CHALMERS
UNIVERSITY OF TECHNOLOGY

Bin picking and placing of complex geometric objects using convolutional neural networks

Master's thesis in Systems, Control, and Mechatronics

Mahmoud Hanafy and Samuel Ingemarsson

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:NN

Bin picking and placing of complex geometric objects using convolutional neural networks

Mahmoud Hanafy
Samuel Ingemarsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Bin picking and placing of complex geometric objects using convolutional neural networks
Mahmoud Hanafy and Samuel Ingemarsson

Supervisor: Per-Lage Götvall, Volvo Group Trucks Operations, Gothenburg, Sweden
Examiner: Yiannis Karayiannidis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden

Master's Thesis 2018:NN
Department of Electrical Engineering
Division of Division name
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The setup of all equipment used to carry out the task, stereo cameras, robotic arm (UR10), MiR, and objects in bins.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2018

Bin picking and placing of complex geometric objects using convolutional neural networks
Mahmoud Hanafy and Samuel Ingemarsson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Using robotic manipulators to carry out pick and place tasks in industrial settings will not only shorten the process duration but also will decrease its cost and increase the production. The problem with the bin picking solutions of today is that they are made for picking specific components and not designed for handling different objects with complex geometries.

One way to solve this problem is to use convolutional neural networks (CNN), trained for multiple objects, to find the best grasping point on the component. By using a stereo camera setup it is possible to find the point in each camera and then compute the depth using triangulation.
In the developed system in this project, the best object to be picked is determined using CNN, the position and orientation of this object are calculated, then the object is picked and placed on a mobile robot. Robotic Operating System (ROS) acts as a framework to communicate between different robot sensors and actuators to manage the control of the robot.
The approach followed in this project shows that multiple objects with different geometries can be recognised and picked from grasping points determined by the network, however in some cases when there are many objects in the bin, the network suggests a grasping point that is not accurate enough leading to inaccurate object picking and as as a result, inaccurate placing. While doing experiments to check how objects are picked, it was found that when the grasping point is accurately determined, picking and placing processes are performed in an acceptable and accurate way, but when the grasping point is calculated to be at the edge of the object for example, the object either falls or picked in a wrong way that would make it wrongly placed. In the later case the process will take a longer time; since it will be repeated if no object is picked, or the picked object orientation will be checked through another checking algorithm.

Keywords: Pick and Place, Computer vision, Neural networks, Convolutional Neural Networks (CNN), Robotics, Robot Operating Systems (ROS), Universal Robots (UR10)

Acknowledgements

First, we would like to express our gratitude to Volvo Group trucks operations for offering us the opportunity to do our master thesis at Volvo Group.
We would like to thank our supervisor at Volvo Per-Lage Götvall for letting us be a part of this exciting project, and also for his continuous support and guidance. We also would like to thank our supervisor at Chalmers Yiannis Karayiannidis for his support, motivation, and letting the work go smoothly. Mahmoud would like to thank Swedish Institute (SI) for funding his studies at Chalmers, and giving him the opportunity to be a part of network for future global leaders which is an inspiring network personally and professionally.
Our gratitude to our teachers and friends who spare no effort to provide information and help. Last but not least, we would like to thank our families for their endless support throughout the years.

Mahmoud Hanafy and Samuel Ingemarsson, Göteborg, June 2018

Contents

List of Figures

List of Tables

1
Introduction

Due to the vast development in the field of automation and robotics, robots nowadays perform several tasks in industry. One common task in many industries is to pick components from different bins (Bin Picking) then place them together on a table. Although pick and place task is relatively easy for a human, it could be a complicated task for a robot particularly for cases in which the objective is to pick up different objects from different bins.
In a typical bin picking task, the object is recognised by cameras and the robot determines the location and pose of the object. Based on the collected data from the cameras, the robot plans a path towards the object and develop a plan to grasp it. When the object is grasped, the robot plans another path to reach the position of the table where the object is placed softly on. If more than one object are picked at time, or if the object is placed on a table where there is no current capacity for it, that will lead to failure of many tasks, also if the object is picked and placed in a wrong orientation, that will lead to failure of the task. The current technologies are not advanced or reliable enough to pick up and place complex objects in an efficient way. In this thesis project **the objective** is to pick and place complex objects. Complex here means objects that their geometry allow them to be attached together when they are picked, like U-tubes.

1.1 Background

At **Volvo Group**, there are assembly lines in manufacturing plants where assembly processes are done manually. Different components with different sizes are picked by hand from a storage rack which is away from the assembly station and placed on a table to be delivered to the assembly station. This process is done repeatedly and needs to be done in a more efficient way.
This master project provides a solution to enhance the quality of picking objects and placing them smoothly on a table which is mounted on a mobile industrial robot (MiR) by performing two main tasks:

- Recognising different objects using neural networks, the technique used in this thesis is *convolutional neural networks* (CNN).
- One and only one object at time must be picked.
- Placing the picked object in a specific position and with a specific orientation.

Volvo Group wants to enhance the technique of bin picking and placing to increase the quality of pick and place task and make the storage localisation more flexible. Studying the scenario of picking more than one object at time, and placing them on

a table are important tasks to assure the safe and efficient flow of the process.

1.2 Objective

The objective of the thesis is to enhance the quality and efficiency of bin picking and placing task at assembly lines. This is done by using Universal Robots UR10 robotic arm. Objects in the bins are recognised using CNN, grasping points are determined, then the robotic arm picks the object and places it on a kitting trolley table. At the end of this project, the robot should be able to recognise objects in different geometries, pick and place them in a free specific position on the table. As a result, the process of picking and placing will be faster and more efficient.

1.3 Scope

This project focuses mainly on developing and using an efficient computer vision technique, developing a strategy to move the robot from picking to placing locations. The project also focuses on placing the carried object on a specific orientation and place on a kitting trolley table.
The boundaries of the project can be written as:

- Human- Robot interaction is not included in this project.
- The optimisation of trajectory path will not be mainly considered through the project.
- The used robot is UR10 which has a maximum payload limit of 10 kg.
- The project should be done in 20 weeks.

1.4 Contribution

Bin-picking is a common robotics problem, many researches are conducted to provide more efficient solutions. The contribution of this thesis work can be listed as follows:

- The 3 finger end-effector picks one and only one object at a time. Thus the safety for both the robot itself and the human working beside the robot is guaranteed.
- The picked object is placed on a table with a specific position and orientation.

1.5 Literature

Pick and place problem is a very common in the field of control and robotics, there are a lot of research conducted to provide efficient solutions for the problem. The first approach was done by L.G. Roberts in 1963 using simple 2D images using single camera [5]. The goal in his work was to detect polyhedral objects. Later in 1968, Shirai and Suwa worked to achieve a 3D detection of objects [6]. In 1975, Tsuji and Nakamura did work which describes a vision system which can be used in industrial applications [7]. In 1986, Ikeuchi, Horn, Nishihara, Sobalvarro and Nagata published a paper discussing a method to estimate the orientation of an object with respect

to the camera and this is done by using Photometric Stereo and Extended Gaussian Images (EGIs) [8]. Bolles and Horaud described in 1986 a way to determine location of an object using 3D sensors and modules of an object [9]. Recently there are more advanced techniques which are used like the paper [10] presents a solution for picking the object using RGB-D sensor data and by using 3D detection technique using Kinect camera. It also discusses optimisation method for path planning using A* algorithm. In paper [11], Microsoft Kinect sensor is used to calculate the depth in the image. The paper [12] addressed the problem of recognising an object and picking it by determining two mask images, one describing contact points which will grasp the object, the other image is the collision one which show that there is already a picked object and not to pick another one. The paper did not deal with the situation of picking more than one object at time. In paper [13], they presented a vision based robotic system which is capable of detecting and estimating 3D poses of an object based on the Fast Directional Chamfer Matching (FDCM) algorithm. The approach used in this paper provide reliable geometric features for different materials that have different characteristics. The paper addressed the problem of picking and placing but did not consider the case when two object are picked at the same time. Grasping an object requires knowing the dimension and geometry of the object. The desired grasping avoid disturbing forces and moments resulting from inertia and gravity. The grasping must prevent improper contact between the end effector and the object in order to prevent sliding or missing of the object [14].
Convolutional neural networks (CNNs) is a popular approach for image recognition and has gained much in popularity in recent years because of having increased size of datasets and computation power. This leads to being able to use larger networks (deeper) which have shown a great accuracy [15].

2
Robotics and collaborative robots

The pick and place task in this thesis project is done using UR10, a collaborative industrial robot arm. This chapter aims at providing some necessary basic knowledge of robotics, also it discusses the differences between traditional and collaborative robots. Communication among different sensors and actuators of the system is done using ROS, this will be explained in this chapter.

2.1 Robotic Fundamentals

According to Robot Institute of America (RIA), a scientific definition of a robot is: "A robot is a re-programmable and multifunctional manipulator, devised for the transport of materials, parts, tools or specialised systems with varied and programmed movements, with the aim of carrying out different tasks." [16]
This definition aligns accurately with the purpose of the thesis which is mainly a pick and place robotic problem. There are essential robotic concepts terms that are necessary to discuss at the beginning.

2.1.1 Manipulator classification

A robotic manipulator is composed of number of links connected together through joints. The joints can be revolute or prismatic (linear) joints. Each manipulator has number of degrees of freedom (DoF) which is determined by the number of joints the robot has, however most industrial robots have 6 DoF.
Based of the type and sequence of DoF, manipulators can be classified as Cartesian, spherical, cylindrical, SCARA, and anthropomorphic [17]. An anthropomorphic manipulator can have six revolute joints, the structure of this kind of arms usually base, shoulder, elbow and, wrist. It is usually common to use these kind of manipulators in industries since they can be used in wide applications because of its high dexterous. Three DoF are used for positioning in the world or global frame, and the other three are used for orientation.

2.1.2 Kinematics: Forward and inverse kinematics

One important term in robotics is called Kinematics. Kinematics can be defined as the study of the motion of a body (robot) without considering or studying the cause of this motion. An object in a space has a position and orientation (position and orientation are known as pose), kinematics describes the position, orientation,

velocity, and acceleration of a body, i.e kinematics describes the relationship between the position of robot joints and the pose of the end-effector. Kinematics can be described into two categories: Forward kinematics and inverse kinematics.

- **Forward Kinematics** The joint angles of the robot are known, the pose of the end-effector with respect to the base is calculated as a function of joints.
- **Inverse Kinematics** The pose of the end-effector is known, the joint angles are calculated as a function of joint angles. [18]

2.1.3 Collaborative robots

Robots are providing efficient solutions to industry in many fields, before the presence of "Collaborative robots", the robots used to carry out tasks were separated in a specific area or cage to prevent any direct interaction with humans, and consequently eliminating injuries.
The main reason for this separation is safety. One idea of collaborative robots is to discover different opportunities to reinvent industrial robots in which these new robots be able to work safely with humans and here comes the term collaborative. Collaboration as a word according to oxford dictionary means: The action of working with someone to produce or create something, and that is the main purpose of this kind of robots. Currently there are different companies that pay attention to expand their operations using collaborative robots such as Universal Robots(UR) and KUKA. Different cobots are shown below in figure 2.1

(a) UR10 collaborative robot

(b) KUKA LBR collaborative robot

Figure 2.1: Different collaborative robots. The source of KUKA robot is [1]

2.2 Robot Operating System (ROS)

ROS is a framework that is usually used in robotic applications. It is functioning as an operating system which manages the data transfer between different processes. ROS provides a set of tools that helps in writing and running codes among different computers, also it provide the ability to visualise state information from ROS. ROS was initially developed in Stanford Artificial Intelligence Laboratory (SAIL) in 2007, and currently it has a large community for sharing and documentation of projects. [19]

2.2.1 Advantages of using ROS

There are different frameworks that are similar to what ROS is doing, but there are some advantages that distinguish ROS among other frameworks that are given below:

- Different sets of tools: many tools for visualising and simulation like rqt_gui and RVIZ.
- Supporting large set of sensors and actuators since it has different interface packages.
- ROS is a robust system, in which different nodes can be created receiving and transmitting data, if one node crashes, the system will still operate without this node.
- Programming language flexibility, nodes can be written in C++ or Python.

2.2.2 Communicating with robot

Robotic systems have many actuators to be controlled and also there are different sensors that read different kinds of data form the process. To communicate between all of these data, a robust framework is needed to handle different processes that happen at one time, and that is what ROS mainly is doing. Communicating between different parts of the system is done by publishing and subscribing to data.

2.2.2.1 Publish and subscribe

A node in a system is defined as a process which is performing a specific computations, for example if a system has a radar as one of its components, this radar measures the distance of an object with respect to the vehicle and send (publish) it to a topic. Other components in the system can get (subscribe) to this data to take a specific action. This process can be more clarified in figure 2.2

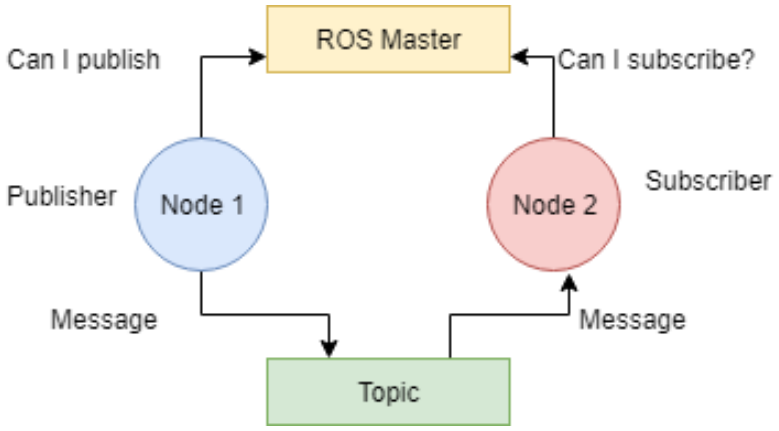


Figure 2.2: Publisher and subscriber communication

ROS master provide naming and registration service for different nodes. For node 1 to publish, it needs permission from the master, and for node 2 to subscribe it needs to notify the ROS master.

2.2.3 Frame transformation

In robotic systems, each object has its own position and orientation in the space, and it is important to determine the pose of each object with respect to the other object, and here comes the important of frames. Communication between different frames in the system is done by tf package on ROS through a broadcaster and a listener.

2.2.3.1 Broadcaster and listener

tf is a ROS package that relate a coordinate of one object's frame to another frame in real time. If for example there are two moving objects in an existing system, and it is required to get the position of object 1 with respect to object 2, the first object may first determine its position with respect to world frame by broadcast its pose with respect to world frame. The object 2 which is supposed to follow object 1 can listen to the pose of object 1 and transformed from object 2 frame to object 1 frame. An example of different frames that can be transformed to each other using tf library is shown in figure 2.3.

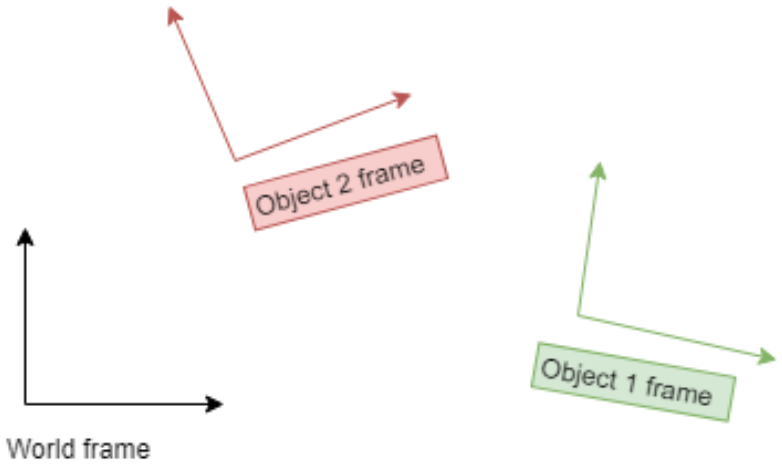


Figure 2.3: Different objects' frames that can be transformed to each other using tf ROS library

2.2.3.2 Creating frames

In ROS, if there is an additional object that is added to the system, a new frame can be created using tf library to be represented in the system. Once the pose of that frame is determined, the frame broadcaster can be run showing its pose in the system. The new created frame can be a fixed or a moving frame.

2.2.4 ROS Visualisation (RViz)

RViz (ROS visualisation) is an important package on ROS which enables displaying the data collected from sensors, and also shows different configurations and states

of the objects in the system. The states are updated in real time, so any change in the system can be monitored in any given time.

2.3 Tools and Equipment

Picking and Placing is a common problem in robotics field, there are many equipment used to carry out such tasks. In this thesis project the equipment in the following sub-sections are used.

2.3.1 Universal Robot UR10

UR10 is the largest collaborative robotic arm from universal robots. The robot can carry out tasks that include payload of weights up to 10 kg. UR10 has 6 DoF (6 revolute joints). UR10 has many advantages as follows:

- In this master project the robot is programmed using URScript programming language, the commands that control the robot are quite simple and flexible.
- UR10 was set up in this project from A to Z, and it was a fast process done in almost one day.
- Working with the robot is safe, once it is about to hit a frame or a human it directly stops and enters the safe mode.

UR10 specifications are shown below in Table 2.1

Table 2.1: UR10 Technical specifications [3]

6-axis robot arm with working radius of 1300 mm	
weight	28.9 kg
Payload	10 kg
Reach	1300 mm
Join ranges	+/- 360°
Speed	Base and shoulder: 120°/s Elbow, Wrist 1, Wrist 2, Wrist 3: 180°/s
Degree of freedom	6 rotating joints
I/O power supply	24 V, 2A in control box 12 V/24 V 600 mA in tool
Material	Aluminium, PP plastic
Temperature	0-50 C
Power supply	100-240 VAC, 50-60 Hz

In this master project the used end-effector is a 3-finger robotiq gripper,which gives the availability to pick up different objects with different configurations, thanks to its different operation modes and the ability to control each finger individually. the end effector is shown in figure 2.4

Figure 2.4: Robotiq 3-finger gripper

2.3.2 Force/Torque sensor

The force/Torque sensor used in this project is 6-axis optoforce, HEX-H model, which is easy to set up and get the readings from it. One advantage of using this sensor is that it is dust and water proof. The sensor gives high resolution readings for forces and torques in x, y and z directions. The sensor and its axis shown in figure 2.5

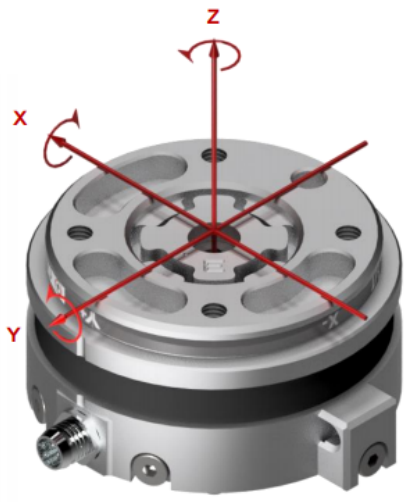


Figure 2.5: The used 6 Axis force/torque sensor

Range of forces and torques that can be measured by HEX-H optoforce is shown in Table 2.2

Table 2.2: Force/torque ranges of optoforce HEX-H model

Forces & Torques	Nominal Capacity	Deflection
Fxy	+/- 200 N	+/- 0.6 mm
Fz	+/- 200 N	+/- 0.25 mm
Txy	+/- 20 Nm	+/- 2°
Tz	+/- 13 Nm	+/- 3.5°

2.3.3 MiR100

MiR stands for Mobile Industrial Robots. It is an autonomous robot that can be controlled to move to specific positions and orientations. The main task for MiR in this project is to move from an initial pose to another pose where placing process of the picked objects will be carried out. The MiR will stay in the placing position until all picked parts are placed, then move back to the initial position. Since it is the era of industry 4.0, in which machines and robots can communicate together to carry out tasks effectively. Communication between the UR10 and MiR is an ideal application for the future industries. Mir is shown below in figure 2.6

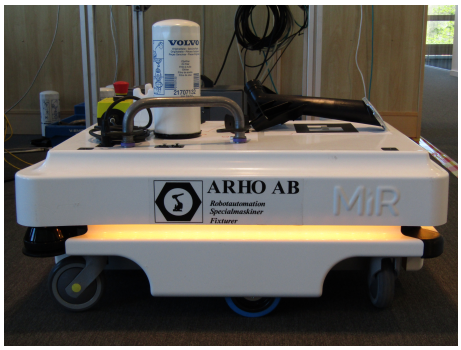


Figure 2.6: MiR100 which is used to transport picked objects from picking position to the assembly area

2.3.4 Equipment setup

For the tasks to be performed, equipment should be setup in a specific way. There are two cameras that are used to capture images and get the the depth of the scene, for that, the two cameras are used in a stereo configuration and set shown below in figure 2.8

Figure 2.7: Stereo camera configuration setup

The full setup of the system is shown in figure 2.8 below

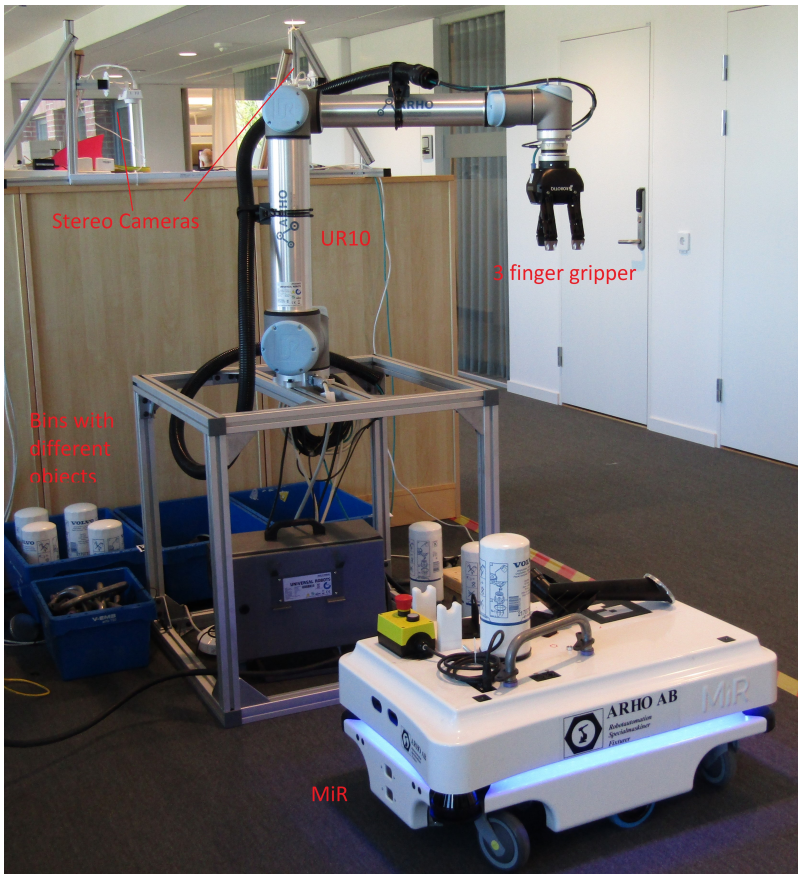


Figure 2.8: The full setup of the system

3

Neural networks

Human brain is the most complicated machine that one can imagine, it can process information in a powerful way, based on learning from previous situations, the brain can make decisions and adjust its actions based on previous data [20]. Artificial Neural Network (ANN) actually tries to mimic the neural networks in the human brain. The first section in this chapter discusses in brief how biological neurons work, and later sections discuss ANN and , Convolutional Neural Networks (CNNs).

3.1 Biological Neurons

The human brain has estimated number of neurons about 10^{12} neurons. Each neuron in the neural system can process complex biological processes. Neurons can transmit and receive signals from each other, the signals are electro-chemical signals. Neurons are connected together with synapses, which is a junction that connects two neurons together. The received signals have weights and are transmitted to the neural cell body through dendrites and soma. Different signal from several neurons are summed up, when this sum reaches a specific threshold excitation value, an output electric signal is transmitted to the output channel axon. At the end of the output channel, there are several terminal which act as synapses for other neurons in the network. [20]. The perceptron model has input signals, each input signal is associated with synapse weight. Each input signal is multiplied by its corresponding weight and added together to give the output of the perceptron model. In next sections, it will be obvious that this process is very similar to what is happening in the ANN perceptron model. The figure that shows biological neural network and the perception model is shown in fig 3.1

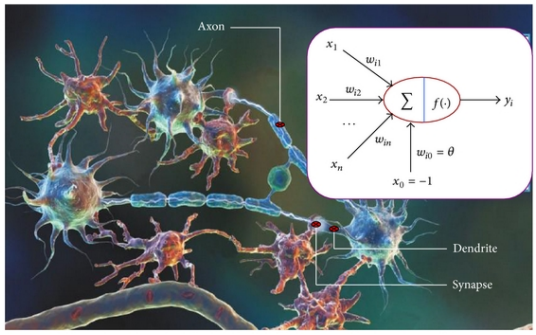


Figure 3.1: Biological neuron showing main elements (Synapse, Dendrite, and axon). The perceptron model is shown where x_0 to x_n are input signals, and w_{i0} to w_{in} are synapses weights. θ assigned to w_{i0} is the threshold excitation value. [2]

3.2 Artificial neural networks

An artificial neural network consists of layers of neurons that are connected with each other as seen in figure 3.2. The input layer feeds the inputs into the neurons of the first hidden layer. A neuron, seen in figure 3.3, consists of different weights w, a bias b and an activation function ξ. The inputs of the neuron are multiplied with the corresponding weights and then summed together with the bias. This sum is then passed into the activation function which is then, depending on the value of the sum and the type of activation function, activated or not activated.

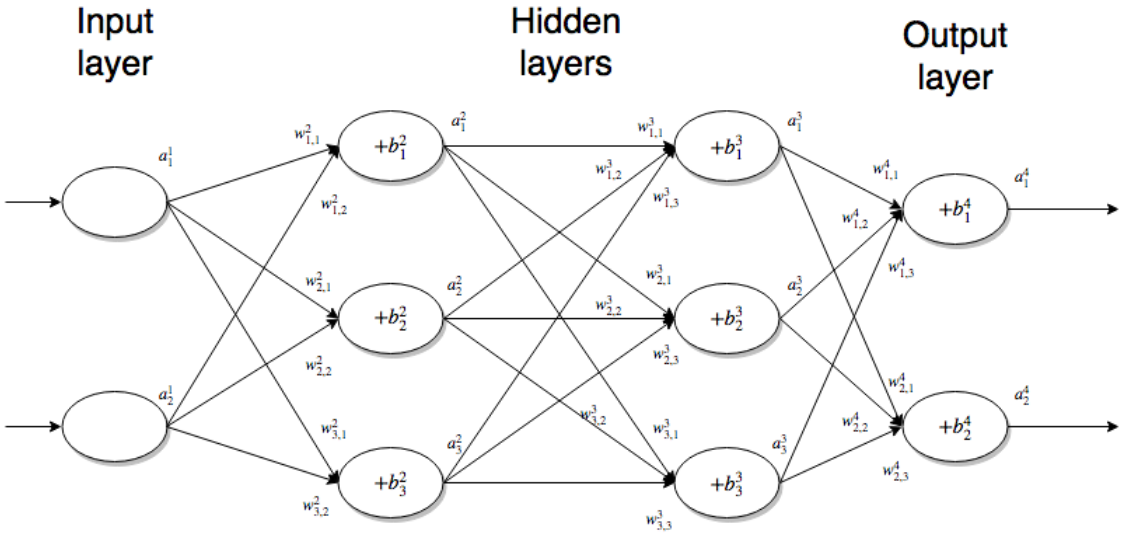


Figure 3.2: The layers of a neural network

The weights use the notation w^l_{jk} where l refers to which layer the weight is for, j the neuron in layer l and k the neuron in layer $l-1$. b^l_j is the bias in layer l at neuron j. a^l_j is the activation in layer l of neuron j. With this notation an activation

in layer l can be written as

$$a_j^l = \xi\left(\sum_k w_{jk}^l a_k^{l-1} + b_k^l)\right). \tag{3.1}$$

By

$$a^l = \xi(w^l a^{l-1} + b^l) = \xi(z_j^l) \tag{3.2}$$

which means that activations a_1^4 and a_2^4 in figure 3.2 can be written as

$$\begin{bmatrix} a_1^4 \\ a_2^4 \end{bmatrix} = \xi\left(\begin{bmatrix} w_{11}^4 & w_{12}^4 & w_{13}^4 \\ w_{21}^4 & w_{22}^4 & w_{23}^4 \end{bmatrix} \begin{bmatrix} a_1^3 \\ a_2^3 \\ a_3^3 \end{bmatrix} + \begin{bmatrix} b_1^4 \\ b_2^4 \end{bmatrix}\right).$$

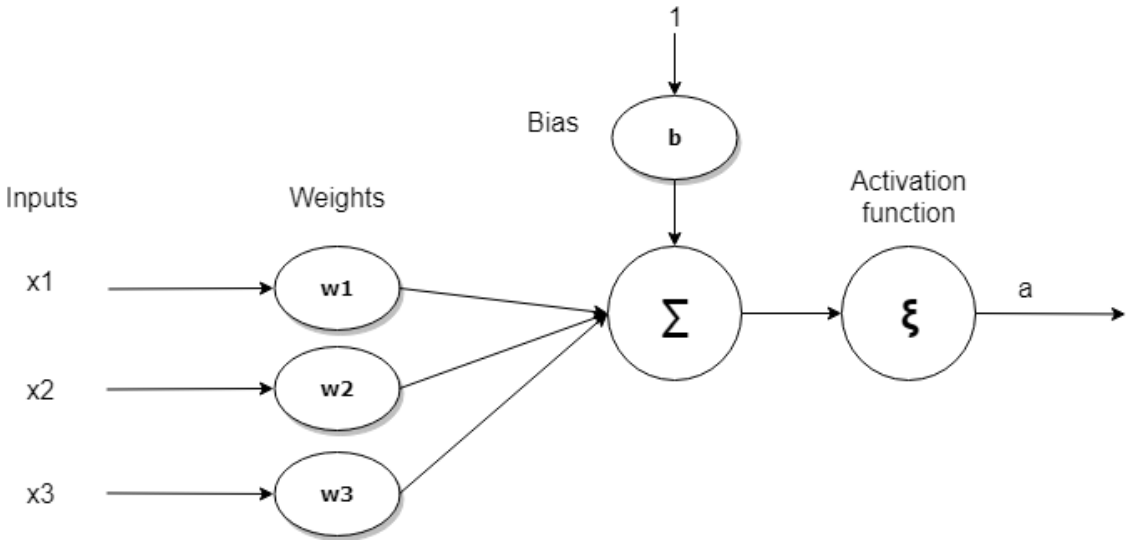


Figure 3.3: Simple model of a neuron

There are lots of different activation functions Common activation function is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and the rectified linear unit (ReLU) $f(x) = max(0, x)$.

3.2.1 Training

For a neural network to behave as wanted, give the correct output, the weights and biases chosen in such a way. To find these parameters the network have to be trained. To train the network, first, a function is needed that decides how good our network behaves. A cost function is a function that compares the output from the network with a known true output. The mean square error (MSE)

$$C(w, b) = \frac{1}{2n}\sum_x (a^{(L)}(x) - y)^2 \tag{3.3}$$

is a commonly used cost function where $a^{(L)}(x)$ is the activation of the output layer from the network when x is the input and y which is the desired output. By

training the network it is possible to find weights w and biases b that makes the cost $C \approx 0$. To find the optimal weights and biases an optimiser algorithm and *backpropagation* are used. The optimiser *gradient descent* works using the gradient of the cost function which shows in which direction the parameters should move to decrease the cost function as seen in figure 3.4. For one and two parameters it can be easily thought as a ball that is sliding down a hill to the minimum point but for millions of parameters, as typical network has, it is much harder to imagine.

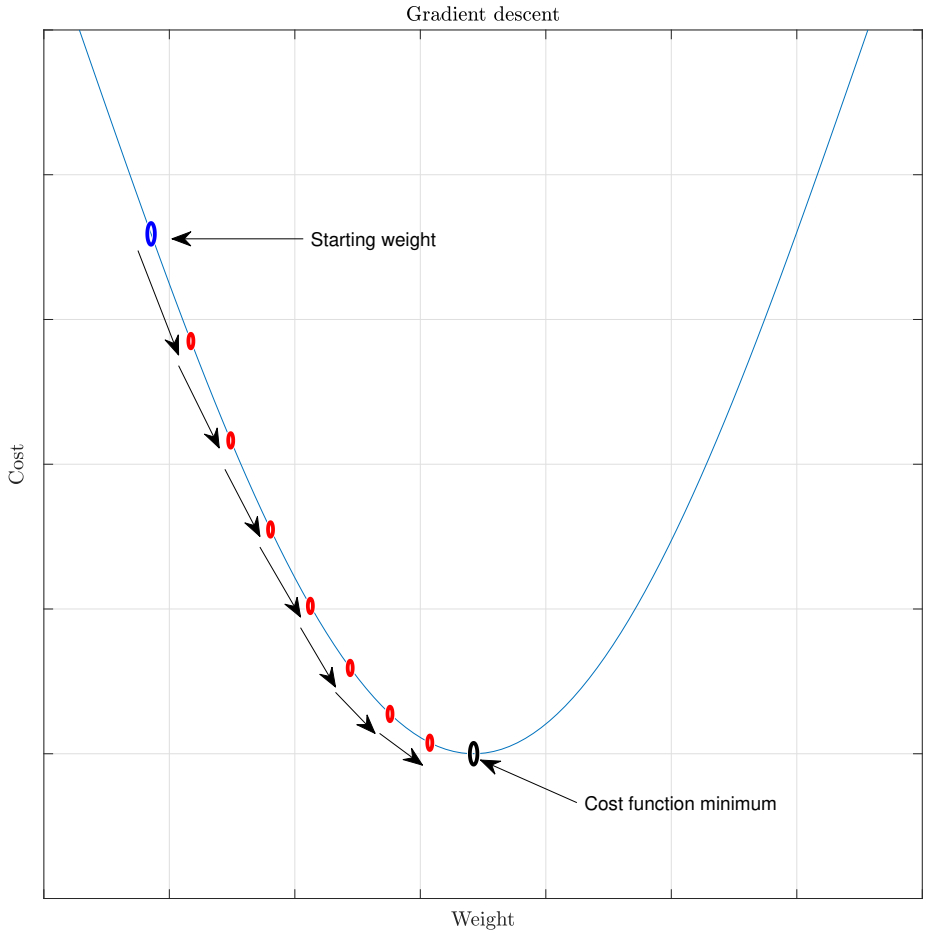


Figure 3.4: Gradient descent for one parameter.

The gradient is a vector with the partial derivative of the cost C with respect to the weights $\frac{\partial C}{\partial W}$ and the biases $\frac{\partial C}{\partial B}$. The training is done by using the training data consisting of training inputs x together with true outputs y and finding the averaged gradient

$$\frac{\sum_x \nabla C_x}{n} = \nabla C \tag{3.4}$$

of all the gradients for each training example. When the gradient has been computed

each weight and bias are then updated according to

$$\begin{aligned} w &= w - \eta \frac{\partial C}{\partial w} \\ b &= b - \eta \frac{\partial C}{\partial b} \end{aligned} \tag{3.5}$$

where η is the learning rate factor that decides how large each step is. By having a too low rate the learning will be slow and by having a too large rate gradient descent may fail to converge or even diverge.
The problem with gradient descent is that it takes a long time to compute the gradient when the size of training examples is big. A solution to this is to use *stochastic gradient descent*. By choosing a random subset of training inputs $X_1, X_2, ..., X_m$, often called batch , equation (3.4) now becomes

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C \tag{3.6}$$

which gives the new update rules

$$\begin{aligned} w^l &= w^l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w^l} \\ b^l &= b^l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b^l}. \end{aligned} \tag{3.7}$$

Now it is known how to update weights and biases but how the gradient of each example ∇C is found still not known. This is where the *backpropagation* algorithm comes in. Backpropagation starts by finding the activation of the output layer L and then works itself backwards finding all the activations to the first hidden layer. When all the activations have been found it is possible to find the partial derivatives with respect to the weights and biases. Define

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \xi'(z_j^L) = (a_j^L - y_j) \xi'(z_j^L) \tag{3.8}$$

which can be seen as the error of the neuron J in the output layer L. It can be rewritten in vector form as

$$\delta^L = (a^L - y^L) \odot \xi'(z^L) \tag{3.9}$$

, where $\odot$ is elementwise multiplication.
Next step is to define and compute the error in layer l in terms of the next layer $l+1$

$$\delta_j^l = \frac{\partial C}{\partial z_J^l} = \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \tag{3.10}$$

which in

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \xi'(z^l). \tag{3.11}$$

With the use of equation (3.9) and (3.11) it is now possible to compute the activation in the output layer L and back propagate to layer $L-1$ all the way to the first hidden

layer.
The next step is the compute the elements of the gradient ∇C which are the partial derivatives of all the weights and biases. The partial derivative $\frac{\partial C}{\partial w^l_{jk}}$ is given by

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial z^l_j}{\partial w^l_{jk}} \frac{\partial C}{\partial a^l_j} \frac{\partial a^l_j}{\partial z^l_j} = a^{l-1}_j \delta^l_j \tag{3.12}$$

and describes how much the cost changes depending on the weight. The partial derivative of the $\frac{\partial C}{\partial b^l_j}$ is

$$\frac{\partial C}{\partial b^l_j} = \frac{\partial z^l_j}{\partial b^l_j} \frac{\partial C}{\partial a^l_j} \frac{\partial a^l_j}{\partial z^l_j} = \delta^l_j \tag{3.13}$$

and describes how much the cost changes depending on the bias. The equations (3.9),(3.11), (3.12) and (3.13) are the four *backpropagation* equations needed to compute the gradient.
So with a set of training examples one epoch of training using *stochastic gradient descent* is done as following:

1. **Pick m random training examples**
2. **For each random training example x**
   (a) **Compute activations:** Start by setting the activation of a^{x1} to the training example then compute the activations a^{xl} for all hidden layers by feed forwarding starting from a^{x1}.
   (b) **Compute the δ errors:** Compute the output layer error $\delta^{x,L}$ by using equation (3.9) then $\delta^{x,l}$ for layers $l = L-1, L-2, .., 2$ by back propagating using equation (3.11).
   (c) **Compute partial derivatives:** Compute $\frac{\partial C}{\partial w^l_{jk}}$ and $\frac{\partial C}{\partial b^l_j}$ using equation (3.12) and (3.13).
3. **Gradient descent:** Update the weights and biases according to (3.7) as $w^l = w^l - \frac{\eta}{m} \sum_x (a^{x,l-1}) \delta^{x,l}$ and $b^l = b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$

3.3 Convolutional neural networks

When the input size of neural network is large, as images, it becomes increasingly harder to train and evaluate the network. A way of dealing with this is to use fewer learnable parameters. Instead of having to learn a weight for each individual neuron the weights of a filter are learned. This greatly reduces the computation needed and opens the possibility of deeper networks. A convolutional neural network (CNN) is a type of neural network where the convolution operation is used with these filters. The convolution operation works by sliding a filter, often called kernel, filled with weights over the input, applying a dot product with the parts of the image inside the filter. So if an image with only one colour channel, I

$$I = \begin{bmatrix} a & b & c \\ e & f & g \\ i & j & k \end{bmatrix} \tag{3.14}$$

and a kernel W

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \tag{3.15}$$

are convoluted with each other it will result in

$$C = \begin{bmatrix} aw_{11} + bw_{12} + ew_{21} + fw_{22} & bw_{11} + cw_{12} + fw_{21} + gw_{22} \\ ew_{11} + fw_{12} + iw_{21} + jw_{22} & fw_{11} + gw_{12} + jw_{21} + kw_{22} \end{bmatrix} \tag{3.16}$$

which is now a 2 by 2 matrix. The height or width of the output is determined by

$$Output = \frac{input - W_{size} + 2P}{S} + 1 \tag{3.17}$$

where the input is the height or width of the input, W_{Size} is the kernel size, P is the size of the padding around the input, usually filled with zeroes, that increase the size of the input and S is the stride which is the step size of the kernel.
A simple CNN classifier called LeNet5 [21] is shown below in fig 3.5 that takes an image as input and determines which of the 4 classes the object in the image belongs to. Pooling layers are use the pooling operation to reduce the dimensions of the convolved features but still keeping the most important information. It reduces the size by choosing subsets of numbers. Two common pooling operations are the max pooling which takes the highest values of the subsets and the average pooling which takes the average of the subsets. The full connections seen in the image are neurons that all have weights and biases as seen in previous chapter.

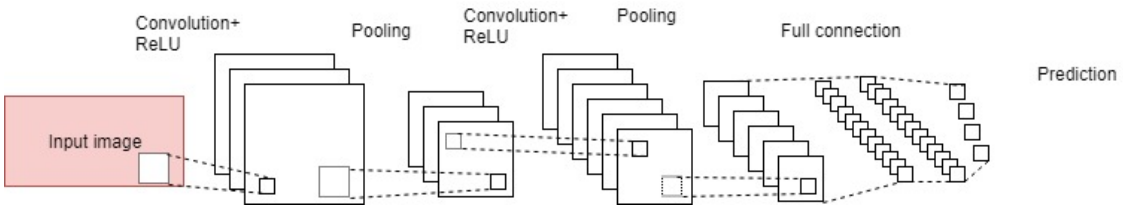


Figure 3.5: The architecture of a simple convolutional neural network

4

Computer vision

This chapter presents ways to find features of images and describes the geometry of cameras and how to use it for triangulation. A visual fiducial system called *apriltags* is also presented.

4.1 Feature extraction

This section defines what an edge is and how they can be extracted from images. An algorithm that finds centre of circles is presented.

4.1.1 Edge detection

An edge is defined as a point where the pixel value (colour) is changing rapidly i.e. the magnitude of the gradient is large. If I denotes an image then the gradient at coordinates (x, y) is

$$\nabla I(x,y) = \begin{pmatrix} I'_x(x,y) \\ I'_y(x,y) \end{pmatrix} \tag{4.1}$$

and the magnitude of that gradient

$$|\nabla I| = \sqrt{(I'_x)^2 + (I'_y)^2}. \tag{4.2}$$

where $I'_x(x,y)$ is the approximated horizontal derivative and $I'_y(x,y)$ the approximated vertical derivative. The orientation of the gradient is

$$\theta = \arctan \frac{I'_y}{I'_x}. \tag{4.3}$$

One way to approximate the gradients is to use the Sobel operator which convolutes two kernels , one over x and one over y, with the image. By sliding the vertical kernel

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

and the horizontal kernel

$$K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

over an image will give the horizontal and vertical approximations as seen in equation (4.4) where I denotes an image and $*$ is the convolution operator.

$$\begin{aligned} I'_x &\approx K_x * I \\ I'_y &\approx K_y * I. \end{aligned} \tag{4.4}$$

The Sobel operator is computationally cheap but the edges found can be several pixels thick.

One way to refine the edges is to use the *Canny* edge detector [22] which is a edge detection algorithm that uses the Sobel operator but only finds the strongest edges. The detector works as following:

1. **Remove noise:** Typically using a Gaussian filter.
2. **Find gradient magnitude and orientation:** Using horizontal and vertical Sobel operator.
3. **Non maximum suppression:** Remove pixels which are not considered an edge. Every edge is compared to its neighbours edges in the gradient orientation. If the edge is not the local maximum it is removed. This makes the the edges thinner.
4. **Hysteresis thresholding:** Two threshold values are chosen. Pixels with a magnitude above the higher threshold are considered strong edges and those below non-edges. Pixels that are in between the threshold and connected with pixels that are above the higher threshold are also considered strong edges.

So by using the Canny edge detector only the strongest edges of the image is returned.

4.1.2 Circle recognition

One way to detect circles in images is to use the circular Hough transform [23].

If the radius of the circle is known the centre of the circle can be found in the following way:

1. Create an empty array of the same size as the image
2. Compute the all edges of the image
3. For all found edge pixels:
   (a) Place a circle with the same radius as the known circle.
   (b) Increase the value in the empty array for border pixels of the placed circle.
4. High values in the accumulator array correspond the centres of circles.

Figure 4.1 shows how the the centre point is detected for the blue circle using 4 edge points. The red circles are where the value of the array is increased.

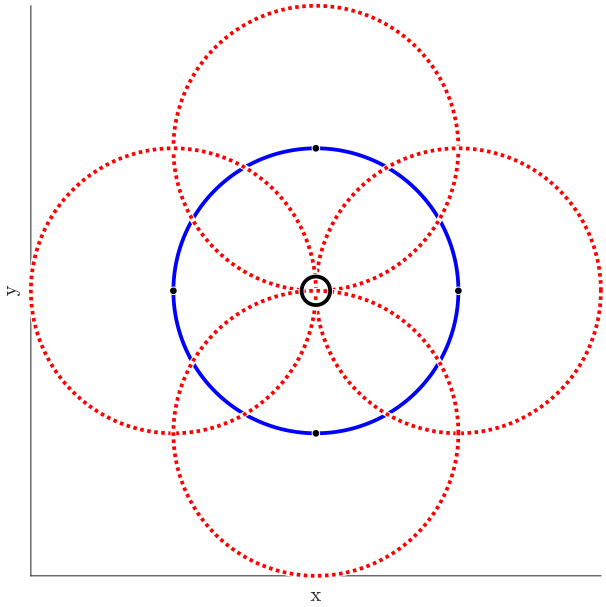


Figure 4.1: The accumulator seen at 4 edge pixels. The border pixels of the accumulator each cast one vote. The centre of the circle is the point where most votes has been cast.

4.2 Camera geometry

To describe the mathematical relationship between 3D coordinates of a point on an object and the projection of this point on the image plane of pinhole camera, the pinhole camera model is used. The pinhole camera model relates image coordinates (x, y) to global 3D coordinates (X, Y, Z)

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.5}$$

where

$$P = K[R|t] \tag{4.6}$$

is the projection matrix consisting of the intrinsic and extrinsic camera matrices. The intrinsic camera matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

describes the internal parameters of the camera and consists of the focal lengths f_x and f_y in pixel units and the principal point c_x and c_y. The extrinsic matrix $[R|t]$

describes the rotation and translation of the camera. The pinhole camera model is shown below in figure 4.2

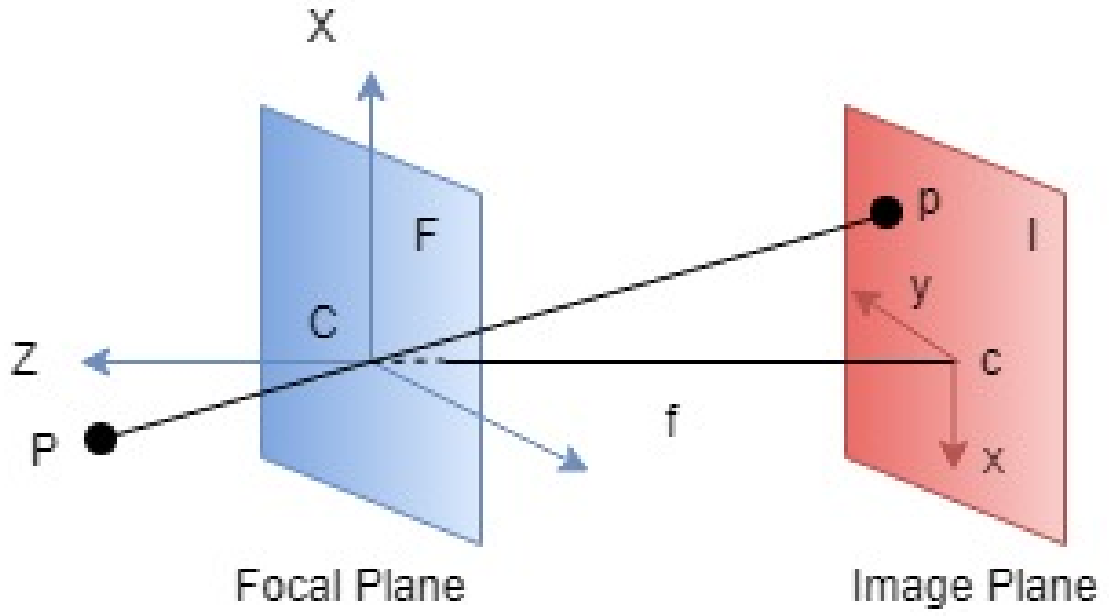


Figure 4.2: The pinhole camera model consists of two planes, focal plane and image plane. The point C is called optical centre, c is the principal point. The distance between the optical centre and principal point is the focal length f. p is an image point which is corresponding to a world point P.

4.2.1 Triangulation

To know the depth of an object inside a bin, the image pixels need to be computed. The calculation is done using pinhole equation (4.5). If P is rewritten as

$$P_n = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

then the camera equation can be reformulated as

$$\begin{bmatrix} a_1 & a_2 & a_3 & -x \\ b_1 & b_2 & b_3 & -y \\ c_1 & c_2 & c_3 & -1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ \lambda \end{bmatrix} = \begin{bmatrix} -a_4 \\ -b_4 \\ -c_4 \end{bmatrix}. \tag{4.8}$$

This gives a system with 4 unknown parameters and only 3 equations. By using image points from two cameras of the same global point it is possible to combine two systems together as:

$$\underbrace{\begin{bmatrix} a_1 & a_2 & a_3 & -x & 0 \\ b_1 & b_2 & b_3 & -y & 0 \\ c_1 & c_2 & c_3 & -1 & 0 \\ \tilde{a_1} & \tilde{a_2} & \tilde{a_3} & 0 & -\tilde{x} \\ \tilde{b_1} & \tilde{b_2} & \tilde{b_3} & 0 & -\tilde{y} \\ \tilde{c_1} & \tilde{c_2} & \tilde{c_3} & 0 & -1 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ \lambda \\ \tilde{\lambda} \end{bmatrix}}_{\theta} = \underbrace{\begin{bmatrix} -a_4 \\ -b_4 \\ -c_4 \\ -\tilde{a_4} \\ -\tilde{b_4} \\ -\tilde{c_4} \end{bmatrix}}_{v}. \tag{4.9}$$

This gives a system with five unknown parameters and six equations. The unknown parameters θ can now be solved by first removing one row of the system and then left dividing matrix M giving the equation:

$$\theta = M^{-1}v. \tag{4.10}$$

By solving (4.10) the 3 global coordinates and the two λ scaling factors

4.3 AprilTags

AprilTag [24] is a visual fiducial system that can be detected and recognised by a camera to give special information for different tags. The tags gives information of the position and orientation and contains an unique id. Unlike other 2D barcodes, AprilTag can be detected even if the captured image is at low resolution and also can be detected at longer ranges than other 2D barcodes like QR.
In the captured image, the tag is recognised by detecting a four sided region in the image which is known as quads. Detecting these quads happen through the following steps:

- The lines in the image are detected by computing the gradient magnitude and direction.
- After calculating the line segment for each component, the sequence of line segments that form a four sided shape is to be found. The AprilTag approach is based on a recursive depth-first search to find the four lines which form a four sided shape, the quad.
- A matrix that transform points from the center of AprilTag to x and y image coordinates is called homography matrix is calculated using the Direct Linear Transform (DLT) algorithm.

Different AprilTags are shown below in figure 4.3

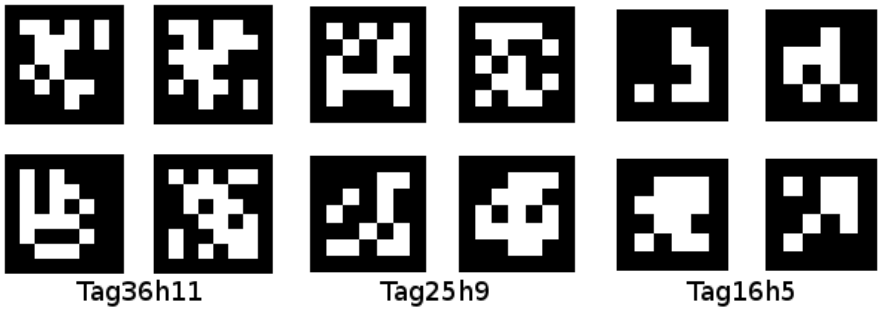


Figure 4.3: Different AprilTag Families

When an AprilTag is scanned by a camera, it gives different information that are used while performing picking and placing process. The centre and corners coordinates in pixels can be generated when a AprilTag is decoded, also the family and the ID can be known. Below in figure 4.4 is an example of how AprilTag is read by a web cam and how data are generated from it.

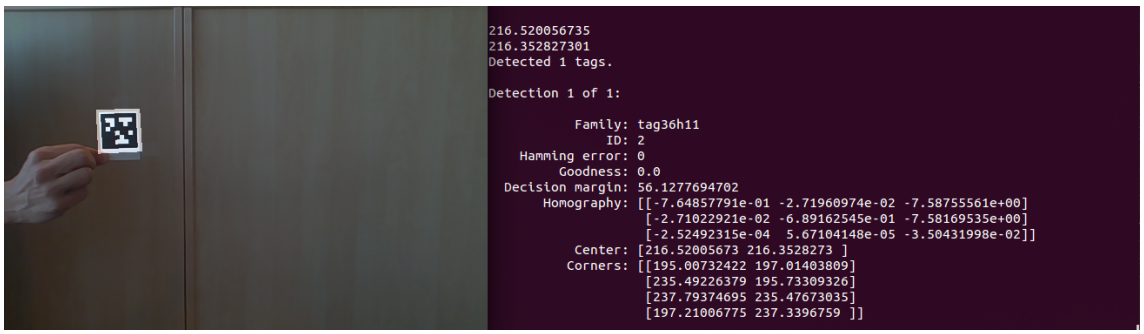


Figure 4.4: Different examples of AprilTags.

5

Determining the best gripping points

To find the best gripping point for the four different objects two ways are used. For the two oil pipes and the pickup a neural network is used and for the oil filter a circle detection algorithm is used. This chapter first presents the developed neural network, how it was created and how it finds the grasping points, and then how to find the grasping points for oil filters. When the points have been found computation of the depth of the points is discussed.

5.1 The neural network

The created network is a fully convolutional network with 7 layers and seen in figure 5.1. The chosen structure of the network is based on two human pose estimation papers [25], [26]. Their network computes heat maps for body limb key points. Our network should compute heat maps for the best graspable points of parts. By using a concatenation between hidden layer 1 and 5, often called skip connection, the network will look for both high and low level features. It is important that the networks learn to detect high level features such as complete components, parts of the component, washers for the oil pipes, but also low level features such as edges, intersections and different colour reflections. The number of filters and filter sizes were chosen manually by trying different compositions.

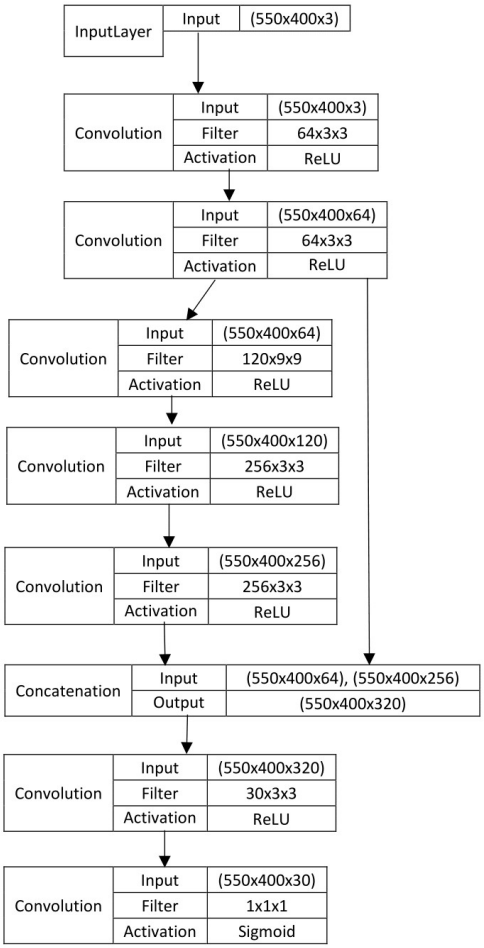


Figure 5.1: The created network

Three networks of the structure shown was created and implemented using the open source python library *Keras* [27] running on top of *Tensorflow* [28]. The training was done using an optimiser called *ADAM* which is gradient based optimiser that uses adaptive learning rates for all the parameters [29] and mean square error cost function, seen in equation (3.3).
To train the network training images had to be created. 3200 images of the u shaped pipe in different positions, different orientations, the number of pipes and lighting conditions was taken. For each of these images a corresponding training target was created. Each target describes what the network should output for its corresponding input and the true output y in the cost function (3.3). A training example for the u shaped pipe can be seen in figure 5.2 and is an image of the same width and height where all the pixels that are good points for picking are set to 1 and the rest are set to 0.

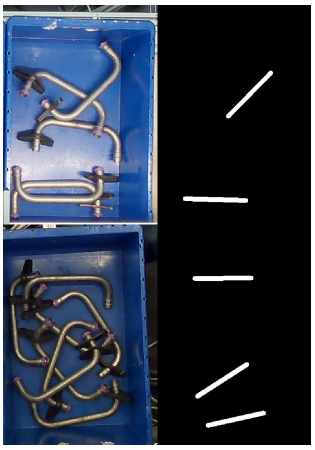


Figure 5.2: Two training images for the u shaped oil pipe.

For the other oil pipe only 1500 training images were created. One training example can be seen in figure 5.3

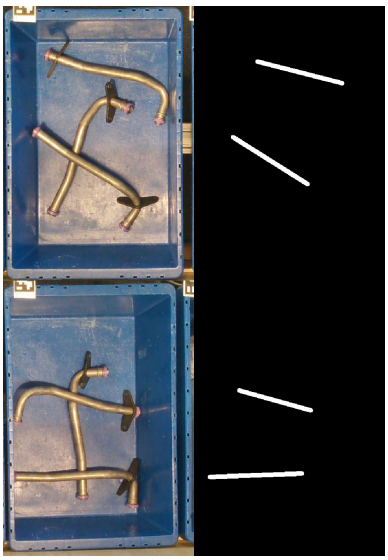


Figure 5.3: Two training examples for the oil pipe.

For the pickup 2000 images were taken and the target image created for each of these. One example can be seen in figure 5.4.

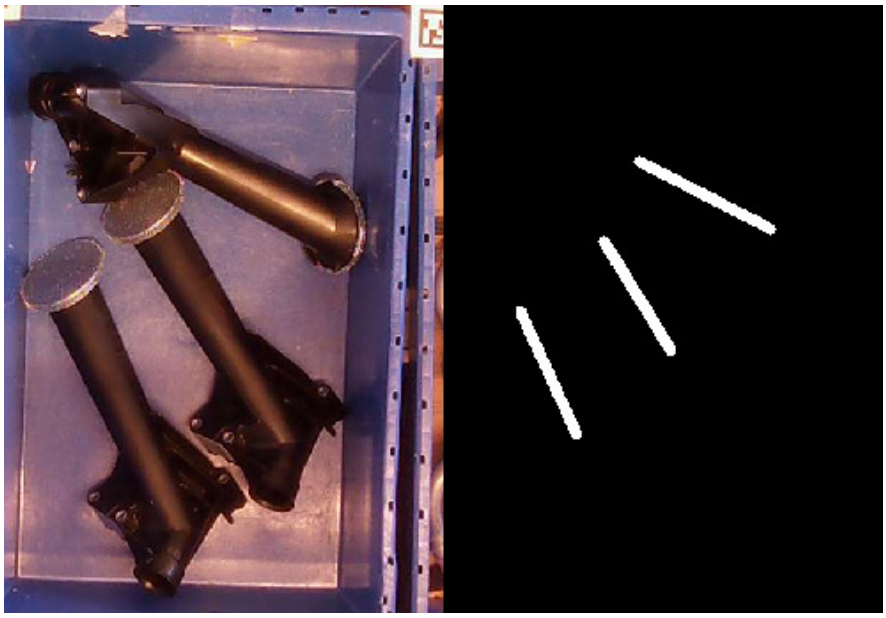


Figure 5.4: A training example for the pickup.

It takes a lot of time to create training data since the parts need to be moved around and the target images manually annotated. Since it is important that the networks learn a general solution it is important that the training data set is large to reduce the risk of the networks becoming over fitted. To improve the size of training data, data augmentation is used. By flipping and shifting the training images randomly the variety of images are increased. Figure 5.5 shows one training example that has been randomly flipped and shifted, the target data have to be flipped and shifted in the same way as the input.

(a) A training input that has been flipped and shifted

(b) A training target that has been flipped and shifted

Figure 5.5: One training example that has been flipped and shifted.

When the training is done the weights and biases of the different filters (kernels) are adjusted. Because of this, the filters are looking for different things in the input and thereafter make the activation functions activate differently. It can therefore be of use to inspect the activation layers when an image is input to see what the network is looking. Figure 5.6 shows the activation for four out of 64 filters in the second hidden layer of the network trained for u shaped pipes. The first activation is not looking for the pipe while the second and third have a very high activation for it. They are also activating differently on the background.

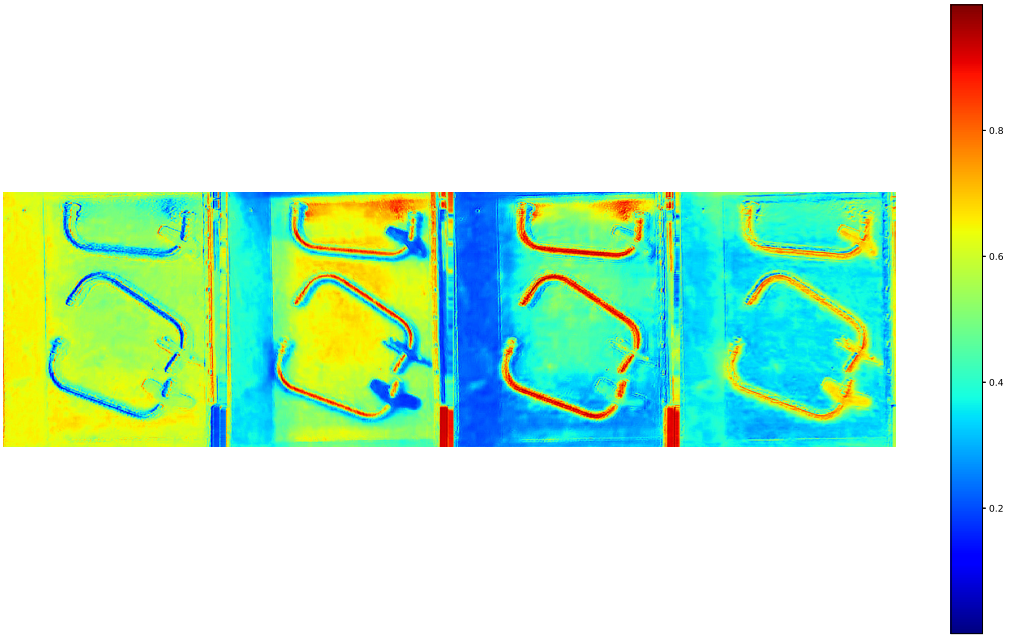


Figure 5.6: The activation's from 4 filters in the second hidden layer showing how the filters activates differently on the input.

The activation's for layer 5, figure 3.2, looks similar to the previous shown layer. Here the edges and colours are also detected

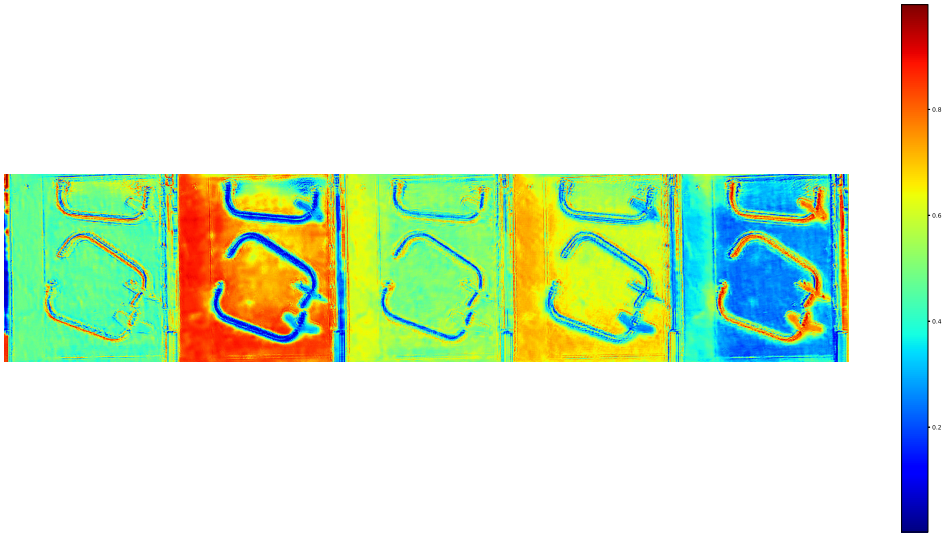


Figure 5.7: Some activation's from hidden layer 5.

Figure 5.8 shows 9 out of 30 activation's for the second last layer. Here

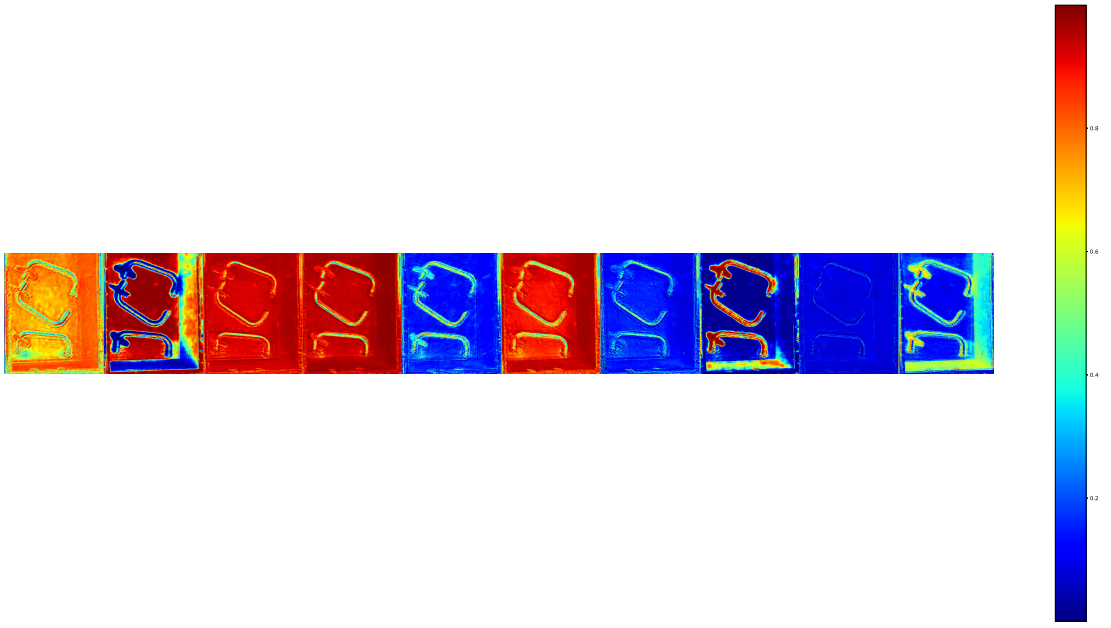


Figure 5.8: Some activation's from hidden layer 6

So from these activation layers it can be seen that the network have learned to detect the parts, the edges of the parts, the colours of the parts and background.

5.2 Finding the points

The grasping points of the objects are computed in two different ways; one way for the oil filter and another for the other parts. The oil filter has a shape that is very easily detected and are always placed in the same height since they are not placed on top of each other, because of these reasons no neural network is used for them. To determine the grasping point of the other parts a CNN is used together with some computer vision techniques.

5.2.1 Detection using CNN

To find the grasping points of the oil suction pipes and the pickup pipe a trained CNN is used. The network is fed with two images, one for each camera, with 3 colour channels. The network then outputs two binary images (channel size 1). The pixels of these images have a float value between 0 and 1 where 1 describes the highest probability. Each of the pixels $I_{x,y}$ are then set to 0 or 1 according to

$$I_{x,y} = \begin{cases} 1, & \text{if} I_{x,y} > threshold \\ 0, & \text{otherwise} \end{cases}$$

Because of the input images looking slightly different the network gives different output. For triangulation to give the correct position the points found in the two images must be the same point. This is solved by first computing the largest contour and centre point for the left image. The centre point is then converted to the right image. By only looking for contours in a area around this points makes sure that the found contours are the same. This procedure can be seen in figure 5.9. The average total time of the procedure is 2.8 seconds of which the network prediction is 1.2 seconds.

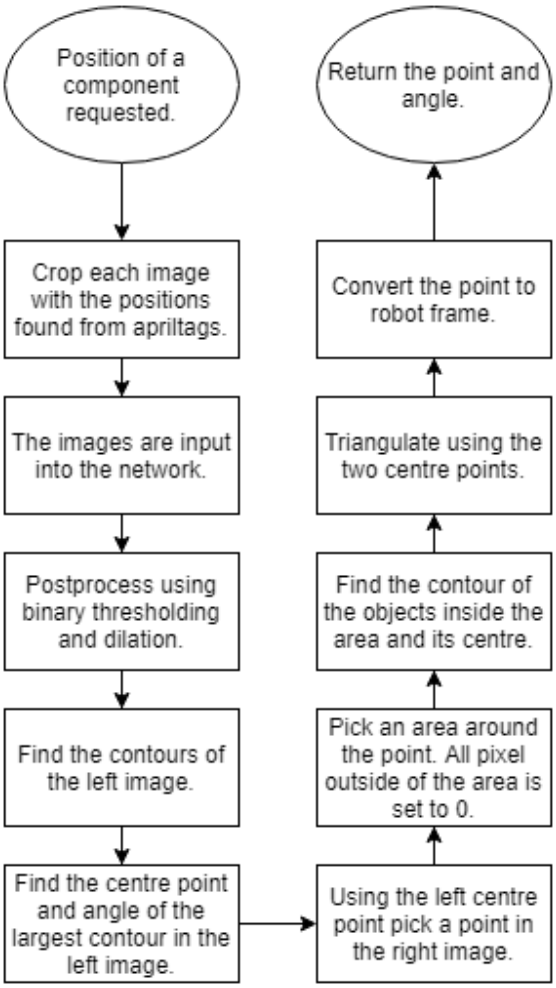


Figure 5.9: The process for finding the gripping points

5.2.1.1 U shaped oil pipe

The u shaped oil suction pipes have such a shape that they can easily cling on to each other. It is therefore important that the correct pipe is picked. A case where three pipes are in the box can be seen below.

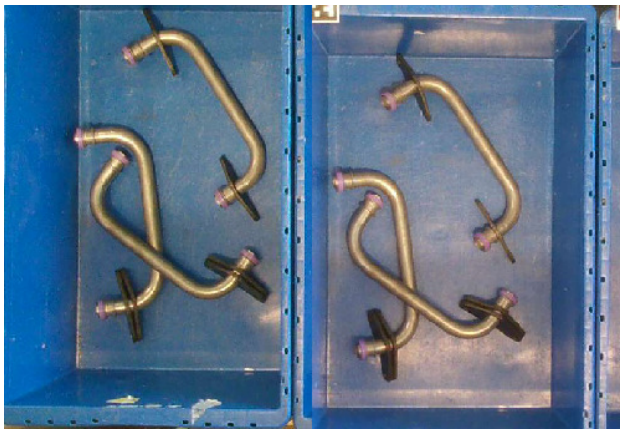


Figure 5.10: One image from each camera that have been cropped. The left and right image is used as input for the network.

The prediction of the network is seen below in figure 5.11. It can be seen that at the intersections the pixels are at or near 0.

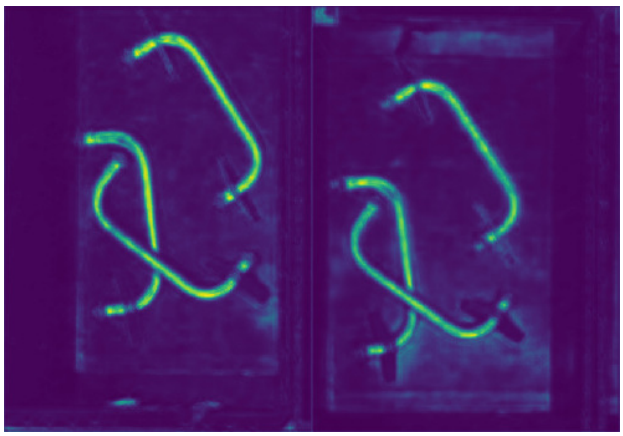


Figure 5.11: The prediction done for images taken with left and right camera.

The thresholded and dilated images seen in figure 5.12. It shows more clearly that the pipes are not connected.

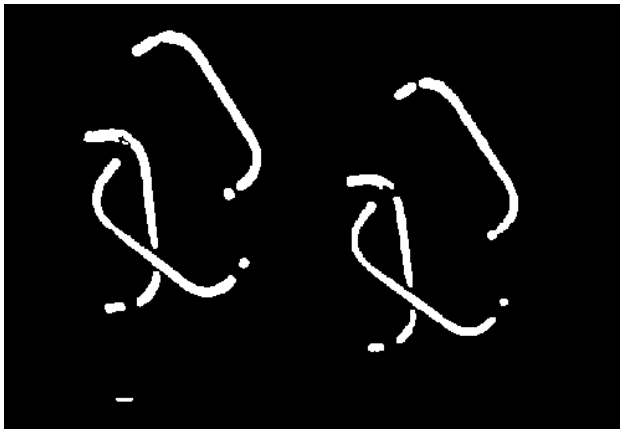


Figure 5.12: The segmented images when binary thresholding and dilation have been applied to the predictions.

The contour is computed for each shape and the longest one chosen. The centre point of this contour is then computed. Because of the two orthogonal parts of the pipe the centre point found has been moved outwards. By only using the contour of an area around this point the new refined point can be moved inwards. These two contours and centre points can be seen below in figure 5.13.

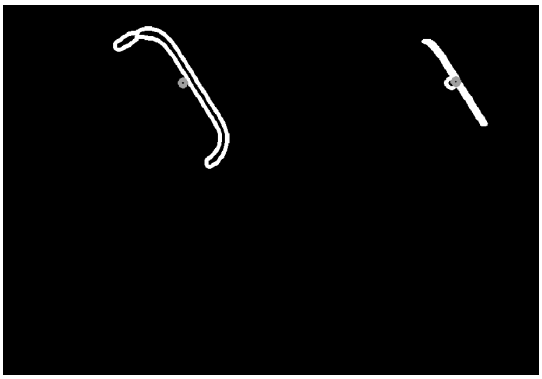


Figure 5.13: Left: The largest contour and its centre point. Right: The refined centre point

Using the found point in the left image it is converted to a similar point in the right image. Everything outside of this area is set to 0. The largest contour and centre point is then found as before.

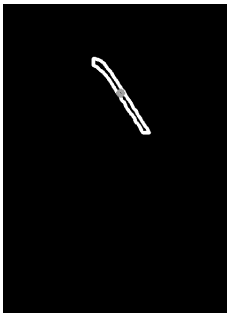


Figure 5.14: The centre point of the pipe in the right image.

For the picking to work in reality it must be able to handle boxes which contain a lot of parts as seen below in figure 5.15.

Figure 5.15: The input images for a case with many parts.

The prediction below shows results looking similar to the case with only three parts.

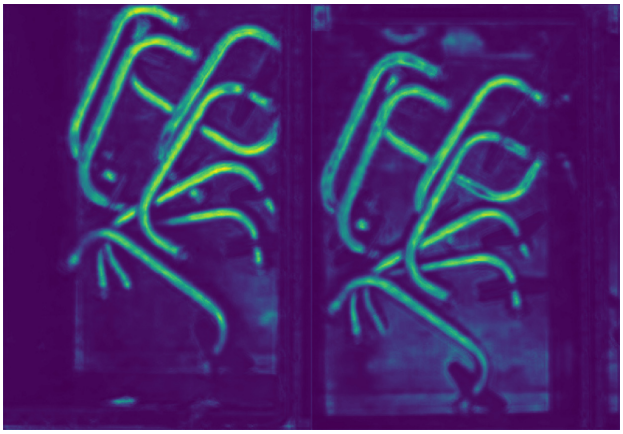


Figure 5.16: The prediction for the case with many parts.

The thresholded and dilated images shown in figure 5.17 also looks as the previous in figure 5.12 but here a higher threshold value and lower dilation size were used. If the same values used in figure 5.12 were used here some segmented parts might merge with each other and become one. This will cause the contour in next step to be wrong.

Figure 5.17: The segmented images for the case with many parts.

The found longest contour seen in figure 5.18 only has one of the two small orthogonal parts of the pipe. This have moved the centre point a bit downwards, away from the middle of the pipe.

Figure 5.18: The centre point of the pipe in the left image and the refined point.

The right centre point is seen below in figure 5.19.

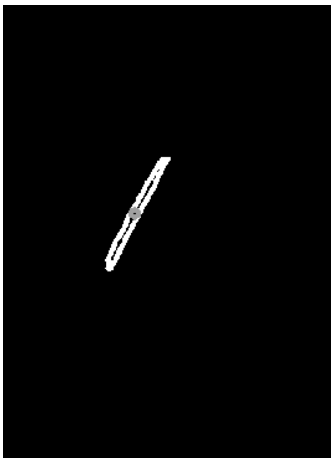


Figure 5.19: The centre point of the pipe in the right image.

This shows that the network have no trouble distinguishing between images with few and many u shaped pipes however the used threshold value had to be higher and the dilation size lower than for the case with few parts. A problem with a high threshold value is that some relevant parts of the prediction is discarded and

causes the segmented images to be more uneven. If the best pickable pipe will have a washer placed on the long part of the pipe the prediction of that area will be low. This will make the contour be divided at that area and it will no longer be the longest contour. Because of this a centre point will be chosen for some other pipe that might not be pickable.

5.2.1.2 Oil pipe

This oil pipe has similar visual properties as the previous u shaped. The largest difference is that this pipe only has one orthogonal part and one washer. The input images of a case with few parts can be seen in figure 5.20.

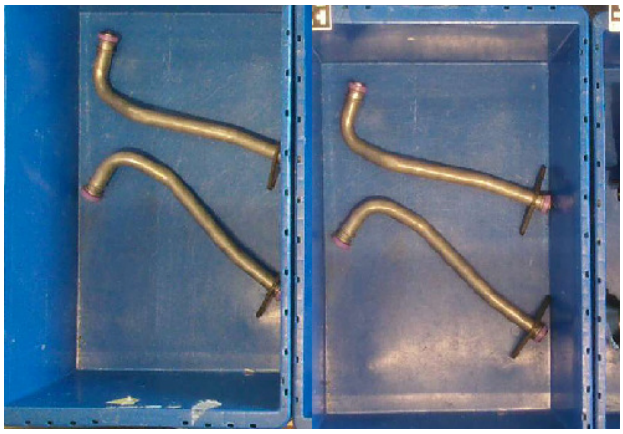


Figure 5.20: The left and right input images of the oil pipe.

From the prediction below 5.21 it can clearly be seen for the bend of the top pipe the network is setting the output low as it should based on the training target but not for the lower pipe. This is probably because of the lower brightness in that area.

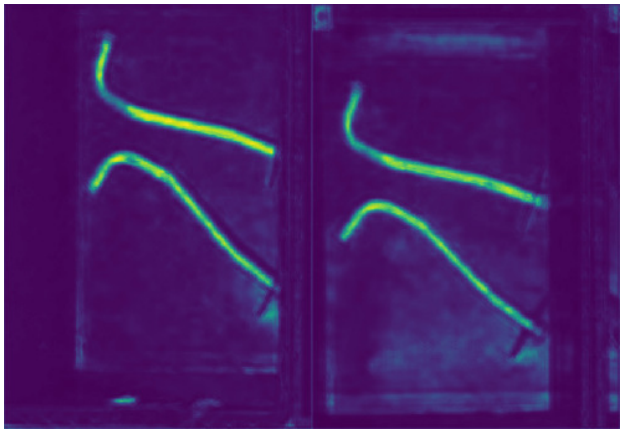


Figure 5.21: The two predictions of the oil pipes.

The segmented image in figure 5.22 will because of the lower output contain 3 large segments instead of 2.

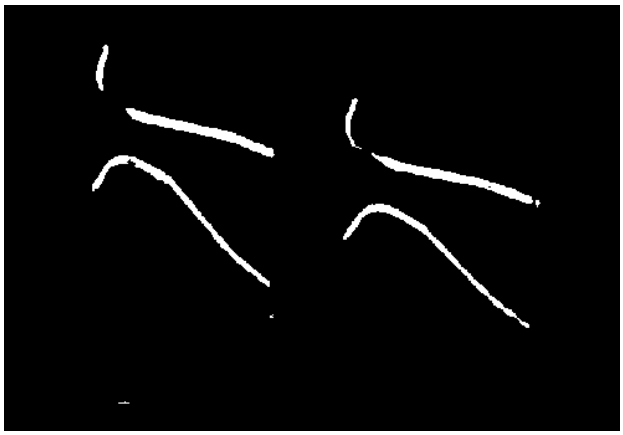


Figure 5.22: The two segmented images of the oil pipe.

The largest contour will be of the pipe that was not split into two segmented parts, seen in figure 5.23.

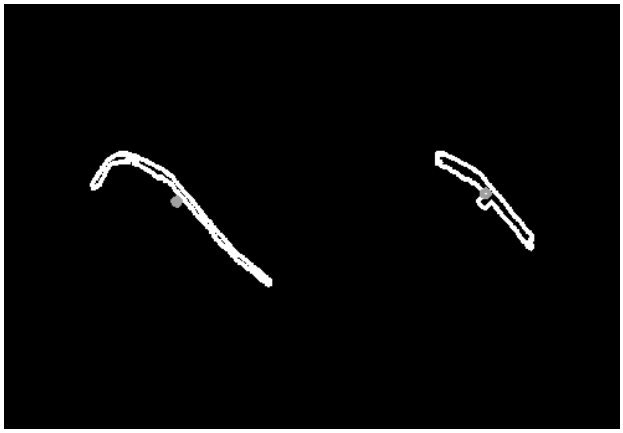


Figure 5.23: The full contour of the left image with the centre point and the smaller one that fits in the chosen area with its refined centre point.

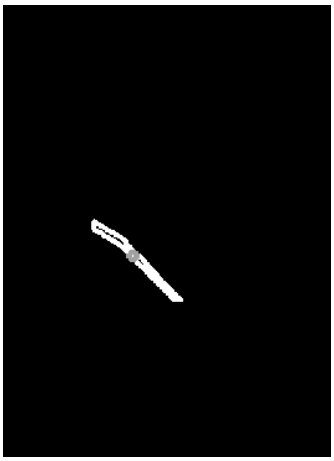


Figure 5.24: The contour of the right image.

Here also a case with several parts in one box is presented. Figure 5.25 shows a case with lots of parts randomly spread out in the boxes.

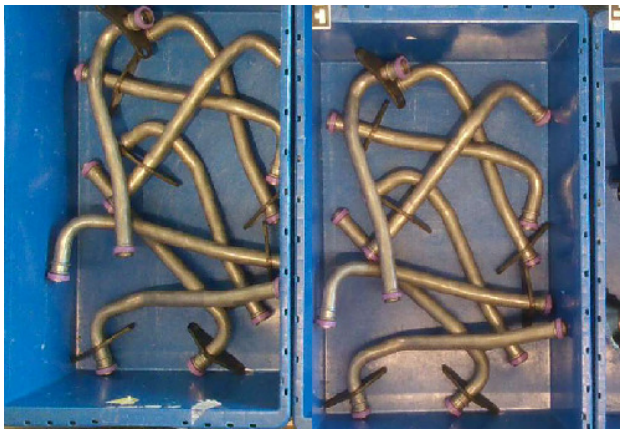


Figure 5.25: The left and right input images with several oil pipes.

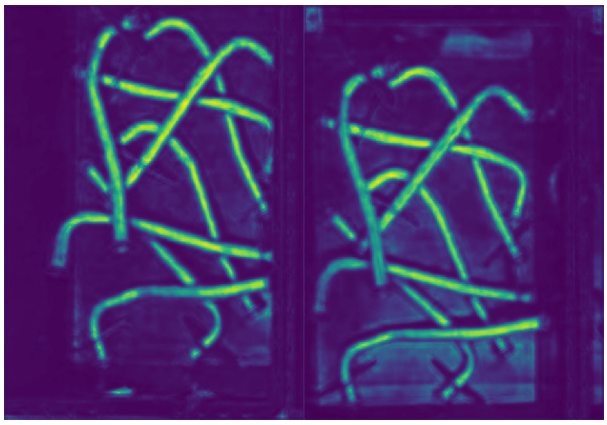


Figure 5.26: The two predictions for the case with several oil pipes.

Here the same threshold values and dilation size were used as for the u shaped oil pipe. This caused some parts in the left image merge with each other. This can be seen in the left image of figure 5.27.

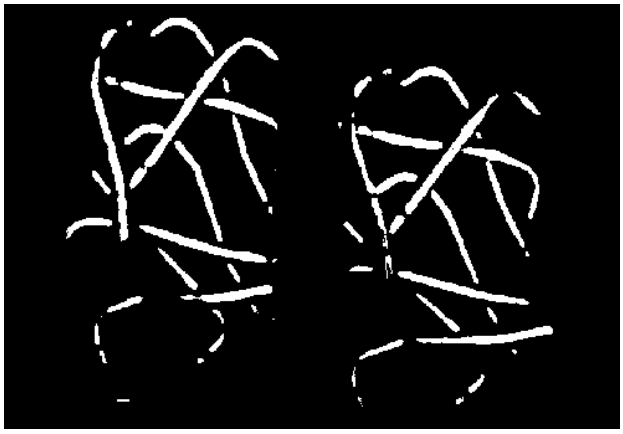


Figure 5.27: The thresholded and dilated image with to large threshold and dilation size.

Because of this the largest contour found was of two pipes. As seen in figure 5.28 the point found was still good but this is not always the case.

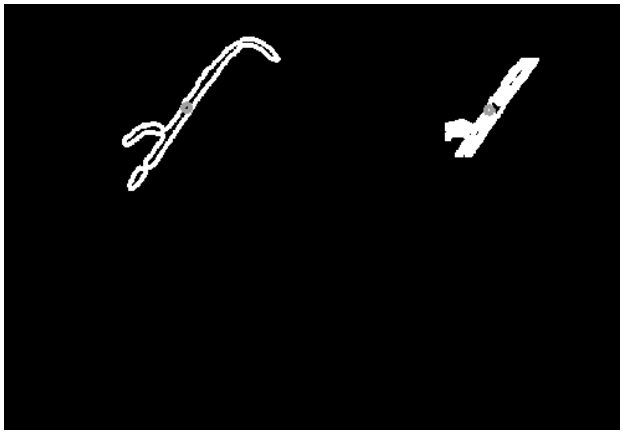


Figure 5.28: The threshold value and dilation size have made the pipes some parts of two oil pipes become connected with each other.

As seen in the right image 5.29 the contour is only of one pipe.

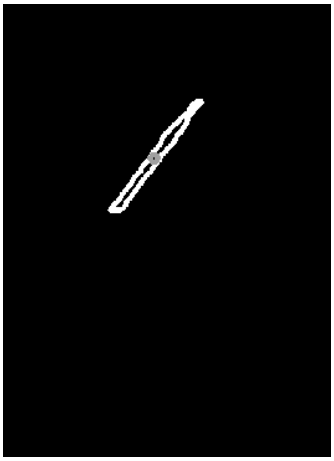


Figure 5.29: The contour and centre point of the right image.

This was a case where the threshold value and dilation size were chosen wrongly. If the threshold value had been chosen to be higher the contours would have been correct.

5.2.1.3 Pickup pipe

The pickup pipe has a complex geometric shape, there is no symmetry. To be picked in an efficient way, it should be picked from a point where it does not tilt so much when picked by the gripper; in order not to fall. The black colour of the pickup pipe can cause problems when pipes are next to each other. In the images the black parts float together merging the pipes into one object with a different geometry. This can result in an incorrect point determination.

Figure 5.30: The input of the pickup pipes for each camera.

The prediction, figure 5.31 shows that the network gives a high output for bright areas of the parts in the input image. It also gives a much lower output for the right image, possibly because of the reflections seen with the left camera is not seen with the right camera.

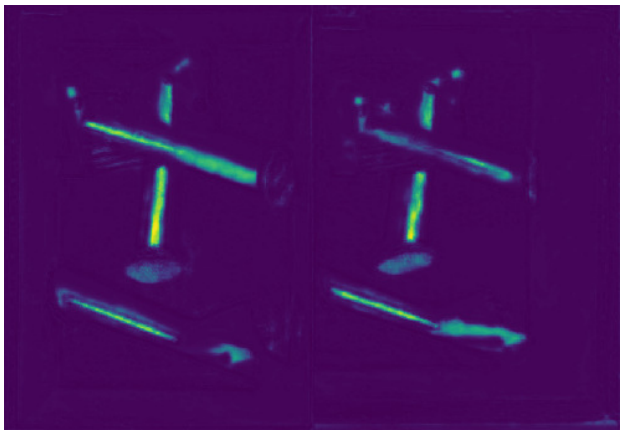


Figure 5.31: The prediction for left and right images.

Because of the low output of the right image, a lower threshold for that one had to be used. Figure 5.32 shows a big difference of the bottom pickup pipe, the found

pick point on the right image is not the same as the found pick point in the left image.

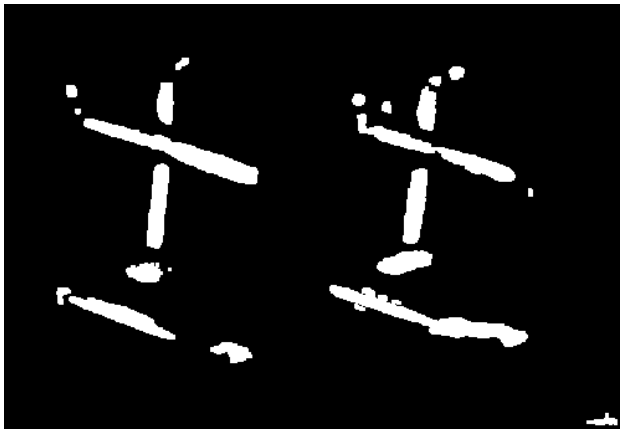


Figure 5.32: The segmented images showing a big difference between the images.

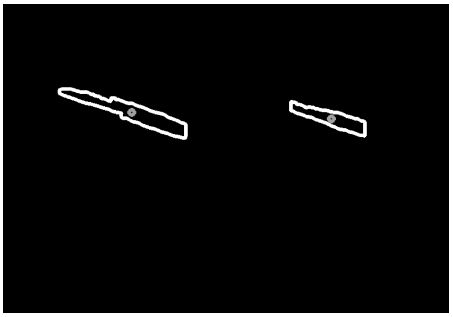


Figure 5.33: The longest contours of left and right image with their centre points.

5.2.2 Oil filter detection

The shape of the oil filters is circular and the radius of them is known which means that the centre of them can easily be detected using the Hough circle detection [23]. The time for computing the grasping point with this algorithm takes an average time of only 0.05 seconds. The Canny edge detection is first applied to the image, returning the edges with the highest gradients then the circles are found of the edges and can be seen in figure 5.34.

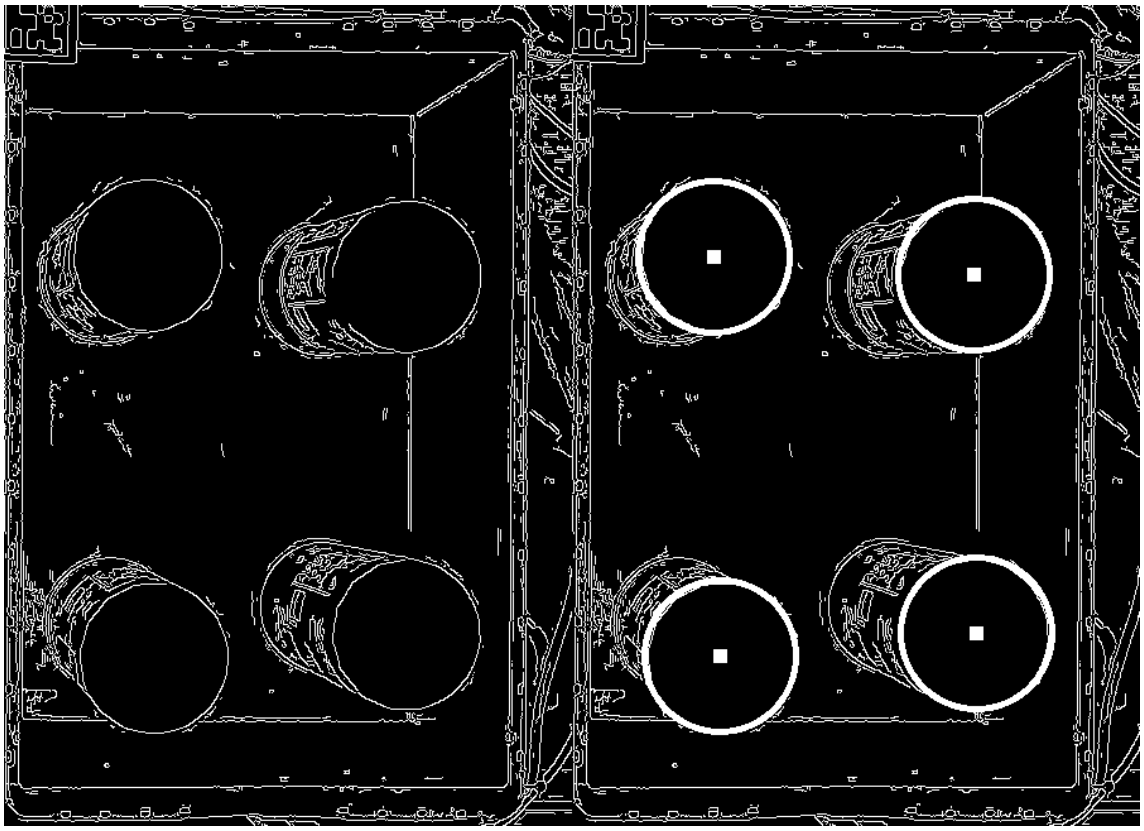


Figure 5.34: The edges of the oil filters after Canny edge detection and the detected circles.

When there are more than one oil filter in the picking box the part which has the largest distance to the other filters is chosen since it should be the easiest one to pick, this can be seen in figure 5.35.

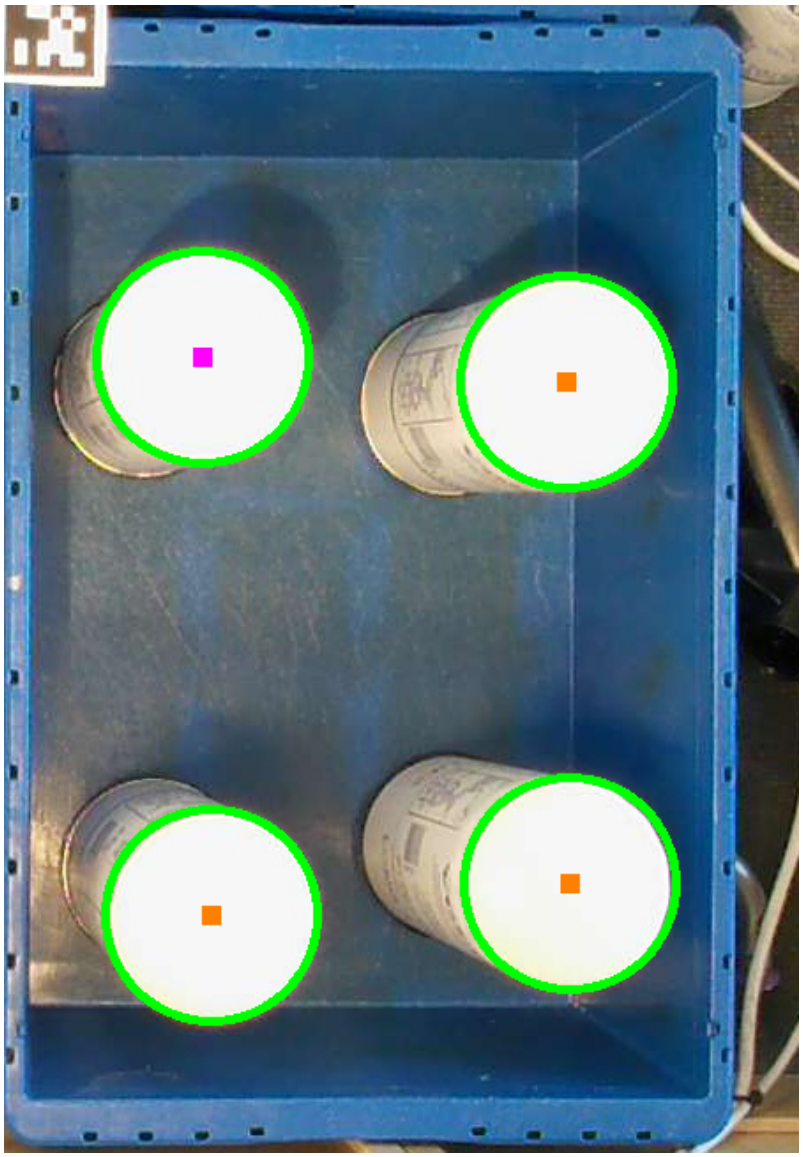


Figure 5.35: The detected centres of the oil filters together with the centre of the oil filter which is most easily picked (purple square).

5.3 Depth estimation

To compute the depth of an object equation (4.10) is used. This means that the image point in both camera frames and the projection matrices P_1 and P_2 are needed. The image points have already been computed in the previous sections but both the intrinsic and extrinsic parameters are still needed for both cameras. To estimate the intrinsic parameters of the camera the camera calibration in the library OpenCV [30] is used, which is based on the algorithm proposed in the paper [31]. The calibration is done in the following way:

1. A flat surface with a chessboard pattern is created.
2. Multiple photos of the pattern in several orientations are taken and can be seen in figure 5.36.

3. The feature points of the photos, the corners of the black and white squares, are detected and can be seen in figure 5.37.

4. The intrinsic parameters and distortion are estimated using the proposed algorithm with the feature points.

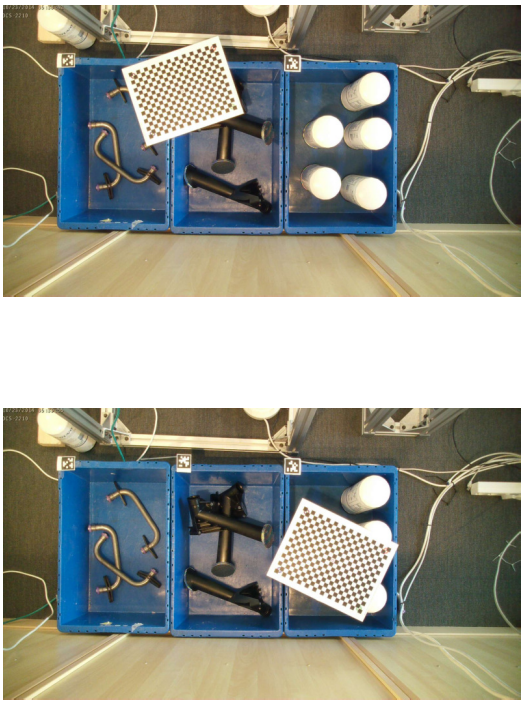


Figure 5.36: Examples of calibration photos with the chessboard pattern in different orientations.

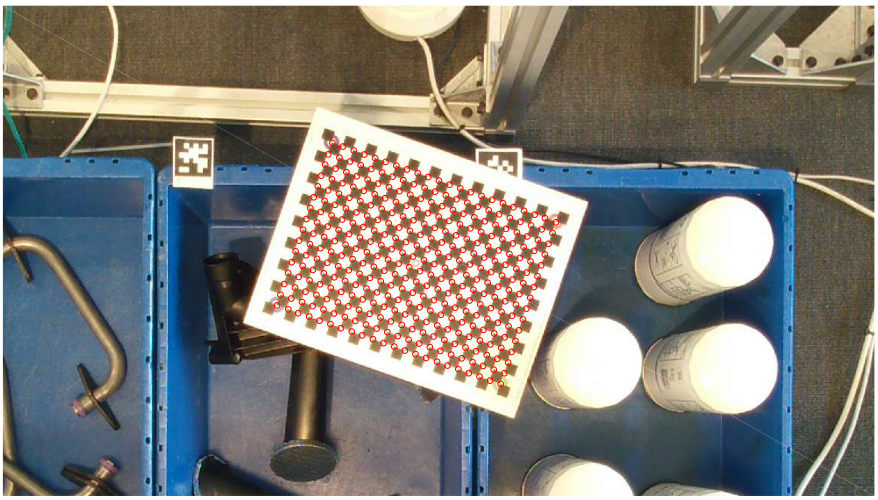


Figure 5.37: The feature points of a chessboard pattern.

The extrinsic matrix for the first camera is just a identity 3 by 3 matrix and a 3 by 1 vector of zeros. The extrinsic matrix for the second camera is estimated by measuring the rotation difference and translation of the second camera. With these two created matrices the projection matrix seen in equation 4.6 can now be created and with this the triangulation can computed using the camera equation 4.5 together with the two previously found points.

6

Picking and Placing Implementation

In this chapter the process of picking different objects from bins and placing them in a specific pose on the MiR is presented. The chapter discusses also AprilTags and how they are used to determine the pose of a specific bin to pick objects from. Picking objects with different geometries is not a simple task for robots, objects can be picked in ways that it makes it hard to place them on the MiR, these (special picking situations) will be discussed and also solutions for them.

6.1 Communication with UR10

In this thesis project, there are several nodes that need to communicate with each other, UR10 is one node of the ROS system. First, the ROS master, UR10 node, and optoforce sensor node are run, in which all information can be shared between different nodes in the system. ROS master manage the communications between publishers, subscribers to different topics. The ROS system that shows different nodes and how the communication in the system happen is shown in figure 6.1

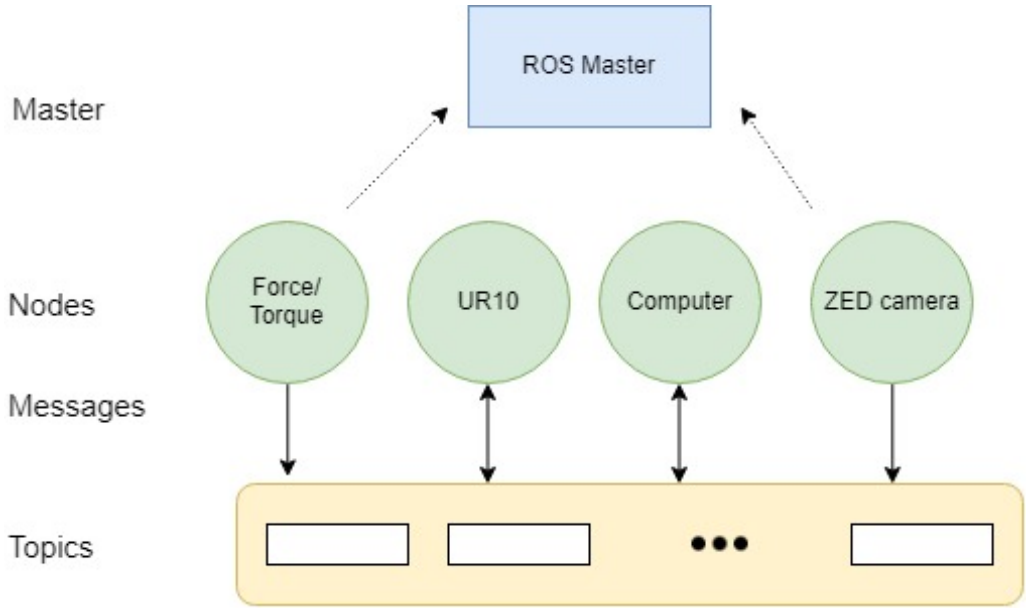


Figure 6.1: ROS system representing nodes used in picking and placing process

As shown above, there are several nodes, for example, force/torque sensor (optoforce) node publishes data to a topic with a specific message type. If data from the optoforce sensor is needed simply the optoforce topic data can be accessed through subscription. Subscriber command can be written as:

```
rospy.Subscriber("/optoforce_0", WrenchStamped
, self.callback_tool)
```

where */optoforce_0* is the topic where force/torque sensor publishes its data, *WrenchStamped* is the message type, and *self.callbacktool* is a function to access specific values from the sensor like force in x direction or torque around y axis. A subscribe process can happen to *Joint_states* in UR10 node to get joints' positions, and also specific position even for joints or the tool can be published to the robot through computer node. The published data from computer to the robot can be written as:

```
self.MovePublisher = rospy.Publisher("/ur_driver/URScript",
String, queue_size=1)
```

and as mentioned before, */urdriver/URScript* is the topic that is published to, in which later, UR10 will subscribe to. String is the message type, and queue_size is to limit the size of messages queue if there are many subscribers, and one of them does receive data slower than others. ZED camera node is mainly used to determine the position of AprilTag placed on MiR, and then place picked objects on specific position by determining fixed offsets in x and y directions. If one node wants to publish data, it notifies the ROS master first, then publish to a topic. If another node wants to subscribe to this topic it also notifies the master, and that is what the two dashed arrows in figure 6.1 shows. At this point both nodes can communicate with each other and share information.
UR10 is controlled through script level. URScript is a programming language used to program the robot with many commands that control velocity and positions of robot joints. Using URScript commands make it easier and more efficient to move the robotic arm to specific position and orientations to pick and place different objects. The following commands in table 6.1 are the most used ones while controlling the arm motion:

Table 6.1: URScript used commands [4]

Command	Description
Movej(q,a,v,t,r)	Moves the arm to joints' positions q with acceleration a, velocity v, time of the move t and blend radius r
Movel(q,a,v,t,r)	Moves the arm to target positions (x,y,z,wx,wy,wz) with acceleration a, velocity v, time of the move t and blend radius r
stopl(a)	Stops the motion of the tool with decreasing acceleration a

6.2 Picking and Placing scenario

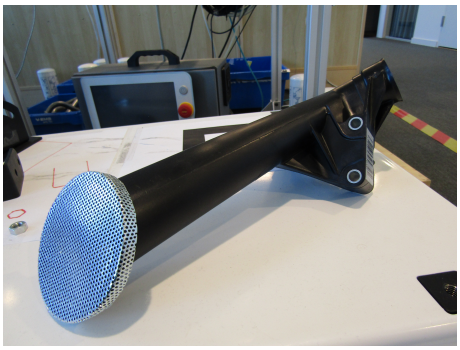
The process can be classified into two parts: Picking and Placing. First, the operator determines all the required objects to be picked, each object has a specific ID as indicated in table 6.2:

Table 6.2: Different objects with their IDs

Object	ID
OilFilter	0
Pickup	1
Oil suction pipe(Utube)	2
Oil suction pipe(Tube)	3

The oil filter and Pickup are shown below in figure 6.2

(a) Oil filter



(b) Pickup

Figure 6.2: Oil filter and pickup

The oil suction pipes are shown in figure 6.3

(a) Oil suction pipe (Utube)

(b) Oil suction pipe(Tube)

Figure 6.3: Two oil suction pipes, the first will be called Utube, and the second tube

For example, if it is required to pick two oil filters and one UTube, the input IDs to the process will be: 0,0, and 2.
When the process starts, an image is taken. By scanning the AprilTags in the captured image, the poses and IDs of bins containing different objects can be known.Now the robot can move to a pre-picking position above the bin that contains first required object. The captured image is cropped, the position and angle of the object to be picked are calculated through inserting the new image through the trained model generated before. The calculated angle is checked to determine how the robot wrist angle will orientate to perfectly pick the objects. There are 4 pre-determined regions addressing different ranges for object orientation as shown in figure 6.4. For example, if the measured angle was 20°, the wrist of the robot will change its orientation to pick the object correctly, in this specific case, the object is in region 0.

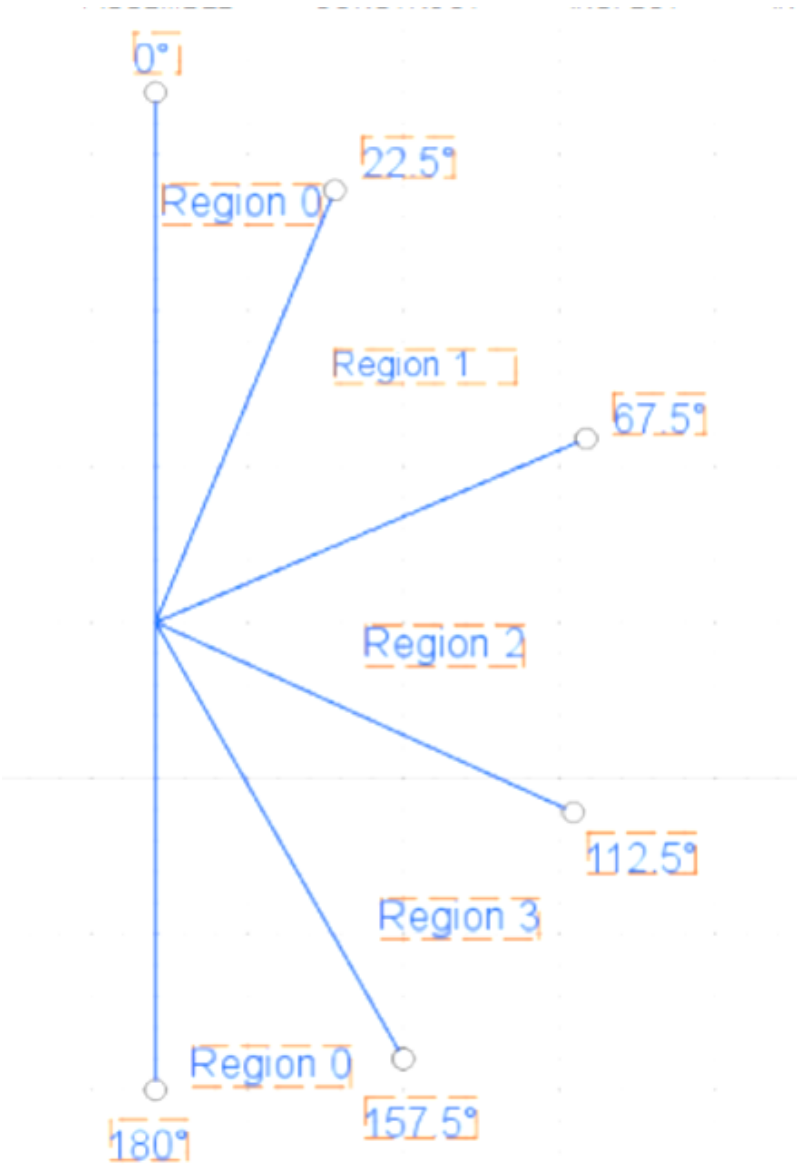


Figure 6.4: Different regions representing different object orientations

The 3 finger gripper moves to the object position, pick it, and move up a bit above the bin to check exactly how the object is picked. The verification process is done using Force/Torque sensor.

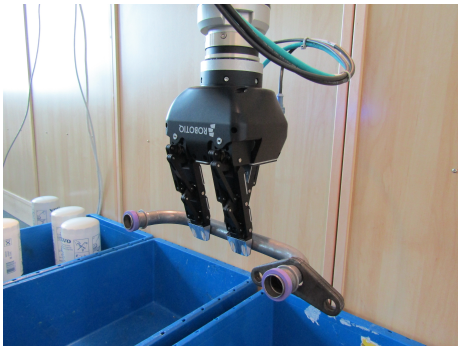
6.2.1 Special picking situations

In Table 6.3 there are three different scenarios that can happen while checking the picked object:

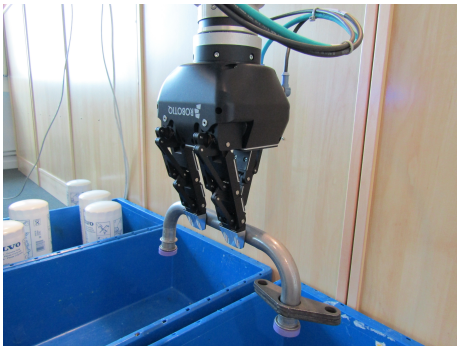
Table 6.3: Different picking scenarios and their responses

Scenario	Response
The object is not picked at all	The robot will move away from camera zone, new image is taken and the process is repeated.
More than one object are picked at the same time	The robot move back to bin post-picking position, leave the objects, move away from camera zone and new image is taken
The object is picked in wrong orientation	The robot move back to bin post-picking position and leave the object,move away from camera zone and new image is taken.

After picking the object, the goal is to place the object at the the end on a specific pose on a table on MiR. Objects can be placed correctly if they are picked correctly. The following figure shows examples of how the Utube is picked in a correct way as shown in figure



(a) Utube is picked and u shape pointing forward



(b) Utube is picked and u shape pointing down

Figure 6.5: 2 examples of picking Utube in a correct way.

In case of picking Utube, tube or a pickup, they may slip from gripper fingers and gripped in a way that they can not be placed correctly on the table, to check if they are picked in a correct orientation or not, a test process is done by moving the the arm to a point above the bin and move to the right, if the Force/Torque sensor measures a change in force measurements in x direction, that means that the object is picked in a wrong way, and if there is no change in force measurements in the

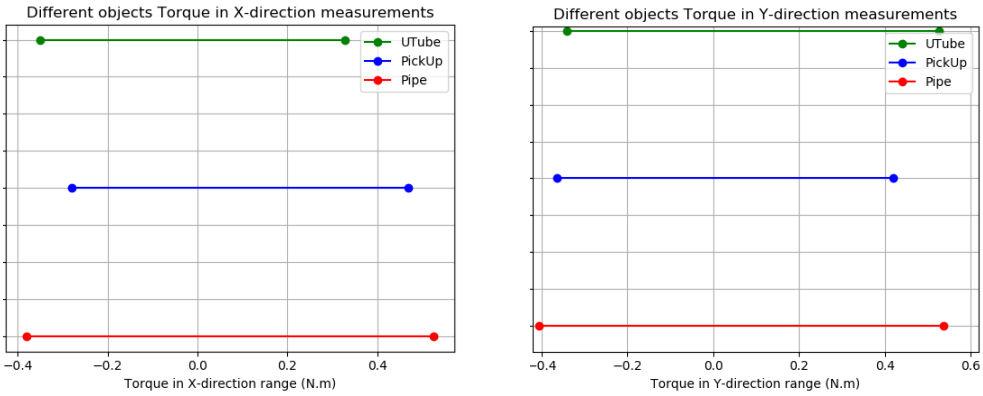
same x direction, the process is continued to place the object. This test process is shown below in figure 6.6

Figure 6.6: While an object (Utube, tube or pickup) is picked, it maybe picked in a wrong way, due to very close measurements value obtained from optoforce sensor, it is not clear how exactly the object is picked, so the check process is executed. The robotic arm goes to a position above the bin and moves in x direction for amount of time, if the object hit the bin, the robot motion stops, and an increase in force in x direction is noticed, then the robot go back to the position above the bin and leave the object. If the object did not hit the bin and the pre-defined amount of time has passed, then the object is picked in a correct way, and will be placed on MiR

Now the picked object is ready to be placed on its specific position on the table put on MiR. Depending on how the object is picked, the orientation of the gripper is changed.

6.2.2 Force/Torque readings

The readings from force/torque sensor measurements are recorded for force in z-direction, torques in x and y directions; since those values are found to give more relevant results that can be used to figure out how are the objects picked so that a proper action should be taken, whether to place the object directly on MiR, or to leave the object if it is picked in a wrong orientation. The graphs showing different measurements for torques are shown below in figure 6.7



(a) Torque range in X-direction **(b)** Torque range in Y-direction

Figure 6.7: Torque range for different objects in XY directions

The forces in Z-direction for different objects are shown in figure 6.8

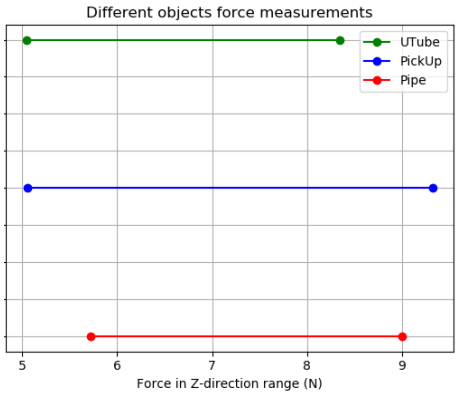


Figure 6.8: Force measurement in Z direction for different objects

6.2.3 Placing

Each object of the four picked ones is placed in a specific position and orientation on the MiR table. The Utube and the tube are placed on special holders as shown in figure 6.9

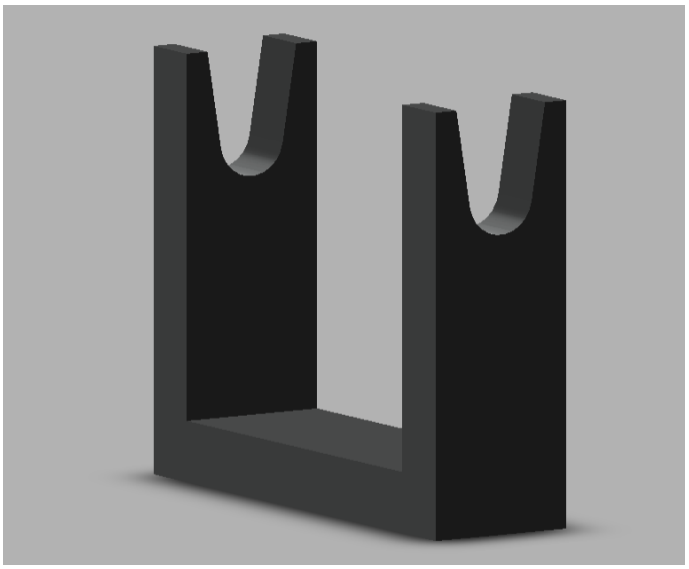


Figure 6.9: CAD design for Utube and tube holder

The oil filters have other holders to be placed on as shown in figure 6.10

Figure 6.10: Oil filter holder

The challenging part here is to make sure that the picked object is correctly placed in the right pose on those holders. By using the force/torque sensor, the data from perfect alignment can be measured and used as a reference. The alignment

force/torque readings of the object on the holder can be measured, if the readings do not match with the reference one, the gripper will re-position the object according to the sensor recorded data. For example,
When all input objects are picked and placed, MiR leaves to starting point. The flowchart of the process is shown below in figure 6.11

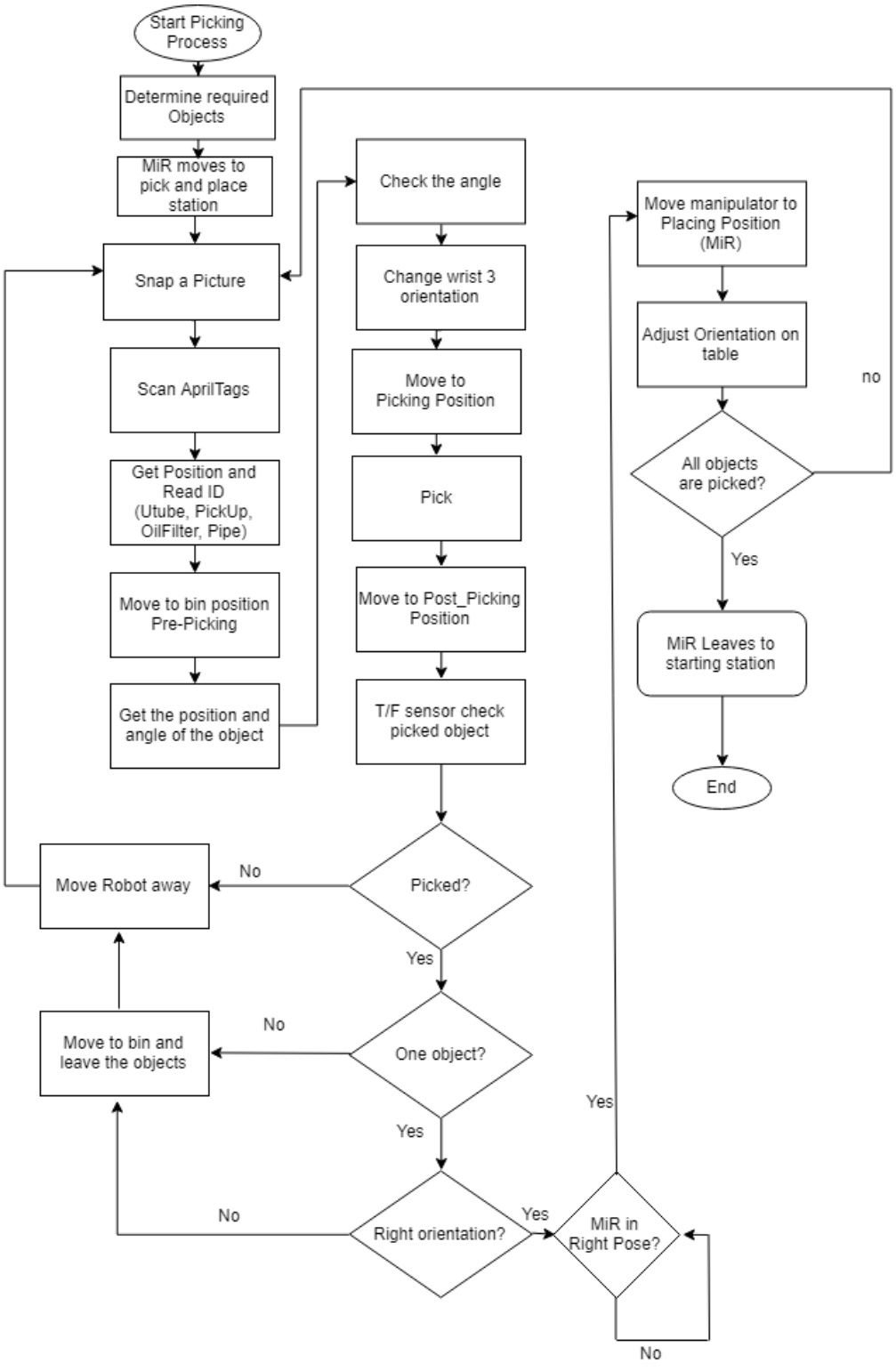


Figure 6.11: Picking and Placing Scenario

6.3 Stereo camera frame and frames transformation

The UR-10 robot has several frames for different joints, ROS provides several packages that can describe the 3D model of the robot, its links, joints, and the environment around it. One important package that ROS has, is called: robot_model. This package has a package inside called URDF. URDF stands for Unified Robot Description Format. The robot can be described using URDF if only it has a tree-structure, which means that the links are rigid and connected together using joints. Since this project aims to provide a solution to a specific task with special setup, the stereo cameras which are used to capture components images are not defined in the robot environment.
A new frame for stereo cameras is created in a point between the two stereo ones using RVIZ as shown in figure 6.12:

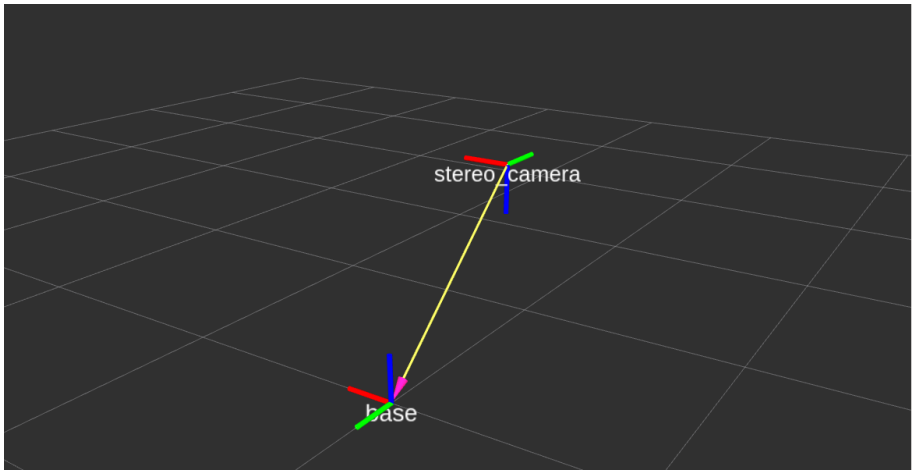


Figure 6.12: A stereo camera frame is created using RVIZ

The position between the base frame of the robot and the stereo camera frame is measured manually and found to be (x=0, y=-805 mm, z= 706 mm), the minus sign in y means the negative y- axis of the stereo camera frame. For the orientation between them, first the xyz for each frame was checked to know where is each axis is. It was found as shown below in figure 6.13:

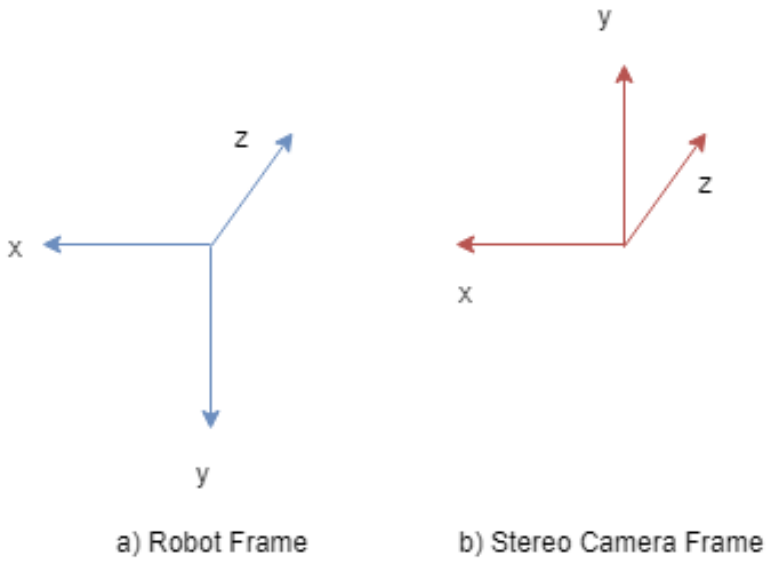


Figure 6.13: a) The xyz coordinates of the robot, b) The xyz coordinates in camera frame

from the figure above, that means that for a point in the stereo camera frame to be transformed into robot base frame, the transformation matrix will be calculated through transformation matrix in 6.1:

$$T_{stereo}^{robot} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\pi & -sin\pi & -805 \\ 0 & sin\pi & cos\pi & 706 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

The tf package in ROS can be used to generate what is called a tf tree. tf tree is a graphical representation for all frames included in the system and their relation to each other as shown below in figure 6.14

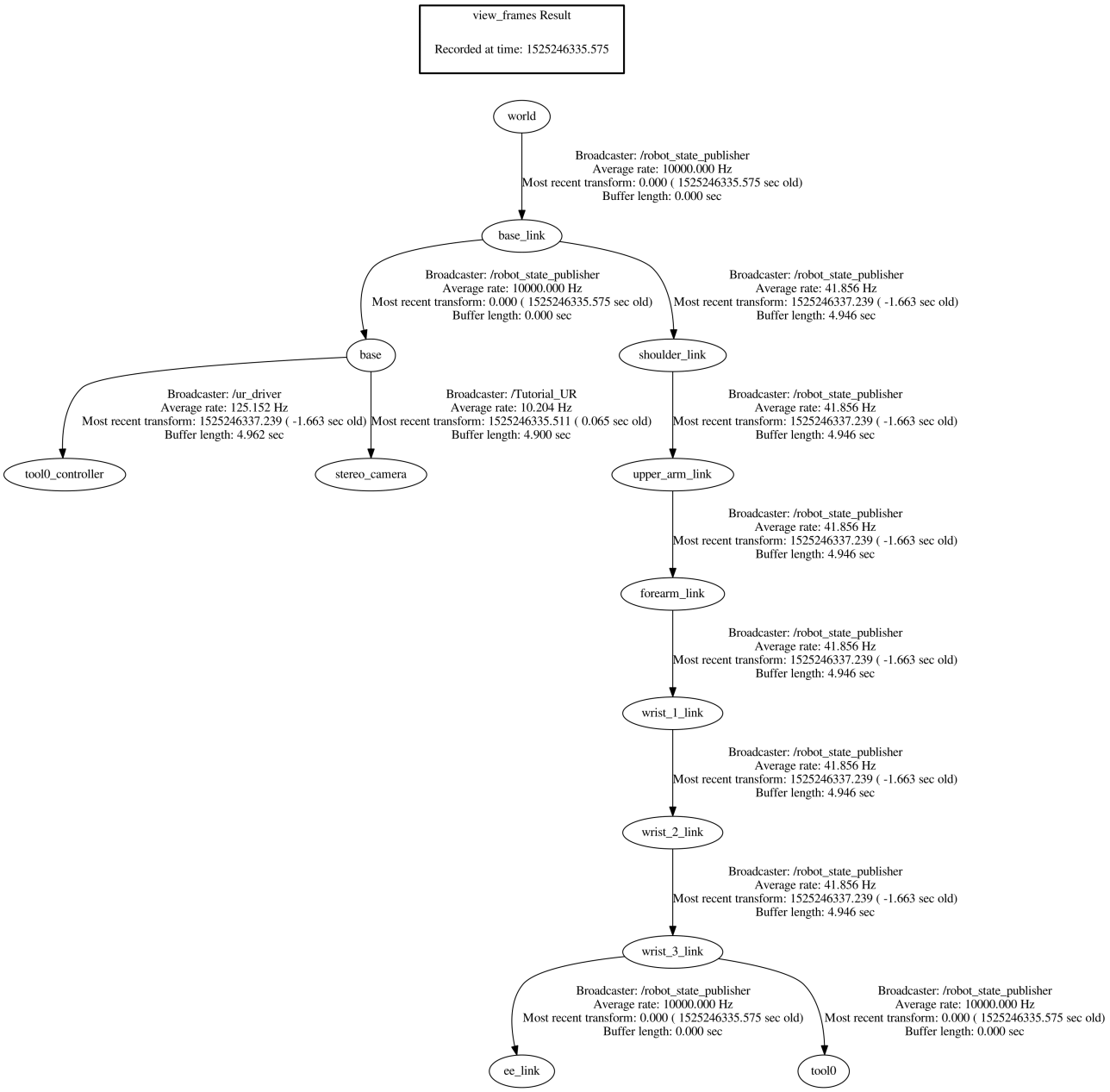


Figure 6.14: Complete tf tree graphical representation using ROS tf package

7

Discussion and Future work

This master project aims at developing a system to recognise different objects using CNN. Since it is a full industrial system, so it is not limited to only object recognition, but also one important goal of the project is to accurately determine grasping point on the object, pick and place it. Through the process there are different scenarios that may prevent carrying out the process in an accurate way, these scenarios are addressed in the projects, and proposed solutions are provided.
Initial hypothesis for picking the objects in a correct way is to pick them from the centre or middle point, and to pick oil filters from above. This assures that objects are stable in the gripper, but also make sure that they will be placed correctly on the holders on the MiR table. Experiment are performed to verify and validate this hypothesis, and it was found that mid points as grasping points are the best for oil pip and pickup, and picking the oil filter from above provide sound and accurate grasp. The scenario of picking more than one object at time can happen while picking Utubes and rarely happen while picking pickups. By measuring the torques and forces with the sensor it is possible to know when several objects have been picked. Two ways can be used for leaving extra picked objects. The first one is that the robot just goes down over the bin and leaves everything. The other way is that the robot goes down in the bin until the objects touches something and hopefully the extra picked object will fall off.

Every time a new photo is taken, it enters the trained network for this specific object, the positions in x, y, and z directions in robot frame are determined in addition to the angle of the object. According to the calculated angle, the end effector changes its orientation according to the calculated angle, and pick the object. The desired placing orientation of each object on the MiR table is known, so when the object is picked from the bin the gripper changes its orientation (wrist 3) and places the object in the right pose. Creating 4 regions for different ranges of angles was found to be enough but can easily be increased with more regions. Also to make gripping process more precise, the degree of openness and closeness of gripper fingers are controlled in a way that the gripper pick one object with a narrow grip. If the gripper is open widely, it may pick more than one object at once.

A challenging task in picking and placing the object is to make sure that it is picked in the correct orientation to be placed in the correct pose. When the object is picked, readings from the force/torque sensor give an indication of how the object is picked, but since each of different components gives very close readings when they are picked in different orientations, whether wrong or correct, so the proposed so-

lution as discussed in chapter 6, is to go through a check process, by moving the manipulator in x direction, and if there was a change in force in x direction, that shows that the object is picked in a wrong way. The solution gives good results while checking the object orientation, but the main disadvantage is that it takes time (up to 5 seconds) to perform the task. Another disadvantage is that it can only detect orientation that is completely wrong, which means that the object could be picked in a wrong way and as a result, it will not be placed on holders on MiR correctly.

In chapter 5 there was a case when the two pipes became merged because of too much information from the prediction was used, determined by the threshold. The network gave too high values for a pipe that should have been low. This means that the threshold has to be chosen higher. One disadvantage choosing higher threshold is that important information is lost and might affect where the centre point is.

Since the images taken with the cameras do not look the same, the network will give a different output. Since the outputs look different, the contours found will not look the same. When choosing the longest contours of the images they might not be of the same component. This was solved by only looking for contours in an area, chosen by the centre point of the left image. By doing this both contours will be of the same component but because of the contours are looking differently the centre point in the right image is not always the same as the left image. This causes an error when estimating the depth

A problem with the camera calibration done is that the computed projection matrices have some errors. If the triangulated points X and Y moves away from the centre of the cameras the computed Z increases to much. For cases when the points are in bins at the edges the error in Z is up to 45 mm wrong. This is partly solved by linearly decreasing Z depending on how far away from the centre the X and Y coordinates are.

A way to make sure that the object is picked in the correct point is to first pick a object anywhere and make sure that only one object is picked. This object is then placed on a table nearby where a camera is placed above. Since this is the only object on the table and the height of the table is known it becomes much easier to estimate the new grasping point. This makes the process more robust but at the cost of taking more time.

Future work

There are several enhancements that could be considered to make picking and placing process much more efficient and reliable. Those suggestions can be written as follows:

- An other type of camera could be used. The two cameras used are simple RGB cameras. With a stereo camera designed for depth estimation the result could be improved. A point cloud could be used as an additional input for

the network. This could make the network become better to determine which components are on top and bottom.
- Additional experimentation can be done of the structure and hyperparameters of the network.
- When the object is at the edge of the bin, it is complicated process to pick the object in this position as shown in figure 7.1

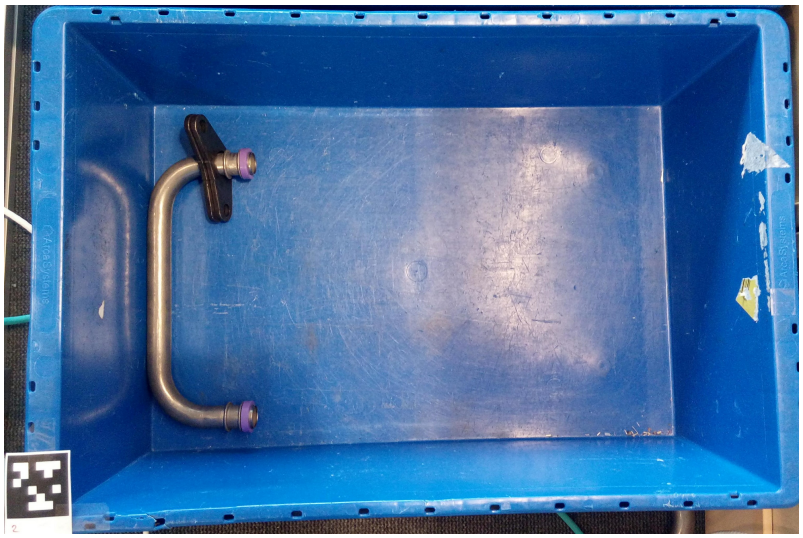


Figure 7.1: The scenario of an object located near the edge of bin makes it hard to be picked by the gripper.

One proposed solution for this is not to grip the object but to grip the bin itself, once gripped, the manipulator will go up, so that the bin is inclined with an angle, which allows the object to go from the bin edge to the middle, and as a result, the object can be picked.
- When an object is picked, the orientation of it needs to be checked to make sure it is placed correctly. Instead of going through the discussed checking process, an extra camera can be placed to check how orientation of the picked object is, if it is picked correctly then it will be placed on the MiR, if not, the robot will either adjust its joints and gripper to the correct orientation or put it back into the bin and a new image will be captured to be processed again through the trained network.

8

Conclusion

The thesis provides a practical solution for the industrial bin picking and placing problem. First, multiple different objects with different geometries are recognised, one with a circle detection algorithm and three using convolutional networks. The convolutional networks have no problem of identifying single parts but in some cases fails when the bin is full of parts.
By using data from force/torque sensor, only one object could be picked at time. By changing the angle of the end effector depending on the calculated angle of the object, and controlling the degree of openness and closeness of the fingers, picking process is done correctly, however in some cases, the gripping point is not calculated in the perfect position on the object for both images, so the object is not picked in a perfect way in those cases.

The readings obtained from force/torque sensor are within the same range for different objects. A checking algorithm is developed to check whether the object is picked in a right orientation or not. The algorithm detects how objects are picked and takes the corresponding action to each case, however the process takes long time to be carried out and can only detect when the orientation error is large.

Bibliography

[1] "Kuka lbr collaborative robot." By KUKA Laboratories [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], from Wikimedia Commons.

[2] "Biological neural network." By Jinxing Lai, Junling Qiu, Zhihua Feng, Jianxun Chen, Haobo Fan [CC BY 4.0 (https://creativecommons.org/licenses/by/4.0)], via Wikimedia Commons.

[3] "Ur10 technical specifications." `http://fab.cba.mit.edu/content/tools/universal_robot_arms/ur10_en.pdf`.

[4] "The urscript programming language." `http://www.sysaxes.com/manuels/scriptmanual_en_3.1.pdf`.

[5] L. Roberts, *Machine Perception of Three-Dimensional Solids*. PhD thesis, Massachusetts Institute of Technology, 1 1963.

[6] Y. Shirai and M. Suwa, "Recognition of polyhedrons with a range finder," in *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, (San Francisco, CA, USA), pp. 80–87, Morgan Kaufmann Publishers Inc., 1971.

[7] S. Tsuji and A. Nakamura, "Recognition of an object in a stack of industrial parts," in *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, (San Francisco, CA, USA), pp. 811–818, Morgan Kaufmann Publishers Inc., 1975.

[8] K. Ikeuchi, H. K. Nishihara, B. K. P. Horn, P. Sobalvarro, and S. Nagata, "Determining grasp configurations using photometric stereo and the prism binocular stereo system," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 46–65, 1986.

[9] R. C. Bolles and P. Horaud, "3dpo: A three- dimensional part orientation system," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 3–26, 1986.

[10] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 816–823, Nov 2013.

[11] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka, "Rigid 3d geometry matching for grasping of known objects in cluttered scenes," *Int. J. Rob. Res.*, vol. 31, pp. 538–553, Apr. 2012.

[12] Y. Domae, H. Okuda, Y. Taguchi, K. Sumi, and T. Hirai, "Fast graspability evaluation on single depth maps for bin picking with general grippers," in *2014*

IEEE International Conference on Robotics and Automation (ICRA), pp. 1997–2004, May 2014.

[13] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa, "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 951–973, 2012.

[14] O. K. Bruno Siciliano, *Springer Handbook of Robotics*. Springer, 1 ed., 2008.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[16] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer Publishing Company, Incorporated, 2nd ed., 2011.

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st ed., 2008.

[18] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[19] A. Martinez and E. Fernndez, *Learning ROS for Robotics Programming*. Packt Publishing, 2013.

[20] J. Kacprzyk and W. Pedrycz, *Springer Handbook of Computational Intelligence*. Springer Publishing Company, Incorporated, 2015.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[22] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679–698, Nov 1986.

[23] D. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111 – 122, 1981.

[24] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[25] T. Pfister, J. Charles, and A. Zisserman, "Flowing convnets for human pose estimation in videos," *CoRR*, vol. abs/1506.02897, 2015.

[26] V. Belagiannis and A. Zisserman, "Recurrent human pose estimation," *CoRR*, vol. abs/1605.02914, 2016.

[27] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[28] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[30] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[31] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, p. 1330–1334, December 2000.