Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Information Technology - Automation and Signal Processing	Spring semester, 2017 Confidential
Author: Simon Marnburg Eriksen	Simon Eriksen …… (signature author)
Instructor: Karl Skretting Supervisor(s): Hafrún Hauksdóttir	
Title of Master's Thesis: Visual guided robotic picking system for the grocery industry Norwegian title: Bruk av robotsyn for plukking av matvarer	
ECTS: 30	
Subject headings: pose estimation, ROS, features, motor control, robotic picking, homography	Pages: 59 + attachments/other: 9 + embedded file Stavanger, 15th of June/2016 Date/year

Visual guided robotic picking system for the grocery industry

Simon Marnburg Eriksen

June 2017

MASTER THESIS

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science
University of Stavanger

Supervisor: Karl Skretting

Abstract

People spend a lot of time and energy doing grocery shopping. Stores have become bigger and more centralised leading to more people having to use cars for their grocery shopping.
The market for online grocery shopping has been increasing rapidly, and many new companies are emerging in a market traditionally ruled by giants. Ordering groceries online can be convenient for the customer, but in the end, someone has to do the picking in a warehouse.

This thesis presents a vision guided robotic picking system designed to pick groceries from vertical shelves. A suggested solution of using the feature detector and descriptor algorithm SIFT to locate and estimate the objects pose is presented. By using a known image of each product, four corner points can be located and used to estimate the homography.
The system is implemented on a vertically mounted 3-axis gantry robot mounted in front of shelves.
The scope of this project extends to controlling the motors on the robot as well as an industrial vacuum system that is used together with suction cups to pick items. A solution for controlling the robot using the open software library SOEM as well as Robotic Operating System (ROS) is presented.

Results show that the pose estimation algorithm can provide a positional accuracy within 1cm on items with a flat surface. This is good enough for the robot to place a suction cup. A short video demonstrating the functionality of the system can be seen using the following link: `https://goo.gl/vQDF44`

Preface

This paper concludes five years at the Department of Electrical Engineering and Computer Science at the University of Stavanger and five months of work at Automatmat. I would like to thank my supervisor Karl Skretting as well as my co-workers at Automatmat who has given me the opportunity to do this exiting project.

Contents

List of abbreviations

SIFT Scale-invariant feature transform

SURF Speeded up robust features

SOEM Simple Open Ethercat Master

EAL Easy Automation Library

ROS Robot Operating System

FLANN Fast Library for Approximate Nearest Neighbors

DoG Difference of Gaussian

HMI Human-machine interface

1. Introduction

This thesis presents a method of using computer vision to guide a robot in picking grocery items from a shelf, including motor control and a ROS system tying the parts together. The task was given by the start-up company Automatmat
The overall goal for Automatmat is to make a feasibility prototype robot that can be used as a demonstration of both hardware and software.

1.1 The company

Automatmat is a lean start-up, and operates as such with minimum overhead. The vision of the company is to bring the benefit of automation into the grocery industry and to revolutionize the inventory management. For more info on the long term vision, please visit www.automatmat.no

Automatmat is currently in phase 2 of its development plan.

- Phase 1 consisted of a market analysis.
- Phase 2 consists of developing a feasibility prototype which this thesis is a part of. Phase 2 ends 15. July 2017
- Phase 3 will be integration in a warehouse in Norway in corporation with a pilot customer.
- Phase 4 is full scale commercialisation.

1.2 Objective

Due to the agile development methodology used in Automatmat, the objective for this thesis has been modified over time. The core problem, computer vision guided picking is the main focus, but it had to be slightly simplified due to the need for a ROS system and a motor control system. The main topics for this thesis has therefore been: Vision guided picking, motor and vacuum control, and development of a ROS system connecting the two.

For the vision guided picking, the following questions are considered:

- Can pose estimation be done reliable with a single camera
- Is the picking by robot reliable enough
- Can the vision system select the point of contact accurately enough

For the motor control, the following questions are considered:

- Can the motors be controlled using open software
- Is the axis movement smooth

For the ROS system, the following questions are considered:

- Is ROS a suitable platform

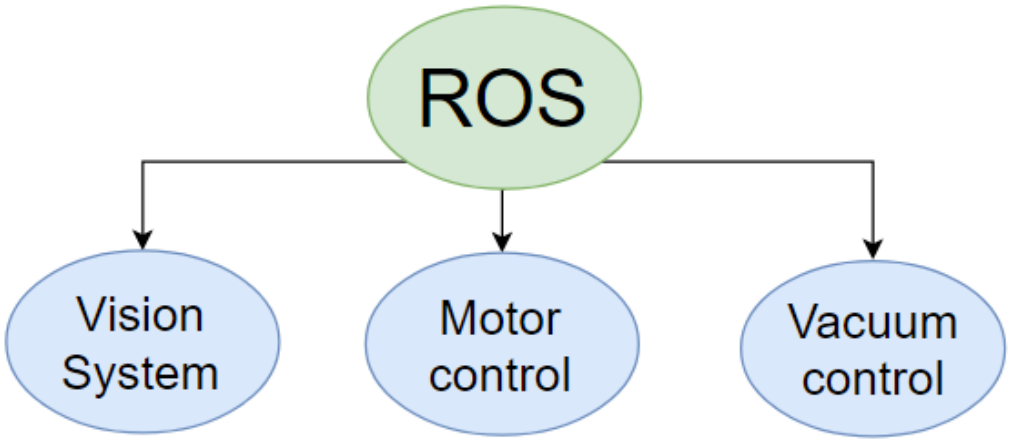

Figure 1.1: Thesis outline

Figure 1.1 shows the outline of the objective. Three main systems tied together using ROS.

Delivery

1. Existing technology overview
2. Design proposal for vision guided picking
3. Selection of cameras
4. Algorithm for picking
5. Motor control software
6. Vacuum control software
7. ROS environment
8. Final report

1.3 Other projects on the same system

This thesis is a part of a bigger development project at Automatmat consisting of two master theses, including this, and one bachelor thesis. The bachelor candidate, Henrik Margnussen, was given the task of choosing motors, communication standards and doing tests on the vacuum system to make an overview of which suction cups is able to lift which products.
The other master candidate Mikal Berge have been working on computer vision algorithms for packing groceries using the same robot that is used for picking.

The bigger development project is expected to be finished one month after the end of this master project. This delay is due to unexpected challenges during development and delayed deliveries from manufactures.

1.4 Motivation

There are some key points that make this project attractive.

- Grocery industry is lagging behind on automation
- The market for online grocery shopping is increasing rapidly
- High minimum wage means automation make sense in Norway
- New technology in computer vision and improved hardware makes it possible to automate the grocery industry

New technology within computer vision and already established online grocery stores makes it a possibility to do this kind of project now.

It is possible to order groceries online and have it delivered to your door or to some pick-up-point. So the user-side has been automated, but on the supplier-side, people still have to walk between the shelves in the warehouse and manually sort out each customers items.
The environmental benefits of completely automated stores will be great for the big cities with fewer cars on the road, less food waste and less wear on the roads due to fewer heavy vehicles.

Why a vision guided system
Automation of production lines, packing robots or similar systems have a long history of using automation to streamline efficiency. The grocery industry should be no exception, but the high variation of items as well as expiration dates and fragile items makes it difficult.

For this project, adaptability is crucial. Groceries come in countless shapes and sizes. Building a specialised implementation for each type of product is not feasible and certainly not scalable. The goal is that the system will be adaptable to a range of products and scalable with the help of computer vision and gantry robots.

The task
A key part of Automatmat's project is to make it cheap enough to do this on large scale. Other solutions use universal robots which is expensive and often slow, but Automatmat aim to make it cheap by using 3-axis robots that can be scalable and specialized for handling vertical shelves. A solution that can handle vertical shelves can be implemented in existing warehouses without a complete re-build. Items can be stocked from the other side of the shelf.

1.5 System requirements

By the given task we can sum up some system requirements:

- Vertical integration of gantry robot
- Able to lift products weighing 1.5kg
- Should position a vacuum cup accurate enough to pick up an item using computer vision
- Control motors and vacuum system using open libraries
- The system should work on a range of products

2. Background

This chapter presents background information to key parts that has been implemented in the solution as well as an overview of what technology has been implemented in other existing solutions.

2.1 What has been done before

Autostore[1] is a Norwegian company that develops automation for warehouses. Their solution have robots moving on top of a 3 dimensional grid of boxes full of product that can be retrieved by the robots to a human picker. As the robot only picks up the whole box, manual labour is needed to pack specific orders.

IAM robotics uses a wheeled robot that navigates between shelves. The robot is equipped with a suction cup on the end of a 6-axis arm and 3d-camera used to locate products.
As stated in [5] "In the long run, speed will be a significant factor in many industrial applications, which may give an advantage to static arms and gantry solutions over wheeled platforms." speed is an important factor in picking solutions. The wheeled robot from IAM robotics is slow and target handling items that have a low throughput like pharmaceuticals.

Ocado is a large online only retailer based in UK that sells groceries and other items. Ocado has been investing a lot into automation of their warehouses and is also developing technology in-house. Their existing solutions only handle boxes and not single items.

As these are all commercial solutions there is not detailed information available about the systems. Existing technology shows a trend of starting at the other end of the spectrum with large boxes, and not individual items.

2.2 The pin hole camera

A monocular camera is often simplified to the pinhole camera model. In theory all the light passes through a single point, which is the pinhole. In practice this is not possible, and lenses further distort the image.

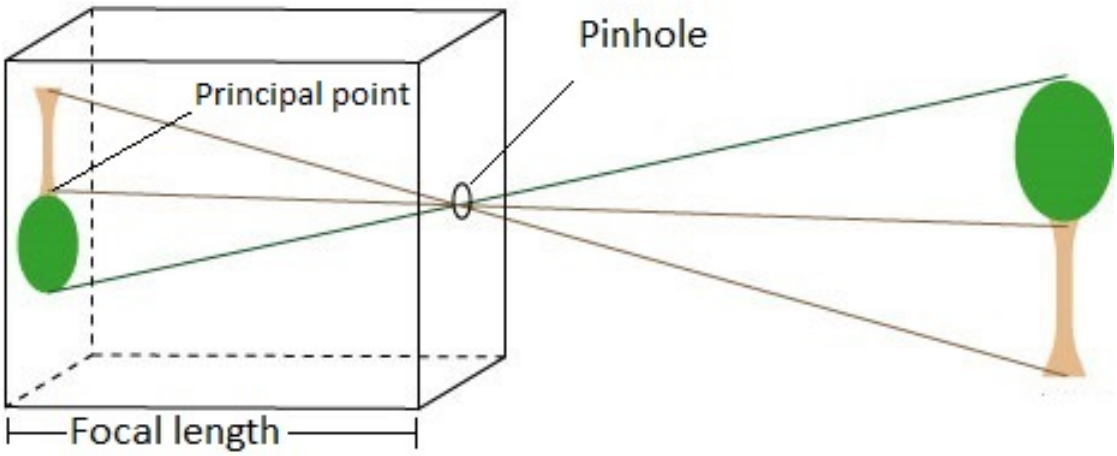

Figure 2.1: The pinhole camera [6]

Figure 2.1 demonstrates how the apparent size of the object changes with distance. This means we can determine the distance by measuring how many pixels the object occupy on the image sensor if we have knowledge about the size of the object.

The center axis seen in figure 2.1 is called the optical axis and the intersection on the camera sensor is called the principal point. The principal point is what defines the center of the image.
All points seen in the real world by the camera is converted to 2d points on the image plane. A point in the real world is defined as $P = (X, Y, Z)$ and a point on the image plane is defined as $p = (x, y)$

$$x = f\frac{X}{Z}, y = f\frac{Y}{Z} \tag{2.1}$$

As seen in equation 2.1, information about depth is lost on the image plane. This means you must have multiple points on the image plane and knowledge about an object to calculate depth information.

2.3 Camera calibration

A camera calibration needs to be done because it is not possible to have a theoretical pinhole camera in the real world and the lens will further distort the image.

ROS has a camera calibration tool built in which has been used in this project to calculate camera matrix and distortion coefficients A flat chequerboard pattern is needed to supply the software with data. The chequerboard can be of arbitrary size, the important thing is that it's flat. By locating points on the chequerboard the software calculates the deviation from what it should look like on a pinhole camera and provides a set of distortion coefficients that can be used to correct for distortion and get an approximation of the pinhole camera.

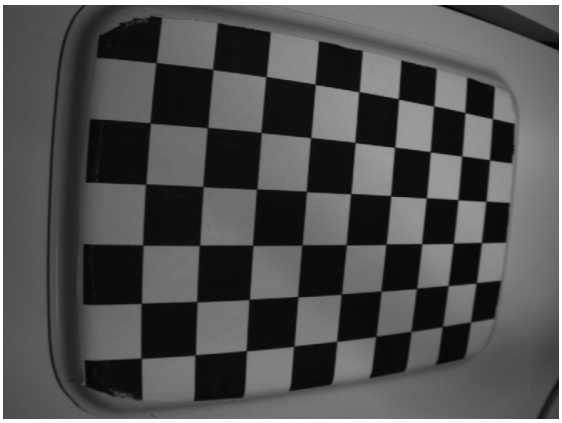

Figure 2.2: Before calibration

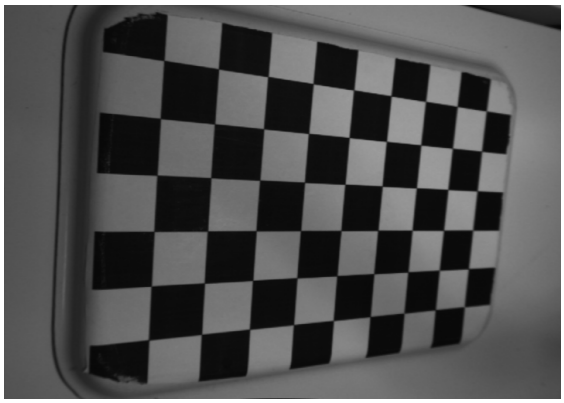

Figure 2.3: After calibration

Figure 2.2 shows an image of a chequerboard. This is one of 14 images used to calibrate the camera in this project. Figure 2.3 shows the same image that has been undistorted using the camera matrix and distortion vector that was found after calibration. Lines that are straight in the real world will now also appear as straight in the picture. Depending on the method used information can be lost near the edges and corners as the whole undistorted image is not a rectangle. The camera matrix is defined as in equation 2.2. f_x and f_y represent the focal length. These will be the same in a pinhole camera, but can differ in an actual camera. cx and cy defines the principal point. Equation 2.3 defines the distortion vector with 5 coefficients.

$$M = \begin{bmatrix} f_x & 0 & cx \\ 0 & f_y & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$dist = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{2.3}$$

2.4 Planar homography

The purpose of using planar homography in this project is to estimate an objects 6 degree of freedom pose in world coordinates.
To find the unique pose of an object using a calibrated camera, homography and a known camera location you need four coplanar and no three collinear points[9]. In other words, given we know the location of the camera, have done a camera calibration and in some way find four points that satisfy the given requirements, we have enough information to calculate an object's 6 degree of freedom pose using planar homography.

The homography matrix describes the relationship between projected point on two planes. The matrix itself have the dimensions 3x3 and is defined as in equation2.4 and has 8 degrees of freedom.

$$H = \begin{bmatrix} h_{12} & h_{12} & h_{13} \\ h_{22} & h_{22} & h_{23} \\ h_{32} & h_{32} & h_{33} \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

Equation 2.5 shows how the homography matrix can be used to transform one point on a plane, to another plane. Figure 2.4 shows an illustration of a transformation between two camera planes.

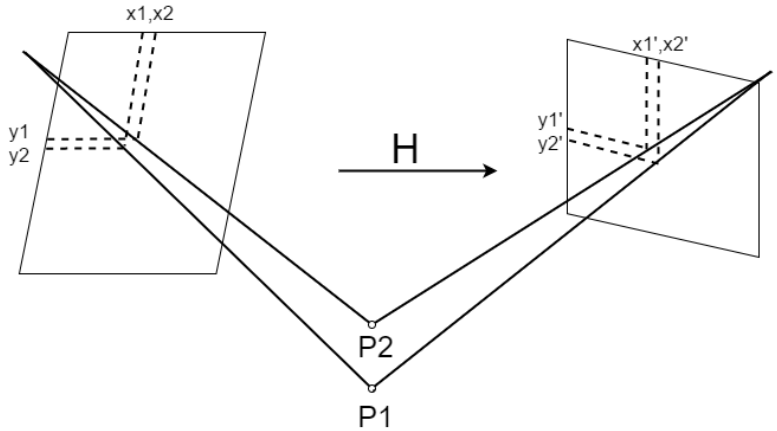

Figure 2.4: Homography illustration

2.5 Outlier rejection

A robust way of rejecting outliers is important when estimating homography using SIFT as there will always be matches found that are not actually corresponding points. The algorithm used in this project is called Random Sample Consensus (RANSAC) and is know to be robust against large quantities of outliers [7]. Implementing RANSAC means randomly sampling a subset of the data and creating a model based on only the subset. The subset model is evaluated towards the whole dataset and given a score based on how many datapoints fit the model within a threshold. This runs for a given amount of iterations and the highest scoring model is chosen as the estimation.

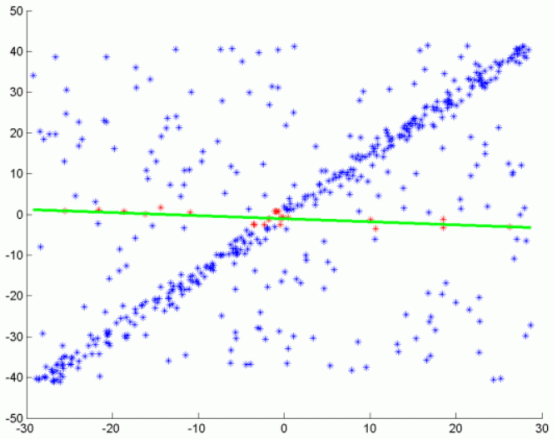

Figure 2.5: RANSAC iteration 1 [8]

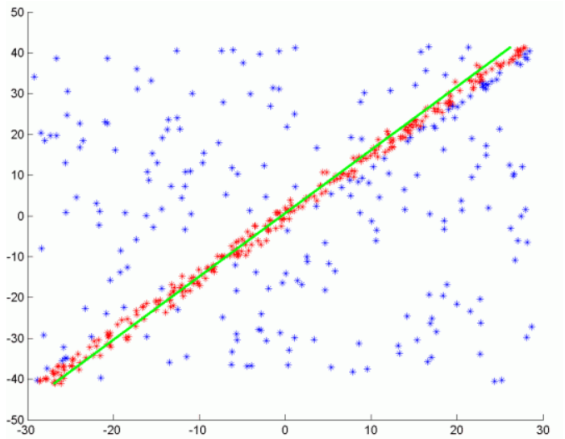

Figure 2.6: RANSAC iteration 2 [8]

Figure 2.5 shows the result during an iteration of RANSAC implemented for line-fitting. The datapoints that fit the model within the threshold are shown in red while the outliers are blue. It is obvious for a human that this is probably not the correct model.
Figure 4 shows a different iteration where the random line model contains a lot more inliers. The model used in figure 2.6 would be chosen over the one in figure 2.5 because it has a higher score.
This line-fitting example demonstrates a simple implementation of the RANSAC algorithm. More complex models can also be estimated like a plane in 3D space.

2.6 Features and descriptors

The purpose of a feature detector is to find interesting areas or points in an image. Interesting points can be corners, edges, texture etc. and are referred to as features or keypoints. Features are often used for tasks like object detection and tracking. Figure 2.7 visualises keypoints found by a feature detecting algorithm. Each feature is assigned a image coordinate, scale and image gradient.

Figure 2.7: Image with keypoints visualised

Descriptors

When a feature has been found, a descriptor will assign a distinct description of the point based on the surrounding area. The purpose of the descriptor is to describe the point so that it can be recognised under different circumstances, like if the same point is rotated or scaled. A good descriptor gives a similar description independent of scale and rotation.

2.7 Feature matching

When keypoints have been found by a feature detector and described by a feature descriptor in two different images. A feature matcher can be used to find keypoints that have similar descriptors in both images. Points that have similar descriptors will hopefully be corresponding points.
When looking at large sets of keypoints from a large number of images, which is usually the case when doing object recognition you can speed up the matching process by implementing algorithms like KD-tree search or Fast Library for Approximate Nearest Neighbors(FLANN)[10].
When comparing only two images, the number of keypoints will often be in the range of a few hundred to a few thousand. It can then make sense to use a brute force matcher with the advantage of guaranteeing the best match.

Figure 2.8: Corresponding points found

Figure 2.8 shows an example where the keypoints and matches are visualized. A SIFT feature detector and descriptor has been used and only the best matches are visible. The product is held upside-down at an angle with a distance that makes the apparent size about half of the training image to demonstrate the invariance to scale and rotation. The lines between the two images shows estimation of which keypoints are corresponding.

2.8 Robotic operating system

Robotic operating system(ROS) is a software framework designed to develop software for robots. It was initially started by Willow garage and Standford Artificial intelligence Laboratory [4] with the purpose of making it easier to collaborate and implement others work. Work was often fragmented and ROS was a step towards unifying and streamlining research and development of robotic and artificial intelligence software.

A new release of ROS is scheduled to be released every year, with a long term support(LTS) version every second year. The release used in this project is called Kinetic Kane and was released May 2016 and will reach its end of life May 2021.

2.9 The components of ROS

Ros nodes

A ROS node is a process that do tasks. Each node is a separate executable that can be compiled and executed independent of other ROS nodes in the system. A ROS node is basically a program that can communicate with other programs through the ROS environment.

Ros message

ROS nodes communicates by sending and receiving messages. A message has to be of a set structure given by what type of message it is. A message type can contain anything from a single integer to multidimensional arrays. The standard collection of messages include String, int8, int32, bool and many more. If the message type you need is not in the standard library, it is easy to create custom message types.

Ros topics

Ros nodes can exchange data by subscribing or publish messages on a topic. The topic is the name of the bus on which the message is transmitted. Only messages of the same type should be published on a single topic, but multiple nodes can publish on the same topic.
Publishing and subscribing is done anonymously so if data of the same type needs to be separated, it could be solved by publishing on a separate topic.

The ROS master

The ROS master have an overview of what topics are being published and what nodes are publishing them. If a node wants to subscribe to a topic it will ask the master where to find it. Data is not funnelled through the master, but it acts more like a DNS server telling nodes where to find what they are looking for.

2.10 EtherCAT

Ethercat is a real-time capable communication protocol developed by Beckhoff [2]. It was important during development to focus on low cycle times (<100μs) which is the rate data can be updated. EtherCAT is based on Ethernet which gives the advantage of cheap cables up to 100m and hardware that are easily available.
The master sends out a standard Ethernet frame, this frame has a EtherCAT telegram inside. When the frame reaches a slave, the data inside is read on the fly and it can also add data to the telegram on the fly. This method of having the data always "moving" contributes to the high speed. It is only the master that can send out Ethernet frames.
Ethercat was chosen as the communication protocol of choice for this project as a result of Henriks study. Figure 2.9 shows the structure of the Ethernet frame with

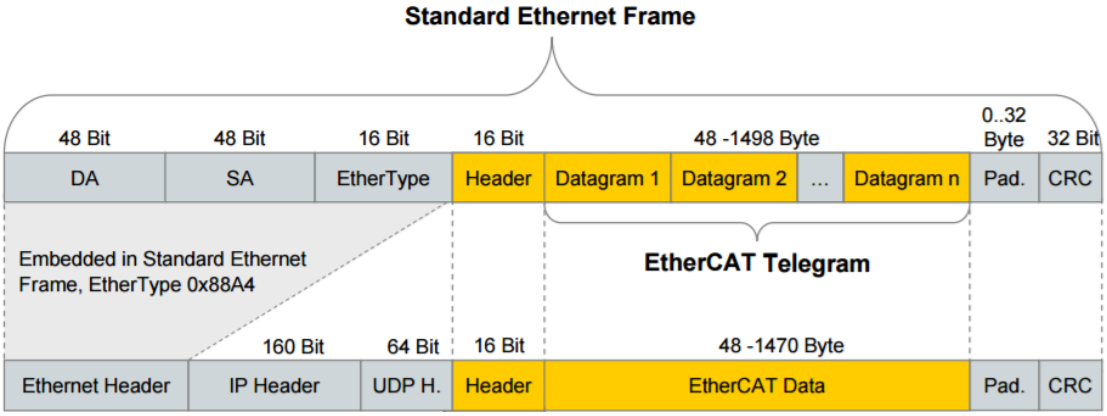

Figure 2.9: EtherCAT structure. Figure by ETG [2]

the EtherCAT telegram inside.

2.11 Libraries used

This section will give a brief overview of the software libraries used in this project.

OpenCV

OpenCV is an open source library that is free for commercial and private use following the BSD licence[3]. Its main uses are computer vision and machine learning. It is written in C/C++ and provides complete python bindings.
OpenCV provide some feature detectors in the standard package and some additional ones including SIFT and SURF if the library is compiled together with a contribution package

Simple Open Ethercat Master

Simple Open Ethercat Master(SOEM) is a open software library for creating a EtherCAT master. It is written in C for use in linux and windows.

Easy Automation Bosch Rexroth

Bosch Rexroth have seen a trend of customers wanting more open software in their applications and have therefore responded by releasing a software library for their drives called Easy Automation Library(EAL). It is written in C and supports the most common operating systems, Windows, Linux and Android. EAL uses the communication protocol Ethernet/ip which is significantly slower than EtherCAT with cycle times around 10ms[2].
This library has not been included in the implementation, but it has been converted to c++ and compiled in a ROS node. This can be used in future development for a human-machine interface(HMI) in parallel with the ROS/SOEM system.

3. Concept development

The following chapter presents different concepts that was assessed in the development process. The choice of concept should be expected to be able to satisfy the system requirements.

3.1 Selection of camera

There where three main concepts of a camera solution that was evaluated during development.

- Monocular camera
- Stereo camera system
- 3d camera system

Monocular camera

A monocular camera system consists of one image sensor and one lens. Using only one monocular camera was from the start the preferred solution because this means the equipment cost goes down together with the simplicity of the system. A challenge is the limited amount of information the camera provides.

Stereo camera system

When you have a camera mounted on a robot that you know the position of there are two approaches to obtaining a stereo image. One is to have a traditional dual camera system and the other takes advantage of the fact that you can move the robot a known distance such that a stereo pair of images can be obtained from a single source.

3d camera system

A 3d camera system can provide information about depth. Can not rely solely on 3d pointclouds as many items are of similar shape and size. What separates them is the texture.

Selection of camera

A decision was made to use a single monocular camera for a few reasons. In a competition hosted by Amazon with the challenge of making a robot that can pick items from shelves about half of the teams did not score a single point[5] and vision was stated as one of the key challenges. Having a simplified vision system increases increased the chance of being able to test a functional prototype in this project. The camera chosen was a Intel Realsense SR300 development kit3.1. This camera also includes structured light based 3d sensor which can be used for future development.

Figure 3.1: Intel realsense SR300

3.2 Selection of control library

EtherCAT was chosen as communication protocol by Henrik as a result of his study. Chosing EAL for motor control would go against his recommendation. The reason this had to be evaluated was because EAL provides abstraction layers and functions that would simplify getting up and running compared to with SOEM.

SOEM was chosen after the manufacturer of motors and drives offered a workshop to get up and running. They could not provide support related to SOEM, but provided essential information about start-up procedures and what the different bits in the input and output registers of the drives represent.

4. Implementation

The following chapter describe the implementation of hardware and software. Figure 4.1 shows an overview of the complete picking system. The blue circles represent the different ROS nodes while the red boxes represent hardware components. All software is written in C++ for ROS kinetic Kame on Ubuntu 16.04.

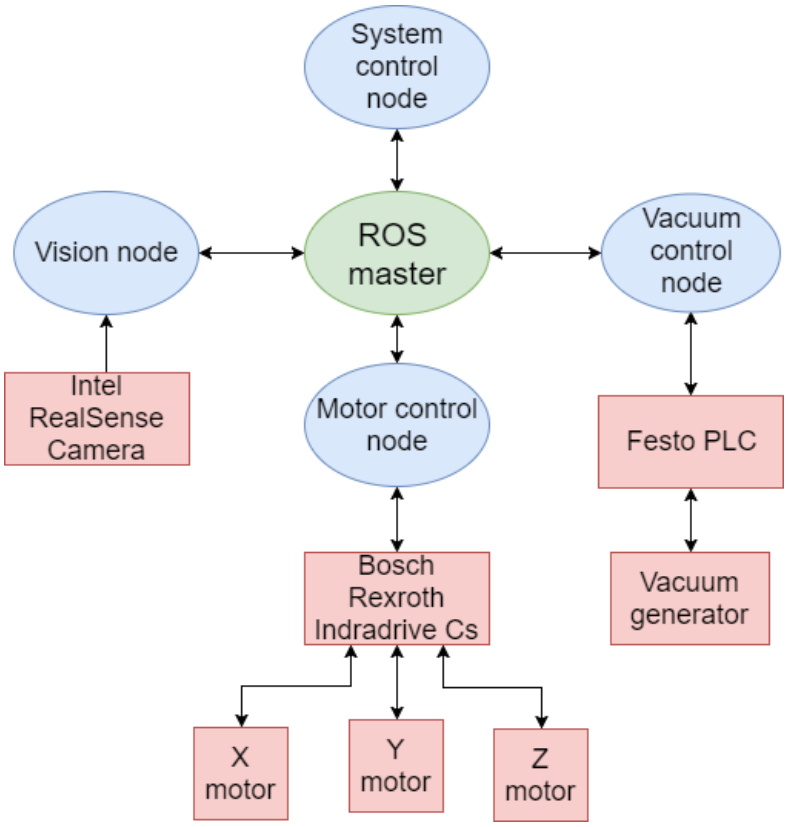

Figure 4.1: System overview

4.1 The robot

The following section will describe the mechanical setup and the robot used. The robot was not assembled when delivered from the manufacturer, the building process has been a collaboration with the other students at Automatmat. The wiring was mostly done by the other master candidate Mikal Berge.

The robot is a 3-axis gantry system designed to be placed with the x-y plane in a horizontal position so that the z axis will move towards the ground. This is a common setup for most pick and place machines. One critical design question for this project is how well this type of gantry system will work when implemented with the x-y plane vertical and the z-axis moving horizontal into the shelves. The robot has a travel range of 1m in x and y direction, and 0.5m in z direction.

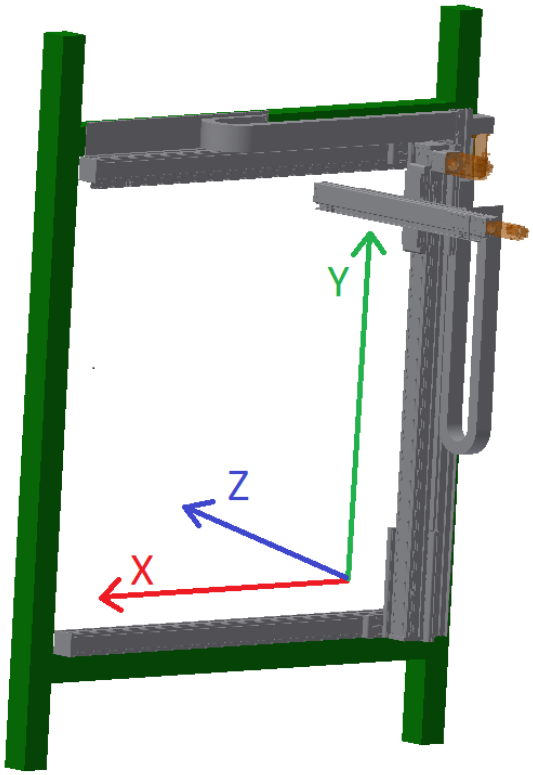

Figure 4.2: CAD drawing of robot supplied by manufacturer together with the mounting frame shown in green

The robot has been mounted using four 80x80mm aluminium profiles, these are

Axis	mm/Rev
X	65
Y	130
Z	10

Table 4.1: Axis data

shown in figure 4.2 as green bars. The horizontal bars are there to give stability for the two x-axis actuators, while the two vertical bars give four points that secure the system to the floor and roof. The horizontal bars have a length of 1350mm and the vertical 2000mm.
The x-axis is closest to the base of the robot and moves the whole y-axis.

A low weight circular aluminium tube profile has been used to mount the vacuum cup on the z-axis to minimize stress introduced on the actuator. This also made it possible to run the vacuum tube inside the axis as seen in figure 5.13. It is only the aluminium tube that moves in the z-direction because of how the z-actuator is mounted.

Table 4.1 shows the amount of axis travel with each revolution of the motor axis. The Y-axis is directly coupled to the axis with a coupling, while the X-axis has a belt that reduces the motor input by a factor of 2. The Z-axis is directly coupled to the screw that runs through the axis. The screw has a pitch of 10 and moves therefore the axis 10mm with each revolution. This data has been registered by testing as the documentation provided by the manufacturer was limited.

4.2 Motors and drives

Both motors and drives are produced by Bosch Rexroth. Choosing this type of motor was a result of the study done by Henrik.
Motors:
x: Has a rated continuous torque of 0.64Nm.
y: Has a rated continuous torque of 1.3Nm. The motor is fitted with a brake that is applied if the motor looses power. This prevents the axis from falling down and causing damage.
z: Rated torque of 0.32Nm. This motor is fitted with a brake in case it make sense to test the robot in a horizontal position with the Z-axis pointing towards the ground. .
A more detailed illustration of the topology of the drives and motors are shown in figure 4.3. The nature of EtherCAT allows for using a single Cat5e cable with

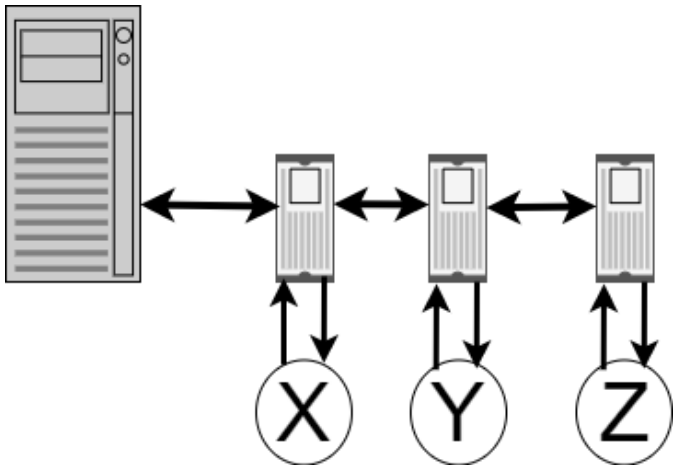

Figure 4.3: Motor connections

rj45 connectors to the first drive which are then connected to the next drive. This is called line topology.
Each motor is connected to the corresponding drive with a encoder cable and a three phase power cable.

Gantry robots will often have to do a procedure called homing when starting from a state of no power. This is often done by moving in one direction until a limit-switch is triggered. The purpose of this is to establish a known position.

All the motors used in this project contain an absolute encoder with battery to keep positional data stored when the power is off. This means the motors will always know how many revolutions and degrees they are from a known point and homing is not needed. It also enables the possibility to implement travel-limits using software and not hardware. Travel-limits is important to prevent the robot from crashing in the mechanical stop point and cause damage.
The travel-limits has been set using the Indraworks DS software, and to a distance of 4cm from the mechanical stop point of all three axis.

4.3 vacuum system

The vacuum system consist of two main parts. A valve terminal with PLC functionality and a vacuum generator. A digital output on the valve terminal turns on or off the vacuum generator by opening and closing a valve inside the vacuum generator. The generator outputs the actual pressure on the vacuum side as a 0-10v analogue signal. This is read by the valve terminal and transmitted in the cyclic EtherCAT data. The valve terminal is supplied with pressured air from a air compressor.

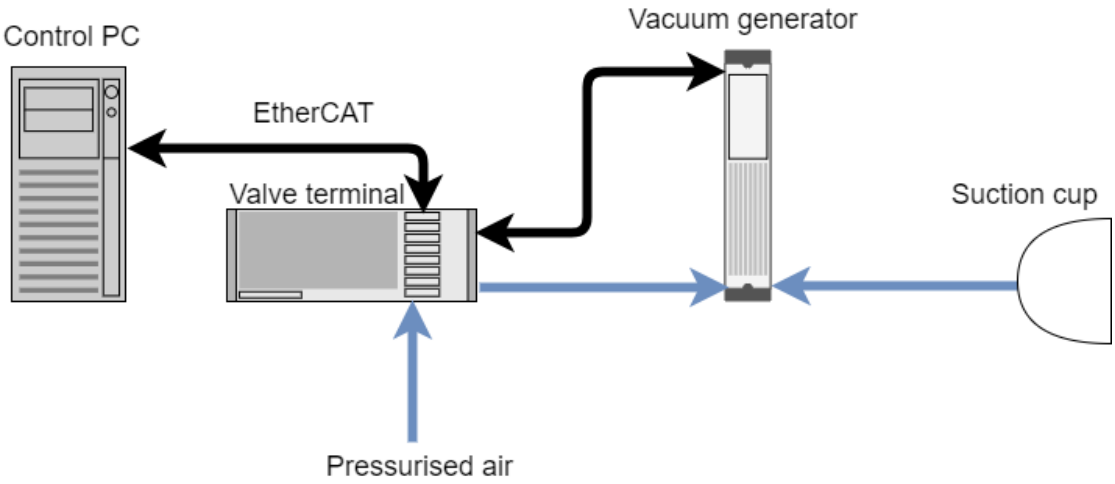

Figure 4.4: Vacuum system

Figure 4.4 shows an overview of the vacuum hardware. Electrical connections are shown in black, while airflow is shown in blue. Figure 4.5 shows how the vacuum cup is mounted on the z-axis. It is possible to change vacuum cup by unscrewing it.

4.4 Camera system

The camera system is a Intel Realsense SR300 development kit which is mounted on the z-axis. The SR300 contains a RGB-sensor, IR-sensor as well as a depth sensor. Only the RGB-sensor has been used for this implementation and with a resolution setting of 680x480(VGA). Communication to the control-pc is done through USB. Because USB 3.0 has a design limit of 5 meter of cable length an active cable is required.
The camera system has a mounting point for standard tripod screws and a joint that can rotate the image sensors from pointing straight forward to 90 degrees

down. This means it is possible to manually alternate between capturing the shelf and the ground where the packing box is located.

Figure 4.5 shows how the camera is mounted on the z-axis.

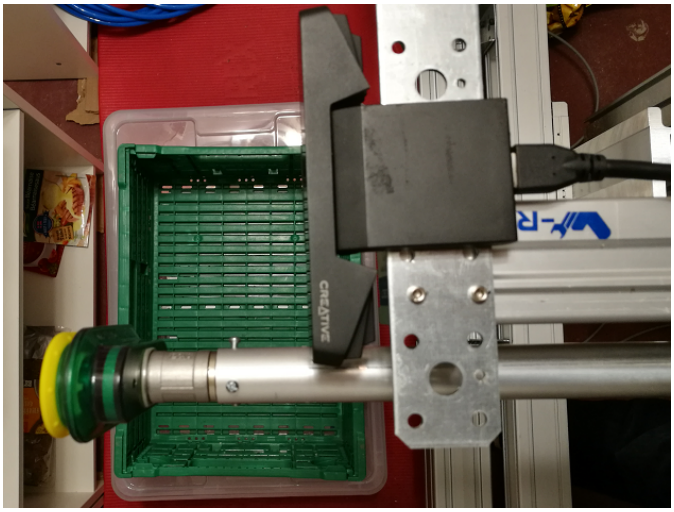

Figure 4.5: Camera system with vacuum cup

4.5 Motor control

One of the requirements for choice of motors and drives was the possibility of using of position control. This means by giving the drives a position they will internally calculate a path and run a closed control loop on the positioning. A high speed control loop between a ROS node and the drives is therefore not needed and simplifies the control software needed for this projects application.

The motors comes with pre-set parameters for the control loop. These settings are good enough for roughly 70 percent of all applications according to the manufacturer If control problems are experienced it is possible to do autotuning in a program provided by Bosch-Rexroth called Indraworks Ds. The software performs autotuning by moving the chosen axis and measuring its frequency response and step response.

4.6 Pose estimation

As the homography describes the relationship between points on two planes, we can use this to estimate the pose of an object. One of the planes is placed on the front facing surface of the product, while the other is placed on the image sensor. This way, the relationship we estimate is the relationship between the camera and the object. By having the camera mounted on the robot we know its position and by having an estimate of the relationship between the camera and the product we can estimate the actual position of the object in world coordinates.

The plane on the products surface is found using SIFT. Each product have one training image. Keypoints and descriptors from each of the training images are calculated and stored when starting the program. Depending on what product should be picked, the keypoints and descriptors from its training image is matched with keypoints and descriptors from the camera feed.

Listing 4.1: Key functions

```
Mat H = findHomography( training_kp, camera_kp, CV_RANSAC );
.
.
perspectiveTransform( training_corners, camera_corners, H);
.
.
solvePnP(training_points,camera_corners,mtx,dist,rvec,tvec);
```

Listing 4.1 shows three key OpenCV functions that have been used for pose estimation. The homography matrix is first estimated using corresponding matches from the training and camera image.
The homography matrix is then used to transform the cornerpoints on the training image, which is on the camera sensor, to the corresponding points on the actual products location.
The last step calculates the rotation and translation vectors from the cornerpoints on the image sensor, to the transformed points found in the second step.

Rejecting bad pose estimation When the homography matrix is estimated, OpenCv can use statistical methods to remove outliers in the dataset. The algorithm used in this thesis is called RANSAC. The implementation of RANSAC in OpenCV when estimating homography is used in the following way.
A random set of four corresponding points are selected and the homography is estimated using only those points. This model is now evaluated towards all corresponding points and a score is made depending of how many corresponding points

fit this model. This goes on a set number of iterations and based on which model got the highest score, a model is selected and the points outside of the threshold is considered outliers.

4.7 ROS framework

The following section will describe the architecture of the ROS system and the functionality of each node in the system. An overview of the most important topics the nodes publish and subscribes to is presented in tables.

4.7.1 Pose estimation node

The pose estimation node is started when the value *true* is published on the topic *start_pose*. The node is only started when needed as it is computational expensive. If a good pose estimation is found it will publish the objects position on the topic *object_position* which is used by the control node.

To filter out some of the inevitable bad pose estimations, some requirements has been implemented.

- Distance must be in the range 30-55cm(within the shelf)
- Rotation in all axes must be within 30°
- No cornerpoint can be far outside the cameras field of view.

As a last step for the node to publish "good pose" there has to be three consecutive pose estimations within ±1cm in distance and ±1.5cm in x,y translation. If all of these requirements are fulfilled the topic "good_pose" will read as True.

Figure 4.6: Visualisation of pose estimation

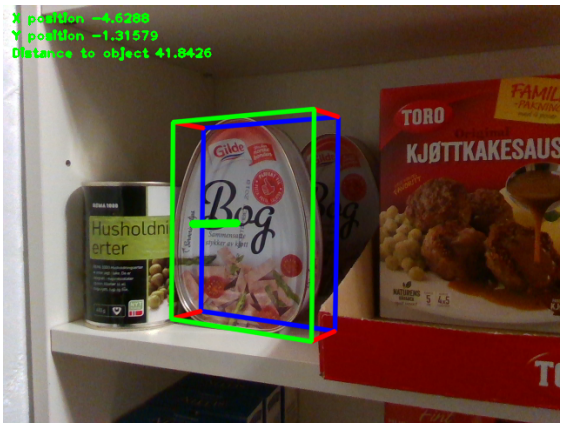

Figure 4.7: Visualisation of pose estimation

Figure 4.6 shows a real-time visualisation of the pose estimation. The green, red and blue lines shows the estimated outline of the target product based on known

dimensions. The yellow line represents the pick-point as a normal on the front-face.
Figure 4.7 visualises pose estimation on a non-rectangular product. The plane used for homography estimation is then the smallest rectangle fitting the object. This rectangle is visualised with green bars.

Message type	**Topic**
Start pose	bool
stop	bool

Table 4.2: Pose estimation node subscribers

Message type	**Topic**
position	object_position
bool	good_pose

Table 4.3: Pose estimation node publishers

position message	
Uint16	x
Uint16	y
Uint16	z

Table 4.4: Contents of position message

4.7.2 Motor control node

The workflow of the motor control node can be seen in figure 4.8. The main structure is a switch-case statement. All three motors are controlled from the same node, but they all have their own switch-case control loop.

Step 0, 10, and 20 are initialization steps and contain a start-up procedure.
In step 0 the target velocity for each motor is set. X and Y is set to 100mm/s while the z-axis is set to 50mm/s. Any error that is present on the drive is also reset in this step by enabling bit 5 in the signal control word.
In step 10 we go to pre-operational mode by enabling bit 14 and 15 in the master control word, and verify the operation before going to step 20.
The drives are set to operational mode in step 20 by also enabling bit 13 in the master control word. Verification is done before going to step 30.

Step 30 is the main operation mode. It means the motor is in position, not moving and ready for a new target position. If a new target position is published in the ROS environment on the topic $target_position$, the target position will be written to the drives and the control sequence goes to step 40.
In step 40 the actual movement is started by toggling bit 0 in the master control word. If movement is confirmed it will go to step 50.
Step 50 waits for the motor to reach its target position and when reached it goes back to step 30 ready for a new position command.
A safety feature has been implemented to only allow X and Y axis movement if the Z axis is not inside the shelves. Moving in X or Y direction while the suction cup is in the shelf can cause damage by crashing in the shelf.

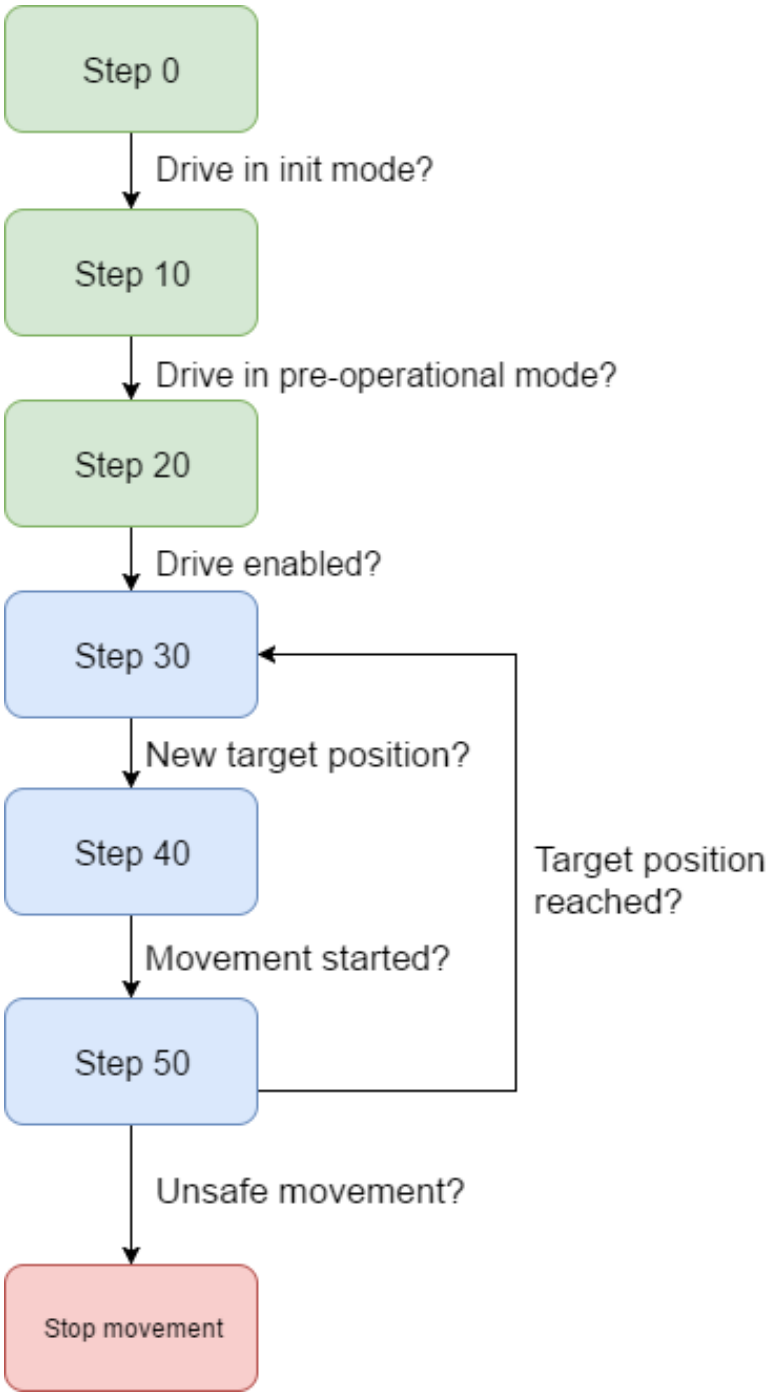

Figure 4.8: Motor control structure

The cyclic data contains the current status code, this code can be used to read error codes. This data is not used in the current implementation, but the drives

will stop if an error is detected as this is handled internally in the drive.
The motor control structure was made with assistance from Bosch-Rexroth during a workshop at their headquarters.

Cyclic inputs	
16 bit	Master control word
16 bit	Signal control word
32 bit	Target position
32 bit	Target velocity

Table 4.5: Cyclic input on drives

The cyclic EtherCAT inputs are shown in table 4.5. It is a total of 96 bits. The contents of the cyclic data is something that is usually set by the master based on a XML file, but because SOEM don't support this type of configuring, the cyclic data has been set using Indraworks DS. The target position is a signed 32 bit integer and interpreted by the drives as 10^{-4}mm. This means the value 1000000 has to be written to move to position 100mm. The target velocity is a unsigned 32 bit integer and interpreted by the drives as $10^{-3}mm/s^2$

Cyclic outputs	
16 bit	Master status word
16 bit	Signal status word
32 bit	Actual position
32 bit	Actual velocity
32 bit	Diagnostics data

Table 4.6: Cyclic outputs on drives

The cyclic outputs are shown in table 4.6. It is a total of 128 bits divided by 5 variables.

Message type	**Topic**
position	uint32
stop	bool

Table 4.7: Motor control node subscribers

Message type	Topic
Position	actual_position
uint32	actual_velocity

Table 4.8: Motor control node publishers

4.7.3 Vacuum control node

The PLC that controls the vacuum generator communicates through EtherCAT. SOEM is used to send and receive the cyclic data inputs and outputs.
The vacuum generator is activated by writing high to the digital output it is connected to. In addition the PLC has a pressure regulator that needs to be opened to a target pressure to provide air to the vacuum generator. The pressure of the air controls how strong vacuum is generated.

The PLC air regulator is opened by publishing a value greater than 0 on the topic *vacuum_signal*. A low value means the regulator will provide low pressure. The vacuum generator is then started by publishing the binary value 0b00000001 on the topic $vacuum_outputs$. Each bit represent one digital output on the PLC. The vacuum generator is connected to output 0 which corresponds to bit 0.

Message type	Topic
Uint8	vacuum_outputs
uint16	vacum_signal

Table 4.9: Vacuum node subscribers

Message type	Topic
bool	good_vacuum
uint16	actual_vacuum_level

Table 4.10: Vacuum node publishers

Table 4.12 shows the cyclic input data provided by the PLC and table 4.11 shows the outputs. The naming can be a bit confusing as it is the cyclic inputs that controls the PLC outputs and vice versa.

Cyclic outputs	
8 bit	Digital inputs
16 bit	Analogue 1
16 bit	Analogue 2
16 bit	Analogue 3

Table 4.11: Cyclic outputs on festo PLC

Cyclic inputs	
8 bit	Digital outputs
16 bit	Regulator set point

Table 4.12: Cyclic input on festo PLC

4.7.4 System control node

The system control node has the job of coordination of the whole system. From receiving a command of the product-ID that needs to be picked, to picking the item from the shelf. The working principle is shown in figure 4.9.

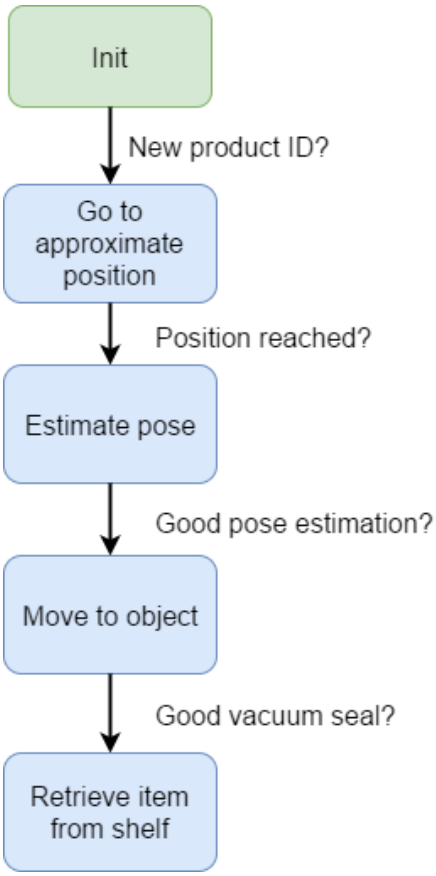

Figure 4.9: System control node

When a new message on the topic "PrID" is published, the control node start by issuing a command to the motor control node to position the robot in the approximate position of the product. When this position has been reached a command will be sent to start the vision node which will find a more precise location of the product and publish that information for the system control node to use. A relative movement command is then issued based on the positional information published by the vision node.
The vacuum generator is switched on when the z-axis is 4cm away from the product. This is done so close to the product to prevent unnecessary use of pressurised air.
If the vacuum node publishes that the vacuum is good, it means that the suction cup has has a good seal and system control node will issue a relative movement command to retract the z-axis.
At this point when the robot has picked up an item, the packing system is supposed to start in order to place the item correct in a box.

4.8 Communication

EtherCAT was chosen as the primary communication protocol used in the system after a study by Henrik.
Listing 4.2 shows the function to write a 32 bit variable to a EtherCAT slave using the cylic outputs. Similar functions are used to write 16 and 8 bit variables. The cyclic data is what connects the ROS software to the internal parameters on the EtherCAT slaves.

Listing 4.2: Write 32 bit over cyclic data

```
void set_output_int32 (uint16 slave_no, uint8 module_index, int32
    value) // Write 32 bits
{
uint8 *data_ptr;
data_ptr = ec_slave[slave_no].outputs;
data_ptr += module_index * 1;
*data_ptr++ = (value >> 0) & 0xFF;
*data_ptr++ = (value >> 8) & 0xFF;
*data_ptr++ = (value >> 16) & 0xFF;
*data_ptr++ = (value >> 24) & 0xFF;
}
```

Listing 4.3 shows how cyclic input data is stored to variables. *ec_slave*[0].*inputs* is the memory address where all input bits are stored. The listing shows how this is broken up to the correct variables. One byte at a time is read and all bytes belonging to the given variable is concatenated. The variables read in the listing is 16 bits, which means each consists of two bytes. As the first variable is 16 bits, +2 has to be added when reading the first byte of the next variable. +2 means we go 2 bytes from the input starting address. This offset is increased further when reading the next variable.

Listing 4.3: Storing cyclic input data

```
control[0] = (*(ec_slave[0].inputs)|*(ec_slave[0].inputs+1)<<8);
status[0]  = (*(ec_slave[0].inputs+2)|*(ec_slave[0].inputs+3)<<8);
```

As SOEM is written in C, it had to be converted to be compiled as C++ to be integrated in the ROS environment.

4.9 Product registration

A great advantage of going with the monocular pose estimation concept is the low need for data when adding a new product to the database. In most cases just a single image of the product will be enough for the SIFT algorithm and matcher to find the same product in a video stream.
The product registration therefore consist of taking a picture of the product directly from the front. This is known as a training image. Taking a picture directly in front of the middle point of the product is difficult without a proper setup. Most images was therefore taken approximately from the front and then manually perspective transformed flat using the image editing software GIMP. The pictures used for this project can be seen in the appendix.
As an alternative to taking pictures and manually editing them, it is possible to use online databases of groceries to source the required data. Mediastore supplies a solution for online grocery shops that provide high quality images of a large group of products. One of the standard images they provide is a picture directly from the front.
The Mediastore database would be a good and fast solution, but it is currently not maintained at the level which is needed by the system. A lot of products are not in the database and some do not have the required front facing picture which is needed by the pose estimation algorithm. The product registration have therefore been done by taking pictures manually.

5. Experiments and results

This chapter presents the experiments done during this project. All experiments were conducted at the robotics lab in Automatmat's office.

5.1 Preliminary test of various feature detectors

The purpose of this test was to get a measurement on how a few different feature detectors compare to each other when used to find the pose of an object that is of interest to this project. The following feature detectors was evaluated.

- ORB
- AKAZE
- BRISK
- KAZE
- SIFT
- SURF

Test setup
A set of 5 images with different qualities was acquired using the a Allied Vision Manta camera. The resolution on all images was 780x580 pixels. A training image was taken following the procedure in 4.9.
It should be noted that all experiments was using the OpenCV implementation of each algorithm. If one algorithm is faster in theory is not that relevant as OpenCV could use a suboptimal implementation.
The following list gives a description of the five images.

1. Control
2. Underexposed
3. Rotated
4. Out of focus
5. Partially visible product

Results

Figure 5.1 shows how many keypoints were found in each of the five images with

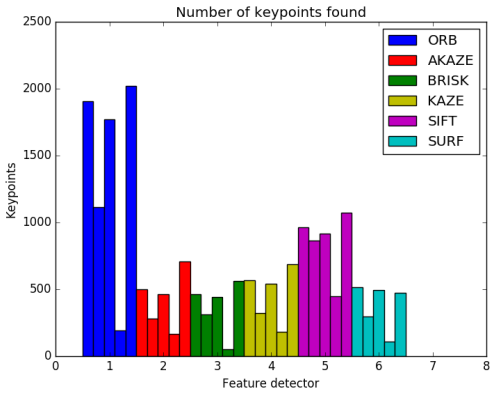

Figure 5.1: Amount of keypoints found

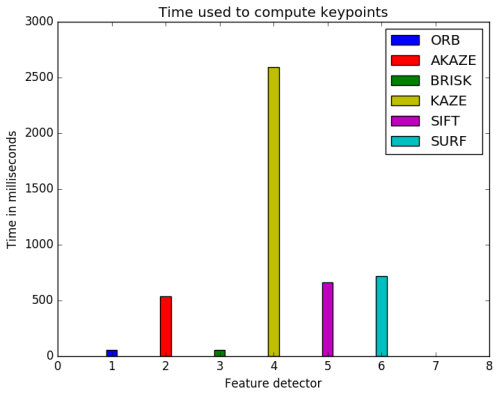

Figure 5.2: Time used to find the keypoints

the respective algorithms. Figure 5.2 shows how much time passed while detecting features. Each column is the result of looking for keypoints in all five images.

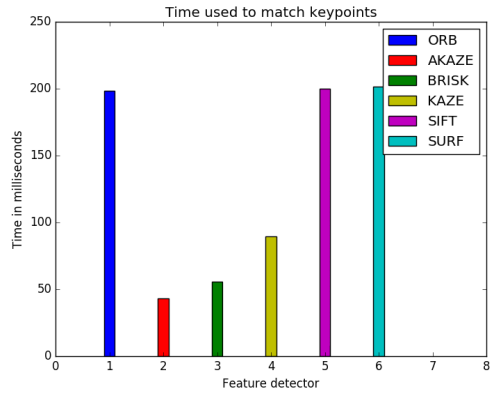

Figure 5.3: Time used to match the keypoints

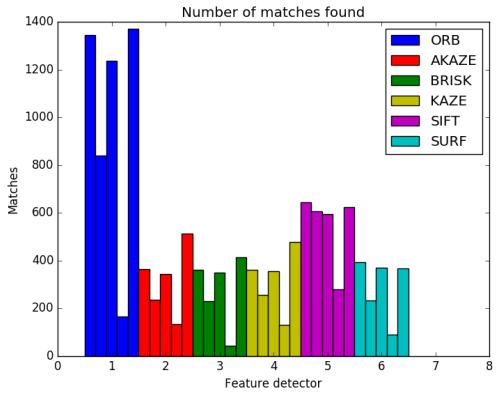

Figure 5.4: Number of matches foundt

Figure 5.3 shows the time spent matching all keypoints using a brute-force matcher. Figure 5.4 shows the amount of matches that was found between the corresponding picture and the training image. It is important to note that the amount of outliers is not evaluated in this test.

Analysis
ORB stood out with the highest number of keypoints found and matches found in four of the five images. SIFT provided the highest number in the most challenging picture. The results from this test does not provide information about the number of outliers in the matches.

5.2 Test on finding pose

The purpose of the following test was to get a more realistic measure on how the various feature detectors and descriptors work in practice by visualising the estimated outline of a product
Test setup
Using the matches found in the previous test, the homography was estimated and used to project lines on the target product. The result is evaluated based on how good the visualised outline of the product fits with the actual outline.
Results

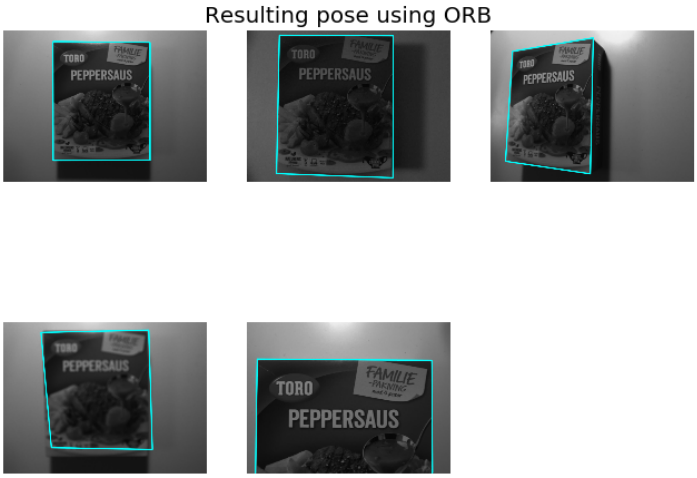

Figure 5.5: ORB

Figure 5.6: System overview

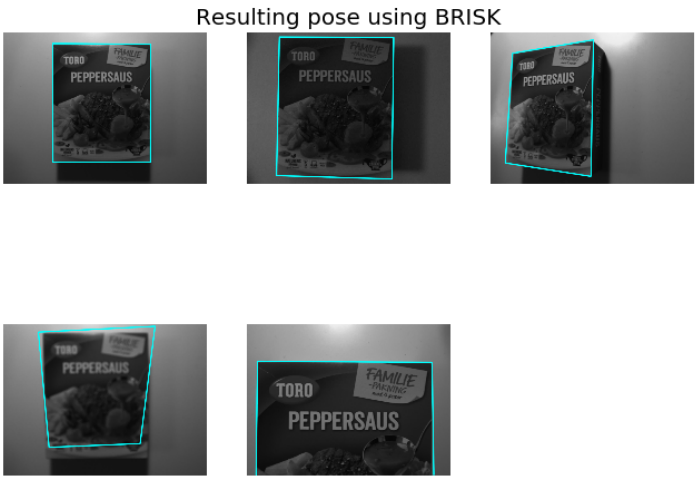

Figure 5.7: System overview

Figure 5.8: System overview

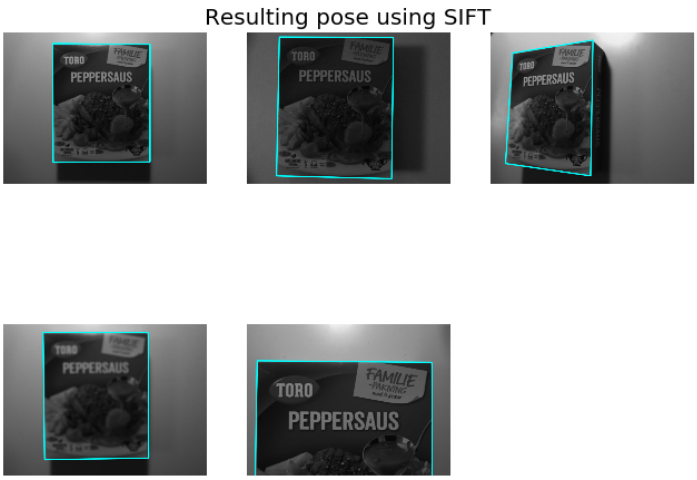

Figure 5.9: System overview

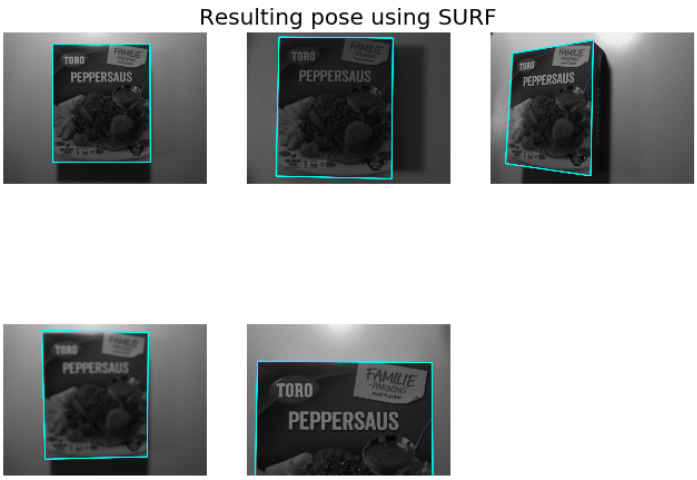

Figure 5.10: System overview

Analysis
AKAZE, SIFT and SURF all provided good estimation of the products outline. SIFT provided marginally the best result. The other algorithms resulted in a bad outline on the image that was out of focus. The reason for the bad estimations is the large amount of outliers in the matches.

5.3 Test of mechanical structure

As stated in the system requirements it is wanted that the robot is able to process groceries weighing up to 1.5 Kg. The following test was done with the intention of figuring out how different loads affect the mechanical structure and robot.

Test setup
A selection of four products weighing 500g, 1000g, 1500g and 3000g was selected for this experiment. One product at a time was attached to the suction cup at the end of the z-axis by using the vacuum system. The distance of the movement downwards on the end of the z-axis due to the added weight was measured. Each product was applied with the z-axis completely retracted and extended. Results is evaluated based on how much the endpoint is moved down, compared to when no weight is applied.

Results

Endpoint displacement		
Weight applied	Retracted	Extended
500g	2mm	2mm
1000g	3mm	4mm
1500g	4mm	5mm
3000g	-	-

Table 5.1: Endpoint displacement test

Analysis
Table 5.1 shows the displacement measured in the two positions. 3000g was too much for the motor brake on the y-axis to handle and it started to slip. It is probable that most of the displacement happens as a result of the aluminium tube or its joint bending because of the similar the displacement is when retracted and extended.

5.4 Test of motor control

The motors and drives was ordered separately from the robot it is therefore of interest looking at how well the performance is in stability and positioning accuracy.

Test setup
One axis at a time was moved to a target position. The speed was set to 50mm/s and acceleration to 100mm/s^2. The actual position provided by the motor control ROS node was plotted against time. Results is evaluated based on how linear the movement is and how the axis settles at the target position. **Results**
Figure 5.11 shows the position of the x-axis in mm plotted against time. The axis was positioned at 500mm and issued a target position of 300mm. Due to the overshoot seen in figure 5.11 an autotuning procedure was conducted on the axis using Indraworks DS. Figure 5.12 shows the results after autotuning. The x-axis drive automatically shut down the motor at 42.5 seconds due to excessive deviation error.
Figure 5.13 shows a position plot of the z-axis when issued a target position of 150mm.

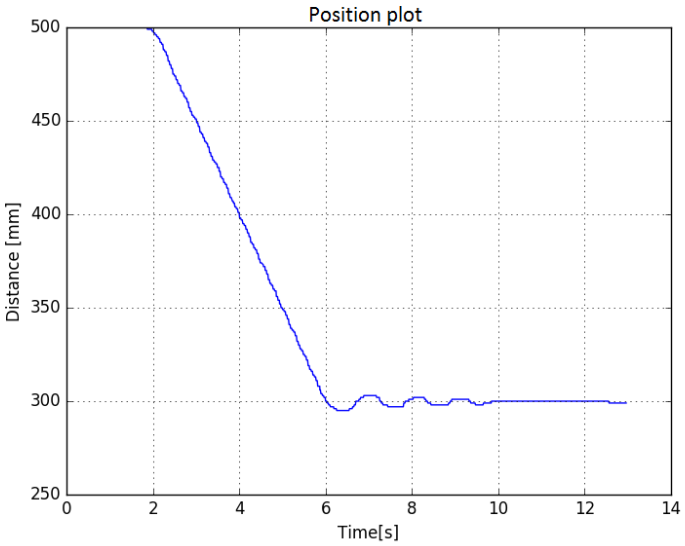

Figure 5.11: Before autotuning

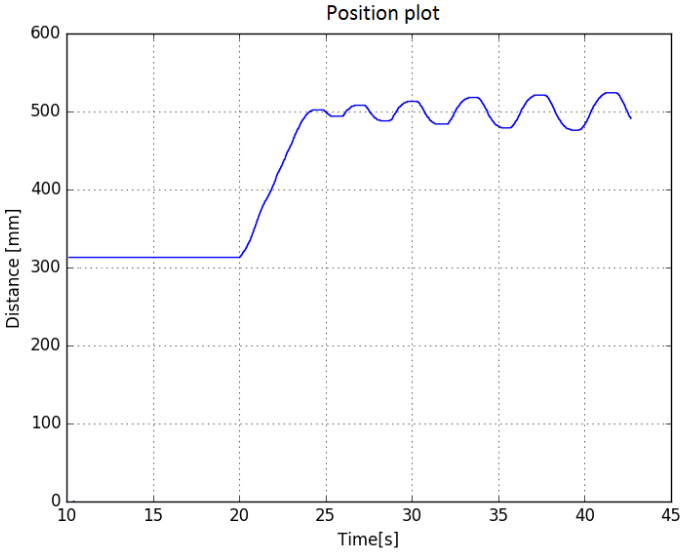

Figure 5.12: After autotuning

Analysis Both Z and Y axis showed a linear path and no sign of overshoot or oscillations. The X axis showed oscillations when reaching the target position and it was therefore conducted autotuning. The results after autotuning showed even more oscillations and the axis became unstable and was automatically shut down. Control parameters was then reverted to standard for use in experiments.

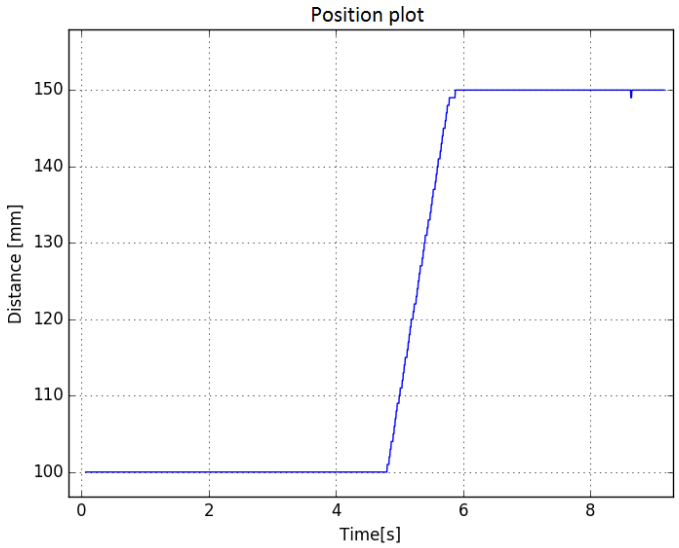

Figure 5.13: Z-axis movement

The direct cause of the overshoot is bad control parameters, but the underlying cause is probably the need for a gearbox. The very low rpm the motor has to operate on, combined with high friction on the x-axis is not optimal for an electric motor. The manufacturer later confirmed the need for gearboxes.

5.5 Pose estimation tests

The purpose of these tests was to determine the accuracy of the pose estimation and also reveal weak areas of the algorithm.

Test setup

A total of ten products with various shapes and size was selected to do experiments on. Training images was acquired using the SR300 camera mounted on the robot. The training images used for the experiments can be found in appendix section

For the first distance test one product at a time was placed at the front of the shelf, 34cm from the camera. The estimated distance was then logged. Then each item was placed 15cm further back in the shelf, 49cm away from the camera. The estimated distance was then logged again. The logged value was the first pose estimation evaluated by the algorithm as good.

For the second distance test one product at a time was placed 34cm directly in front of the camera, and then moved 15cm in X direction. The estimated x and y translation together with the distance was then logged.

In the third test each item was placed directly in front of the camera and rotated by 40° around the Y-axis. The estimated rotation was then logged.

The results are evaluated based on how close the estimated pose is to the actual pose.

Results

Distance test		
Product ID	Estimated distance front	Estimated distance back
0	34.1cm	49.2cm
1	34.0cm	48.8cm
2	34.5cm	49.6cm
3	35.2cm	49.5cm
4	33.9cm	48.5cm
5	35.2cm	40cm - unstable
6	33.6cm	48.8cm
7	34.5cm	49.4cm
8	34.3cm	49.9cm
9	36.7cm	52.5cm

Table 5.2: Distance test 1

Distance test			
Product ID	Measured x translation	Measured y translation 2	Measured distance
0	14.8cm	0.2cm	33.5cm
1	14.7cm	-0.1cm	33.8cm
2	14.8cm	-0.2cm	33.7cm
3	14.5cm	-0.4cm	34.3cm
4	14.4cm	-0.2cm	33.5cm
5	not stable	not stable	not stable
6	14.9cm	-0.5cm	33.4cm
7	14.8cm	-0.4cm	32.0cm
8	15.1cm	-0.2cm	34.7cm
9	15.6cm	-0.2cm	36.0cm

Table 5.3: Distance test 2

Rotation test	
Product ID	Estimated rotation
0	39°
1	37°
2	38°
3	-
4	40°
5	-
6	41°
7	41°
8	42°
9	34°

Table 5.4: Rotation test

Analysis
Results show that the algorithm struggles with the two objects that are round, but provide good results on the rest. Results can probably be improved as there are uncertainties in the accuracy of the manual measured dimensions of the products, also when placing the product manually it is expected to have an uncertainty ±2mm.

5.6 Test of actual picking

The purpose of these tests was to demonstrate the functionality of the system and to identify weaknesses.

Test setup

The system was started as instructed in the start-up procedure manual (8.3) and the test items was placed in the shelf in front of the robot. The approximate position of each item was saved in a array in the system control node. The approximate position was specified so that the product would be in the cameras field of view when the robot moved to the position, but not directly in front of the suction cup. The suction cup used is shown in figure 5.14

Results is evaluated based on successful or not successful pick. A successful pick means the robot is able to locate and pick up the wanted product and retrieve it out the the shelf.

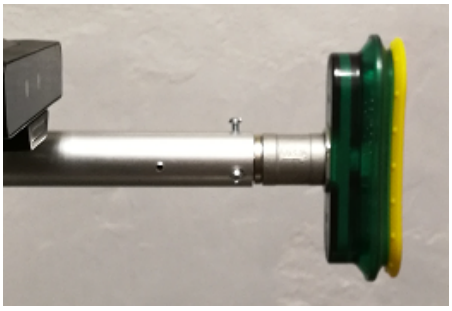

Figure 5.14: Suction cup

Result analysis

All products that gave a good result in the pose estimation tests was picked successfully. Two products was not picked, both of them have a circular shape. The circular shape makes it difficult for both the pose estimation and for the suction cup to get a good seal.

Product number 2 was picked successfully, but only with a suction cup that is better suited for flexible products shown in figure 5.15.

Picking test	
Product ID	Successful pick
0	Yes
1	Yes
2	Yes*
3	No
4	Yes
5	No
6	Yes
7	Yes
8	Yes
9	Yes

Table 5.5: Picking test

Figure 5.15: Suction cup

Video
A short video demonstrating the functionality of the system and a selection of the picking tests done on the final system can be seen at the following link: `https://goo.gl/vQDF44`
The video starts by showing movement across the x-y plane and then movement of the z axis. The rest of the clips are experiments on different products. Each test is initiated by clicking on the wanted product in a simple user interface the publishes the corresponding productID, the rest of the picking sequence is automated as described in the implementation chapter. The robots speed was set to a low value of 50mm/s as a precaution. The last clip demonstrates the speed of the pose estimation algorithm on a regular laptop.

6. Discussion

In the following chapter contains discussion about the system and results from experiments.

6.1 Hardware

Mounting the robot sideways have been successful and tests done when running the robot have shown that the mechanical structure and the robot itself can withstand the forces required by the specifications. It is unknown if this mounting position affects the lifetime of the robot as this is outside the standard operating conditions, but it should be noted the vertical mounting probably introduces more stress than a horizontal mounting.

Camera system
The current solution requires the operator to manually change between looking at the shelf to looking at the ground which is needed for the packing solution. This implementation has been working fine as the two systems are separated and in a testing phase. A simple solution of using two cameras is suggested for future development.
The camera chosen for this project has not been a limitation and is ready for future development as it also has a structured IR light projector that is used in combination with IR camera sensor to generate a depth image.

6.2 Experiments

6.2.1 Preliminary tests and mechanical setup

Two preliminary test was conducted to chose a feature detector for the system. The results from test 1 did not provide enough information to make a decision as it did not reveal how many of the features found was actually corresponding points.
A second test was therefore performed with a more practical approach of doing actual pose estimation. Three algorithms stood out as the best. SIFT,SURF and AKAZE. SIFT was chosen as it was the most robust during testing. The high score of AKAZE is useful finding as it is not licensed like SIFT and SURF. SIFT was chosen for this project as it provided the best result, licensing was not considered because it is not a commercial product and it was of interest to see the performance using the best algorithm available.

6.2.2 Motor and vacuum control

The motor control system has been working as intended, but experiments showed that the x-axis was not moving smoothly and consistently overshot when moving to a target position. After tuning the control-loop parameters an improvement was not registered and it became worse. It is probable that the cause is the gearing ratio from motor-axis to motor axis.
Y-axis will get a temperature warning from just holding its position. This finding revealed the potential for improvement in the motor control software. The proposed solution is adding functionality that applies the motor brake when the y-axis is in its intended position. This should eliminate the high current produced by the drives needed to maintain its position. When the brake is applied no current is needed as the brake will cause enough friction to hold the y-axis in place.
Motors overheat. This finding is important and a response is critical as it halts full-scale testing on the robot. The proposed solution is adding a gearbox on both the x-axis and y-axis to lower the amount of travel per revolution on the motor axis.

6.2.3 Pose estimation

Experiments done have shown good results on the pose estimation given the object has a flat surface. The positional accuracy is roughly ±1cm.

Having a pose estimation algorithm that can handle items of any shape would be beneficial, but looking back at the original objective of making a feasibility prototype, this limitation of the system has been evaluated as not critical at this stage in the development process. There are a lot of products in a store which has a flat surface. It makes sense to start with the easy products as the goal is not to make a single robotic system that can handle all products.

6.3 Test of actual picking

The most important experiment was the actual picking. 8 out of 10 selected items was picked successfully. The two items that failed has a circular shape and was known to be difficult. The result of this test is exiting for the project as it demonstrates actual functionality

7. Conclusion and future work

A Vision guided robotic picking system for groceries has been implemented and tested on a three-axis gantry robot system. With basic knowledge about a product the position and orientation can be estimated using a monocular camera.
The pose estimation algorithm has been evaluated and confirmed by testing as accurate enough to guide the robot for picking items, but has the limitation of being designed to find objects that have a flat surface.

Experiments have shown that the position of items with a flat surface and in the cameras field of view can generally be found within ±1cm.

Motor control over EtherCAT using open software has been successful and proprietary software is only used when commissioning the drives and motors. Tests done have shown that gearboxes are needed for the robot to operate continuously and smoothly.

The vision system, motor control and vacuum control has all implemented in a single ROS system which simplifies future development.
This project demonstrates the basics of an innovative idea that can have a big impact on how people do their grocery shopping in the future.

7.1 Recommendations for future work

As phase 2 of the bigger development project at Automatmat ends one month after this master project is complete this section can be divided in two parts, work that should be done before the end of phase 2 and work that should to be done in phase 3.

Phase 2

A gearbox should be acquired for both x and y axis. It is not as critical with the X-axis, but is definitely needed on the Y-axis as just holding the position requires too much standstill torque and the motor quickly overheats. With the implementation of a 1:10 ratio gearbox the torque required will have a tenfold reduction and it should not overheat as the motor will not be close to its maximum torque.

Even with the addition of a gearbox, the software should also be improved to apply the internal motorbrake of the Y-axis motor when standing still for extended periods without having to shut off the drives.

Picking and packing algorithms should be implemented to a single system and a common database for products should be made.

Error reaction and handling using SOEM library should be implemented. The diagnostics code is part of the cyclic data and is published in the ROS environment, but is currently not used.

Phase 3

For the system to be able to handle most of the products in a grocery store it has to be compatible with a large variety in shape and size of products. A pose estimation algorithm that is capable of this should be developed. The camera system on the robot has 3D vision that can be used in such a system.
As the system gets more industrialised, EAL could be used to create a human-machine interface in parallel with ROS.

8. Appendix

8.1 References

[1] Autostoresystem.com | THE SYSTEM. http://autostoresystem.com/thesystem.

[2] EtherCAT Technology Group | EtherCAT. https://www.ethercat.org/en/technology.html.

[3] License - OpenCV library. http://opencv.org/license.html. [2017-06-14].

[4] ROS.org | History. http://www.ros.org/history/.

[5] N. Correll, S. Member, K. E. Bekris, D. Berenson, O. Brock, S. Member, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. First Amazon Picking Challenge. pages 1–17, 2016.

[6] DrBob at the English language Wikipedia. File:Pinhole-camera.png - Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Pinhole-camera.png.

[7] D. A. Forsyth and J. Ponce. *COMPUTER VISION A MODERN APPROACH*, volume XXXIII. 2012.

[8] Haldir. File:RANSAC LINIE Animiert.gif - Wikimedia Commons. https://commons.wikimedia.org/wiki/File:RANSAC{_}LINIE{_}Animiert.gif.

[9] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2024–2030, 2006.

[10] C. Vision, O. Library, A. Kaehler, and G. Bradski. *Learning openCV 3*.

8.2 Original task

Objective
The objective is to develop algorithms for vision guided picking, product location, and pose estimation of items to be picked as well as selection of point of contact.

For the vision guided picking, the following questions should be had in mind:

- Can items be reliable picked with a single camera setup
- Is the picking by robot reliable enough
- Can the vision system select the point of contact accurately enough
- Is it possible to pick out of supplier boxes from shelf's

Delivery

1. Literature overview
2. Design proposal for vision guided picking
3. Selection of cameras
4. Identification of available space
5. Architecture proposal for algorithms for bin picking
6. Algorithm for bin picking
7. Final report

8.3 System start-up procedure

In order to perform an item pick the following steps should be followed.

Here follows a description of each step in the process.

1. Connect both ethernet cables leading to the drives and Festo PLC
2. Connect source for pressured air.
3. Power on drives and PLC.
4. Start ROS master by the command *roscore* in a terminal window.
5. Start control node with the command *rosrun ethercat_soem control*
6. Start motor control node with the command *rosrun ethercat_soem robotcontrol*
7. Start vacuum control node with the command *rosrun vacuum vacuum*
8. Start pose estimation node with the command *rosrun pose pose*

The system should now be ready to pick up items by publishing productID on the topic $PrID$

In case of error:

- F4002: RTD telegram failure. Try to minimize CPU-usage on the control pc. Restart robotcontrol node.
- F4034: Emergency stop is activated

8.4 List of figures

8.5 List of tables

8.6 Source code

Click on the pin to download the source code, this requires a compatible PDF reader. Adobe reader has been tested and is known to work.

8.7 Training images

PrID: 1

PrID: 2

PrID: 3

PrID: 4

PrID: 5

PrID: 6

PrID: 7

PrID: 8

PrID: 9

PrID: 10

8.8 Test images