Bohg A. Morales, T. A. and Kragic, D. (2014a). Data-Driven Grasp Synthesis—A Survey. *IEEE Transactions on Robotics*.

Jabalameli, A., Ettehadi, N. and Behal, A. (2019). Edge-Based Recognition of Novel Objects for Robotic Grasping. *SAI Computer Vision Conference (CVC)*.

A. Sahbani S. El-Khoury, P. B. (2012). An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*.

A. M. Okamura, N. S. and Cutkosky, M. R. (2000). An overview of dexterous manipulation. *IEEE International Conference*.

Park, Y. C. and Starr, G. P. (1992). Grasp Synthesis of Polygonal Objects Using a Three-Fingered Robot Hand. *The International Journal of Robotics Research*.

Howard, W. S. and Kumar, V. (1996). On the stability of grasped objects. *IEEE Transactions on Robotics and Automation*.

Diankov, R. (2010). Automated construction of robotic manipulation programs. *Ph.D. dissertation, Robotics Inst., Carnegie Mellon Univ.*

J.Weisz and Allen, P. K. (2012). Pose error robust grasping from contact wrench space metrics. *Proc. IEEE Int. Conf. Robot. Autom.*

R. Detry, N. P. and Piater, J. H. (2009). Probabilistic Framework for 3D Visual Object Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ciocarlie, M. and Allenc, P. (2009). “Hand posture subspaces for dexterous robotic grasping". *Int. J. Robot. Res.*

J. Bohg A. Morales, T. A. and Kragic, D. (2014b). Data Driven Grasp Synthesis A Survey. *IIEEE Transactions on Robotics*.

A. Saxena, J. D. and Ng, A. Y. (2008). Robotic grasping of novel objects using vision. *Int. J. Robot. Res.*

Nguyen, V. D. (1986). Constructing force-closure grasps. *Robotics and Automation . Proceedings. 1986 IEEE International Conference*.

GmbH, F. E. (2017a). *Franka page*. [Online; accessed 08-March-2019]. URL: `https://www.franka.de/technology`.

– (2017b). *Franka Control Interface Documentation*. [Online; accessed 08-March-2019]. URL: `https://frankaemika.github.io/docs/index.html`.

Zeng, G. and Hemami, A. (1997). *An Overview of Robot Force Control*. Robotica.

Dede, M. İ. C. (2003). Position/Force Control of Robot. *Masters thesis at The Graduate School Of Natural Aand Applied Sciences Of The Middle East Technical University*.

Ha, Q. P., Nquyen, H. Q., Rye, D. C. and Durrant-Whyte, H. F. (1998). Robust Impedance Control of Excavator Dynamics. *Australian Centre for Field Robotics*.

Fisher, W. D. and Mutjaba, M. S. (1991). *Hybrid Position/force Control: A Correct Formulation*. Hewlett-Packard Company.

Fodor, G. and Tevesz, G. (1999). *Hybrid Position and Force Control Algorithm Expansion of a Robot Control System*. Periodica Polytechnica.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). GradientBased Learning Applied to Document Recognition. *Proceedings of the IEEE*.

Wikipedia (2018). *Intel RealSense*. [Online; accessed 10-March-2019]. URL: `https://en.wikipedia.org/wiki/Intel_RealSense`.

ActiveRobots (2018). *franka emika datasheet*. [Online; accessed 09-March-2020]. URL: `https://s3-eu-central-1.amazonaws.com/franka-de-uploads-staging/uploads/2018/05/2018-05-datasheet-panda.pdf`.

Delowar, H., Genci, C., Mitsuru, J. and Shin-ichiro, K. (2017). Pick-place of dynamic objects by robot manipulator based on deep learning and easy user interface teaching systems. *Industrial Robot*.