Real-time Computer Vision in Industrial Automation

Joakim Cedergren

Jonathan Berglund

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6130
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2021

Abstract

The field of computer vision is growing larger and larger every day, fuelled by lower cost, higher computational power, and more sophisticated cameras. This trend, in tandem with the ongoing rise of Industry 4.0, provides new and exciting opportunities to innovate in areas where this technology has not yet been widely adopted or explored.

This thesis aims to investigate the ways in which real-time computer vision can be utilized in the field of industrial automation, and the benefits and challenges that it brings. Firstly, a literature survey was carried out, exploring previous research to identify trends and applications used in the field today. Secondly, a range of experiments were evaluated to give an overview of the parameters that affect the performance of a vision-based system. Finally, a prototype for a pick-and-place application was implemented to compare a vision-based system with a traditional system.

The results show great potential for the adaptation of real-time computer vision in the field of industrial automation. There is a promising opportunity for computer vision to be able to take a larger role in the field and replace many of the more traditional systems, based on its similar performance and unique characteristics.

Acknowledgements

First and foremost, we would like to thank our two supervisors Daniel Jovanovski at Beckhoff and Anders Robertsson at the Department of Automatic Control. Daniel has provided fantastic support throughout the entire project, helping us progress through invaluable feedback on our work and radiating excitement about the thesis. Anders has given us much appreciated input and guidance during the course of the project, providing unique perspectives and knowledge concerning the thesis.

We would also like to thank Beckhoff's Malmö office for giving us the opportunity to write this thesis. Even though the current global pandemic has hindered our ability to work at the office, we have received fantastic support and resources providing us to work remotely with great success.

Finally, a big thank you to Henrik Sandstedt and Josef Nilsson for reviewing and providing feedback on this report.

Contents

Acronyms

GPIO	General-purpose Input/Output
I/O	Input/Output
IPC	Industry PC
MA	Moving Average
QR Code	Quick Response Code
ROI	Region of Interest
RTOS	Real-Time Operating System
PLC	Programmable Logic Controller

1

Introduction

This chapter will present a brief background to the project and its objectives, as well as an outline of the thesis.

1.1 Background

Ever since the first industrial revolution in the mid-18th century, engineers have tried to push the industry further towards higher productivity enabled by new technologies; the first revolution enabled by steam power, the second by electrical power, and the third by electronics.

Today, with the ongoing rise of Industry 4.0, the revolution based on smart technology, further productivity will be achieved with less human intervention. Engineers seek new ways of expanding and further develop towards this goal, as done during previous industrial revolutions. One of the areas that are being explored is computer vision; a field that may become increasingly important in the near future where more sophisticated cameras, higher bandwidth and faster computation are present.

Beckhoff and Existing Products

Beckhoff is an international company focusing on industrial automation systems. With a worldwide presence in more than 75 countries, they are a significant actor in the automation industry. Beckhoff's products have, since its beginning in the 1980s, revolved around a PC-based control system, providing both hardware and software for its clients. [Beckhoff, 2021c]

One of the latest additions to Beckhoff's software portfolio is TwinCAT Vision, a computer vision system that is unique on the market due to its real-time execution on the same PLC as other tasks, such as motion control [Beckhoff, 2021d]. This real-time aspect is one that enables new possibilities in the field of industrial automation. Since this technology is new, the implications and impact it will have on the industry are not yet discovered and explored. Beckhoff is therefore very keen

on investigating the use of real-time computer vision in industrial applications, to discover the new opportunities this technology enables.

1.2 Objectives

The goal of this project is to further explore what possibilities a real-time computer vision system can open up in industrial automation. This is done through a list of objectives. The objectives of this project is separated into two categories: primary- and secondary objectives. The primary objectives are the main focus of the project, whilst the secondary objectives will be carried out if time allows for it.

Primary Objectives

- There are four main fields of work in industrial automation that are subject to application of computer vision: machine control, robotic guidance, quality control, and code scanning. How can real-time computer vision be applied to each of these four fields?
- Which of the aforementioned fields can benefit the most from real-time computer vision?
- What challenges within the field of industrial automation are not solvable with non-real-time computer vision? What is the limiting factor, and can these challenges be solved with real-time computer vision?
- Implement real-time computer vision on a process that benefits from the implementation.

Secondary Objectives

- Implement real-time computer vision on additional processes.
- Investigate the possibility of closing control loops with real-time computer vision as feedback.

1.3 Thesis Outline

Introduction

In the introduction, the thesis' objectives is presented. A wider context regarding the industry and the company is also given to better understand the motivation for this project.

Background

The background contains relevant theory. This includes a description of industrial automation, typical concepts used in computer vision, filtering methods, and a definition of real-time computer vision.

Literature Survey

This chapter presents the literature survey that was done to identify trends and applications for real-time vision.

Experiments

This chapter contains the background, the aim, and the results for each of the experiments that were performed. The objective is to evaluate the parameters that effect the performance of a vision-based system. The setup, including the software and the hardware, is also presented in the beginning of the chapter.

Prototype

This chapter provides a description of the hardware and software used to create the chosen application, as well as the resulting performance of the system.

Discussion

This chapter contains a discussion regarding the results from the literature survey, experiments, and prototype.

Conclusion

This chapter presents a conclusion to the work presented in this project, as well as future work.

2

Background

This chapter will give a short introduction to industrial automation and the fundamental concepts of computer vision. A description of methods for filtering of measurement noise is also given, as well as the difference between real-time and non-real-time computer vision.

2.1 Industrial Automation

Industrial automation is the act of coordinating mechanical, hydraulic, electrical computational, chemical, biochemical, and/or biological units to produce a given product. This is automated through the integration of measurement, control, and IT in order to run and manage the complex system and processes that the production is composed of [Jeppsson, 2021].

There are four main areas in the field of automation that are subject to implementation of computer vision; robotic guidance, quality control, code scanning, and machine control. These areas will be described in further detail in the upcoming sections.

Robotic Guidance

Robotic guidance refers to guidance of both robotic arms as well as mobile robots. In order for a robot to navigate in an environment, it needs information regarding its position relative to nearby objects [Courtney et al., 1984]. This is achieved through sensory feedback, such as radar [Arreguin, 2008], ultrasonic distance sensors, and computer vision. The data from these sensors can be fed to the control system in order to calculate the position of the robot and/or object in relation to its surroundings. A typical application within robotic guidance is path planning [LaValle, 2006]. This aims to find the shortest, collision-free path between two points. By doing so, the robot can optimize its movement, letting it move at a high speed on its way from one workpiece to another, and at a slower and more precise speed as it gets closer to the object.

Quality Control

Quality control refers to the act of verifying that the manufactured products meet the quality requirements set out by the manufacturer. Methods used to achieve this are many, but a common approach is to measure the product in some way, e.g., measuring dimensions, angles, and surface roughness. The traditional way of quality checking is to manually check a sample in a batch in order to ensure the quality of the entire batch. This method is flawed, since it may lead to high-quality products being rejected or low-quality products passing, depending on what sample is chosen. It is therefore beneficial to quality check every single product, which needs to be done in-line with the manufacturing process; a quality control that can be done with computer vision. By introducing this automation of quality control, the manufacturer can decrease quality check time, increase the certainty that each product is of high quality, and increase efficiency in the system [Tiwari, 2016].

Code Scanning

Code scanning refers to the act of decoding information stored in a binary format, typically done by either a laser scanner or a camera. The codes are commonly used for tracking parts, in inventory applications, and in administrative tasks [Fernández-Caramés and Fraga-Lamas, 2018]. There are many different ways of encoding information on the market, but the most common ones are bar codes and QR codes.

Barcode The barcode is a sequence of black and white stripes that contain information encoded in the width of the stripes, see example in Figure 2.1. The code is divided into several segments with an equal number of stripes, where each segment encodes a character [Pavlidis et al., 1990].

There exist several different variations to the bar code with different complexity and variance, such as two- and three-dimensional bar codes, where the linear bar code is the most common as seen in Figure 2.1 [Diachok et al., 2018].

Figure 2.1: A linear barcode.

QR Code The QR code is a two-dimensional grid that can store information. The grid can be seen as a square matrix where each square is either black or white, see Figure 2.2. The amount of data stored is determined by the grid size, where a larger grid holds more information. The QR code is created by an encoder that generates a QR code based on the data supplied. A scanner can then decode the QR code and retrieve the data. It was initially developed for tracking parts in vehicle manufacturing but has since found many other applications. It was developed due

to the limitations of a regular bar code, which can represent a maximum of 20 alphanumerical characters [Tiwari, 2016].

Figure 2.2: A simple QR-code.

Machine Control

Machine control refers to the act of controlling a machine or process, usually based on sensory feedback in an industrial environment. The goal of the controller is to drive any type of actuator to a certain state, based on the given feedback. This ranges from velocity control of a conveyor belt with an incremental encoder as feedback, to solenoid actuation for release of a substance in a chemical process with temperature as feedback. Robotic guidance, quality control, and code scanning may be regarded as a type of machine control, however, they are excluded from this category in this thesis.

2.2 Computer Vision

Image Representation

The section describes what an image is and how it can be represented. An image consists of a 2D array of pixels where each individual pixel element describes the light in a specific location in the image. There are different formats used to represent a pixel, and below is a description of the three most commonly used in computer graphics.

RGB Color Model The RGB color model is a three-channel color model, where each pixel has a value for red, green, and blue. These three colors are then combined in varying proportions to achieve a certain color. The RGB color model is widely used in display technologies since a pixel typically consists of a red, a green, and

a blue LED, making it simple to vary the intensity of each LED. The disadvantage of this model is that it is sensitive to non-uniform illumination and its non-linear relationship between colors [Yu et al., 2020].

HSV Color Model The Hue-Saturation-Value (HSV) color model was introduced in the 1970s by researchers aiming at designing a color model which interprets color in a manner more similar to how humans perceive color [Joblove and Greenberg, 1978]. Unlike the RGB model, the HSV model separates brightness and color, making it less sensitive to non-uniform illumination, resulting in a more accurate color description. HSV is often represented in a conical coordinate system of color, see Figure 2.3. Hue is the angle around the center axis and represents the dominant color. Saturation is the radius and indicates how true the color is, that is: how far it is from grey. The brightness value expresses the brightness of the color, ranging from 0 to 100, where 100 corresponds to full color and 0 to pure black.

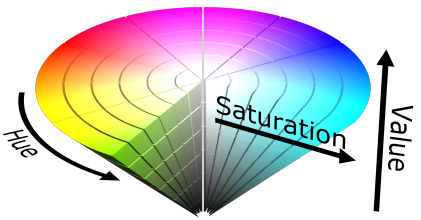


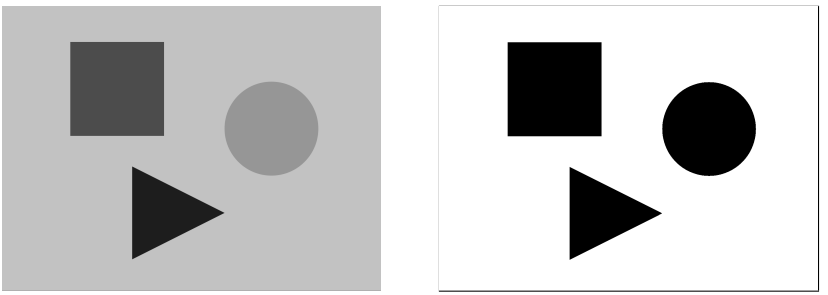
Figure 2.3: HSV Color Model [SharkD, 2015].

Grayscale Image A grayscale image is a one-channel model, where the value of the pixel varies in intensity from black to white. This means that, compared to the RGB- and HSV model, each pixel has only one value. It is often used when the goal is to find a certain attribute in an image.

Image Segmentation

Image segmentation is the process of converting an image of any representation, e.g., an grayscale image, to a binary image where each pixel can assume the value 0 or 1. This is done in order to simplify the image and highlight areas of interest, with the goal of creating an image that is easy for the computer to analyze. This process is one of the most important in the field of computer vision, since a great binary image is key to creating accurate object detection, finding edges, and other image analyses.

Thresholding Thresholding is a very popular method of segmenting an image. The base concept is to set a pixel threshold, where all pixels exceeding the value are set to black or white, and all pixels below the threshold are set to the opposite color. Worth noting is that this method of segmenting has an implicit assumption that there is a significant difference in pixel intensity between the object of interest and its background [Qingmao Hu et al., 2006]. The threshold value can be chosen in various different ways, with the simplest being a binary threshold that uses a fixed threshold value for the entire image, as illustrated in Figure 2.4.



(a) Original image. (b) Binary threshold of 150.

Figure 2.4: Segmenting a gray scale image using a binary threshold.

Blurring

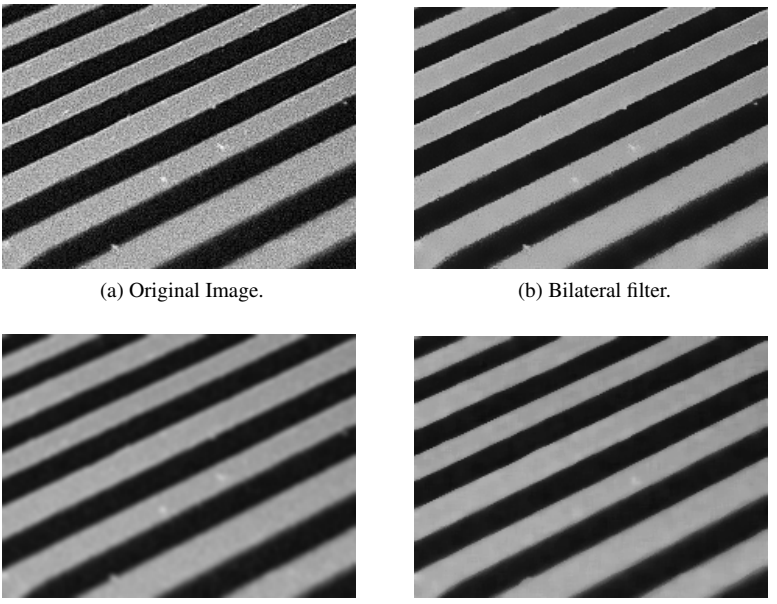
In many cases blurring can be a helpful method for removing noise from an image, making an image more suitable for object detection. It can smoothen out pixels around the edges of an object, creating a more even shape. Three of the most common filtering methods are bilateral-, Gaussian-, and median blurring, which are illustrated in Figure 2.5.

Bilateral The Bilateral filter is a good choice when the goal is to create a smoother image but without losing its distinct edges. The filter calculates a value for each pixel based on a weighted average of the surrounding pixels. The weight is based on how close the surrounding pixel is as well as how large the difference in intensity is [Beckhoff, 2021a].

Gaussian The Gaussian filter is similar to the bilateral filter in the way that it creates a weighted average based on the surrounding pixels and that distant pixels receive a lower weight. The difference, however, is that the filter's weights follow a Gaussian bell curve. This results in a filter where smaller details are removed while larger structures are preserved [Beckhoff, 2021a].

Median A median filter creates a list of values surrounding each pixel, the list is then sorted and the median value of the list is chosen as the new value. One way in

which the median filter is superior to the filters that use a weighted average is that outliers are ignored and are not mixed in into the filtered value. As a result, this filter is very efficient at removing outliers and so-called salt-and-pepper disturbances [Beckhoff, 2021a].



(a) Original Image.

(b) Bilateral filter.

(c) Gaussian filter.

(d) Median filter.

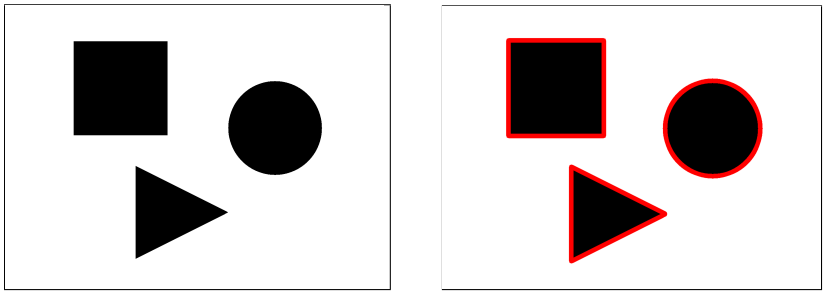
Figure 2.5: Example of the different blurring methods.

Edge Detection

Edge detection aims at locating edges in an image, using algorithms to identify contrasts where there is a steep difference in pixel intensity. It can, for instance, be be used to detect a car-license plate, identity a fingerprint, or finding various shapes as seen in Figure 2.6. There is a wide variety of algorithms to choose from but three of the most commonly used are Prewitt, Sobel, and Canny [Hagara et al., 2020].

Object Detection

Object detection is the concept of identifying objects such as cars, humans or a workpiece in a digital image. It couples localization with classification, aiming at finding characteristics, e.g., shape and colour, of a certain object [Solovyev et al., 2021].

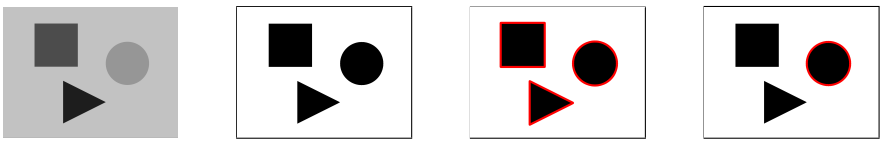


(a) Original image.

(b) Original image with detected edges marked in red.

Figure 2.6: Detecting edges in a binary image.

In order to find an object, a sequence of steps is usually performed. Let's assume that the goal is to find a circle in an image. The first step is to convert the image to a grayscale image. The grayscale image is then binarized and run through an edge detection algorithm. The edges which form a loop and encloses an area are saved in a list of potential objects. The last step is then to examine all the objects that the edge detection has enclosed and compare their attributes against a range of criteria that characterize a circle, e.g., circularity. The objects that remain are presented and the sequence is finished. The sequence is visualized in Figure 2.7.



(a) Original Image.

(b) Image Segmentation.

(c) Edge Detection.

(d) Object Detection.

Figure 2.7: Example of a computer vision sequence which detects circles in an image.

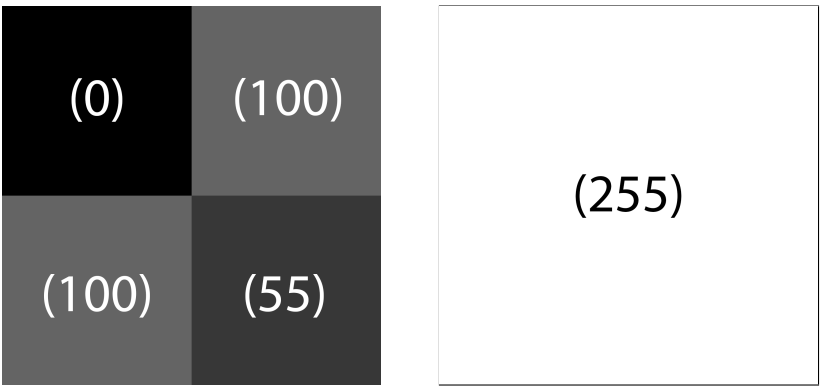
Region of Interest (ROI)

A common practice in the field of computer vision is to define a region of interest (ROI) for the image to be processed. Defining an ROI means selecting an area within the image on which the image processing will be performed. Pixels outside the ROI are excluded from the processing. By defining an ROI that only includes the relevant sections of the image, an increase in performance can be achieved since calculations are performed on fewer pixels.

Binning

Binning is a method used in cameras where the signal from multiple pixels is summed up to form a new pixel. Since more light is registered when adding several pixels together, the light intensity can be increased. A byproduct of binning is that it reduces the resolution of the image since smaller pixels are combined to create a new bigger pixel [Srinivas and Wilson, 2002].

Binning is typically done along both the x- and y-axis but can be done independently as well. An example of a 2x2 binning is shown in Figure 2.8, where four pixels in x- and y direction are summed together to form a new pixel.



(a) The four pixels to bin. (b) The resulting pixel from binning.

Figure 2.8: An example of 2x2 binning.

2.3 Filtering Methods

Kalman Filter

Constant Velocity Model The state vector for a constant velocity model can be expressed as

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.1}$$

where

$$\begin{cases} x_1 = \text{position} \\ \dot{x}_1 = x_2 \\ \ddot{x}_1 = 0 \end{cases} \tag{2.2}$$

As the name implies, the constant velocity model assumes that the velocity is constant during a sampling period Δt . This can be expressed as

$$x(k+1) = \Phi(k)x(k) + w(k) \tag{2.3}$$

where w is the process noise with covariance matrix Q, and Φ is the transition matrix from k to $(k+1)$, which can be expressed as

$$\Phi(k) = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

Given the assumption that the system can only measure position, a measurement model with the measurement vector y can be modelled as

$$y(k) = Cx(k) + v(k) \tag{2.5}$$

where the measurement matrix C and the measurement noise $v(k)$ with the covariance matrix R can be expressed as

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{2.6}$$

$$R = \begin{bmatrix} \sigma_x^2 \end{bmatrix} \tag{2.7}$$

where σ_x is the standard deviation of the measurement noise [Saho, 2018].

Kalman Filter Tracking Given the constant velocity model stated in the previous section, a Kalman filter tracker can predict and estimate the state vector using the Kalman filter equations, which can be expressed as

$$\tilde{x}(k) = \Phi(k)\hat{x}(k-1) \tag{2.8}$$

$$\hat{x}(k) = \tilde{x}(k) + K(k)(y(k) - C\tilde{x}(k)) \tag{2.9}$$

where the Kalman gain K is calculated as

$$K(k) = \tilde{P}(k)C^T(C\tilde{P}(k)C^T + R)^{-1} \tag{2.10}$$

The covariance matrix of errors $\tilde{P}(k)$ is calculated as

$$\tilde{P}(k) = \Phi(k)\hat{P}(k-1)\Phi^T(k) + Q \tag{2.11}$$

$$\hat{P}(k) = \tilde{P}(k) - K(k)C\tilde{P}(k) \tag{2.12}$$

The random acceleration process noise is chosen as the covariance matrix of process noise Q, which is expressed as

$$Q = Q_{RA} = \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 \\ \Delta t^3/2 & \Delta t^2 \end{bmatrix} \sigma_q^2 \tag{2.13}$$

There is no precise method for determining the variance σ_q, however it is typically set based on the assumed target motion [Saho, 2018].

Moving Average Filter

A moving average filter reduces the noise by taking the average value of the last k data points p, as described in the following equation.

$$MA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{2.14}$$

Exponential Filter

An exponential filter reduces the noise by combining the current value with the previous value. The smoothing factor, α, determines how closely the filter follows the raw data. A low value returns a smooth signal but the filter becomes less sensitive to change, a high value returns a filtered signal very similar to the raw data. To avoid instability and oscillations, α should be chosen within the range of $0 \leq \alpha \leq 1$.

$$s_k = \alpha x_k + (1 - \alpha) s_{k-1} \tag{2.15}$$

2.4 Difference between real-time and non-real-time vision

In this report, real-time vision refers to a vision system that is scheduled and executed on a real-time system. This means that the designer is in complete control of the cycle time and priority order of the different processes, which are then scheduled and executed in a deterministic manner by the real-time system. Such a real-time system is often implemented on a dedicated real-time operating system, or in a general operating system's kernel space.

This differs from non-real-time vision where the host operating system's scheduler determines when the process is allowed to execute, and for how long. Such a scheduler does not prioritize in the same manner as a real-time system, meaning that there may be high variance in execution time and latency based on what other processes are running on the system.

3

Literature Survey

This chapter will present the method and results of the literature survey done to identify the trends of using real-time vision in industrial automation. The results are then summarized and three main themes are defined.

3.1 Method

In order to find existing usages of computer vision in relevant applications, a literature survey was conducted. This was done by searching for articles to find studies that meet the following criteria:

- Studied an application in any of the four automation areas mentioned in Section 2.1
- Used computer vision in its application
- Explicitly stated that a real-time implementation was used

Three different platforms were used for the article search: IEEE Xplore, Google Scholar, and ScienceDirect. The search was done in the following steps:

1. Search for keywords in both title and full text. Keywords used include but are not limited to "real-time", "vision", "RTOS". Note that some searches resulted in thousands of results, therefore only the most relevant results were considered.
2. The titles of the search results were assessed to determine if they could meet the criteria.
3. If the title seemed relevant, the corresponding abstract was assessed to judge whether it could meet the criteria.
4. If the abstract seemed relevant, a search for keywords and a brief review of the full text were done.

5. If the full article was relevant, it was summarized and presented in the result Section 3.2.

6. If the article were summarized, the reference list was assessed by following Steps 2–6.

3.2 Results

In this section the result from the literature search is presented. The results are divided into the four automation areas, and further into discrete- and continuous processes. A discrete process refers to a process that executes its action based on one or a few images taken a moment back in time. A continuous process refers to a process where the output is controlled by a continuously updated feed of images.

Robotic Guidance

Discrete Processes In 2019, a study on using computer vision and deep-learning for a pick-and-place robot was conducted at the University of Malaysia, Pahang [Farag et al., 2019]. The study used a SCARA robot and a real-time vision system with a deep-learning model trained to recognize a cylindrical object. The camera was placed with a top-view of the work area in order to allow the robot to move the object from the grasp- to the place region, see Figure 3.1 for the experimental setup. The result of 50 experiments showed that the system achieved 100% successful rate for both pick and place operations, with a detection time of 0.38 s [Farag et al., 2019].

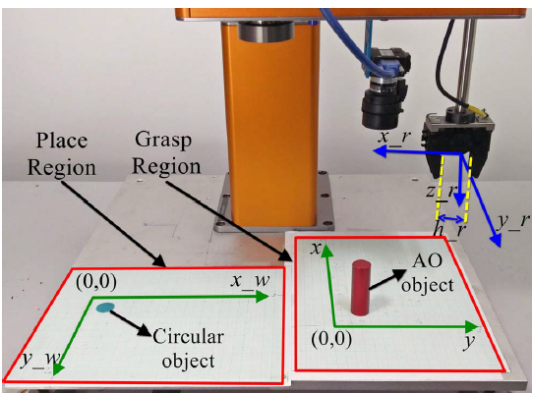


Figure 3.1: Experimental setup for a SCARA pick-and-place robot [Farag et al., 2019], © 2019 IEEE.

Continuous Processes In a study conducted at the Polytechnic University of Valencia, real-time vision was used for guiding robots on a simulated factory floor [Zaera et al., 2001]. The experiment had a camera mounted in the ceiling to provide a top view of the entire floor where both mobile robots and distributed, immovable machines were present. The robot assignment was to supply and gather products to and from the machines. In order to guarantee a collision-free system, the implementation had real-time constraints. By utilizing real-time vision, a system where a central unit successfully guides the robots without collisions was achieved [Zaera et al., 2001].

A PC-based delta parallel manipulator with real-time computer vision was presented at the 2018 WRC Symposium on Advanced Robotics and Automation [Liu et al., 2018]. The system used a delta robot for sorting circles from one conveyor belt to another, using images to determine the circle's position as seen in Figure 3.2. The real-time image acquisition and processing time had a time distribution between 12.1 and 13 ms, achieving adequate real-time performance. The resulting system could pick and place the circles with a frequency of 120 times per minute with a conveyor belt speed of 0.2 m/s, with an average horizontal distance of 30 cm and lifting distance of 20 cm. The controller had the largest control latency of 35.43 μs, and the vision system could process up to 75 frames per second [Liu et al., 2018].

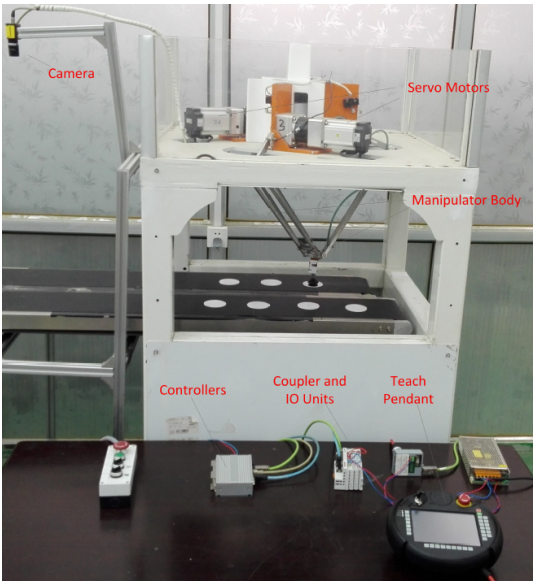


Figure 3.2: Experimental setup for a delta sorting robot [Liu et al., 2018], © 2018 IEEE.

Quality Control

Continuous Processes In a study conducted at the University of Maribor, a plastic granulate sorting system was investigated [Peršak et al., 2020]. The system used real-time computer vision to analyze transparent plastic granulates passing on a transparent, back-lit conveyor belt, see Figure 3.3. The goal was to remove defective granulates at the end of the conveyor belt using pressurized air. Real-time computer vision was used to achieve high enough performance needed for the sorting. The resulting system had an average of only 0.94% contamination in the accepted material. The granulate image analysis which classified defective and accepted granulates had 100% accuracy, meaning it could theoretically separate with 100% accuracy. Any errors that occurred during sorting was therefore a result of other factors, such as adhesion of the granulates to the conveyor and cohesion forces between the granulates [Peršak et al., 2020].

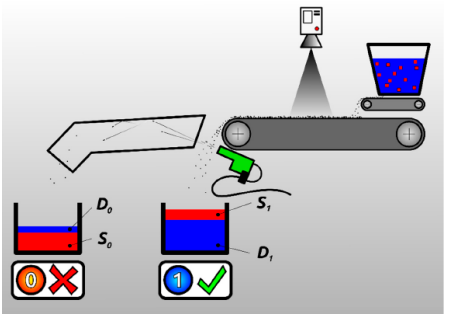


Figure 3.3: Experimental setup for plastic granulate separation [Peršak et al., 2020].

At the San Pablo Catholic University, a study with the goal of automating the selection process of chestnuts was carried out. The experiments used real-time computer vision where a camera was placed with a top-view of two production lines. The production lines had two rollers that rotated the chestnuts as they went by, enabling the camera to capture a 360-degree view of each nut. Some typical defect characteristics were then classified and discarded, such as cracked, rancid, or chipped nuts. The resulting system showed a promising result with a defect detection accuracy of 87% [Álvarez-Valera et al., 2014].

A study was conducted at the RWTH Aachen University in Germany with the purpose of developing tools and concepts for real-time quality control of non-crimp fabrics [Schmitt et al., 2015]. The study aims at providing a solution that has the ability to provide its results in a deterministic rate, something that was not commercially available for optical measurement systems at the time of the study. The system inspects the fabric and locates the position and orientation of the fiber to determine if it satisfies the requirements. All investigated combinations had an uncertainty level below 0.3, which is below the criteria for maximum measurement uncertainty

in industrial usage. The study also investigated how different factors influence the uncertainty of the measurement and came to the conclusion that the material type is the most critical parameter influencing the random-based error whereas aperture, resolution, filter algorithm, and material type had the largest effect on the systematic error [Schmitt et al., 2015].

In 2016, a real-time steel bar recognition and counting system was designed at the University of Jinan in collaboration with Beijing Normal University [Zhao et al., 2016]. The system used a high-speed camera to acquire an image of the steel bars laying on a chain bed, which was transmitted to an IPC. The IPC performed several image processing techniques in order to count the number of bars and sent a stop signal to a conveyor belt when the number reaches the set threshold, see Figure 3.4. In order to satisfy the real-time requirements, different image processing algorithms were compared to find the best one in terms of speed and accuracy. The final design showed a correct recognition rate of 96% for bars with a diameter larger than 15mm and 94% for bars smaller than 15mm [Zhao et al., 2016].

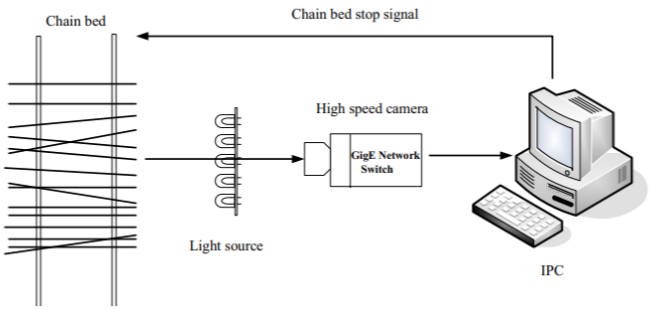


Figure 3.4: Experimental setup for steel bar counting system [Zhao et al., 2016] © 2016 IEEE.

In a study published at the International Computer Science and Engineering Conference in 2019, computer vision was used for quality control in e-jet printing for micro scale fabrication [Lies et al., 2018]. The authors proposed a system where a real-time vision system was mounted to view an e-jet horizontally in order to analyze the flow from the nozzle to the workpiece, see Figure 3.5. The study showed that real-time vision had convincing results for monitoring and real-time quality control, and may be further used in a feedback control system for advanced e-jet printing [Lies et al., 2018].

Code Scanning

Continuous Processes A study conducted in 2017 presents a solution to track the position of mobile robots through QR codes coupled with other sensors

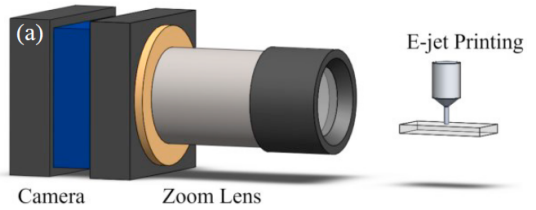


Figure 3.5: Experimental setup for e-jet monitoring system [Lies et al., 2018].

[Nazemzadeh et al., 2017]. The mobile robot used odometry and gyroscope data for continuous position tracking and computer vision to scan the ground in a frontal cone. The vision system searched for QR codes mounted on the floor in order to adjust the robot's position and direction. The experimental setup is shown in Figure 3.6. The study implemented four different algorithms for determining position, where two of them met the real-time requirements and did not miss a deadline. The results showed that a decent accuracy can be achieved and was mainly limited by the low-cost sensors used [Nazemzadeh et al., 2017].

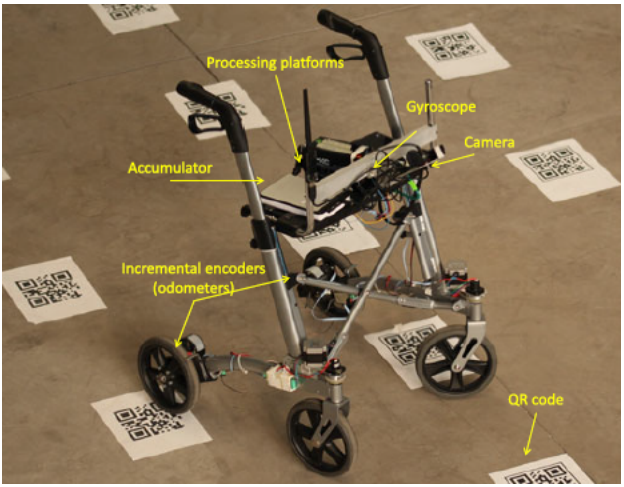


Figure 3.6: Experimental setup for mobile robot with floor mounted QR codes as guidance [Nazemzadeh et al., 2017], © 2017 IEEE.

A vision-based robotic localization system was developed at Pusan National University in 2012 [Rusdinar et al., 2012]. The study proposes a system where an upward-faced camera was mounted on a mobile robot, with custom codes mounted on the ceiling. The real-time vision system scans the ceiling for the codes in order to calculate its position in the room, in combination with data from an odometer and an extended Kalman filter. The results of the study show that it is possible

to successfully localize and navigate a vehicle indoors using the proposed setup [Rusdinar et al., 2012].

In 2013, at the International Conference on Robotics and Biomimetics (ROBIO), a novel real-time navigation methodology was presented [Dzodzo et al., 2013]. In this study, localization was achieved by having a camera pointed at 2D codes mounted to the ceiling, see Figure 3.7. The motivation for this approach arises from an effort at finding a solution where the 2D codes are less sensitive to damage, by equipment or personnel, compared to more common placement such as the floor. Gradient correction and a Kalman-like averaging were added to compensate for non-uniform lightning conditions, thus increasing the system reliability [Dzodzo et al., 2013].

Figure 3.7: Experimental setup for ceiling mounted navigation codes [Dzodzo et al., 2013], © 2013 IEEE.

Machine Control

Discrete Processes An implementation of real-time computer vision for a monomer filling system was created by the King Mongkut's University of Technology Thonburi in 2019 [Kulpraneet and Natsupakpong, 2019]. The study used two cameras to capture four different ROI of back-lit glass molds, as well as a Cartesian robot to pick and place the molds in the filling position. The setup is shown in Figure 3.8. The vision system identified the fill level and stopped filling when the assigned level was reached. The resulting system was capable of processing 52 frames per second by using a real-time system, and did successfully fill the glass molds to the full level. Compared to the existing manual process with skilled workers, the system achieved a 60% productivity increase [Kulpraneet and Natsupakpong, 2019].

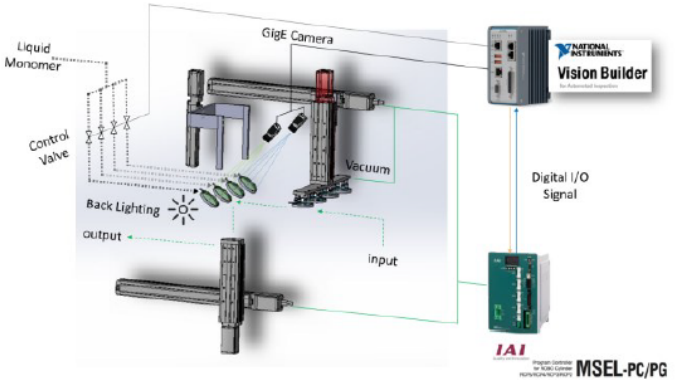


Figure 3.8: Experimental setup for a glass mold filling machine [Kulpraneet and Natsupakpong, 2019], © 2019 IEEE.

Continuous Processes In 2014, a study on using real-time computer vision for variable spray control of fertilizer was conducted at the Chongqing University of Science and Technology [Zhang and Song, 2014]. In this system, vision was used in order to identify the shape or texture of a farmland crop. The goal of the study was to create a solution that saves fertilizer and improves efficiency. The resulting system shows that it is possible to use computer vision in dynamic conditions in order to control the spray flow rate for each individual crop [Zhang and Song, 2014].

3.3 Summary of Applications

To get a better overview of the different applications found in the previous section, a summary was made for each article as seen in Table 3.1.

Themes

Based on the applications listed in Table 3.1, the following themes were found in the different areas. The themes, and which articles have solutions regarding them, are found in Table 3.2.

Feature detection Feature detection can be found in a lot of industrial applications, especially in quality control. The process aims at finding predetermined characteristics in order to classify an object as, e.g., accepted or rejected. In most cases you have an ideal product which the sample is compared to in order to find similarities and differences, examples include identification of shape, size, orientation, and position.

Position and orientation tracking Another theme is the use of computer vision in order to track the position, orientation, and direction of different objects. This is

Automation Area	Process Type	Application	ID
Robotic Guidance	Discrete	Object position detection for pick-and-place robot	R1
	Continuous	Mobile robot localization through top-view vision of floor. Real-time critical	R2
		Object position detection for pick-and-place robot. Real-time critical	R3
Quality Control	Continuous	Plastic granulate defect detection and separation. Real-time critical	Q1
		Chestnut shape detection	Q2
		Position and orientation detection. Real-time critical	Q3
		Object counting. Real-time critical	Q4
		Fluid flow detection. Real-time critical	Q5
Code Scanning	Continuous	Position tracking for mobile robot through floor-mounted QR codes. Real-time critical	C1
		Position tracking for mobile robot through ceiling-mounted codes	C2
		Position tracking for mobile robot through ceiling-mounted codes	C3
Machine Control	Discrete	Pick-and-place and fill level check. Real-time critical	M1
	Continuous	Crop shape and texture identification	M2

Table 3.1: A summary of each article presented in Section 3.2. The articles are numbered in order of appearance in the report.

typically done by finding codes or objects in an image and transform their position on the image to real-world coordinates. The use case of these coordinates is often to manipulate the objects in some manner or to determine where the camera is located in relation to a fixed point in the environment.

Real-time critical All of the presented articles utilize real-time vision, however, the applications can be split up as either real-time critical or not. Real-time critical applications are applications that need to be implemented in a real-time fashion in order to achieve their objective, either because they need high performance or rely on deterministic time conditions, e.g., a task that needs to be executed reliably with a period of exactly 10 ms.

Theme	Article ID
Feature detection	M1 M2 Q1 Q2 Q4 Q5
Position and orientation tracking	R1 R2 R3 Q3 C1 C2 C3
Real-time critical	R2 R3 M1 Q1 Q3 Q4 Q5 C1

Table 3.2: The three major themes and which articles they apply to.

4

Experiments

In this chapter, ten different experiments are presented. These experiments have been chosen with the intent to increase the understanding of what parameters affect the performance of a vision based system for the chosen application. The test setup, including both hardware and software, is presented at the beginning of the chapter.

4.1 Selected Experiments

On the basis of the literature survey in Chapter 3, pick-and-place was selected to be implemented using a real-time vision system. More specifically, a pick-and-place system with an object traveling along a conveyor belt, a down-facing camera capturing the object's position, and a robot grabbing the object further down the conveyor belt. Such a system typically uses an encoder mounted on the conveyor belt's motor axis to measure the velocity of the objects.

The proposed implementation made an attempt to replace the encoder with a velocity measured in a real-time vision system. Before the robot was added to the system, a set of experiments were performed to find which parameters that affect the performance of the vision-based system. An overview of the conducted experiments is presented in Table 4.1.

4.2 General Test Setup

Most of the experiments presented in this chapter used the same base setup, but with slight alterations for each experiment. The base setup is described in this section, and alterations for each experiment is described in its method section. An overview of the setup's architecture is provided in Figure 4.1.

Hardware

The hardware setup used consisted of a linear unit, a camera, ring light, and an IPC with input- and output cards. The linear unit was powered by a servo motor,

Experiment	Section
Time Resolution	4.3
Velocity	4.4
Noise	4.5
Filtering	4.6
Camera Resolution	4.7
Blurring	4.8
Position Lag Error	4.9
Direction Dependency	4.10
Extrapolation of Position	4.11
Response Time	4.12

Table 4.1: Overview of experiments.

connected via a planetary gearbox. A list of models for the components used is found in Table 4.2, and a picture of the setup is shown in Figure 4.2.

Component	Manufacturer	Model
IPC	Beckhoff	CX1800-0511-1009
Input Card	Beckhoff	EL1258
Output Card	Beckhoff	EL2252
Linear Unit	Rollco	RHL80
Planetary Gearbox	Beckhoff	AG3210, NP015S MF2-020
Servo Motor	Beckhoff	AM8021-0B20-0000
Servo Motor I/O Card	Beckhoff	AX5203
Camera	Allied Vision	Mako G-040B
Camera Lens	Computar	H0514-MP2
Ring Light	LAT Elektronik	SAW3 1102

Table 4.2: List of hardware components used in the general setup.

The IPC is the core of the system, running the TwinCAT PLC system. The linear unit and camera was connected to the IPC through ethernet, communicating with the PLC using the EtherCAT protocol. This allowed the IPC to control the linear unit and retrieve pictures from the camera. The camera was also connected to the IPC through three GPIOs. This enabled faster triggering of the camera to take a picture compared to triggering over ethernet.

The input- and output cards used is Beckhoff's EL1258 and EL2252 respectively. These were connected to the IPC via an EtherCAT bus. The input card had its own distributed clock which was synchronized with the PLC. This allowed the card to record the timestamp of an up- or downward flank on any of its inputs in-

between two PLC cycles. Each input had a filter that removed voltage changes that are held for <1 μs, and a distributed clock precision of «1 μs. The output card also had its own distributed clock synchronized with the PLC clock. This card has a functionality of switching a port high or low at a certain time independently from the PLC cycles, with a switching time of <1 μs and distributed clock precision of «1 μs.

The camera used was a GigE camera with a resolution of 728x544. It used a global shutter and had a maximum frame rate of 286 fps at full resolution. The camera also had four dedicated GPIOs, one for input and three for output. These were connected to the IPCs input- and output card, respectively. The input was configured for triggering the camera, and two outputs were configured for indicating when the camera was exposing and when it was ready to be triggered again. The camera was set up to acquire a top-view of the linear unit where approximately 400 mm of the linear unit was visible in the frame. The ring light was mounted around the camera lens to increase the available light in the frame, and thus enabling shorter exposure times.

The linear unit is an aluminium profile with a monorail on top with a mounted slider. The slider has one degree of freedom along the length of the monorail, and has a stroke length of 1 500 mm. The position of the slider was controlled by a servo motor mounted to the linear unit via a planetary gearbox. The servo motor used an absolute encoder for determining the slider's position, which filters the signal internally.

A high contrast black circular printout was mounted on the slider, which was tracked by the vision software.

Software

General System The IPC was running TwinCAT 3 on a Windows-based operating system. TwinCAT runs as a real-time system, which allows the user to separate different tasks such as vision and motion with their own specific cycle times and priorities. A list of all of the specific tasks, cycle times, priorities, and CPU cores used in the general setup can be found in Table 4.3.

Task	Cycle Time [ms]	Priority	CPU Core
Image Processing	30	21	2
Camera Triggering	0.400	12	2
Linear Unit Control	10	20	3

Table 4.3: Tasks used in the general setup's real-time system.

Camera Configuration In order to be able to transform points in an image to position in the real world, a camera calibration is needed. This was done through

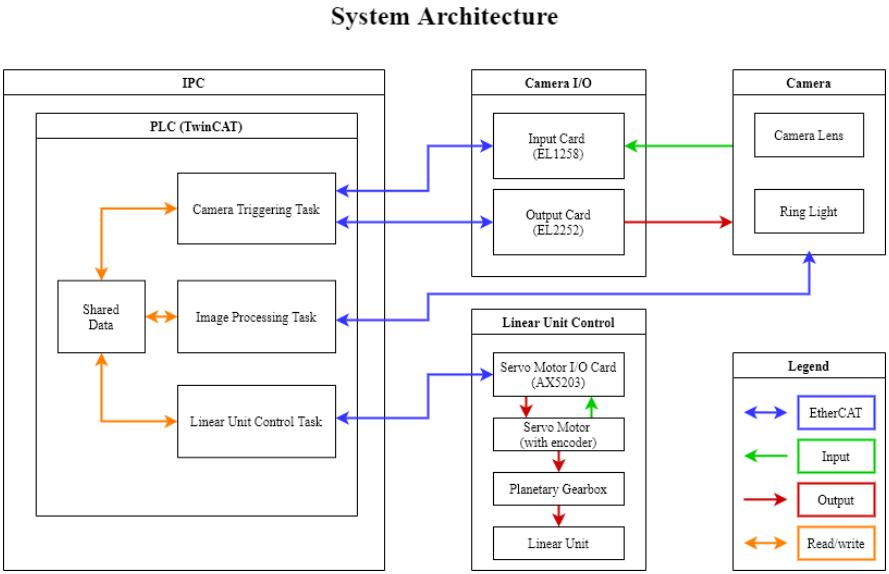


Figure 4.1: System architecture for the general test setup.

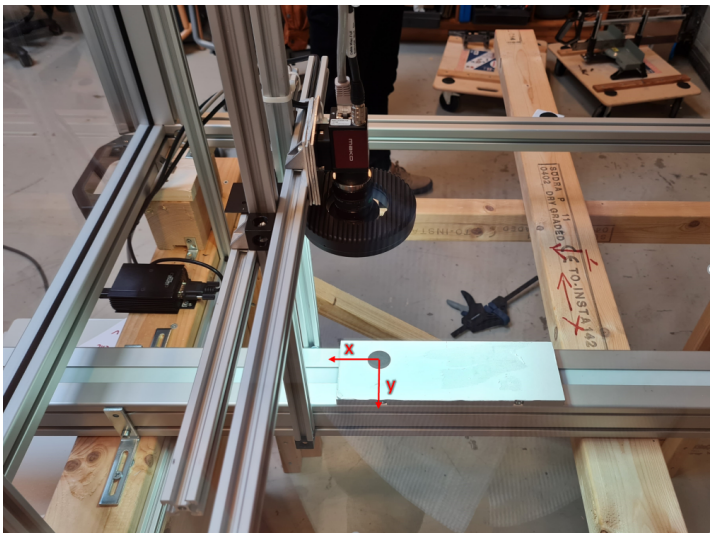
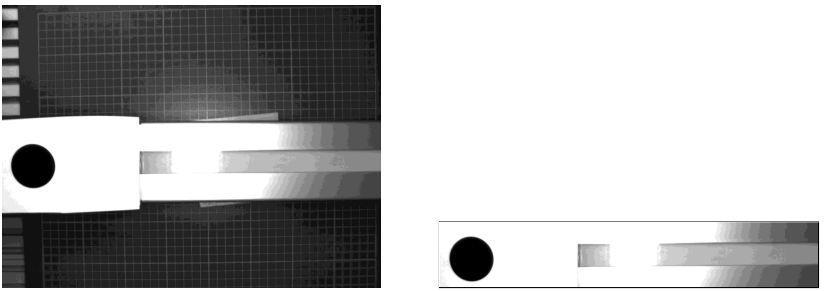


Figure 4.2: View of the general setup.

TwinCAT's internal tools, where multiple images are taken of a checkered board at different angles and positions. This produces four matrices that describe how the image's pixels relates to the real world.

An ROI was also selected in order to reduce the execution time for the image processing task. This was chosen as the smallest area along the linear axis where the circular object will travel across the image. Figure 4.3 shows the image before and after the ROI was applied.

The exposure time used in the general setup was 3.2 ms.



(a) Original image. (b) ROI.

Figure 4.3: Before and after ROI was applied to the image.

Camera Triggering Task This task was responsible of continuously triggering the camera. The task was designed as a state machine with the following states and transitions:

- **Waiting to Trigger** - Waiting for the camera's signal that indicates that it is ready to be triggered. This initiates a state transition to the trigger-state.
- **Trigger** - Sets the camera trigger signal high and waits for an upward flank of the camera's exposing signal. When this is received, the timestamp of the signal is saved and a state transition to the exposing state is initiated.
- **Exposing** - Pulls the camera trigger signal low and waits for the exposing signal to be low. When these two signals are low, the previously saved timestamp is made visible for the image processing task and a state transition to waiting to trigger state is initiated.

A simplified version of the state machine is illustrated in Figure 4.4.

Image Processing Task This task was responsible for processing an image to determine the current position and velocity of the object. This was achieved through the following algorithm:

1. Acquire the image if there is a new image available.

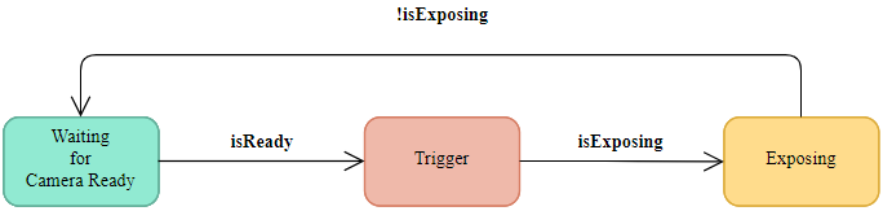


Figure 4.4: Camera Triggering Task state machine.

2. Convert the image to a grayscale image.
3. Segment the image using a binary threshold.
4. Locate continuous, connected edges. Returns the contours of objects with an area above a certain value.
5. Calculate the centerpoint of the tracked object's contour.
6. Transform the centerpoint from pixel location to real world coordinates.
7. Calculate velocity by using difference in distance and time between previous and current centerpoint.

Linear Unit Control Task This task was responsible for controlling the linear unit used in the tests. Its main purpose was to move the trolley back and forth across the camera's field of view between two set points with a set velocity. The task was designed as a state machine with the following states and transitions:

- **Initialize** - Initializes variables and data structures. The state changes to "enable" when user sets an enable signal to true.
- **Enable** - Starts controlling the position of the trolley, and the state changes to "idle".
- **Idle** - Wait for user to start a test with given velocities and positions. When a test is started, the new positions and velocities are used and the state changes to "move to start position".
- **Move to Start position** - Moves the trolley to the start point with a given velocity. When done, the state changes to "move forward".
- **Move Forward** - Moves the trolley to the end point with a given velocity. When done, the state changes to "move backward".
- **Move Backward** - Moves the trolley back to the start point with a given velocity. When done, the state changes to "idle".

A simplified version of the state machine is illustrated in Figure 4.5.

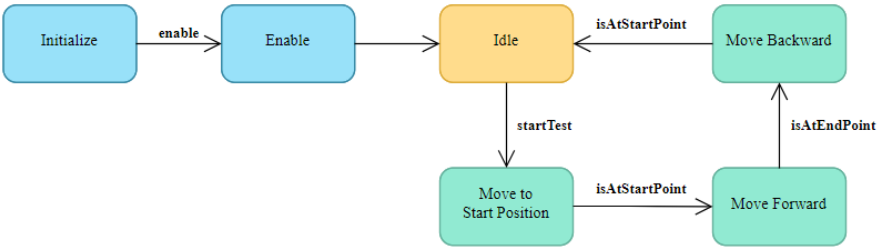


Figure 4.5: Linear Unit Control Task state machine.

4.3 Time Resolution

Background

Since time is needed to calculate the velocity of the system, the resolution of time has an impact on the accuracy. It was believed that an increased resolution of the timestamp for each individual frame increases the accuracy of the measurement. Therefore, this experiment tested how time resolution affects the calculated velocities of the system.

Aim

To investigate the effect time resolution has on the resulting velocity.

Method

For this experiment, the trolley moved from the left edge of the frame to the right, and back, at a velocity of 100 mm/s. This was done five times for each sample time. The data for the time period where the object travels with stationary speed was then extracted and processed.

For Tests 1 – 3 in this experiment, an I/O card without distributed clocks was used instead of the I/O cards listed in the general setup. With this I/O card, the sample time is directly coupled to the cycle time of the Camera Triggering Task task, since it is unable to acquire the timestamp of a signal switch that occurs in-between two PLC cycles. Three tests were executed with this I/O card, where the cycle time (and thus sample time) of the task was changed according to Table 4.4.

Lastly, for Test 4, the I/O card listed in the general setup was used, allowing for a sample time of 1 μs.

Results

The results are presented as a box plot in Figure 4.6. A reference on what a box plot is composed of can be found in Appendix A. As indicated by the figure, a higher time resolution does not result in a more accurate measurement. A probable reason is given in the discussion.

Test	Sample time [μs]
1	200
2	600
3	1 200
4	1

Table 4.4: Sample times used for the tests in the time resolution experiment.

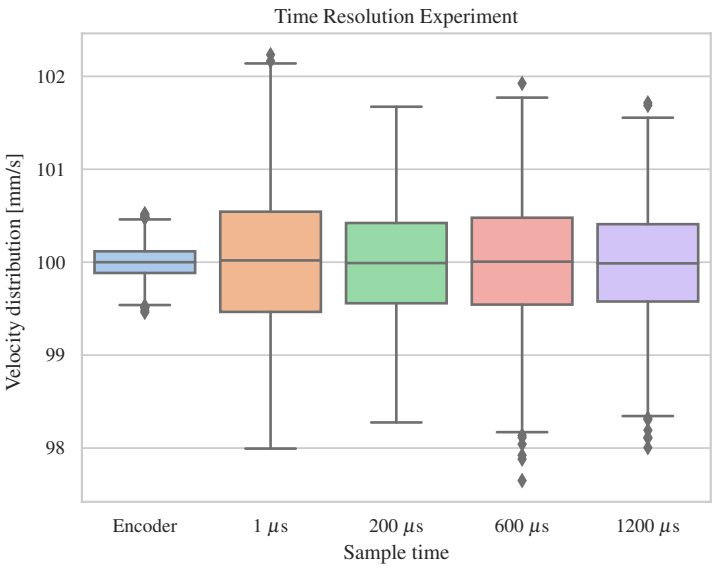


Figure 4.6: Results from the time resolution experiment.

4.4 Velocity

Background

Since an accurate velocity estimation is desired when extrapolating future position of an object, it was interesting to investigate how the object's actual velocity impacts the calculated velocity. It was believed that the velocity has an upper limit where it is no longer feasible to measure velocity with a high enough accuracy. Therefore, this experiment tested how the velocity magnitude affects the accuracy of the measured velocity.

Aim

To investigate the effect the actual velocity of the object has on the calculated velocity.

Method

At high speeds the trolley traverses the FOV relatively fast and few samples are measured. In order to get a higher sample size the cycle time for the Image Processing Task was reduced to 15 ms compared to the general setup of 30 ms. The different velocities that were tested are presented in Table 4.5.

Test	Velocity [mm/s]
1	25
2	50
3	100
4	200
5	400
6	600

Table 4.5: Velocities used for the tests in the velocity measurement experiment.

Results

Figure 4.7 presents the unexpected result that the accuracy increases at higher velocities.

Discussion

A potential explanation for the increase in accuracy at higher velocities is that the noise in position measurement makes up a larger proportion of the velocity at a lower speed.

4.5 Noise

Background

It was believed that some of the uncertainty in the measurements originates from noise in the reading of the position. The estimation of the center point may vary due to inconsistent edge detection. It was also of interest to see if the object's relative position in the image influences the magnitude of the noise.

Aim

To analyze the effect signal noise has on the measured position in different parts of the image.

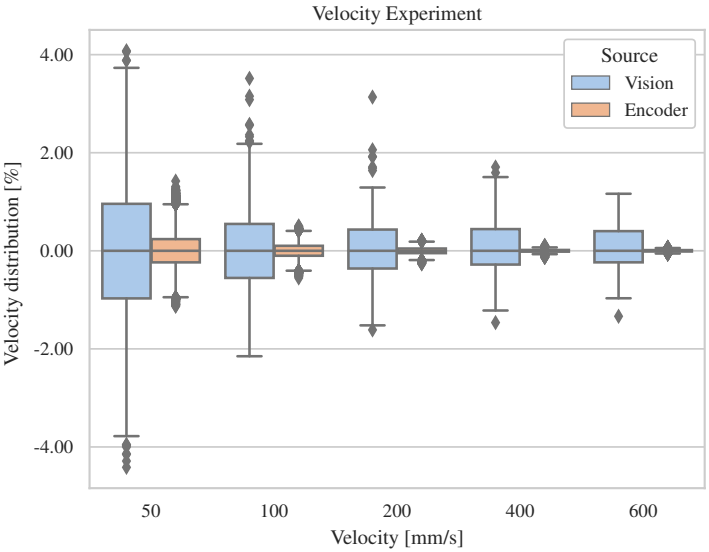


Figure 4.7: Results from the velocity experiment.

Method

The object was pre-positioned in five different parts of the image. The chosen positions have different exposures due to lighting conditions and were positioned at different distances and angles relative to the camera. The chosen positions are presented in Table 4.6. Note that the ROI spanned approximately $\pm$ 185 mm from the picture center.

Test	Position from picture center [mm]
1	-120
2	-60
3	0
4	+60
5	+120

Table 4.6: Positions used for the tests in the noise experiment.

Results

Figure 4.8 shows that the uncertainty derived from noise in the measurement signal is relatively low, with a standard deviation less than 0.00324 mm. The experiment

also indicated that the magnitude of the noise is independent from the position in the image.

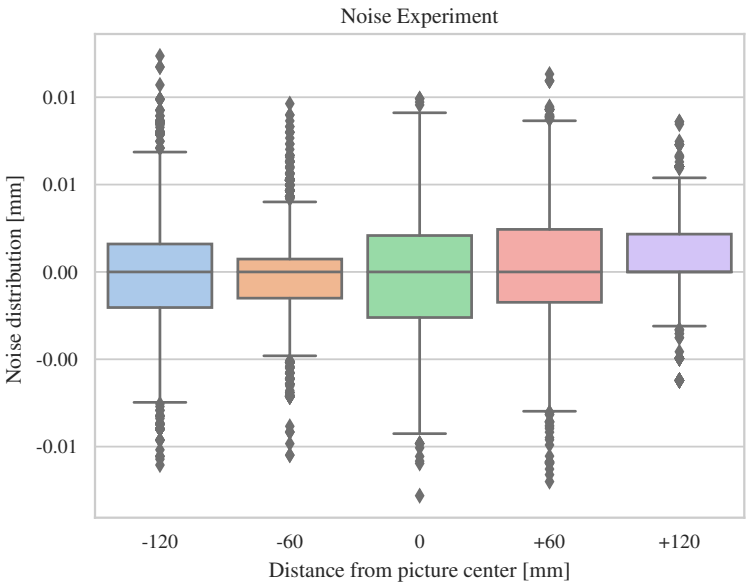


Figure 4.8: Results from the noise experiment.

4.6 Filtering

Background

As with any measured signal, the position of the object is noisy. In order to smoothen out this noise, low pass filtering is often used. There are many different filtering methods to use, which process the input in different ways. It was therefore interesting to investigate how the filtering methods affect the spread of the velocity, compared to the raw input signal.

Aim

To investigate how different filters affect the spread of the velocity.

Method

For this experiment, the trolley passed back and forth across the FOV at a stationary speed of 100 mm/s five times. The object's position in the current frame and time

elapsed since the previous frame is extracted from these five tests. These data sets were then processed to create the different filtered signals. The filtering algorithms used were an exponential filter, a Kalman filter, and a moving average filter.

Kalman Filter The model used for the Kalman filter was a constant velocity model with position-only measuring as described in Section 2.3. Based on the expected velocity of 100 mm/s, the initial values of the state vector was chosen as

$$x(0) = \begin{bmatrix} 0 \\ 100 \end{bmatrix} \tag{4.1}$$

The covariance matrix R of the measurement noise $v(k)$, i.e., velocity noise, with the standard deviation σ_x, was calculated based on the results presented in Figure 4.6, subplot 1 μs.

$$R = \begin{bmatrix} \sigma_x^2 \end{bmatrix} = \begin{bmatrix} 0.770^2 \end{bmatrix} \tag{4.2}$$

The covariance matrix Q of the process noise $w(k)$ was assumed to be the same as the measurement noise, meaning that $\sigma_q = \sigma_x$. Δt is the time between the two latest acquired frames.

$$Q = Q_{RA} = \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 \\ \Delta t^3/2 & \Delta t^2 \end{bmatrix} \sigma_q^2 = \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 \\ \Delta t^3/2 & \Delta t^2 \end{bmatrix} 0.770^2 \tag{4.3}$$

Based on empirical observations, the initial value of $\tilde{P}$ was chosen as

$$\tilde{P}(0) = \begin{bmatrix} 10000 & 1 \\ 1 & 1 \end{bmatrix} \tag{4.4}$$

The predicted velocity $\tilde{x}_2(k)$ and estimated velocity $\hat{x}_2(k)$ was then calculated by using Equations 2.8 and 2.9 and continuously updating the Kalman gain $K(k)$, predicted error covariance $\tilde{P}(k)$, and estimated error covariance $\hat{P}(k)$ according to Equations 2.10, 2.11, and 2.12. Φ and Q as stated in Equations 2.4 and 2.13 were also updated continuously, since Δt is not constant.

Exponential Filter The model used for the exponential filter was the model presented in Section 2.3. The smoothing factor α in Equation 2.15 was chosen as $\alpha = 0.2$, based on empirical observations. This results in the model being described as the following equation.

$$s_k = 0.2x_k + (1 - 0.2)s_{k-1} \tag{4.5}$$

Moving Average Filter The model used for the moving average filter is the model presented in Section 2.3. The last k data points p to average was chosen as $k = 15$ for Equation 2.14, since 15 samples was the maximum amount of data points gathered

when tracking the slider across the FOV at the maximum velocity. This results in the model being described as the following equation.

$$MA_k = \frac{1}{15} \sum_{i=n-15+1}^{n} p_i \tag{4.6}$$

Results

As expected, the experiment showed that a filter can greatly reduce the measurement uncertainty. As seen in Figure 4.9, the Kalman filter was the superior filter in this use case with a standard deviation of 0.0907 mm, to be compared with the unfiltered signal which has a standard deviation of 0.7321 mm. Figure 4.10 shows how the filters' velocity changes over the course of one traverse across the FOV.

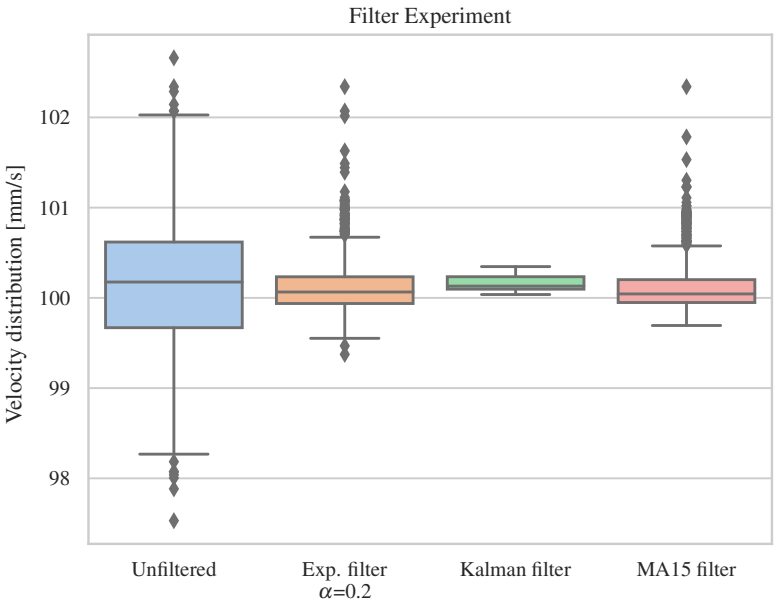


Figure 4.9: Results from the filtering experiment, velocity distribution.

4.7 Camera Resolution

Background

When you want to find a suitable camera for a vision project, you usually have to choose between a fast or a high resolution camera. With a higher resolution, more

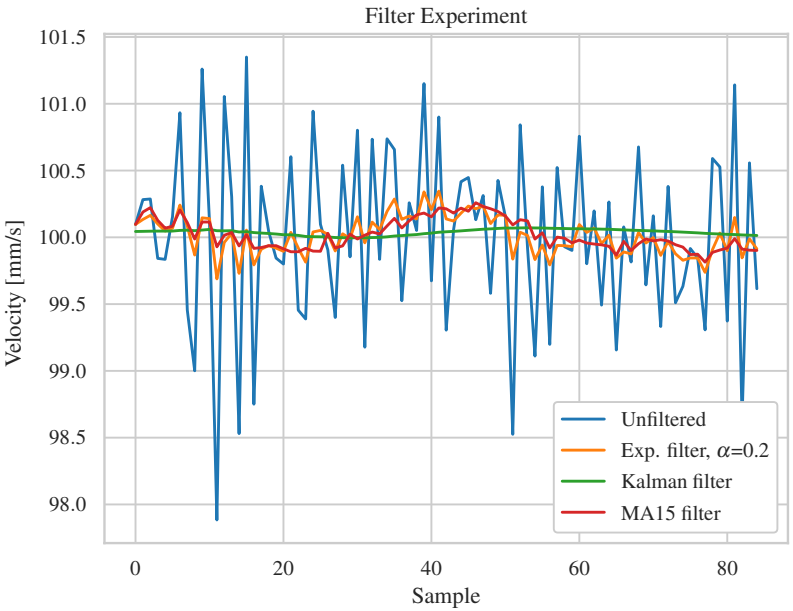


Figure 4.10: Results from the filtering experiment, one traverse across the FOV.

pixels are taken into consideration and thus, a more precise localization of the edge of the object can be found. A more accurate edge estimation should result in a more accurate center point estimation and thus a more precise velocity estimate.

Aim

To investigate if the camera resolution has an impact on the accuracy of the velocity measurement.

Method

For this experiment, binning was utilized in order to reduce the camera resolution. This allows for the possibility to test different resolutions using the same camera and setup, which is favorable when you want to single out a certain parameter. When the image is binned, the overall exposure is increased. To keep the brightness constant for all tests, the exposure time was reduced a factor equal to the amount of pixels that are merged into one. The exposure times and binning modes tested is presented in Table 4.7.

A considered alternative to using binning were to test different cameras with different resolutions, however this adds uncertainty to the experiment since there is a risk that other contributing factors are added.

To measure the velocity, the trolley traveled back and forth across the FOV five times at 100 mm/s for each binning mode.

Binning Mode	Exposure Time [ms]
1x1	3.2
2x2	0.80
4x4	0.20

Table 4.7: Exposure times used for the different binning modes in the camera resolution experiment.

Results

The results from the experiment showed that the distribution of the calculated velocity increases rapidly with decreasing resolution, as seen in Figure 4.11.

When plotting the standard deviation of these results, as presented in Figure 4.12, it is a line with the slope 0.77, which means that the accuracy increases as the camera resolution increases, as expected.

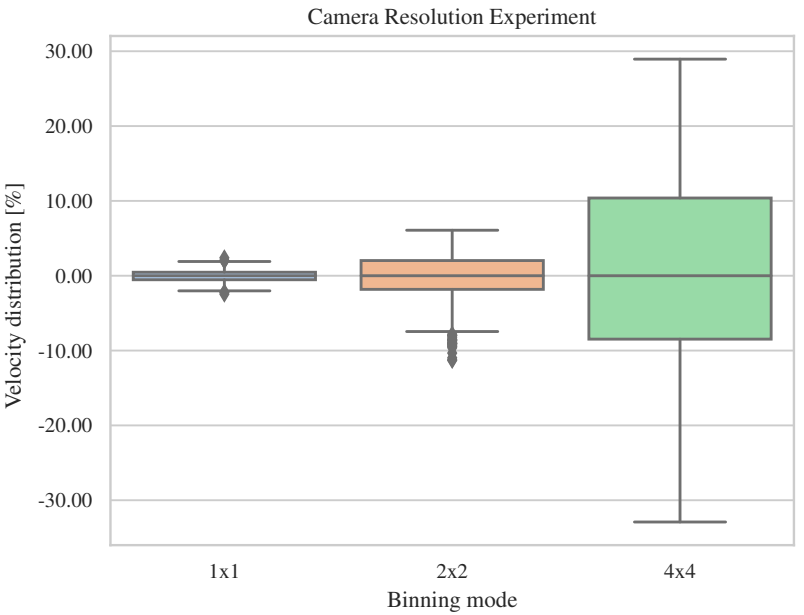


Figure 4.11: Results from the camera resolution experiment, velocity distribution.

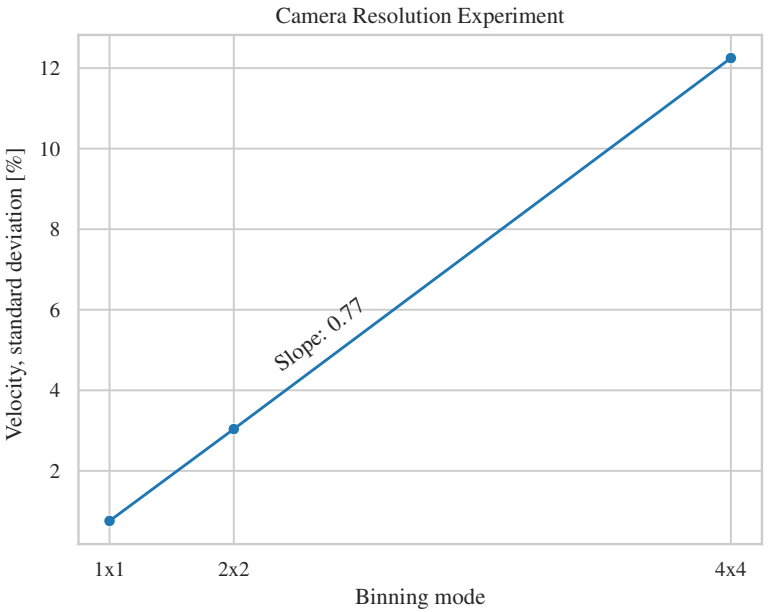


Figure 4.12: Results from the camera resolution experiment, plot of standard deviation.

4.8 Blurring

Background

Blurring is a standard method used to remove noise and improve edge detection in an image. The downside however, is that it is a computational heavy method and can lead to an increased execution time. This experiment was therefore looking both at the effect blurring has on the accuracy as well as the execution time.

Aim

To investigate if the blurring can reduce noise in the measured position and thus improve accuracy for the velocity, as well as the blurring effect on the execution time.

Method

For this experiment, three different blurring methods were investigated: Bilateral-, Gaussian-, and Median filter. Since this experiment did not aim at finding the optimal solution but rather at investigating the effect and comparing different options, it

was decided that the default values, provided in Beckhoff documentation [Beckhoff, 2021a], were to be used as the input parameters for each blurring method.

Results

The velocity distribution was almost identical to the no blur experiment for all the different blurring methods, as seen in Figure 4.13. However, the execution time varies considerably, ranging from 2 ms to 16 ms for the different blurring methods as presented in Table 4.8.

Discussion

It is believed that the blurring has a negligible effect on velocity due to the already high contrast object, which is black on a white background. Using blurring in this case would therefore be unnecessary and only worsen the image processing execution time.

Blur Method	Execution Time [ms]
No blur	1.2
Bilateral filter	16
Gaussian filter	2.0
Median filter	13

Table 4.8: Measured execution times for the different blurring methods.

4.9 Position Lag Error

Background

A common phenomena for objects in motion is that a lag error is introduced, which is the error between the measured position and the actual position due to signal- and mechanical latency. It was believed that this error is increased with greater velocities, which was studied in this experiment.

Aim

To investigate the magnitude of the lag error in the position measurement.

Method

The experiment was set up to have the trolley moving at a stationary velocity when it enters the FOV, and moves back and forth across the FOV five times for each velocity. During the movement, the lag error was calculated as the difference between the position calculated from the image processing and the position from the encoder. Five different velocities were tested, which can be found in Table 4.9.

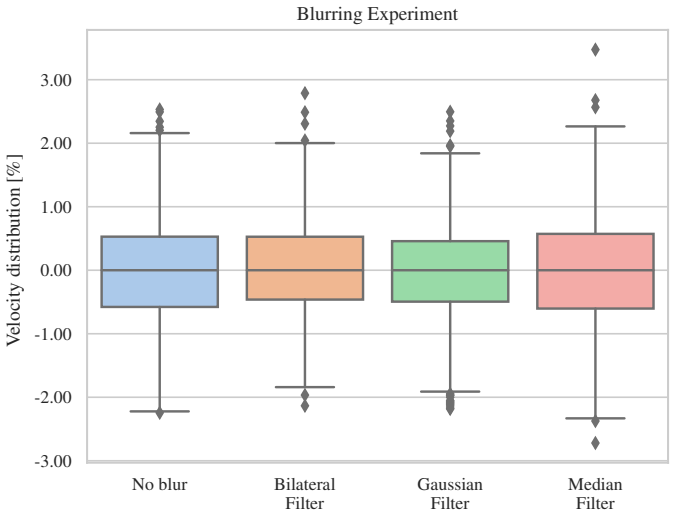


Figure 4.13: Results from the blurring experiment.

Test	Velocity [mm/s]
1	50
2	100
3	200
4	400
5	600

Table 4.9: Velocities used for the tests in the position lag error experiment.

Results

As expected, Figure 4.14 shows that the position lag increases at higher speeds. However, the experiment also discovered an interesting phenomena where the magnitude of the lag differed depending on the direction of the movement of the linear unit, this observation was further investigated in Experiment 4.10.

4.10 Direction Dependency

Background

A phenomena observed in Experiment 4.9 was that the lag error differed significantly between the left and right movement direction across the image. A possible

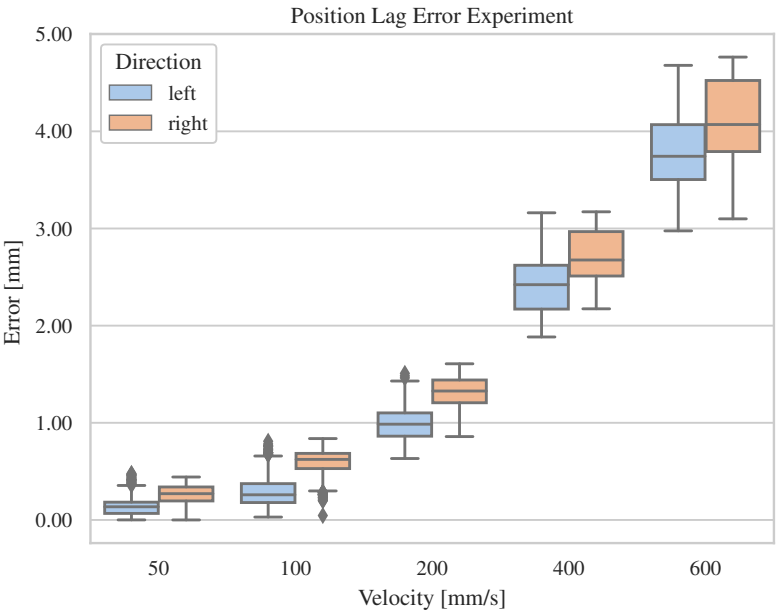


Figure 4.14: Results from the lag error experiment.

factor that contributes to this was the image processing, which may be direction dependant. This experiment was therefore looking at whether the image processing introduces this difference or not.

Aim

To investigate whether the image processing introduces a directional error.

Method

For this experiment, the trolley was positioned at a position approximately right below the camera with an approaching direction from the left. The trolley was then moved to -100 mm and back to center. It was then moved to position +100 mm and back to center. The main point of interest was to observe the position error after each of the two movements, since the first movement approaches the center from the left, and the second movement approaches from the right. The velocity used was 100 mm/s.

To analyze the direction dependency of the image processing, the image was analyzed both normally and after being rotated 180 degrees.

Results

The results showed that there was a substantial error between the position extracted from the image compared to the encoder when approaching from different directions, as seen in Figure 4.15. The image processing does not seem to affect the direction error as the direction error can be observed in the rotated image as well, see Figure 4.16.

Discussion

A possible explanation of this is backlash in the servo motor transmission when changing direction, resulting in the encoder recording a motion that is not propagated to the linear unit's trolley.

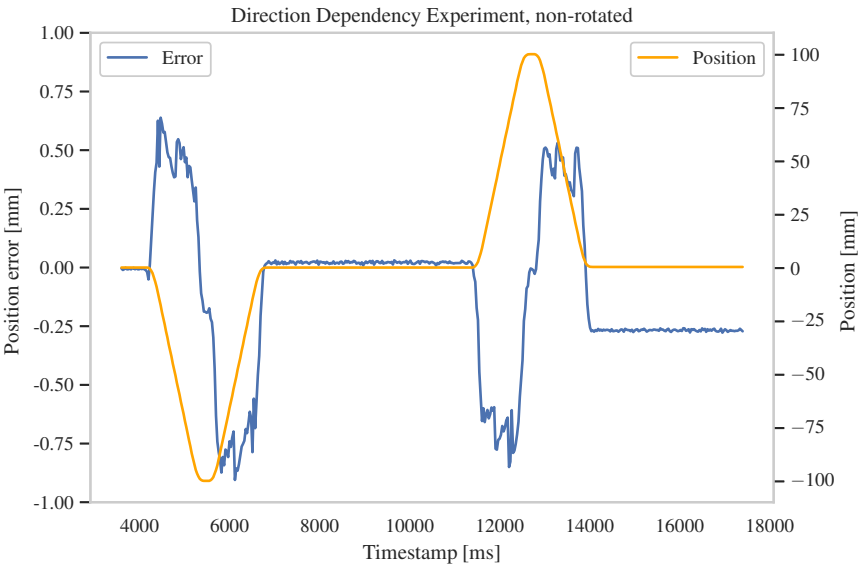


Figure 4.15: Results from the direction dependency experiment, non-rotated image.

4.11 Extrapolation of Position

Background

One of the identified applications for real-time computer vision found in Chapter 3 was a pick-and-place robot. By extrapolating the position where an object should be at a given time in the future, a robot outside the FOV further down the line can

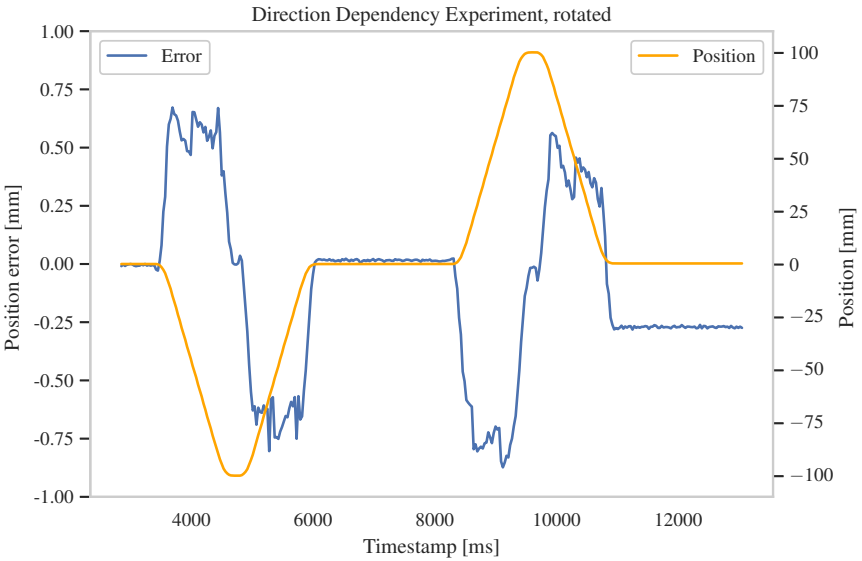


Figure 4.16: Results from the direction dependency experiment, rotated image.

pick up the object. This is under the assumption that the extrapolated position is accurate enough for the given object and application. It was therefore interesting to determine the accuracy of this extrapolation.

Aim

To investigate the accuracy of the extrapolated position.

Method

For this experiment, the trolley moved across the FOV and further along the linear unit at a stationary velocity. The velocity was calculated continuously when the trolley was within the FOV by using the position difference and time difference between the two latest images. After leaving the FOV, the estimated position was extrapolated based on the estimated velocity and time passed. Three different velocities was calculated using a simple average, median, and the best performing filter, i.e., Kalman filter from Section 4.6.

The position estimation was then compared with the actual position measured by the encoder. The difference between the measured and estimated position was calculated between exiting the FOV and 325 mm past the FOV. This was done five times each for three different velocities, as presented in Table 4.10.

Test	Velocity [mm/s]
1	100
2	200
3	400

Table 4.10: Velocities used for the tests in the extrapolation experiment.

Results

The results showed that Kalman had the best performance, with the lowest spread and an average error per meter of about 0.4 mm, see Figure 4.18. The median filter came second with an average error per meter of about 0.55 mm, see Figure 4.19. The worst performing filter was the simple average filter having an average error per meter of about 1.2 mm, see Figure 4.20.

To visualize how the extrapolation error increases with increasing distance from FOV, a single data set's extrapolation using Kalman-, median-, and average filter is plotted, see Figure 4.17.

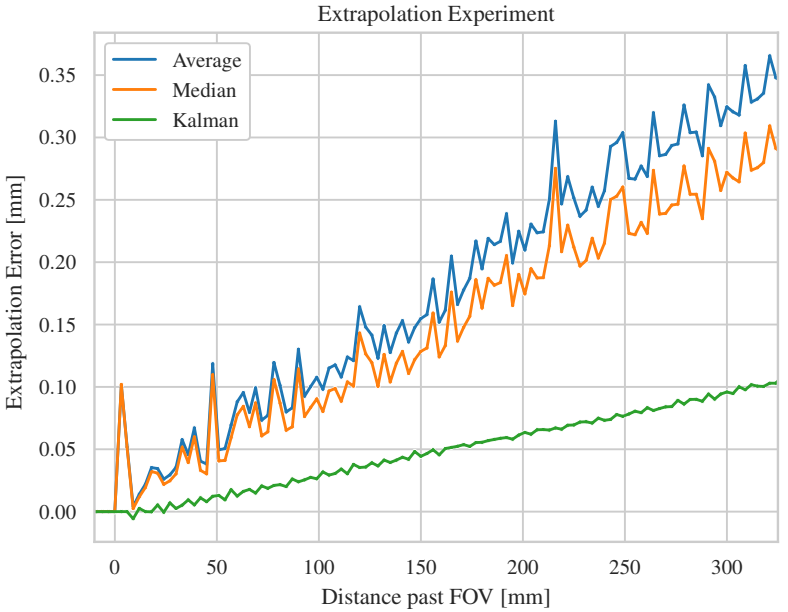


Figure 4.17: Results from the extrapolation experiment. Kalman, median and average filter's extrapolation for one traverse.

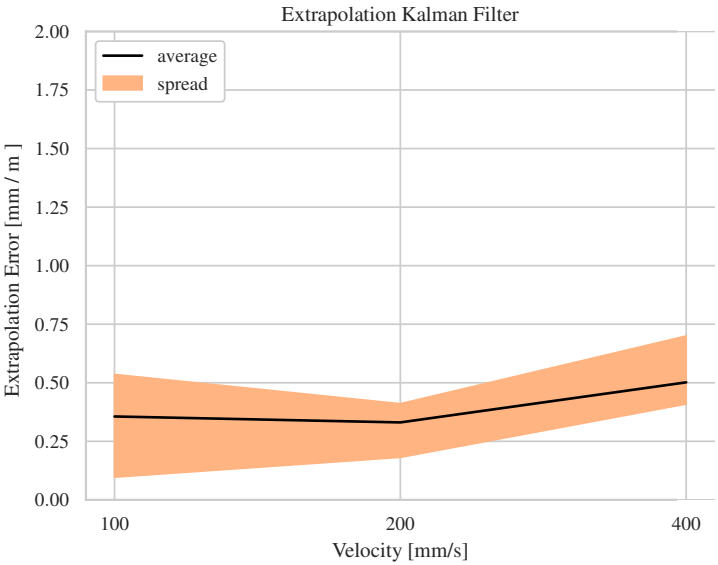


Figure 4.18: Results from the extrapolation experiment, Kalman filter.

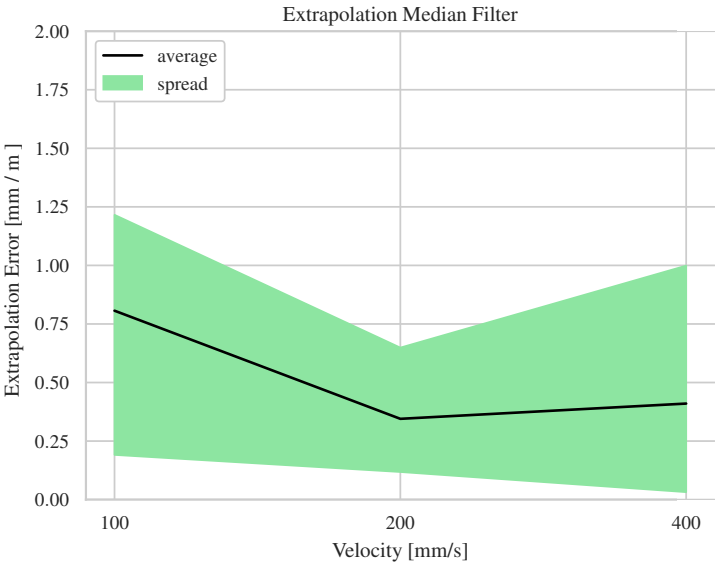


Figure 4.19: Results from the extrapolation experiment, Median filter.

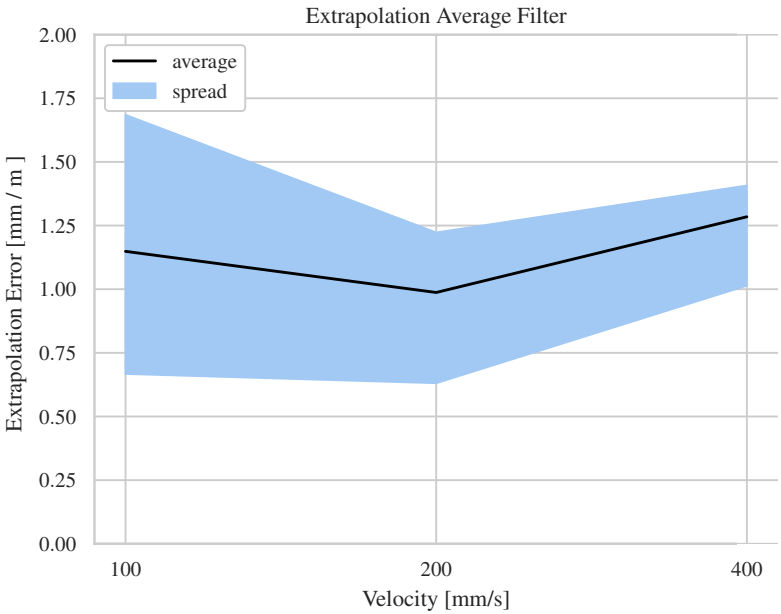


Figure 4.20: Results from the extrapolation experiment, Average filter.

4.12 Response Time

Background

For a pick-and-place application it is important that not only the input data for position and velocity is accurate, but also knowing how old the data were. That is: how long it takes from the moment an image is captured until a measurement has been calculated. This is called the response time, and this experiments aimed to show how large the response time is and its biggest contributors.

Aim

To investigate the response time of the system.

Method

The trolley was placed in the middle of the FOV where the object can be detected. The response time was then measured by logging the timestamps of the beginning and end of both the camera triggering task and image processing task. In addition to this, the timestamps of major events within the task was logged to further see how long time the different parts of the code takes to execute. To see how the image size

influence the response time, the experiment was done using two images, the large image being twice the size of the smaller one.

Results

As indicated by Figures 4.21 and 4.22, the exposure makes up a large proportion of the total response time. However, one should keep in mind that the exposure time is highly variable and can be controlled and set by the developer.

Figures 4.23 and 4.24, show a breakdown of the constituents of the image processing task. It is clear that the method for finding the object is the heaviest task, making up roughly 50% of the entire image processing task.

How the image size influences the response time can be seen by comparing Figures 4.23 and 4.24, the experiment shows that the tasks for calculating the center point, calculating the velocity, retrieving the image and other remains the same regardless of the image size, meanwhile, the task time for finding the object and converting the image to grayscale is almost doubled as the image size increases with a factor of two.

An additional comparison can be seen by looking at Figures 4.21 and 4.22, which shows that the triggering time and exposure time remains constant with an increased image size, whereas the image processing task almost doubles.

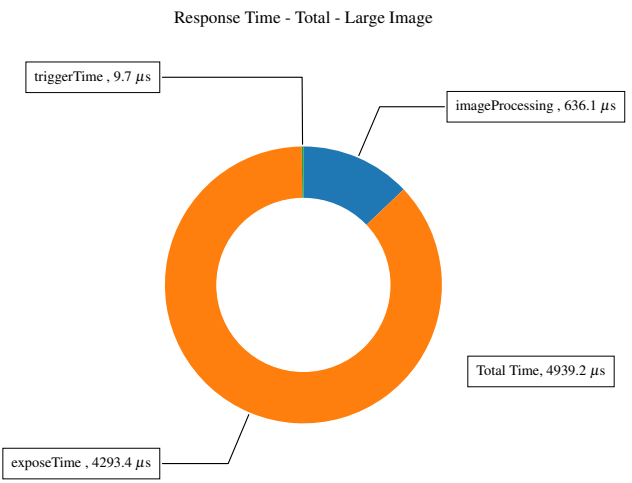


Figure 4.21: Results from the response time experiment. Total execution time, large image.

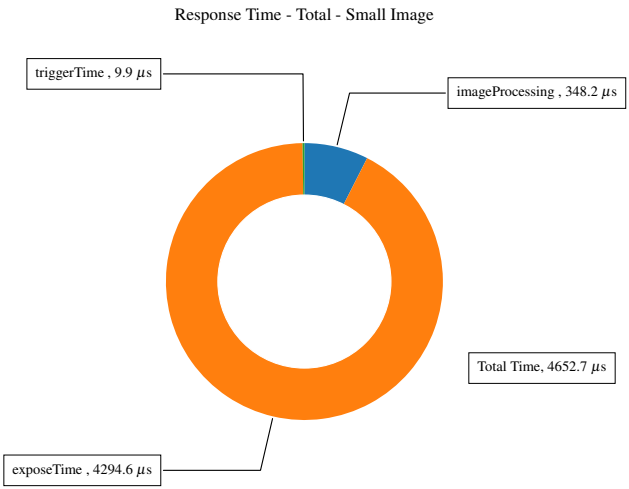


Figure 4.22: Results from the response time experiment. Total execution time, small image.

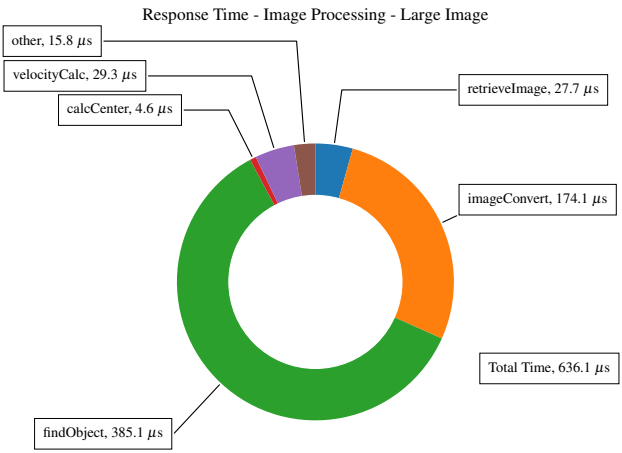


Figure 4.23: Results from the response time experiment. Image Processing task's sub components, large image.

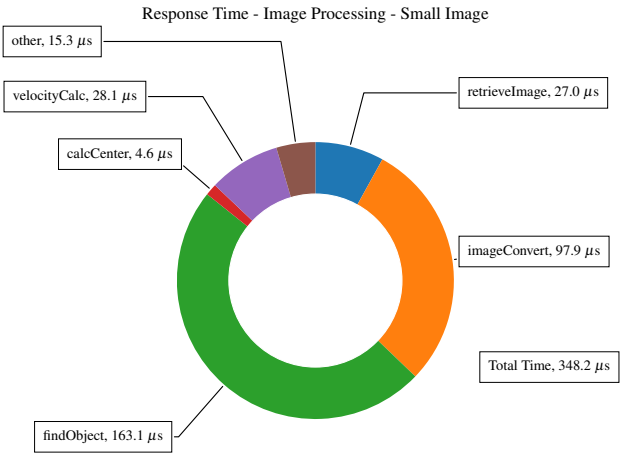


Figure 4.24: Results from the response time experiment. Image Processing task's sub components, small image.

5

Prototype

In this chapter, a prototype which utilizes real-time computer-vision for a pick-and-place application is presented. To evaluate its performance, the vision-based prototype is compared to a traditional pick-and-place solution.

5.1 Background

Based on the experimental results presented in Chapter 4, it seemed promising to use a real-time computer vision system for accurate position and velocity measurements. To see whether these results correlate to a successful pick-and-place application, it was interesting to test it practically by creating such an application. This prototype has therefore implemented a pick-and-place application where the extrapolated position was predicted using the vision-based velocity estimation. To see whether it has similar performance compared to using an encoder, tests were also be done using the velocity from an encoder.

5.2 Aim

To implement a pick-and-place prototype which utilizes real-time computer vision to replace a conveyor belt encoder for position extrapolation.

5.3 Setup

The setup for this prototype was based on the general test setup described in Section 4.2. Only the differences from this setup will be mentioned in this section.

Hardware

In comparison with the general test setup, there was an addition of a delta robot, an object to pick up, and a fixture for the object. The linear unit acted as a conveyor belt and moves the trolley across the camera FOV into the delta workspace. The

fixture for the object was mounted on the trolley, which lifted the object up into the delta robot workspace. The camera was placed in front of the delta robot, resulting in a FOV of 74x175 mm at the object's height. The delta robot workspace was in the form of a cylinder with a radius of 180 mm and the height of 70 mm. The complete setup is presented in Figure 5.3.

The delta robot consisted of three arms and has 4 degrees of freedom: movement along the x-, y- and z-axis as well as rotation around the z-axis. The arms were connected with universal joints to a servo motor which was attached to a base plate mounted to a cage. At the end of the arms there was an end-effector with a two-finger force-closure grasp, which can be seen in Figure 5.2. The object that was picked up was a lightweight 3D printed object with a rectangular shape and a scaled reference on one of its sides, which can be seen in Figure 5.1.

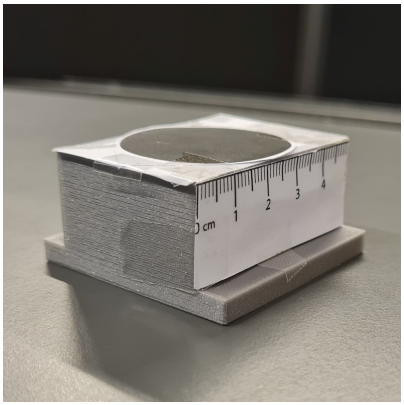


Figure 5.1: Pick up-object for the prototype.

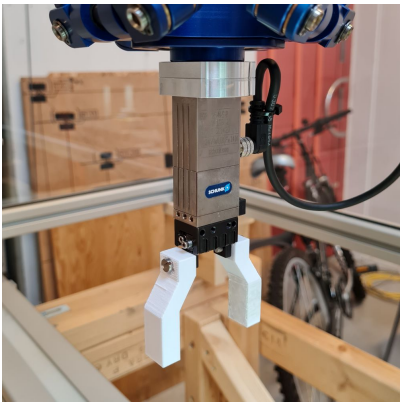


Figure 5.2: End effector of the prototype's delta robot.

Software

In comparison with the software in the general setup, the linear unit control task was not used. Instead, a new task called "motion control" was used for controlling the motion of both the linear unit and the delta robot. A separate task for updating the extrapolated position was also introduced. The cycle times, priorities and, CPU cores for all the tasks are as presented in Table 5.1.

Camera Configuration Since the camera and ring light were mounted closer to the tracked object compared to previous tests, it was possible to decrease the exposure time due to less dissipation of light from the ring light. The exposure time was thus reduced to 0.497 ms.

Update Extrapolated Position Task This task was responsible for updating the extrapolated position. The extrapolated position was based on the object's last posi-

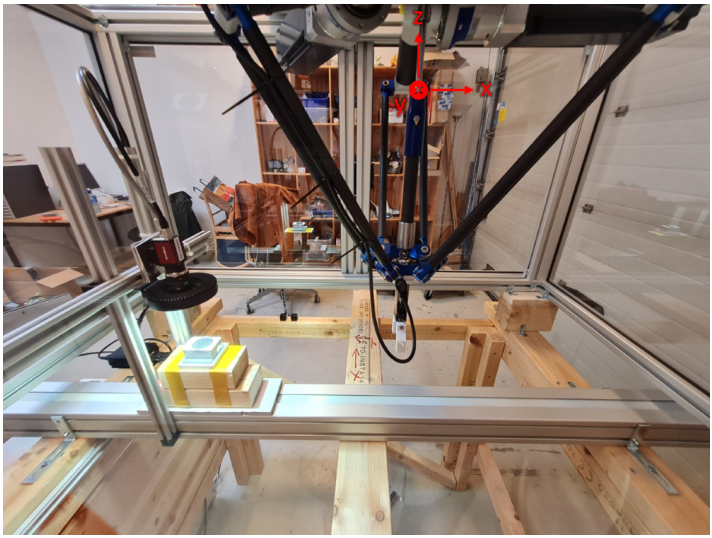


Figure 5.3: Experimental setup for the pick-and-place prototype.

Task	Cycle Time [ms]	Priority	CPU Cores
Motion Control Task	10	15	1
Camera Triggering Task	1	2	2
Image Processing Task	10	17	3
Update Extrapolated Position Task	2	3	2

Table 5.1: Tasks used in prototype's real-time system.

tion in the camera's FOV. This position was incremented by the estimated distance travelled since the last task cycle. The velocity used for calculating the distance travelled was either the velocity estimated by the Kalman filter, or the velocity from the encoder. The design of the Kalman filter is described in Section 4.6.

Virtual Axis The extrapolated positions calculated in the "update extrapolated position" task were fed to a virtual encoder that was connected to a virtual axis. This enables better and more robust functions to be used within the TwinCAT 3 software, such as coupling two axes' movements together.

Motion Control Task This task was responsible for the motion of both the linear unit and the delta robot. Its main purpose was to start the motion of the linear unit which acted as a conveyor belt, which moves across the camera's FOV that tracked an object which was later picked up by the delta robot. The task was designed as a state machine with the following states and transitions.

- **Wait for start** - Waits for the user start the sequence. The state changes to "move to start positions" when the user sets an enable signal to true.
- **Move to start positions** - Moves the linear unit and delta robot to their respective start positions. The state changes to "start conveyor belt" when the start positions are reached.
- **Start conveyor belt** - Initiates the linear unit's movement across the camera's FOV and delta robot's workspace. The state changes to "setup axis coupling" when the trolley has travelled beyond the camera's FOV.
- **Setup axis coupling** - Initializes the coupling of the virtual axis with extrapolated positions and the delta robot's x-axis, and the state changes to "couple axes".
- **Couple axes** - Couples the two axes together when the virtual axis' position is within $\pm$ 5 mm from the delta robot's position. The coupling causes the delta's x-axis to follow the exact movement of the virtual axis. The state changes to "move down to object" when the two axes are in sync with each other.
- **Move down to object** - Moves the delta robot down in the z-direction. The state changes to "close gripper" when the robot has reached its set position.
- **Close gripper** - Closes the delta robot's gripper. The state changes to "move up from object" when the grip has been closed.
- **Move up from object** - Moves the delta robot up in the z-direction. The state changes to "uncouple axes" when the movement has been started.
- **Uncouple axes** - Uncouples the virtual axis and the delta robot's x-axis. The state changes to "move to drop off position" when the two axes have been uncoupled.
- **Move to drop off position** - Moves the delta robot and linear unit to the drop off position. The state changes to "move down to drop" when the positions have been reached.
- **Move down to drop** - Moves the delta robot down in the z-direction. The state changes to "open gripper" when the robot has reached its set position.
- **Open gripper** - Opens the delta robot's gripper. The state changes to "move up from drop" when the grip has been opened.
- **Move up from drop** - Moves the delta robot up in the z-direction. The state changes to "wait for start" when the movement has been started.

A simplified version of the state machine is illustrated in Figure 5.4.

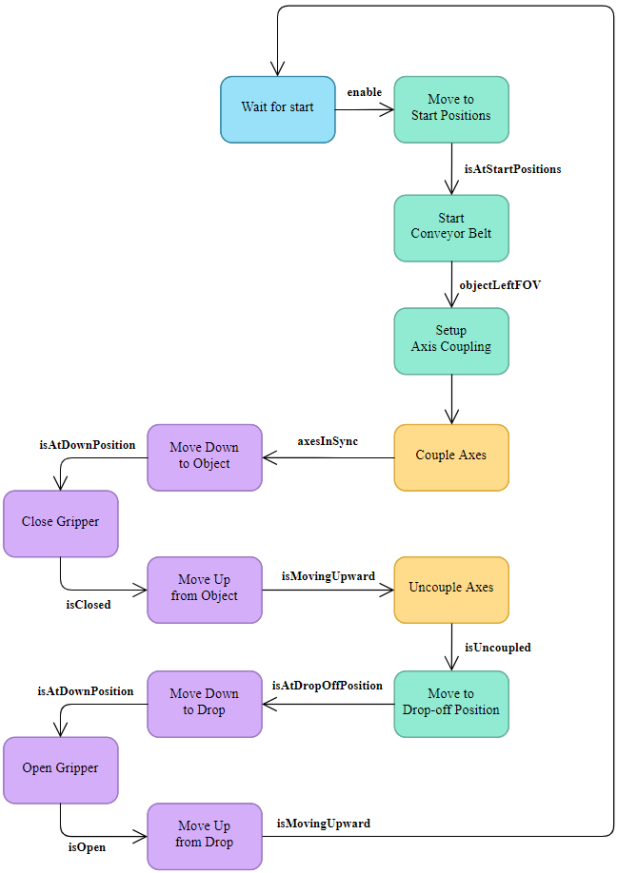


Figure 5.4: Motion Control Task state machine.

5.4 Method

The experiment was done by letting the linear axis travel at a fixed velocity across the camera's FOV into the delta workspace where the robot stands waiting at its outermost position. When the linear axis was below the delta robot, the delta robot coupled with the virtual axis which moved based on the extrapolated positions. The delta robot then utilized 50 mm to get in sync with the virtual axis and as soon as it was, the pick-up sequence began. The delta robot maintained its speed as it moved down and grasps the object. When the gripper was closed, it moved up again and the sequence was finished.

To compare the vision-based velocity estimation against a traditional encoder, identical tests were done using both extrapolation using velocity from an encoder and from the vision software. To evaluate the performance, the gripper's offset from

the object's centerline perpendicular to the motion direction was recorded. The experiments were performed at four different speeds which can be seen in Table 5.2.

Test	Velocity [mm/s]
1	25
2	50
3	100
4	200

Table 5.2: Velocities used for the tests in the pick-and-place experiment.

5.5 Results

The results show that the vision-based system could successfully pick up the object, demonstrating adequate performance and achieving a hit rate of 100 percent. Figure 5.5 shows the average error for each speed as well as the spread between the minimum and maximum error. A positive value on the y-axis means the delta picked the object too early, ahead of the centerline. Additionally, it was possible to see a trend, with a downwards sloping line, where the gripper picked the object further and further back as the speed increased, most likely due to lag.

The sequence from start to when the object was picked up is visualized in Figure 5.6 – 5.11.

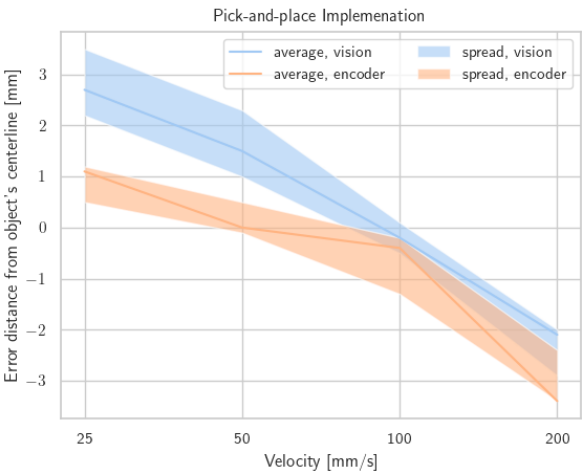


Figure 5.5: Results from the pick-and-place implementation.

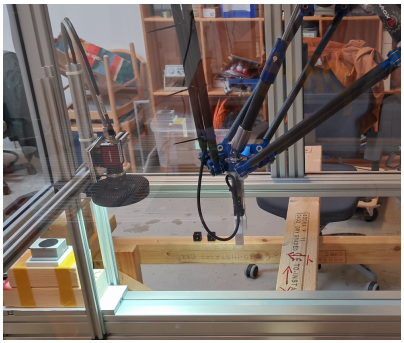


Figure 5.6: Start position.

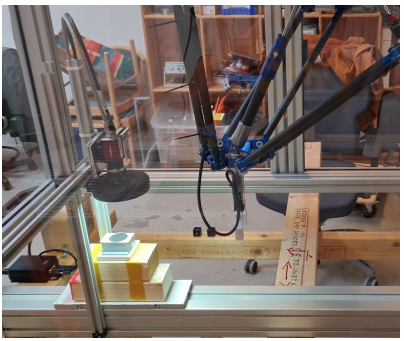


Figure 5.7: Object in FOV.

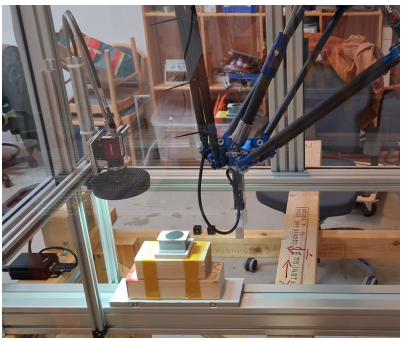


Figure 5.8: Object leaves FOV, extrapolation starts.

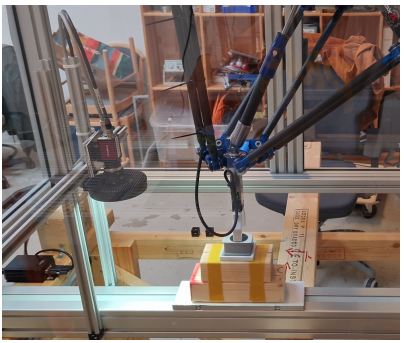


Figure 5.9: Delta robot couples to the virtual axis.

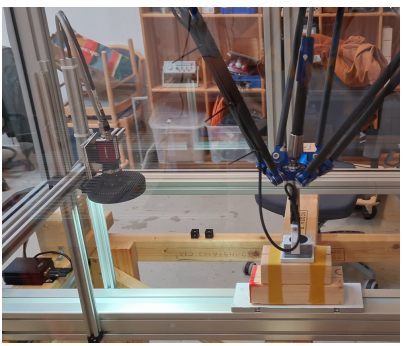


Figure 5.10: Delta robot moves down to grab object.

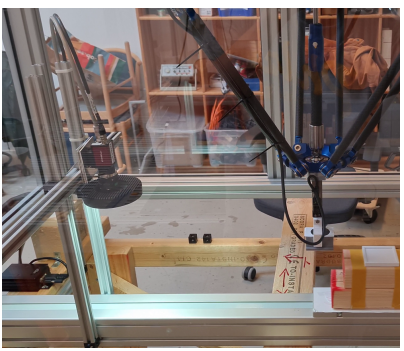


Figure 5.11: Object has been grabbed.

6

Discussion

In this chapter, the results from the literature survey, experiments, and prototype are analyzed and discussed. The challenges and benefits of using real-time computer vision are also assessed with its ability to have an impact in the field of industrial automation.

6.1 Integrated vision system

As presented in Section 1.1, the main focus of this thesis was to study the benefits that can be attained when using a platform where a vision system is integrated within the control system. This is how TwinCAT is different from most other systems in the industry and what makes it stand out.

Control systems found on the market typically does not allow a vision system to be integrated with the control system, but does enable an external vision system to be connected through a generic interface. Some examples used in the industry are ABB's "Externally Guided Motion" interface [ABB, 2021], and Franka Emika's "Franka Control Interface" [Franka Emika GmbH, 2017a]. By running an external vision system connected through a generic interface, a communication delay is introduced when exchanging data between with the control system. This means that time between actuation of the control system based on new data from the vision system is increased. This communication delay is naturally eliminated on an integrated vision system.

The update frequency on these interfaces are at best 4 ms for position guidance and position stream using the Externally Guided Motion interface [ABB, 2021], and 1 ms for the Franka Control Interface [Franka Emika GmbH, 2017b], as compared to TwinCAT's PLC cycle time which can be set as low as 50 μs [Beckhoff, 2021b]. Another advantage of running an integrated vision software on the control system is that the need for synchronization between two separate systems is eliminated, thus reducing complexity and erasing a potential point of failure.

This systemic difference that TwinCAT brings in comparison with other control systems was the main point of interest when evaluating the results in this project.

Comparisons between different algorithms, although they were made, were not the main focus of this thesis, but rather the evaluation of how the systemic difference can provide new advantages and use cases.

6.2 Literature survey

Real-time vision applications

The literature survey showed that there was a wide range of applications where real-time computer vision can be utilized within the four fields of industrial automation. There were three broader themes that were identified which are feature detection, position and orientation tracking, and real-time critical applications, as presented in Section 3.3. Feature detection was typically found in applications within quality control and machine control, whereas position and orientation tracking was most often found in robotic guidance and code scanning. Real-time critical applications were usually implementations which are sensitive to jitter and latency.

It is worth noting that there is a possibility that there are many more different types of applications where real-time computer vision is utilized, since this thesis only has searched for applications in academic papers. It would therefore be very interesting to conduct a study that focuses more on the actual implementations found in different machines and production lines in the industry, in order to broaden the scope and get a better understanding of how it is utilized practically in real production scenarios. Such study could, e.g., interview industry experts, and visit production facilities.

Benefits

It is difficult to determine exactly which area may benefit the most from real-time computer vision. If, e.g., one wants to improve throughput in an entire production line, it is most important to evaluate and improve the weakest link in the chain. Whether that is robotic guidance or code scanning needs to be determined on a case-by-case basis.

However, it is possible to argue that the applications that were found in several instances in the literature survey are fields that benefit heavily from the implementation of real-time vision. With such reasoning, the applications that benefit the most from real-time computer vision were position and orientation tracking for pick-and-place, and code scanning for mobile robot position tracking.

On the contrary, one can argue that it is the areas where fewer applications were found that could potentially benefit the most from real-time computer vision, since it may be areas where it is not yet widely used today. If it is not widely used due to not being properly explored, then it is possible that there is more room for improvement with new, interesting applications compared to the more studied fields. If this is the case, then machine control is the area where that potentially benefits the most from using computer vision.

Challenges

When evaluating the literature survey, it is possible to say that real-time critical applications is an area where real-time computer vision provides a solution that non-real-time vision cannot solve. Such applications will simply not be solvable in a reliable manner for a non-real-time vision system. When observing the general themes for all the applications as presented in Table 3.2, the majority of them were real-time critical applications - typically applications with high performance, and low latency constraints.

Worth noting is that most of the applications presented in the literature survey were done in a controlled lab environment. In actual production situations, the environment may look quite different. One case is a dirty environment with lots of dust flowing around. Such environments have difficulties utilizing computer vision due to the camera lens getting dirtier and dirtier as time goes on, and will therefore not be able to utilize neither non-real-time vision nor real-time vision. Likewise goes for machines where one simply cannot mount a camera at the place needed to measure the desired state.

System Evaluations

Some alternative real-time systems found in the literature study are LabVIEW RT operating system used by [Schmitt et al., 2015] and real-time Linux distribution used by [Álvarez-Valera et al., 2014]. A real-time system running in parallel with a Windows OS, similar to TwinCAT's implementation, was used in the application by [Liu et al., 2018]. The Windows extension called Kithara Real-time Suite (KRTX) was utilized and the author highlighted the value of using a Windows-based system, since it since most programs can be run on Windows and thus avoiding many incompatibility problems. An advantage with TwinCAT and other PLC systems compared to KRTX is that PLCs are typically optimized for industrial automation, which may ease the control and integration of sensors and actuators in industrial automation, in comparison with a generic RTOS.

In the articles found in the literature study, several implementations used additional hardware specially responsible for image processing. For instance, a separate laptop was used by [Dzodzo et al., 2013] and an FPGA-module was designed by [Schmitt et al., 2015]. A major advantage of using separate, dedicated hardware is that the processing speed is generally higher than running the image processing on the same CPU as other tasks. This is especially true when implementing the image processing on an FPGA. However, as previously mentioned in Section 6.1, the downside is that the separation of tasks between different hardware introduces a communication delay, which may cause issues based on the application.

6.3 Experiment Evaluation

Camera Resolution

One of the most distinct findings found in the experiments was the relationship between camera resolution and measurement accuracy. Experiment 4.7 showed that by increasing the resolution by a factor of four, one increased the measurement accuracy by a factor of three. Thus, if high accuracy is needed for a certain use case, a straightforward solution is to opt for a higher resolution camera. However, one should take into consideration that there might be an upper limit, where increased resolution gives diminishing returns. One should also keep in mind that a higher resolution camera often comes at the expense of speed, both in terms of frame rate and data transfer time.

When comparing these findings with the camera resolutions used in previous research, it can be argued that the importance of a high degree of accuracy is largely dependant on the industrial application. The pick-and-place application presented by [Liu et al., 2018], which is similar to the prototype presented in Chapter 5, used a camera resolution of 659x494 pixels which is 18% lower than the resolution used in the prototype presented in this thesis. Despite this, both implementations achieve adequate performance. On the other hand, there were several high performance applications found in the literature survey that utilized a higher camera resolution. Some examples of these applications were plastic granulate sorting [Peršak et al., 2020], flow monitoring in an e-jet printer [Lies et al., 2018], and inspection of fiber position and orientation in non-crimp fabrics [Schmitt et al., 2015], which used a camera resolution of 2590x1942, 1279x1025, and 2048x2048 pixels respectively. Each of these three applications measured small details in an image, which demanded a high degree of measurement accuracy. Worth noting is that camera placement and magnification also affected the space resolution in the plane that was measured, as demonstrated by [Lies et al., 2018] which achieved a resolution of 0.2735 μm at a distance of 86 mm from the lens with a horizontal FOV of 0.35 mm at the highest lens magnification. Based on these comparisons, it is believed that an adequate camera should be selected after evaluating the performance requirements of the application, where higher demands on measurement accuracy calls for a higher camera resolution, closer camera placement, or higher magnification.

Filtering

When constructing a low pass filter, one aspect which needs to be considered is the trade-off between speed and smoothness, since a filter that removes noise to a high degree oftentimes is slow at reacting to change. This is especially true when there is a large jump between two sample points and the filter needs to catch up. However, in the use case of this project, where the conveyor belt was expected to move at a constant speed, the rise time is not as critical. Therefore, in Experiment 4.6, where different filtering methods were compared, rise time was not taken into con-

sideration. The Kalman filter used in the implementation had a good performance, however, there is still room for improvement. More time could have been spent refining the parameters, especially the initial predicted error covariance, $\tilde{P}(0)$, which was determined empirically. Another alternative is to not reset the filter after each run, which would result in the performance being less dependent on the accuracy of the initial values of $\tilde{P}(0)$. The standard deviation of the process noise was assumed to be the same as the standard deviation of the measurement noise, which is an assumption that may not be accurate and could also be further examined to achieve a better result.

Time Resolution

One experiment that had surprising results was Experiment 4.3, time resolution. It was believed that a higher time resolution would result in a more accurate velocity, however, this did not end up being the case. A possible explanation of this can be the fact that the system was run in real-time. For the experiment, the task responsible for collecting the timestamps did so by polling the GPIO indicating whether the camera was exposing every cycle for state change. This polling was done after the camera was triggered, and when an upward flank occurred, the timestamp was recorded. By increasing the task cycle time, the window in which the upward flank can occur will increase, leading to less resolution. However, if it is assumed that the time between triggering the camera and the camera indicating that it is exposing is constant, then the amount of task cycles passed between triggering and recording the upward flank will also be more or less constant regardless of the task cycle time. Since the velocity is calculated based on the time difference between the past two measurements, this time drift will be eliminated.

Direction Dependency

A phenomenon observed in Experiment 4.9, was that the lag error varied depending on the direction in which the object was headed. The subsequent experiment, Experiment 4.10, showed that the cause of the error was not introduced by the image processing algorithm and instead was believed to be a mechanical issue. A conceivable explanation presented in the results was that the error originates from backlash in the servo motor's transmission when changing direction, resulting in the encoder recording a motion that was not propagated to the linear unit's trolley. If this error will cause a problem for a certain application, the developer could simply opt for a power transmission system with a smaller degree of backlash. There might exist other similar mechanical issues not observed in this project which can be avoided using a vision-based system.

The ability to measure these movements, which an encoder is unable to, can prove to be useful in other applications as well. Applications that combined odometry from an incremental encoder with a vision system that scans for QR-code 'anchors' were done by both [Nazemzadeh et al., 2017] and [Rusdinar et al., 2012],

as presented in the literature survey. These implementations resulted in a more robust position measurement by correcting the inaccuracies in the estimated position through localizing a QR-code at a known position in space. This concept of anchoring used in these two applications can be applied to a conveyor belt with a encoder based position measurement in a similar manner. By attaching a QR-code on a fixed position on the belt, the camera may scan for this position and compensate for any discrepancies caused by mechanical issues.

Response Time

When comparing the results from Experiment 4.12 with the pick-and-place implementations proposed by [Farag et al., 2019] and [Liu et al., 2018] as presented in Chapter 3, it seems that the implementation was relatively efficient. The results showed a response time of approximately 4.9 ms, outperforming the implementation by [Farag et al., 2019] and [Liu et al., 2018], which had a response time of 380 ms and 12.1 - 13 ms respectively. This corresponds to an ability to process up to 204 frames per second as compared with 52 frames per second for [Farag et al., 2019] and 75 frames per second for [Liu et al., 2018].

It is important to note that this comparison may not be entirely fair, since the difference in response time may be attributed to many different factors, such as hardware and software design choices. In this thesis, the hardware used enabled a fast exposure time and high CPU clock speed. The setup used for the experiments also reduced execution time by using both a binary threshold and no blurring for object detection, in comparison with a computational heavy deep-learning algorithm. These factors may differ from the other two implementations, which may have a significant contribution to the difference in execution time.

6.4 Implementation Evaluation

The experiments show that it was possible to use vision-based measurements of position and velocity and that it can be used as feedback for an application. However, one should keep in mind that all the experiments were done in a lab environment under fixed conditions and that other issues could appear in a real application in an industrial environment. Potential issues that might appear is varying light conditions, dirty environments, and/or tracking of non-uniform objects.

The object detection algorithm used was fairly basic, using only a binary threshold. Despite this, a hit rate of 100 percent was achieved. This was achieved by fine-tuning the threshold for the unique object that was tracked, which had a high contrast with its black color on a white background. In other cases, the object shape may be much more irregular such as detecting plastic granulates [Peršak et al., 2020], chestnuts [Álvarez-Valera et al., 2014], or steel bars [Zhao et al., 2016]. In these instances, the detection algorithm needed to be more sophisticated in order to achieve a high hit rate of 100%, 87%, and 94–96% respectively. Because of the increased

complexity of these algorithms, it is believed that they are more robust and reliable at varied conditions than a simple binary threshold. This should be taken into consideration when implementing the prototype in an industrial setting. It is worth noting that a change to a more sophisticated detection algorithm would likely affect the complexity and execution time of the vision software.

When observing the results from the pick-and-place experiment, one can see that the encoder-based extrapolation slightly outperformed the vision-based extrapolation both in regards to error distribution for every tested velocity, as well as error increase between the different velocities. Even though the encoder-based extrapolation showed a better performance, the results showed that a vision-based extrapolation was possible to use in this type of pick-and-place application and possibly others as well, given that the application is within the tolerances of the vision system's accuracy. It is worth noting that the prototype was not optimized for higher accuracy, meaning that it is possible to get an even higher resolution than the one presented. This can be achieved by implementing changes based on the factors found to affect the accuracy, e.g., higher camera resolution.

One can also observe in the results that the error for both the encoder-based- and vision-based extrapolation increased with an increased velocity. This is believed to be a lag error, which was expected based on the results found in Experiment 4.9. Worth noting is that the amplitude of the lag error was significantly higher in the prototype compared to the amplitude in Experiment 4.9. A possible explanation for this is that the prototype had two compounded lag errors: a lag error between the delta robot and the virtual axis, and a lag error between the virtual axis and the linear unit. In comparison, the experiment only had a single lag error between the linear axis and the vision-based position. A possible way to reduce the magnitude of the lag were to use a feed forward controller. Another alternative, if the lag was consistent, was to introduce an offset that compensates for the lag error. This offset could either be a static value or dependent on the velocity.

Another interesting observation is that the delta robot picked up the object ahead of its center line at low velocities. This was believed to be a calibration error caused when measuring the translations and rotations for the conversion of the vision-system's coordinate system to the delta robot's coordinate system. One way of eliminating this was to simply use more sophisticated tools for calibration, e.g., touch probes. Such tools were not available when calibrating the prototype's system.

6.5 Advantages for vision based systems

One aspect, that does not show in the experiments, was the fact that vision measured the object's speed directly whereas an encoder measured the object's speed indirectly, by assuming that the object moved at the same speed as the conveyor belt's driving axle. There could be cases where this aspect could be of advantage for the vision-based system. For example, the vision-based system has the ability

to detect if an object starts to slide on the belt or if the belt itself is slipping on the driving axle.

This could prove useful in implementations such as the plastic granulate sorting machine proposed by [Peršak et al., 2020]. This achieved a 100% object classification accuracy, but introduced sorting errors due to the physical aspects of the system. There is a possibility that one of these errors were caused by slippage of the conveyor belt on the axle, leading to an inaccurate position estimation and thus an improperly timed sorting maneuver. It would therefore be interesting to compare the encoder-based velocity with a vision-based velocity, in order to investigate whether or not that was a cause of the timing error.

The vision-based system could also be used in applications where it is difficult to measure an object's speed in a indirect method, e.g., vibratory conveyors. In addition to this, there is a possibility to replace other sensors such as an encoder in situations where a camera is already required, e.g., in quality control or position measurements.

A second aspect is the flexibility created as a result of vision systems being mainly based on software rather than hardware. Most production lines are built to be able to produce many types of articles, so it is often of value to have a low changeover time. In many cases, it is a lot easier to switch the input parameters to a software program rather than adjusting the hardware.

A third aspect is the lack of wear and tear of a software-based system compared to systems that rely on hardware-based measurement systems. Therefore, one could argue that once the system is up and running less time will be required for service and maintenance which is very desirable in high volume production where production interruptions are very costly.

7

Conclusion

7.1 Objectives

This master thesis has examined the benefits and challenges of using real-time computer vision in industrial automation. The literature survey has provided an overview of applications and examples of where and how computer vision can be applied, as well as the applications which required implementation in a real-time fashion. Two broader themes were found where real-time computer vision was utilized, namely feature detection and position and orientation tracking.

Unfortunately, the results regarding which of the fields that could benefit the most from computer vision were non-conclusive, since it was highly subjective based on what parameters were deemed most important. However, it was clear that applications with high constraints on latency and jitter are the main areas where real-time computer vision was used compared to non-real-time, as these were the majority of the applications found in the survey.

Furthermore, a range of experiments were performed in order to outline the parameters which influence the performance of a real-time vision-based system. The most distinct finding was the relationship between camera resolution and measurement accuracy and also the unexpected result that a higher time resolution does not result in a more accurate velocity estimation. Finally, a prototype for a pick-and-place application was implemented to demonstrate the possibility for a vision-based system to replace more traditional implementations, in this case, an encoder.

7.2 Future Work

This project mainly focused on applications made in a controlled lab environment; both in the literature survey as well as the experiments and prototype. It would therefore be interesting to further study how real-time computer vision is used practically in the industry today through, e.g., interviews with industry experts, in order to get a more complete view of the subject. In addition to this, it would be interesting to

see how the prototype and lessons learned from the experiments translates into a real world industrial application.

Regarding the experiments, only one way of calculating velocity was used, namely using time- and position difference of the object between two frames. Another method that would be interesting to compare the used method to is optical flow, where one determines velocity based on the perceived motion of individual pixels. This method was considered for the experiments but ruled out due to time constraints.

The implementation used in the prototype may also be further developed in order to create a more reliable and higher-performing system, if it is to be used in an industrial environment. The solution should then be further optimized based on the results from the experiments, e.g., use a higher resolution camera.

Another area to further study is how real-time computer vision can be used to close control loops. This was a secondary objective in this thesis that was not achieved since it did not suit the chosen application to implement.

A

Box Plot

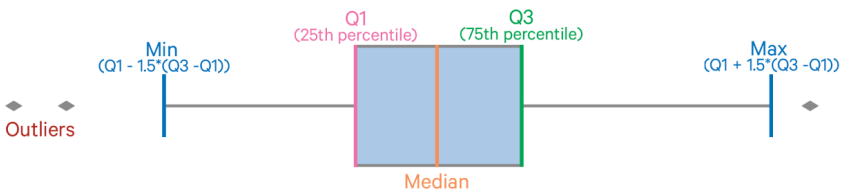


Figure A.1: Visual representation of what a box plot is composed of.

Bibliography

ABB (2021). *Application Manual - Externally Guided Motion*. URL: https://abb.sluzba.cz/Pages/Public/OmniCoreRoboticsDocumentationRW7/Controllers/RobotWare/en/3HAC073318-001.pdf (visited on 2021-07-19).

Álvarez-Valera, H. H., E. Bolivar-Vilca, C. Cervantes-Jilaja, E. E. Cuadros-Zegarra, D. Barrios-Aranibar, and R. Patiño-Escarcina (2014). "Automation of chestnuts selection process using computer vision in real time". In: *2014 33rd International Conference of the Chilean Computer Science Society (SCCC),* Talca, Chile, Nov. 8–14, 2014, pp. 87–91. DOI: 10.1109/SCCC.2014.21.

Arreguin, J. M. R. (2008). *Automation and Robotics*. I-Tech Education and Publishing, Vienna. ISBN: 978-3-902613-41-7.

Beckhoff (2021a). *Beckhoff documentation*. URL: https://infosys.beckhoff.com/english.php?content=../content/1033/tf7xxx_tc3_vision/9007207516833419.html&id= (visited on 2021-03-26).

Beckhoff (2021b). *Beckhoff documentation*. URL: https://infosys.beckhoff.com/english.php?content=../content/1033/tcsystemmanager/basics/TcSysMgr_ConfigRT_Intro2.htm (visited on 2021-07-19).

Beckhoff (2021c). *Company*. URL: https://www.beckhoff.com/en-en/company/ (visited on 2021-02-04).

Beckhoff (2021d). *TwinCAt Vision*. URL: https://www.beckhoff.com/en-en/products/automation/twincat-vision/ (visited on 2021-02-04).

Courtney, J., M. Magee, and J. Aggarwal (1984). "Robot guidance using computer vision". *Pattern Recognition* **17**:6, pp. 585–592. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(84)90012-8. URL: https://www.sciencedirect.com/science/article/pii/0031320384900128.

Diachok, R., R. Dunets, and H. Klym (2018). "System of detection and scanning bar codes from Raspberry Pi web camera". In: *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Kyiv, Ukraine, May 24–27, 2018, pp. 184–187. DOI: 10.1109/DESSERT.2018.8409124.

Dzodzo, B., L. Han, X. Chen, H. Qian, and Y. Xu (2013). "Realtime 2D code based localization for indoor robot navigation". In: *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO),* Shenzhen, China, Dec. 12–14, 2013, pp. 486–492. DOI: 10.1109/ROBIO.2013.6739507.

Farag, M., A. N. A. Ghafar, and M. H. ALSIBAI (2019). "Real-time robotic grasping and localization using deep learning-based object detection technique". In: *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS),* Selangor, Malaysia, Jun. 29–29, 2019, pp. 139–144. DOI: 10.1109/I2CACIS.2019.8825093.

Fernández-Caramés, T. M. and P. Fraga-Lamas (2018). "A review on human-centered IoT-Connected Smart Labels for the Industry 4.0". *IEEE Access* **6**, pp. 25939–25957. DOI: 10.1109/ACCESS.2018.2833501.

Franka Emika GmbH (2017a). *Franka Control Interface Documentation*. URL: https://frankaemika.github.io/docs/ (visited on 2021-07-19).

Franka Emika GmbH (2017b). *Franka Control Interface Documentation*. URL: https://frankaemika.github.io/docs/overview.html (visited on 2021-07-19).

Hagara, M., R. Stojanović, T. Bagala, P. Kubinec, and O. Ondráček (2020). "Grayscale image formats for edge detection and for its FPGA implementation". *Microprocessors and Microsystems* **75**, p. 103056. ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2020.103056. URL: https://www.sciencedirect.com/science/article/pii/S0141933119305034.

Jeppsson, U. (2021). *Lecture Slides: Introduction*. EIEN50 Automation, Advanced Course. Division of Industrial Electrical Engineering and Automation, Lund University. URL: https://www.iea.lth.se/aut/lectures/1_Introduction_21.pdf. Last visited on 2021-05-26.

Joblove, G. H. and D. Greenberg (1978). "Color spaces for computer graphics". *SIGGRAPH Comput. Graph.* **12**:3, pp. 20–25. ISSN: 0097-8930. DOI: 10.1145/965139.807362. URL: https://doi.org/10.1145/965139.807362.

Kulpraneet, W. and S. Natsupakpong (2019). "Automatic monomer filling system using machine vision". In: *2019 23rd International Computer Science and Engineering Conference (ICSEC),* Phuket, Thailand, Oct. 30–Nov. 1, 2019, pp. 92–96. DOI: 10.1109/ICSEC47112.2019.8974786.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. DOI: 10.1017/CBO9780511546877.

Lies, B. T., Y. Cai, E. Spahr, K. Lin, and H. Qin (2018). "Machine vision assisted micro-filament detection for real-time monitoring of electrohydrodynamic inkjet printing". *Procedia Manufacturing* **26**. *46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA*, College Station, USA, Jun. 12–22, 2018, pp. 29–39. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2018.07.004. URL: https://www.sciencedirect.com/science/article/pii/S2351978918306747. License: Creative Commons CC BY NC ND, https://creativecommons.org/licenses/by-nc-nd/4.0/.

Liu, Y., T. Hu, H. Ni, C. Zhang, B. Lin, and E. Judd (2018). "Design of a pc-based open industrial robot control system integrated with real-time machine vision". In: *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA),* Beijing, China, Aug. 16–16, 2018, pp. 1–6. DOI: 10.1109/WRC-SARA.2018.8584214.

Nazemzadeh, P., D. Fontanelli, D. Macii, and L. Palopoli (2017). "Indoor localization of mobile robots through qr code detection and dead reckoning data fusion". *IEEE/ASME Transactions on Mechatronics* **22**:6, pp. 2588–2599. DOI: 10.1109/TMECH.2017.2762598.

Pavlidis, T., J. Swartz, and Y. P. Wang (1990). "Fundamentals of bar code information theory". *Computer* **23**:4, pp. 74–86. DOI: 10.1109/2.55471.

Peršak, T., B. Viltužnik, J. Hernavs, and S. Klancnik (2020). "Vision-based sorting systems for transparent plastic granulate". *Applied Sciences* **10**, p. 4269. DOI: 10.3390/app10124269. License: Creative Commons CC BY 4.0, https://creativecommons.org/licenses/by/4.0/.

Qingmao Hu, Z. Hou, and W. L. Nowinski (2006). "Supervised range-constrained thresholding". *IEEE Transactions on Image Processing* **15**:1, pp. 228–240. DOI: 10.1109/TIP.2005.860348.

Rusdinar, A., J. Kim, J. Lee, and S. Kim (2012). "Implementation of real-time positioning system using extended Kalman filter and artificial landmark on ceiling". *Journal of Mechanical Science and Technology* **26**. DOI: 10.1007/s12206-011-1251-9.

Saho, K. (2018). "Kalman filter for moving object tracking: performance analysis and filter design". In: ISBN: 978-953-51-3827-3. DOI: 10.5772/intechopen.71731.

Schmitt, R., T. Fürtjes, B. Abbas, P. Abel, W. Kimmelmann, P. Kosse, and A. Buratti (2015). "Real-time machine vision system for an automated quality monitoring in mass production of multiaxial non-crimp fabrics". *IFAC-PapersOnLine* **48**:3. *15th IFAC Symposium on Information Control Problems in Manufacturing,* Ottawa, Canada, May 11–13, 2015, pp. 2393–2398. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.06.446. URL: https://www.sciencedirect.com/science/article/pii/S2405896315006850.

SharkD (2015). *HSV color solid cone*. URL: https://commons.wikimedia.org/w/index.php?curid=8421620. License: Creative Commons CC BY-SA 3.0.

Solovyev, R., W. Wang, and T. Gabruseva (2021). "Weighted boxes fusion: ensembling boxes from different object detection models". *Image and Vision Computing*, p. 104117. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2021.104117. URL: https://www.sciencedirect.com/science/article/pii/S0262885621000226.

Srinivas, Y. and D. L. Wilson (2002). "Image quality evaluation of flat panel detector binning in x-ray fluoroscopy". In: *Proceedings IEEE International Symposium on Biomedical Imaging,* Washington, DC, USA, Jul. 07–10, 2002, pp. 177–180. DOI: 10.1109/ISBI.2002.1029222.

Tiwari, S. (2016). "An introduction to qr code technology". In: *2016 International Conference on Information Technology (ICIT),* Bhubaneswar, India, Dec. 22–24, 2016, pp. 39–44. DOI: 10.1109/ICIT.2016.021.

Yu, L.-S., S.-Y. Chou, H.-Y. Wu, Y.-C. Chen, and Y.-H. Chen (2020). "Rapid and semi-quantitative colorimetric loop-mediated isothermal amplification detection of asfv via hsv color model transformation". *Journal of Microbiology, Immunology and Infection*. ISSN: 1684-1182. DOI: https://doi.org/10.1016/j.jmii.2020.08.003. URL: https://www.sciencedirect.com/science/article/pii/S1684118220301845.

Zaera, M., M. Esteve, C. E. Palau, J. C. Guerri, F. J. Martinez, and P. F. de Cordoba (2001). "Real-time scheduling and guidance of mobile robots on factory floors using Monte Carlo methods under windows nt". In: *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597),* Antibes-Juan les Pins, France, Oct. 15–18, 2001, 67–74 vol.1. DOI: 10.1109/ETFA.2001.996355.

Zhang, R. and L. Song (2014). "Study of variable spray control system based on machine vision". In: *2014 IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing,* London, UK, Aug. 18–20, 2014, pp. 455–458. DOI: 10.1109/ICCI-CC.2014.6921498.

Zhao, J., X. Xia, H. Wang, and S. Kong (2016). "Design of real-time steel bars recognition system based on machine vision". In: *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC),* Hangzhou, China, Aug. 27–28, 2016. Vol. 01, pp. 505–509. DOI: 10.1109/IHMSC.2016.75.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2021

Document Number
TFRT-6130

Author(s)
Joakim Cedergren
Jonathan Berglund

Supervisor
Daniel Jovanovski, Beckhoff, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Real-time Computer Vision in Industrial Automation

Abstract

The field of computer vision is growing larger and larger every day, fuelled by lower cost, higher computational power, and more sophisticated cameras. This trend, in tandem with the ongoing rise of Industry 4.0, provides new and exciting opportunities to innovate in areas where this technology has not yet been widely adopted or explored.

This thesis aims to investigate the ways in which real-time computer vision can be utilized in the field of industrial automation, and the benefits and challenges that it brings. Firstly, a literature survey was carried out, exploring previous research to identify trends and applications used in the field today. Secondly, a range of experiments were evaluated to give an overview of the parameters that affect the performance of a vision-based system. Finally, a prototype for a pick-and-place application was implemented to compare a vision-based system with a traditional system.

The results show great potential for the adaptation of real-time computer vision in the field of industrial automation. There is a promising opportunity for computer vision to be able to take a larger role in the field and replace many of the more traditional systems, based on its similar performance and unique characteristics.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-81

Recipient's notes

Security classification

http://www.control.lth.se/publications/