Tynes, Odd Arne Skjeret
Furnes, Pål-André
Melaas, Marius Høyer

Redesigning the Point Cloud Acquisition for Sort™

Bachelor's thesis in Electrical Engineering
Supervisor: Hatledal, Lars Ivar
Co-supervisor: Coates, Erlend Magnus Lervik

May 2022

Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Department of ICT and Natural Sciences

NTNU
Kunnskap for en bedre verden

Tynes, Odd Arne Skjeret
Furnes, Pål-André
Melaas, Marius Høyer

Redesigning the Point Cloud Acquisition for Sort™

Bachelor’s thesis in Electrical Engineering
Supervisor: Hatledal, Lars Ivar
Co-supervisor: Coates, Erlend Magnus Lervik
May 2022

Norwegian University of Science and Technology
Department of ICT and Natural Sciences

NTNU
Kunnskap for en bedre verden

NTNU
Knowledge for a better world

Solwr

IELEA2920 - Bachelor thesis

Redesigning the Point Cloud Acquisition for Sort™

Odd Arne Tynes, Pål-André Furnes,
Marius Høyer Melaas

Bachelor thesis

May 2022

Summary

This bachelor thesis is a detailed explanation of how the bachelor group went about redesigning the point cloud acquisition for Solwr's Sort™, a pallet sorting machine designed for the logistics industry. The current camera solution is using six Intel® RealSense™ L515 cameras, which are discontinued.

The project started with a research period investigating the current system at Sort, and generated ideas through several brainstorm meetings. The group presented the ideas to Solwr on a regular basis in order to get their feedback, and to decide which designs to pursue further. During the progress meetings with Solwr the group presented both ambitious and simple designs. Based on Solwr's feedback, some of the designs would be simulated, collecting more detailed information about their positive and negative aspects.

The group researched the leading manufacturers in 3D vision. Each camera and design was evaluated based on delivery time, cost, complexity, performance, sorting time and feasibility. The goal was to design a solution using industrial cameras that at least performs equally to the current solution, in the above mentioned domains. Sort™ has recently been commercialized and therefore Solwr wished to use industrial cameras in the new point cloud acquisition.

In order to compare the different cameras and placements, the group utilized 3D visualizations through CAD, and programmed their own field of view verification software. The software allowed for different tests in cameras and placements, without having the need to build prototypes for each one. The software has two different modes, graphical FOV and point cloud verification. The graphical FOV mode produces graphs showing a camera's field of view. The point cloud verification mode generates a point cloud representing a pallet, and displays whether the points are seen or not, given a certain camera placement.

Based on the camera comparison result and meetings with Solwr, a simulation was made. The simulator gave more realistic data to analyze, and the basis needed to build a prototype. The simulated cameras were given the same resolution and the calculated distance as the group's chosen camera. The data collected allowed the group to further confirm their proposed solution.

Finally, a new design was chosen based on the simulator, 3D-models, calculations and FOV software. Thereafter, a working prototype was built using the current Intel cameras, in Solwr's warehouse. Fortunately, Solwr had components from previous installations and prototypes, and therefore there were no need to buy additional hardware. The data gathered from the prototype was used in order to form the final conclusion. The thesis concludes that the new point cloud acquisition is a viable solution, both mechanically and economically, considering Solwr's requirements. Solwr has now ordered two Zivid One^{+} Large cameras per the group's proposed solution. Further testing will be the deciding factor for the new design of Sort™. However, due to a two month delivery time of the ordered cameras, the final testing will be executed by Solwr.

Sammendrag

Denne bacheloroppgaven er en detaljert forklaring på hvordan bachelorgruppen gikk frem for å redesigne punktsky-anskaffelsen for Solwr's Sort™, en pallsorteringsmaskin designet for logistikkbransjen. Den nåværende kameraløsningen bruker seks Intel® RealSense™ L515 kameraer, som har gått ut av produksjon.

Prosjektet startet med en forskningsperiode som undersøkte dagens Sort™ og genererte ideer gjennom flere idémyldringer. Gruppen presenterte ideene for Solwr med jevne mellomrom for å få tilbakemeldinger og for å bestemme hvilke design å forfølge videre. Under fremdriftsmøtene med Solwr presenterte gruppen både ambisiøse og enkle design. Basert på Solwrs tilbakemelding vil noen av designene bli simulert for å samle inn mer detaljert informasjon om deres positive og negative aspekter.

Gruppen undersøkte de ledende produsentene innen 3D-kameraer. Hvert kamera og design ble evaluert basert på leveringstid, kostnad, kompleksitet, ytelse, sorteringstid og gjennomførbarhet. Målet var å designe en løsning ved hjelp av industrikameraer som minst yter likt som dagens løsning, i de ovennevnte domenene. Sort™ har nylig blitt kommersialisert og derfor ønsket Solwr å bruke industrielle kameraer i den nye punktsky-anskaffelsen.

For å sammenligne de forskjellige kameraene og plasseringene brukte gruppen 3D-modeller og programmerte sin egen verifiseringsprogramvare for synsfelt. Programvaren gjorde det mulig å gjøre forskjellige tester angående kameraer og plasseringer uten at det er nødvendig bygge prototyper for hver enkelt. Programvaren har to forskjellige moduser, grafisk FOV og punktskyverifisering. Den grafiske FOV-modusen produserer grafer som viser et kameras synsfelt. Punktskyverifiseringsmodusen genererer en punktsky som representerer en pall og viser om punktene er sett eller ikke, gitt en bestemt kameraplassering.

Basert på kamerasammenligningen og møter med Solwr, ble det laget en simulering. Simulatoren ga mer realistiske data å analysere, samt grunnlaget for å bygge en prototype. De simulerte kameraene hadde samme oppløsning og avstand som gruppens valgte kamera. Dataene som ble samlet inn gjorde at gruppen kunne bekrefte deres løsning ytterligere.

Til slutt ble et design valgt basert på simulatoren, 3D-modeller, beregninger og FOV-programvare. Deretter ble det bygget en fungerende prototype ved bruk av de nåværende Intel-kameraene, i Solwrs lager. Solwr hadde komponenter fra tidligere installasjoner og prototyper, og derfor var det ikke nødvendig med innkjøp av flere komponenter. Dataene gruppen samlet fra prototypen ble brukt for å danne den endelige konklusjonen. Avhandlingen konkluderer med at den nye punktsky-anskaffelsen er en levedyktig løsning både mekanisk og økonomisk, i forhold til Solwrs krav. Solwr har nå bestilt to Zivid One^{+} Large kamera per gruppens foreslåtte løsning. Videre tester vil være avgjørende for det neste designet av Sort™. Grunnet en to måneders leveringstid på de bestilte kameraene vil den endelige testingen bli utført av Solwr.

Preface

This thesis is submitted as part of a bachelor degree in electrical engineering at the Norwegian University of Science and Technology (NTNU). The assignment is provided by Solwr, a Norwegian logtech company, in collaboration with NTNU Ålesund.

The students collaborating on this thesis are all graduating in the spring of 2022. They have a variety of backgrounds and qualifications, with one having a bachelor degree in physics, another having a certificate of apprenticeship in automation, and the third having certificates of apprenticeship in both automation and as an electrician. During their education at NTNU Ålesund, they have all taken subjects that have prepared them for this thesis. Together, they have a wide foundation in both theoretical and physical aspects beneficial to this thesis.

The work on this thesis has provided a good look into the development of a industrial product. It has allowed the group to put a lot of the theoretical knowledge they have accrued over their studies into practice while gaining experience about working with established companies.

None of the students started working for Solwr after the thesis, However, one of the students did have a part time job in Solwr during the bachelor period.

We would like to thank our advisors, Lars Ivar Hatledal and Erlend Magnus Lervik Coates, as well as all engineers employed at our employer, Solwr. The group made a presentation video of the results and software during this thesis, and can be viewed at This Youtube link, or at the URL: https://youtu.be/_Ffkh-aWm6I

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter contains the background, motivation, scope, contribution, outline and related work for the bachelor thesis. It is meant to give the reader insight into the issue solved in the thesis and provide reasoning for doing so.

The background explains the historical development of cameras and how they are used in the industry. The motivation details the group's reason for solving the issue, as well as Solwr's. The scope defines the limits of this thesis, explicitly stating which areas of Sort™ the group is redesigning. The contribution states what the group has found during the bachelor period and what areas their work has contributed to. The outline is meant to guide the user through the structure of the thesis, explaining what each other chapter contains. Finally, related work is similar papers that share some of the issue solved in this thesis.

1.1 Background

Before the advent of the Industrial Revolution in the eighteenth to nineteenth century, most hard labor was done by humans or animals [4]. However, the invention of the steam engine, the age of science and mass production, and the rise of digital technology revolutionized the world.

In modern society, the importance of robots, sensors and automated systems are growing rapidly. In the last decade the technology has had big leaps in advancements and computation power [5]. Processes are automatized and the need for automation engineers is increasing. As the technology gets cheaper and more advanced, the importance of ethical thinking is crucial for the worlds future, due to the coexistence between humans, robots, and automated systems.

In the early days of automated systems, relays were used until the invention of the PLC, which to this day is widely used in the automation industry [1]. However, the PLC, sensors and I/O used today have evolved into more then just binary signals. An up and coming trend is the utilization of cameras and high level programming, to replace sensors and to create a adaptable dynamic systems. Examples of use cases are: Product Assembly, Defect Detection, 3D Vision Systems, Predictive Maintenance and Safety and Security Standards etc.

Solwr is Norwegian logtech (Logistics and Technology) company that develops software and manufactures robots for the logistics industry such as Asko and H.I Giørtz.

Recently Solwr was two individual companies named ”Driw”, and ”Currence Robotics”, which have now combined. Currence Robotics was the original employer of this thesis, and is the owner of the pallet sorter robot called Sort™, located in Ålesund.

Solwr has offices in Oslo, Trondheim and headquarters located in Ålesund, Norway. Solwr offers a plug-and-play product portfolio of software and hardware. Customers of Solwr include some of Norway’s top retailers [6]. They have approximately one hundred employees with a wide spread expertise of mathematicians, engineers, scientists, full stack developers, intelligent system developers, logistic experts and a marketing team. In addition to this, they are partly funded by Innovation Norway and the Norwegian Research Council [7].

The camera currently used at Sort™ is the Intel® RealSense™ L515 LiDAR camera, which generates point clouds to classify an assortment of used pallets. Intel, the manufacturer, has decided to discontinue a number of cameras, including the L515, with no official reason [8]. According to framos.com it was ”*to focus on their existing D400 series Stereo Vision product line*” [9]. However, Solwr ordered a substantial amount of L515 cameras in order to keep building Sort™, while researching new cameras and solutions.

1.2 Motivation

The motivation of this thesis was Solwr's need of a new point cloud acquisition for their sorting robot, named Sort™. The current cameras used on Sort™ are now discontinued, and Solwr therefore is in need of new cameras and a new camera solution.

The current cameras are defined as consumer cameras due to the low cost and availability. Solwr needed six cameras to generate the resolution required in their point cloud. Sort™ has been in development as a beta prototype at H.I Giørtz (H.I.G) in Ålesund since 2019, and has now evolved into a commercial product. Therefore, Solwr wanted an industrial camera for better performance, stability, with a reliable manufacturer. The group's research in this thesis focuses on high-end industrial cameras with significantly improved resolution, as well as specifications from specialized camera manufacturers. Industrial cameras, on the other hand, are far more expensive, creating an incentive to reduce the number of cameras required as much as feasible.

The new solution will be measured in delivery time, cost, complexity, performance, sorting time and feasibility. The goal is to design a solution using industrial cameras that at least performs equally to the current solution, in the above mentioned domains. In this thesis the planned tasks is to CAD, simulate, calculate, and plan a feasible solution. Furthermore to build a prototype and finally document a proof of concept. The main motivation to maintain a relatively low cost solution is for Solwr to stay economically competitive, since Sort™ is now a commercial product.

According to Solwr, most professional camera manufactures offer to perform testing on the companies behalf, or alternatively provides the company with cameras to perform own tests, for a small shipping fee. Solwr also mentioned that in most cases they prefer to do their own testing, due to it being more time efficient and result effective. One of the thesis' goals was to come to a conclusion on a camera based on Solwr's requirements, and to order one for Solwr to test. Utilizing the prototype produced for this thesis, as well as full scale testing on site, at H.I.G.

The bachelor group's personal motivation for this thesis was personal development as engineers, due to the relevancy of their education. Furthermore, to physically and theoretically implement knowledge learned during the bachelor studies.

1.3 Scope

The bachelor thesis is researched and written by three students at NTNU Ålesund in Norway, during the period January 10th through May 20th, 2022. All three students are graduating in the spring of 2022, in the field of automation engineering.
The scope of this thesis include 3D modeling, data collection and analysis, software development, simulator-, PLC-, and HMI programming, electrical wiring, and a prototype assembly. The group will focus on redesigning and prototyping the point cloud acquisition of Sort™. The other areas of the system will be left unchanged.

The bachelor group will brainstorm and make several sketches of different solutions. Thereafter, the sketches will be modeled in a browser-based 3D-modelling software, named Onshape. In addition to this, the group will develop their own software for FOV verification using ray casting. Based on the mentors' and employer's feedback, several solutions will be simulated for further research. The simulation will be built in Webots. Webots supports user-defined robots, cameras, and rangefinders. It can also be programmed using several different programming languages, while Python being the group's choice for all high level programming. Thereafter, a prototype will be built based on the most promising solution. The hardware for the prototype will be provided by Solwr. The motor and drive setup, PLC code, and HMI will be created in structured text by the bachelor group using Microsoft Visual Studio, with TwinCAT 3 integration [R].

The group will use GitHub to exchange software both with each other and the employer. This is a development and version control tool. Because everything is saved online and in successive versions, using GitHub makes data loss highly unlikely. See risk chart in the project preliminary report [R].

1.4 Contributions

During the bachelor project period, the group has done contributions in line with the motivation and scope of this thesis. The group has done research, tests, and simulations towards presenting a new image acquisition for Solwr, meeting Solwr's requirements. This solution is contributing in research and development of a commercial product, in the large industry of logistics. While doing the research of this thesis, the group has also found solutions that are not viable, which is, in the group's opinion, just as important. Choosing a camera was done by comparing the leading 3D manufacturer's cameras objectively, and filling out a comparison tables within this thesis. I.e the information gathered may be used to find suitable cameras, or exclude them, for other point cloud acquisitions.

The group has made their own FOV software, that has given the group great contribution to their research. This software can also be used outside the scope of this thesis, using other 3D objects. This tool can be used in several camera based applications. For instance, to quickly test a one- or multi-camera setup.

The group has built a prototype for Solwr to use for extended testing, and have detailed their work in how-to-guides in the appendix. These guides might be helpful for anyone using a similar setup to recreate the work described in the thesis.

The group has made a simulator. They created an environment for inserting camera specs, and outputting a simulated point cloud based on the cameras' positions. The Webots simulator has been a big part of the research done in this thesis, and can be used by others facing similar challenges.

The group's simulation and FOV software provided data, which the group offer to other thesis' as useful insight, and resource to build new software/projects.

1.5 Thesis outline

The thesis is written in a chronological manner, giving the reader an idea of how the group proceeded in order to reach their conclusion. A list of tables, figures as well as an interactive table of contents is placed at the beginning of the thesis.

Chapter 2 provides information needed in order to fully understand the rest of the thesis. It provides explanations for terms, mathematical methods and formulas and hardware explanations.

Chapter 3 contains the methods in which the group has reached their conclusion and details how the current solution of Sort™ functions. It details the workings of their own software, the possible designs they considered, the possible cameras they considered, as well as their simulator. Each part's results provided the group with the theoretical foundation on which they based their conclusion.

Chapter 4 gives the reader insight into the building process and programming of the group's prototype. It was based on the results from chapter 3, and was built in Solwr's own warehouse. Included in the appendix and referenced from the chapter, there is a guide anyone can use in order to replicate the group's results.

Chapter 5 is the results of the work outlined in chapters 3 and 4. Chapter 5 includes the results and the thought process, which lead to the group's proposed solution.

Chapter 6 is where the group discusses the results, possible shortcomings, things to remember and think about and their ideas for improvement they themselves did not get around to.

Chapter 7 contains the final conclusion. This is where the group gives their recommendation for how Solwr should proceed in regards to Sort™. In addition to this, it explains the remaining work needed before the proposed solution can be fully implemented into Sort™.

The appendix is a collection of documents the group has deemed helpful for the reader and/or anyone wanting to replicate the results of this thesis.

1.6 Related work

Cameras and camera acquisitions are topics which is widely researched, since it is rapidly evolving and automating society. In this chapter, research carried out in the industry on potential problems that are relevant to this thesis are discussed. The aim of this chapter is to summarize the contributions from various researchers across the domains of camera technology, stability, and accuracy in a industrial environment. With a prospect to understand the scope of these thesis's, which in turn influences the choices of this thesis.

1.6.1 Applications of High-Precision Optical Imaging Systems for Small Unmanned Aerial Systems in Maritime Environments

This thesis for the degree of Philosophiae Doctor and written by Christopher Dahlin Rodin with the university of Norwegian University of Science and Technology. This thesis provided useful research regarding cameras exposed for various maritime environments. It explains several ways an image can be distorted by the environment a camera is placed within, and how the image is effected. There are several methods presented towards preventing distorted images, both by utilizing software and hardware. This thesis provided valuable knowledge since the group decided on a move-able camera in the new acquisition, which may cause some distortion in the images when moved at high velocity [10].

1.6.2 Image space coverage model for deployment of multi-camera networks

A master thesis written by Eslam Samir Eissa with the University of Windsor has described a similar issue in regards to area covered by cameras. His thesis is based on a multi-camera setup and how to place each camera in order to see the entire surface of a 3D-object. His result is a model that places cameras with maximum coverage in mind. [11]

Chapter 2

Preliminaries

This chapter contains explanations for terms, mathematical methods, formulas, concepts and hardware used in this bachelor thesis.

2.1 Terminology

Acronyms

IPC Industrial Personal Computer

PLC Programmable Logic Controller

POU Programmable Organization Unit

PRG Program

FB Function Block

FUN Function

DUTs data unit types

ENUM enumeration

GVL Global

I/O Input / Output

HMI Human Machine Interface

FOV Field of view

ToF Time of flight

LiDAR Light Detection and Ranging

VAT Value added tax

Notations

V AC Voltage Alternating Current

V DV Voltage Direct Current

2.2 Theory

This section covers concepts, methods and mathematical formulas used in the bachelor thesis. It is meant to give the reader a theoretical foundation of the concepts used in the thesis.

2.2.1 Point clouds

Point clouds are a method of showing 3D information. When a sensor scans a 3D object it returns points detected along the surface of the object. In addition to the position of each point on the surface in 3D space, the color of each point can be added to the point cloud to reproduce both the shape and color of the object.

2.2.2 Ray casting

Ray casting is a technique used in computer graphics. The technique determines what objects are visible from a certain view point, like from a camera. Rays with origins in the camera lens are cast outward until they reach a surface. The computer renders objects whose surface has been reached by a ray [12].

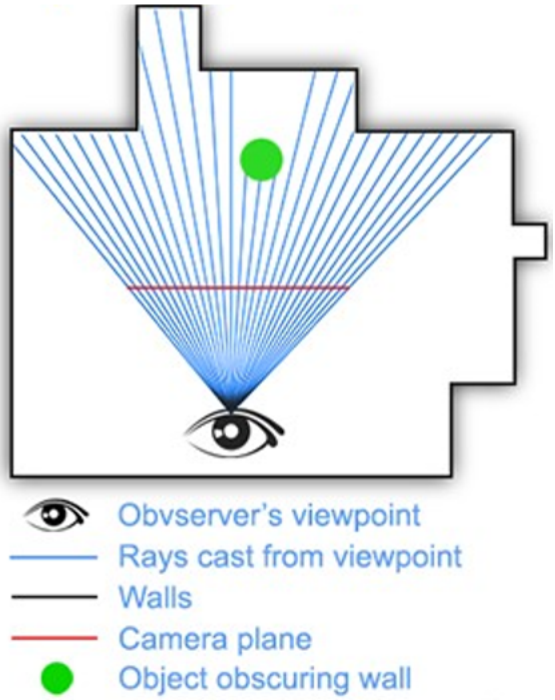


Figure 1: Ray casting, visualized

2.2.3 Asynchronous programming

Asynchronous programming is a programming method where a computer performs several tasks simultaneously. This method uses more of the computational power available to the computer, but makes tasks that can be divided into several sub tasks, less time-consuming [13].

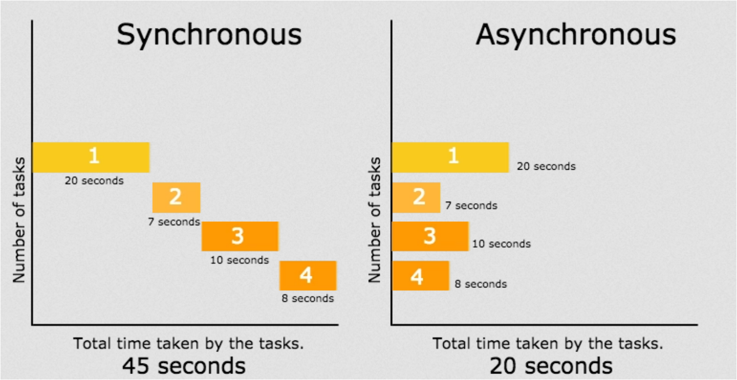


Figure 2: Asynchronous programming, visualized

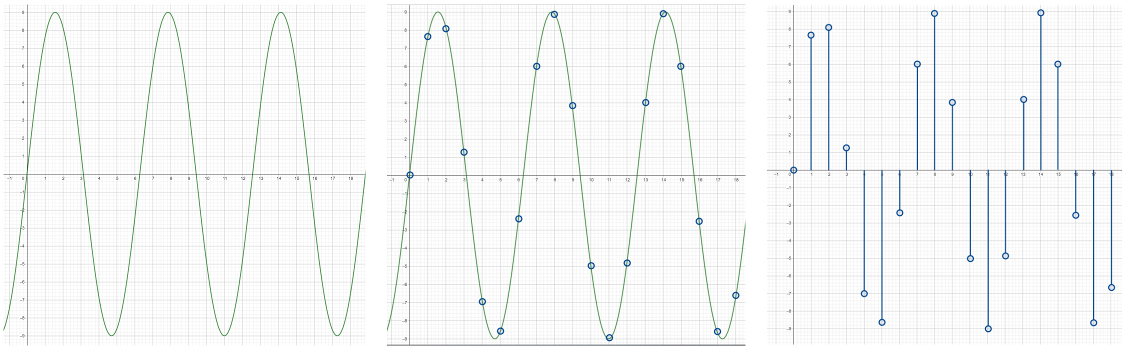
2.2.4 Formulas and mathematical methods

Discretization of functions

Typically, when graphing functions, one would use a continuous parameter value. i.e the output for all possible parameter values are presented. It is impossible to store all possible values when digitally analyzing functions.

Instead, one normally uses discrete parameter values, and stores the values of the functions at those values in memory. This is called sampling. The resolution can be adjusted to be higher or lower, depending on the sampling frequency. A higher frequency gives a higher resolution, and a frequency approaching infinity gives seemingly continuous values.

Figure 3 illustrates this method. It shows a continuous sine function with a sample per one unit along the x-axis. The output of this discretization can be seen in sub figure 3c.



(a) Continuous function (b) Sampled function (c) Samples

Figure 3: Sampling of continuous function

Direction of vectors

In order to construct a vector, one needs two points in space. The vector between them contains information about the direction of movement needed to move from one point to the other.

$$Vector = (x_1, y_1, z_1) - (x_2, y_2, z_2) \quad [14] \tag{1}$$

Where x, y and z are the points' coordinates.

Length of vectors

Vectors have both direction and magnitude. In order to calculate it's magnitude, or length, one can use the following formula:

$$Magnitude = \sqrt{a^2 + b^2 + c^2} \quad [14] \tag{2}$$

Where a, b and c are the vector components.

Euclidean space

Euclidean space is a fundamental space of geometry. It represent physical space. It can represent both 2D- and 3D-coordinate systems. It was the first way of representing physical space. It remains the de facto standard today [15].

Rotation matrices

Rotational matrices are used in linear algebra in order to perform rotations in euclidean space [16]. An example of such a matrix is as follows:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This rotational matrix performs a rotation of θ degrees/radians around the z-axis.
For rotations along the x- and y-axes. The rotations would look like this:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

Rotation matrices for x- and y-axes, respectively

Normal vector
A normal vector is a vector that is perpendicular to a surface at a given point. It can be constructed using no less than three points, or two vectors, that exists on the surface it is perpendicular to. The formula for a normal vector is:

$$\vec{N} = \vec{AB} \times \vec{AC} \quad [14] \tag{3}$$

Where $\vec{AB}$ and $\vec{AC}$ are vectors pointing from the same origin to two different points.

In order to simplify usage of this vector, the normal vector can be converted to a unit normal vector, represented by $\hat{N}$. This operation leaves the direction of the vector unchanged, but changes the length to 1. This is done by dividing all components by the total length of the vector.

$$\hat{N} = \frac{\vec{N}}{|N|} = \frac{(a,b,c)}{\sqrt{a^2+b^2+c^2}} \quad [14] \tag{4}$$

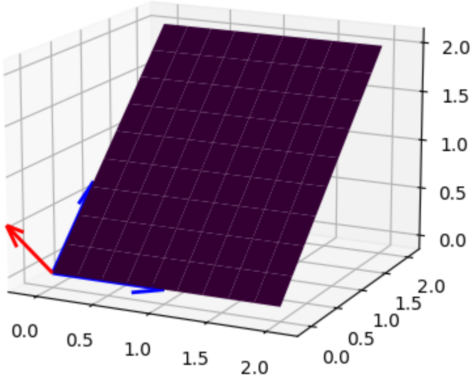


Figure 4: Normal vector in red perpendicular to plane

Planar equation
A plane in 3D-space can be represented as an equation. The equation for any plane is:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \quad [14] \tag{5}$$

Where a, b and c are the components of a normal vector, x, y and z make up a point in 3D-space and d is a constant. Alternatively, it can be written as such:

$$a \cdot (x - x_0) + b \cdot (y - y_0) + c \cdot (z - z_0) = 0 \quad [14] \tag{6}$$

Where a, b and c are the components of a normal vector, x, y and z make up a point in 3D-space and x_0, y_0 and z_0 is a point in the plane.

Points' relative position to planes

Once a unit normal and a point in the plane is obtained, a different point can be determined to be above or below said plane, relative to the normal vector. By using the property of a vector where if it is inverted, it has negative length, one can determine the perpendicular movement from the plane to an arbitrary point. Meaning if the normal vector has positive direction as seen on figure 4 any movement towards a point on the opposite side of the plane would be in the vector's negative direction.

Whether the movement is positive or negative, can be determined by taking the dot product of the normal vector and a vector going from the origin point of the normal vector to an arbitrary point.

$$Amount\ \ of\ \ normal\ \ vectors = (P - O) \cdot \hat{N}\ \ [14] \tag{7}$$

Where P is an arbitrary point, O is the origin of the normal vector and $\hat{N}$ is the unit normal vector.

Should the amount of normal vectors be negative, the movement is opposite to the normal vector. Therefore, the point is below the plane relative to the normal vector.

Converting distances for one camera to another

Seeing that most cameras have different fields of view, two different cameras will see two different pictures from the same view point. In order to calculate what vantage points will provide the same picture, at least along one axis, the group has defined the following formula:

$$Distance_2 = \frac{tan(theta_1) \cdot distance_1}{tan(theta_2)} \tag{8}$$

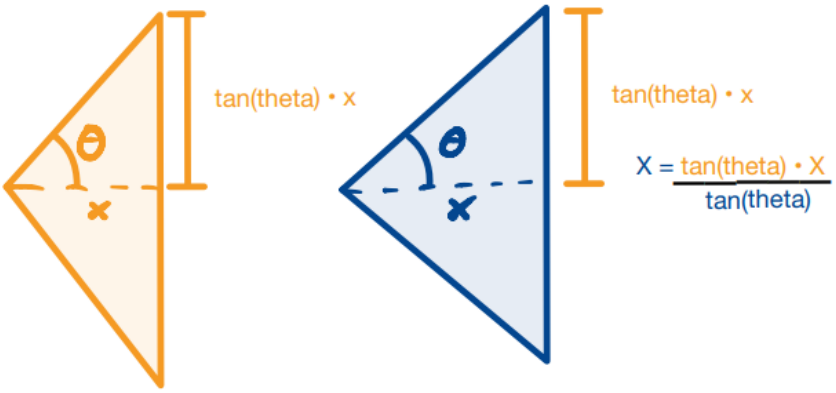


Figure 5: Visualization of formula

Where $distance_1$ is the distances used with the first camera, $distance_2$ is the same distance for the new camera, $theta_1$ is the FOV angle of the first camera and $theta_2$ is the FOV angle of the new cameras.

Defining a coordinate system using vectors

A coordinate system is suspended by n number of axes, with a single point considered as its origin.

A new coordinate system can be created using self-defined vectors representing the axes and aligning the ends of the vectors in the origin.

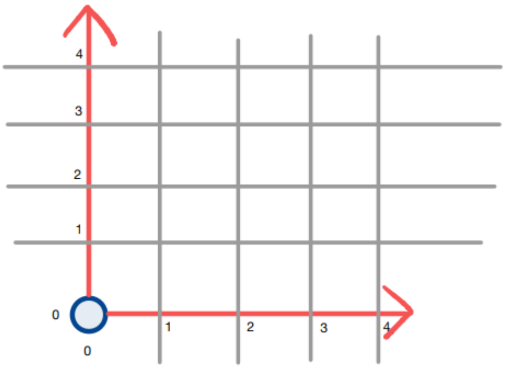


Figure 6: Coordinate system suspended by two vectors

The coordinate system in figure 6 can for example be made using vectors defined in a different coordinate system. For instance, the horizontal vector can be defined as $[x,y,z]$ in the original coordinate system. The vertical vector would then be a vector perpendicular to the first one. The origin in the new coordinate system would be the blue dot, which is an arbitrary point in the original coordinate system.

Using these vectors as the axes, one can now describe location in space using the amount of each vector, moving relative to the origin. One unit along an axis would mean one length of the respective vector. An example of vectors representing a new coordinate system can be seen in figure 7. Any point defined in the new coordinate system can be transformed back to the original system by using equation 9.

$$Point\ \ in\ \ space = x \cdot vector_1 + y \cdot vector_2 + origin \tag{9}$$

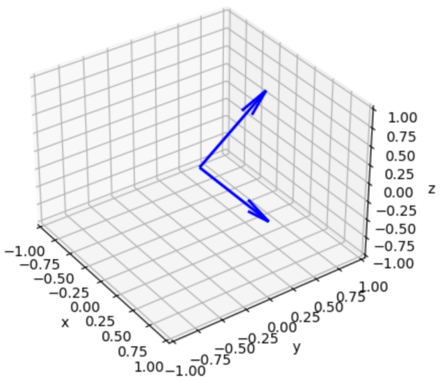


Figure 7: Vectors creating a new coordinate system

Rectangular- and spherical coordinates
Coordinates in euclidean space can be represented in multiple ways. Two of which are rectangular coordinates and spherical coordinates [14].

Rectangular coordinates, or Cartesian coordinates, are based on axes that are perpendicular to one another. For example the x-, y- and z-axis. Positions are defined by moving along each of the axes. E.g (2, 3, 4).

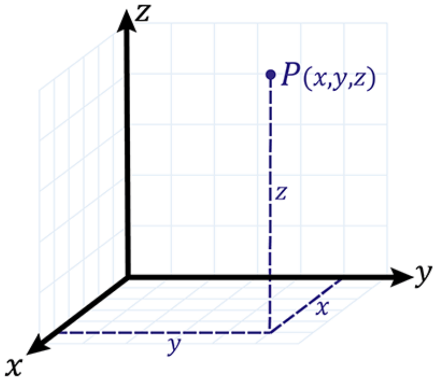


Figure 8: Rectangular coordinates

Spherical coordinates are based on angles and the total length of the vector from an origin to a point on a sphere around it. This way of representing coordinates is preferred when applying rotations. Coordinates are represented by ρ/r (radius of sphere), ϕ (vertical angle) and θ (horizontal angle). E.g (1, 0.5π, *pi*)

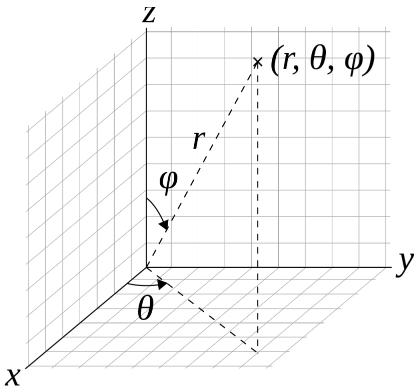


Figure 9: Spherical coordinates

In order to convert between rectangular and spherical coordinates one can use the following equations:

From rectangular- to spherical coordinates:

$$\rho = \sqrt{x^2 + y^2 + z^2} \tag{10}$$

$$\phi = \arccos(\frac{z}{\sqrt{x^2 + y^2 + z^2}}) \tag{11}$$

$$\theta = \arccos(\frac{y}{x}) \tag{12}$$

From spherical- to rectangular coordinates:

$$x = \rho \cdot \sin(\phi) \cdot \cos(\theta) \tag{13}$$

$$y = \rho \cdot \sin(\phi) \cdot \sin(\theta) \tag{14}$$

$$z = \rho \cdot \cos(\phi) \tag{15}$$

2.2.5 3D cameras

There are multiple ways of collecting 3D information using cameras. One way is by using multiple cameras or by using a camera in combination with a different sensor. Using two cameras to gather 3D data is called a stereo camera. Two images are taken at slightly different points and by identifying common features and looking at the displacement, the depth can be calculated for each feature. Instead of using two cameras, one of them can be replaced with a projector that sends out structured light patterns. When these patterns reaches a surface they will be distorted when viewed from a different angle to the projector. The depth can be calculated using this distortion. Instead of a projector it is also possible to use a LiDAR to get information on depth in the image. This type of camera falls under the ToF category, time of flight [17].

2.2.6 Pixel coordinates to world coordinates

In order to go from pixel values with depth to full coordinates, a transformation is needed. This transformation is built on the intrinsic parameters of the camera taking a picture. This info is either known ahead of time or gained trough a calibration of the camera. This information forms a matrix called the calibration matrix or the intrinsic parameter matrix and is denoted by K. The K matrix is built up as follows:

$$\begin{bmatrix} f \cdot s_x & f \cdot s_\theta & o_x \\ 0 & f \cdot s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where the f is the focal length of the camera. s$_x$ and s$_y$ are the size of the pixels, so if they have the same value the pixels are square. s$_\theta$ is the skew of the pixels, usually there is little to no skew. o$_x$ and o$_y$ are the coordinates of the center pixel. If you have the pixel value and distance out to the point, this can be converted into world coordinates by multiplying the pixel coordinate with the inverse of the K matrix.[18]

2.2.7 3D Transformations

In order to move a 3D object, you need to apply a transformation. This transformation can be done with a transformation matrix, T. The transformation is a combination of rotation and translation. For a 3D rotation you need a 4x4 transformation matrix containing a rotation matrix and a translation matrix.

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

Here the R is the 3x3 rotation matrix and T is the translation vector.[18]

2.3 Software, protocols and technology

2.3.1 PLC OPEN

PLC open is an association representing various industries, with a focus on harmonization of control programming, application and interfacing engineering. PLC open is standardizing base functions for a re-usable software across multiple hardware platforms, this is in order to increase efficiency in cost and time [19].

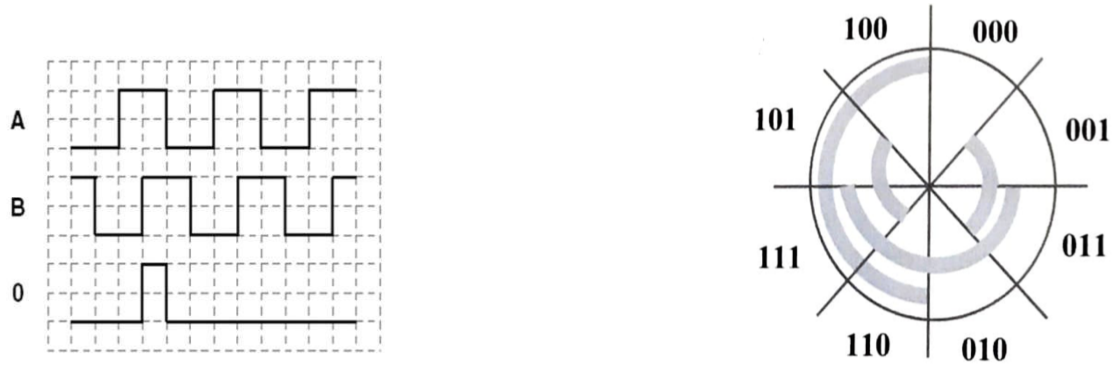
2.3.2 EtherCAT

”*EtherCAT (Ethernet for Control Automation Technology) is a real-time Ethernet field bus that developed by Beckhoff and managed by ETG (EtherCAT Technology Group)”[20].*

EtherCAT is built on standard Ethernet, employing the physical layer of Ethernet, RJ45 connectors, 100BASE-TX, and EBUS to transport LVDS signals as the transmitting media. Full duplex data transport is possible with EtherCAT, with a maximum rate of 100 Mb/s. It also enables redundancy and hot swap technologies in terms of safety and reliability [20].

2.3.3 Encoder

Positioning, RPM, and speed control are common uses for encoders. The use of light is a commonly utilized principle in encoders. In general, an optical encoder is made up of one or more sets of light emitting diodes, photo-detectors (photo transistors), and a disk. The number of holes on the revolving disk determines the encoder's resolution. Incremental and absolute encoders are the two most common types [1].



(a) Pulse train using channel A, B and 0 in an incremental encoder

(b) Absolute encoder disk

Figure 10: Illustrations of a incremental and absolute encoder [1]

Incremental encoder
A PLC can utilize an incremental encoder by counting the number of pulses received. However, a exact reference point is required for the encoder to function. E.g to move an axis until it reaches a set point, and store this position as reference. This action is often referred to as a homing. After determining the axis position, the axis can be moved with the accuracy of the encoder. However, if the encoder loses power a new homing will be required. A common incremental encoder has 3 channels: A, B and 0, and is illustrated in figure 10a. [1]

The pulse train on channels A and B are 90 degrees out of phase with each other. This allows you to figure out which way the encoder turns. The rotation is clockwise if A has a rising edge before B, and vice versa if B is first. Channel 0 is a pulse that is used to keep track of the encoders full rotation. [1]

Absolute encoder

Absolute encoders know the exact position at all times without needing to calibrate to a known reference position. An absolute encoder outputs the actual angle or position as a binary signal. The number of bits is equal to the number of rows of holes in the disk. It is common to arrange the holes in line with Gray coding. Gray code is a method where only one bit changes between the sectors in the disk. Figure 10b is an example of a disk using 3 rows and gray code. Since the disk has 3 rows of holes, the encoder has a resolution of 45°, calculated in equation 16 and 17 [1].

$$Number\ \ of\ \ positions = 2^{number\ \ of\ \ rows\ \ in\ \ disk} = 2^3 = 8 \tag{16}$$

$$Encoder\ \ resolution = \frac{360^\circ}{8} = 45^\circ \tag{17}$$

Chapter 3

Research, software & simulation

This chapter details how the group has found limitations for their camera placement through CAD, considered several designs and chose the most suitable camera. In addition to this, it explains the group's software and how their simulator was built.

Chapter 3 is the main part of the thesis where the group details their workflow in order to reach the final results. The chapter also gives the reader insight into the thought process behind the decisions made.

The subsections 3.3 and 3.4, which detail the choices made regarding the design and camera, include a summary that explain the chosen option and why.

3.1 Current solution

This section contains information about the current version of Sort™. It is meant to provide the reader with context and give an understanding of how the cameras work together with the other parts of the system.

3.1.1 Introduction current Sort™ solution

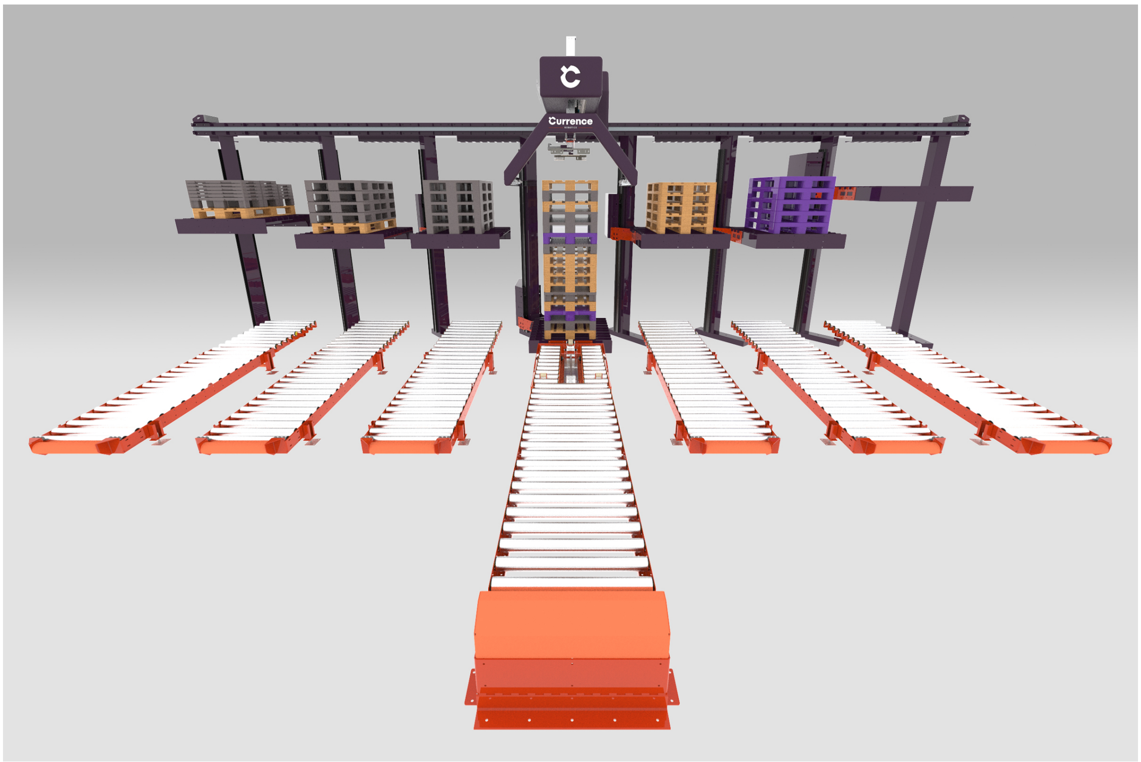


Figure 11: Mockup of Sort™ [2]

Sort™ is a pallet sorter robot, illustrated in figure 11, developed by Solwr, previously known as Currence Robotics. Sort™ was developed with scalability in mind, and therefore is modular to fit both small and large facilities. It can work 24 hours a day, if needed. It can sort all pallets including: Euro pallet, plastic pallets, eco-pallets, half sized pallets and quarter sized pallets[2].

Sort™ is made up of a pallet-transporting conveyor system, numerous out-feed towers, and one in-feed tower for receiving pallet stacks. A robotic gripper is also used to sort each pallet. Sort™ is designed for warehouses and enterprises like H.I. Girtz and ASKO, who receive hundreds of pallets per day of all types.

3.1.2 Work flow

(a) Sorting using forklift [21]

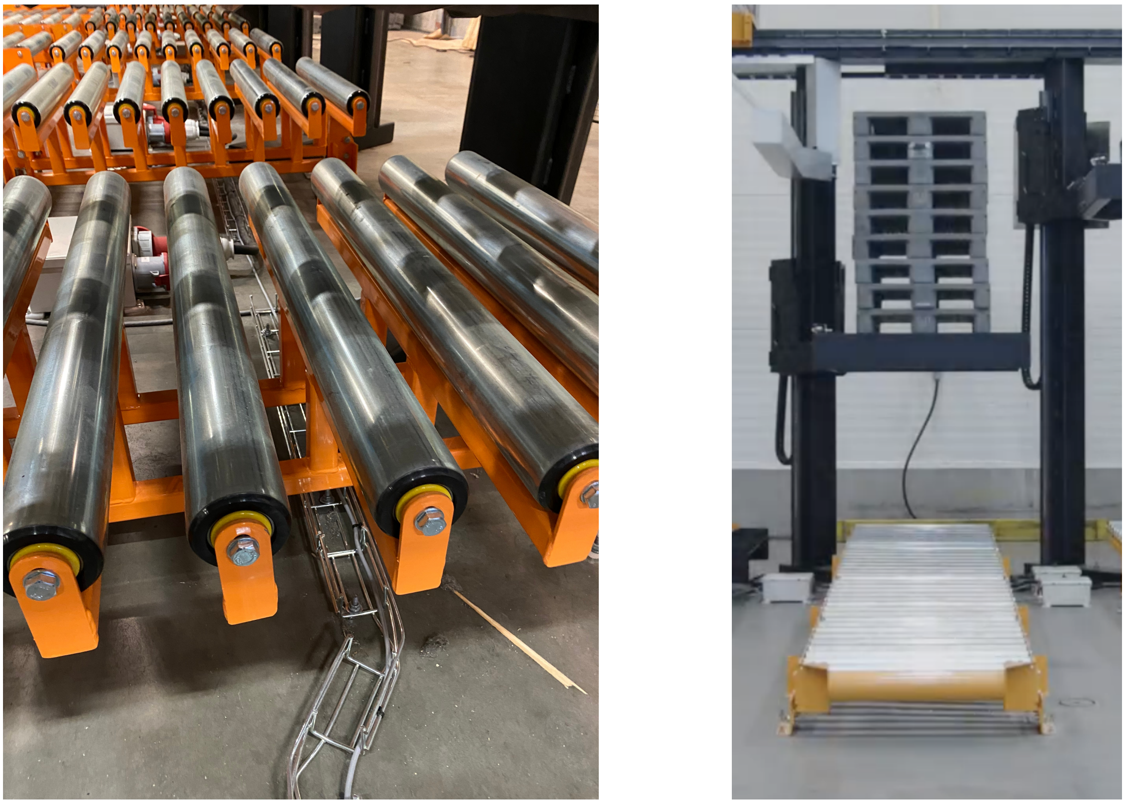
(b) Wooden and plastic pallet [22]

Figure 12: Sorting different pallets using forklift

Traditionally, the incoming pallets must be manually sorted by staff using a forklift. Since this method is repetitious and demands precision over lengthy periods of time, it is time consuming and psychologically demanding. Employees must sort the pallets in addition to looking for faults, and other reasons why the pallet should be eliminated from production. Pallets that have been discarded will be repaired or recycled, thus they must be placed into separate stacks.

Instead, by utilizing Sort™, the employee only needs to sort an assortment of pallets into stacks. The stacks can be from one and up to 17 pallets high. Thereafter, the pallet stack is delivered at the in-feed tower, which is the center conveyor in figure 11.

3.1.3 Conveyor system



(a) Conveyor and resistance rollers

(b) How it is used

Figure 13: Conveyors and the resistance rollers

The conveyor system on the current version of Sort™ is mechanical, which means there are no motors or other forces pushing the pallets forward. This locomotion is due to the angle of each conveyor, and low resistance rollers to gain speed. To prevent pallets gaining too high velocity, some of the rollers are braking rollers, which means rollers with high resistance. The conveyor system will be adapted for each customer, due to each facility and their needs.

3.1.4 Camera portal

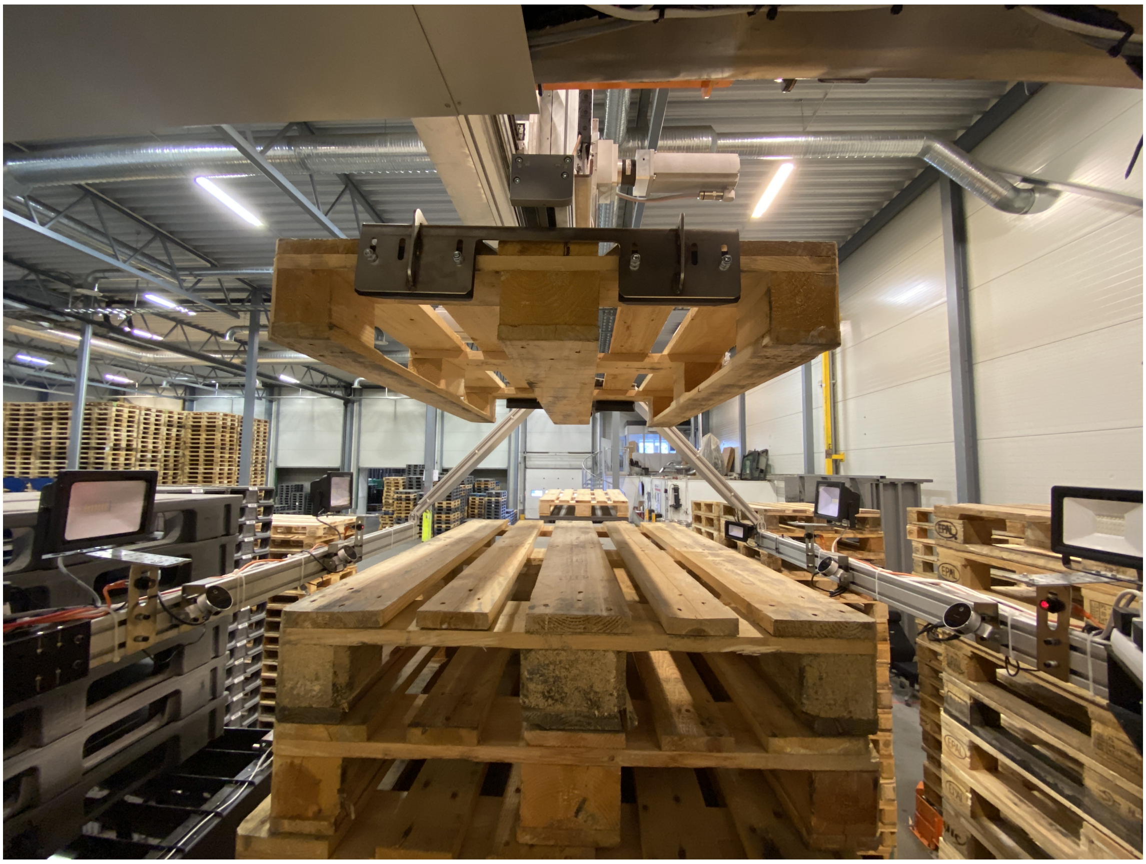


Figure 14: Sort™ in action

The in-feed tower also consist of a frame surrounding the pallet stack, called the camera portal. The camera portal is where six LiDAR cameras are used to create a 3D point cloud of each pallet with today's design. The point cloud in combination with Solwr's image algorithm, the system can determine the type, size, and the state of each pallet. The state is evaluated by cleanliness, rotation in any parts, cracks, missing material and other defects.

3.1.5 Robot gripper

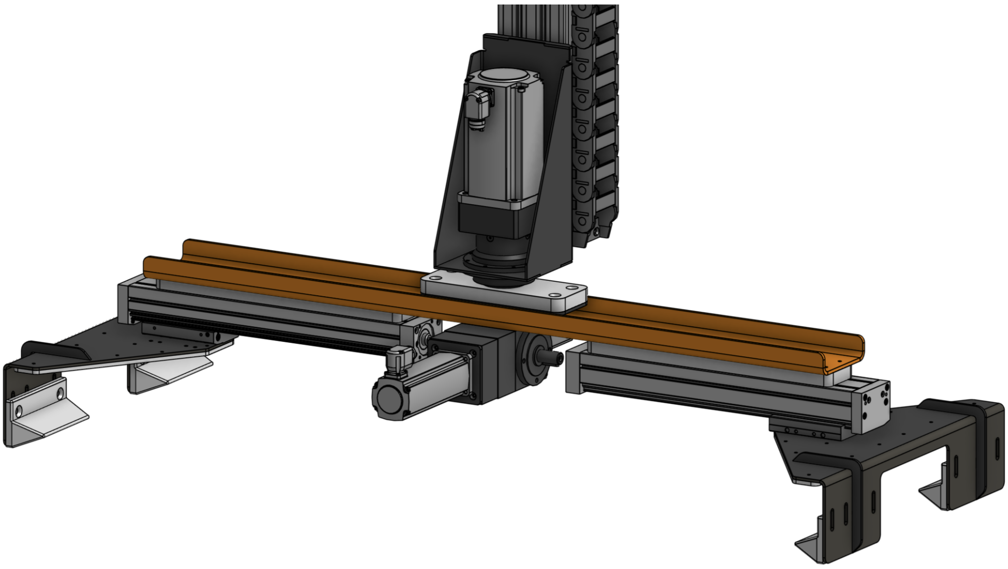


Figure 15: Illustration of the robot gripper

The robot gripper is illustrated in figure 15, and has 5 movable axes to navigate. The robot is fully automated, and is programmed using high level programming, machine learning, and a PLC.

3.1.6 Movable axes

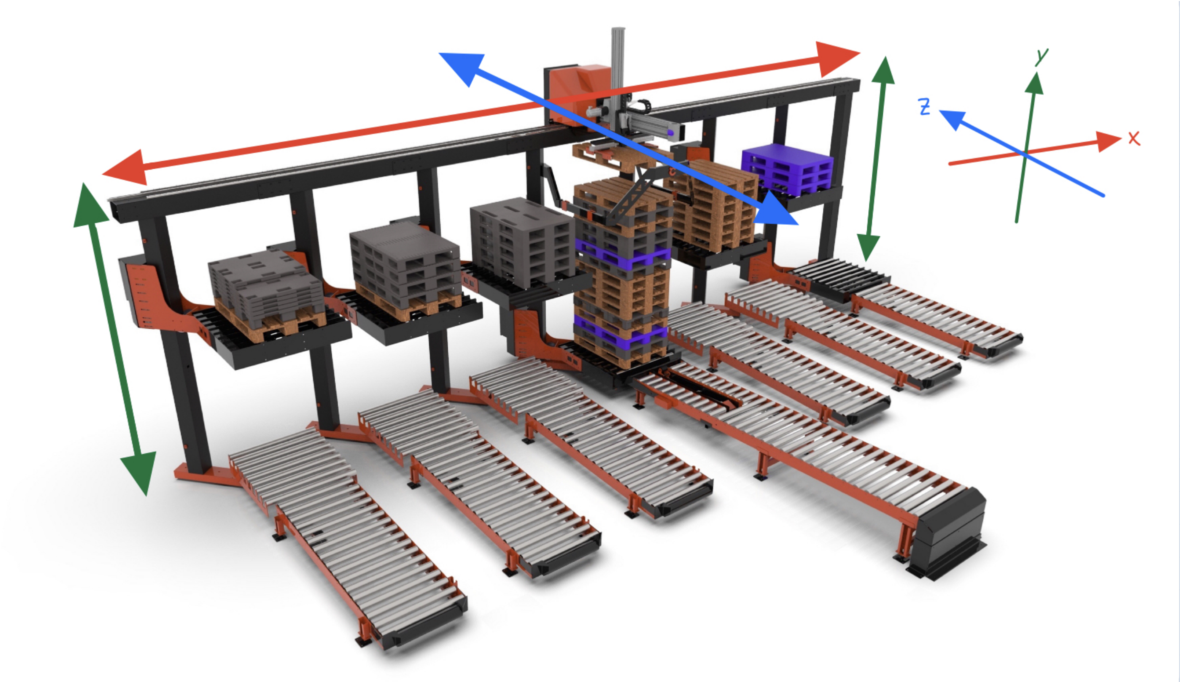


Figure 16: Illustrates axis one, two and three as x, y, and z on Sort™.

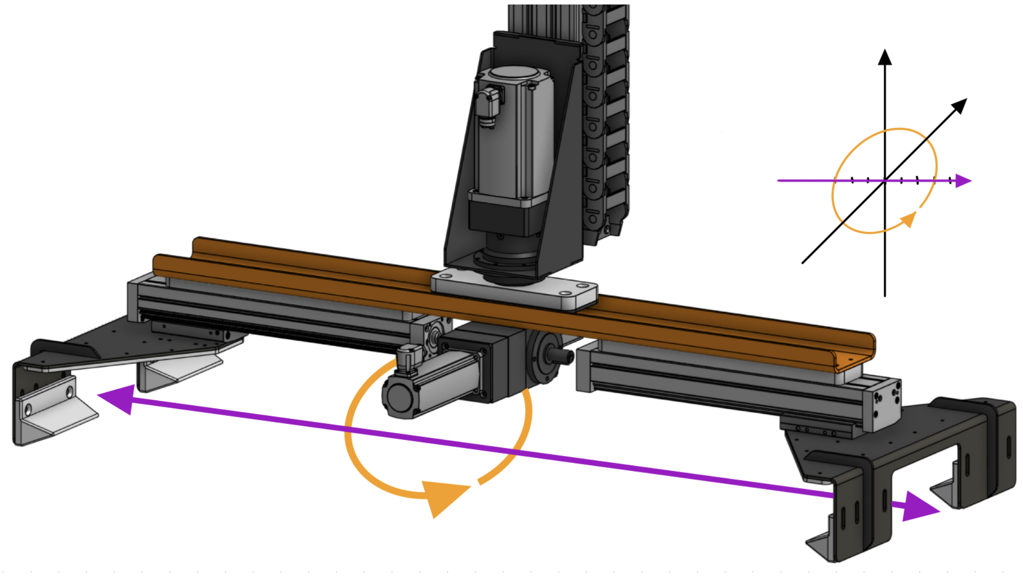


Figure 17: Illustrates axis four and five as the rotating and gripping axis on Sort™.

- Axis 1: Horizontal axis (red)
- Axis 2: Vertical axis (blue)
- Axis 3: The depth axis (green)
- Axis 4: Rotational axis (orange)
- Axis 5: The gripping axis (purple)

3.1.7 Out-feed

Using sensors and high-level programming, the robot counts every pallet within each tower. When a tower's capacity is reached, the out-feed tower is lowered and unloaded for an employee to collect. The stack's maximum capacity can be adjusted throughout development, however it is currently set at 17 pallets.

The group will be working with the version of Sort™ located at H.I. Giørtz, illustrated in figure 11. It was built in 2019 and continuously updated with improvements. This instance of Sort™ was originally built as a prototype, but has since been developed into a commercial product [R].

3.1.8 Current camera solution

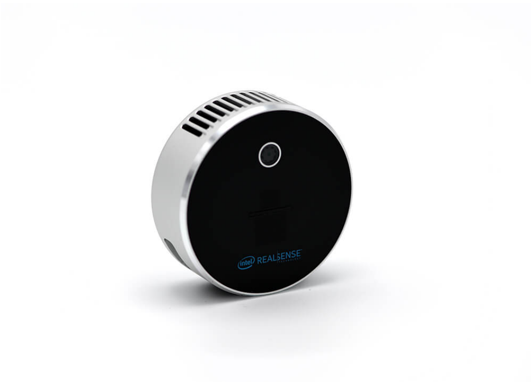


Figure 18: Intel® RealSense™ L515

The camera currently used on Sort™ is the Intel® RealSense™ L515 [23], seen in figure 18. This is an affordable off-the-shelf LIDAR camera, with free plug-and-play software, named Intel® RealSense™ Viewer.

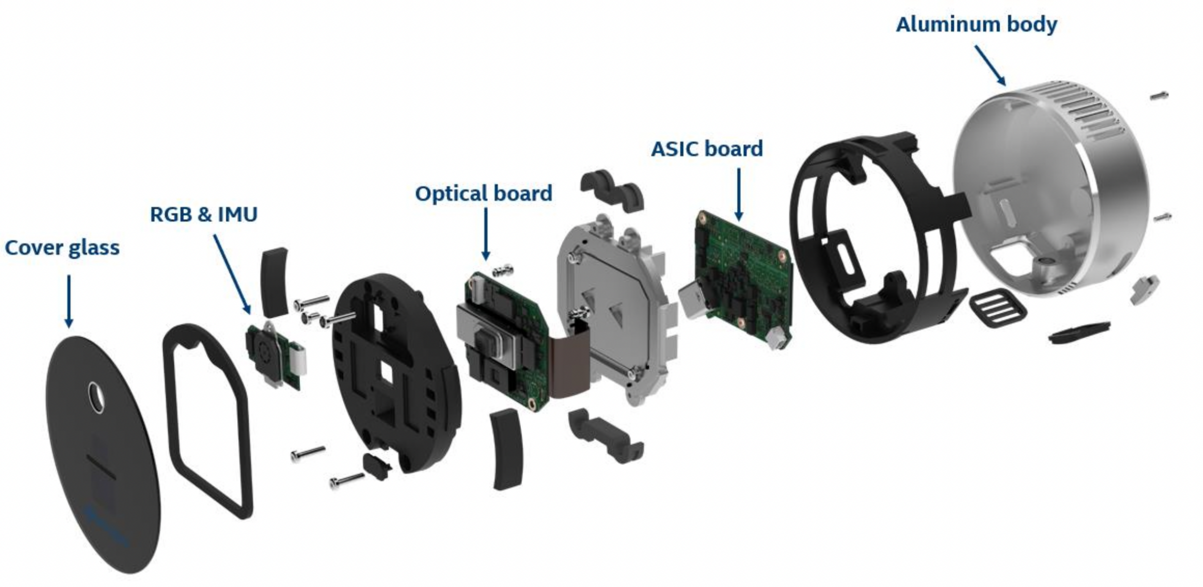


Figure 19: Intel® RealSense™ L515 internal components [B]

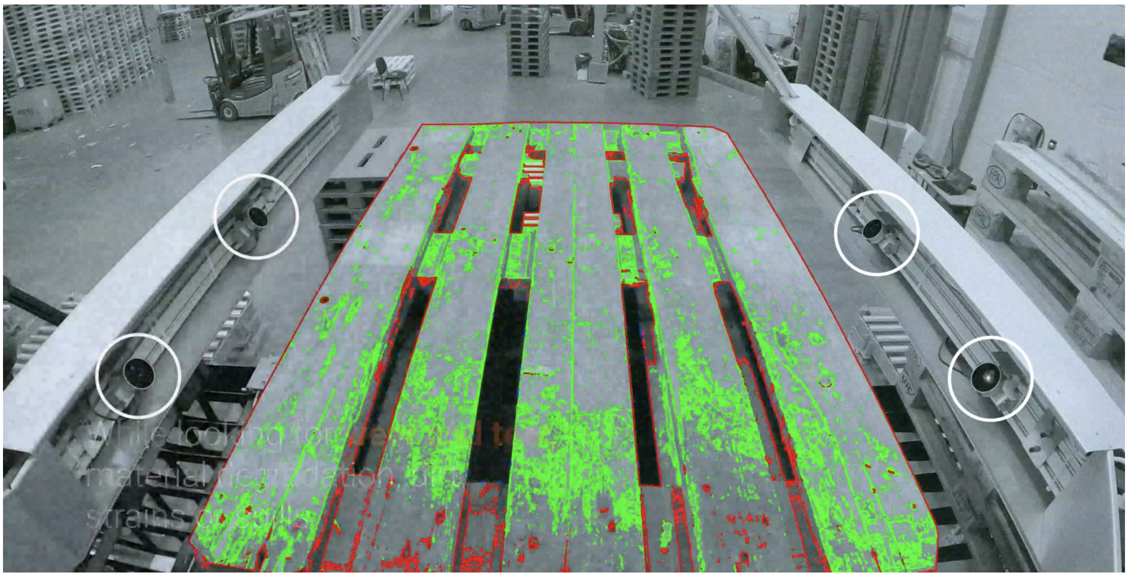
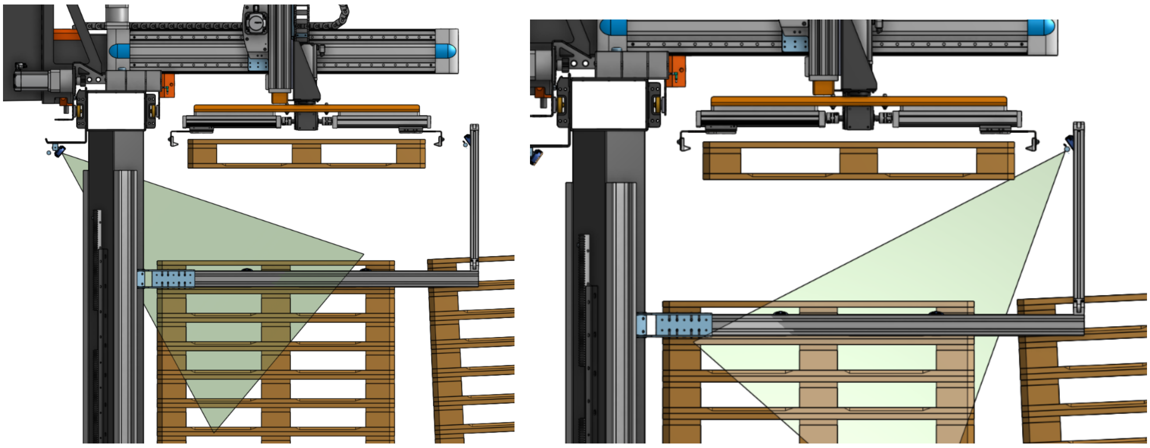


Figure 20: Placement of the four cameras capturing the side and bottom of the pallets.

Six L515 cameras are installed in the current version of Sort™. Four cameras are located inside the camera portal shown in figure 20. They are responsible for capturing the sides and bottom of the incoming pallet. The two remaining cameras are placed in front and behind the incoming pallet, as illustrated in figure 21a and 21b. These cameras capture the four corners, front, back, and the top of the pallet. Together, all images forms a 3D point cloud representing the pallet, and can be used to determine the type and state of the pallet.



(a) Back top camera position and field of view (b) Front top camera position and field of view

Figure 21: Illustration of all current camera placements, and field of view

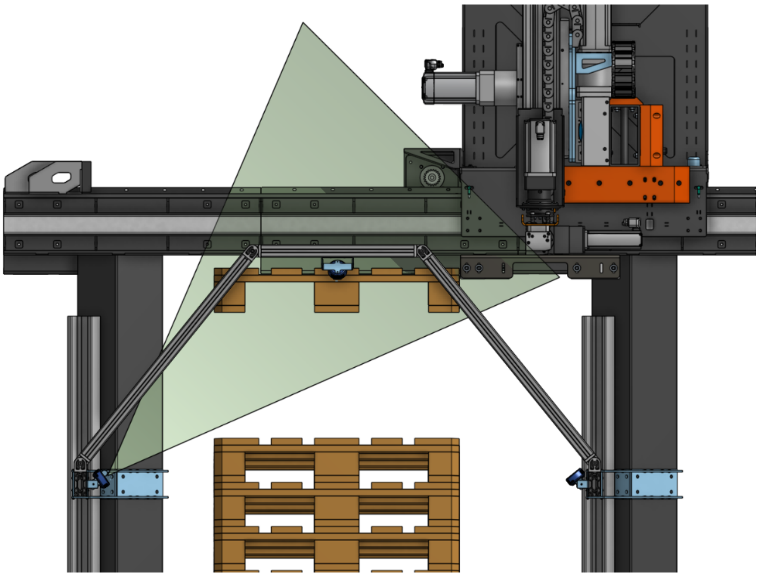


Figure 22: One of four cameras in figure 20 and field of view.

The Intel® RealSense™ L515 cameras are ideally used in the range 0.25 to 9 meters. Their field of view (FOV) is 70° by 55° (±3°). The depth output resolution can be up to 1024x768 pixels with a frame rate of 30 frames per second (FPS) and the depth accuracy is 5mm to 14mm. The resolution and ideal range vary based on the cameras picture format. The different formats and their specifications can be seen in table 1 [B].

Format(Depth Resolution)	Number of depth points per second	FOV	Range @15% reflectivity	Range @95% reflectivity
QVGA (320x240)	2.3M	70° x 55°	0.25 - 3.9m	0.25 - 9m
VGA (640x480)	9.2M	70° x 55°	0.25 - 3.9m	0.25 - 9m
XGA (1024x768)	23.6M	70° x 55°	0.25 - 2.6m	0.25 - 6.5m

Table 1: Intel® RealSense™ L515 depth specifications B

3.1.9 Point cloud

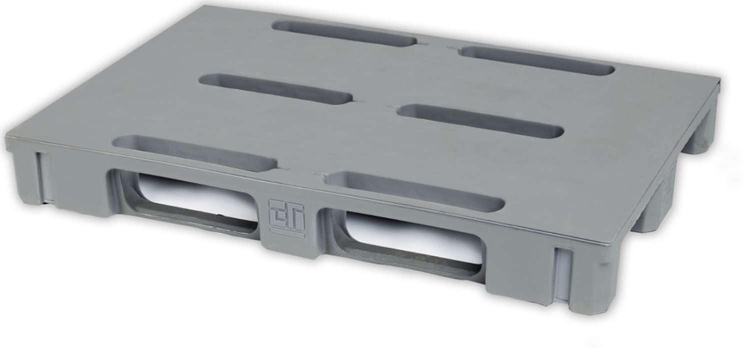
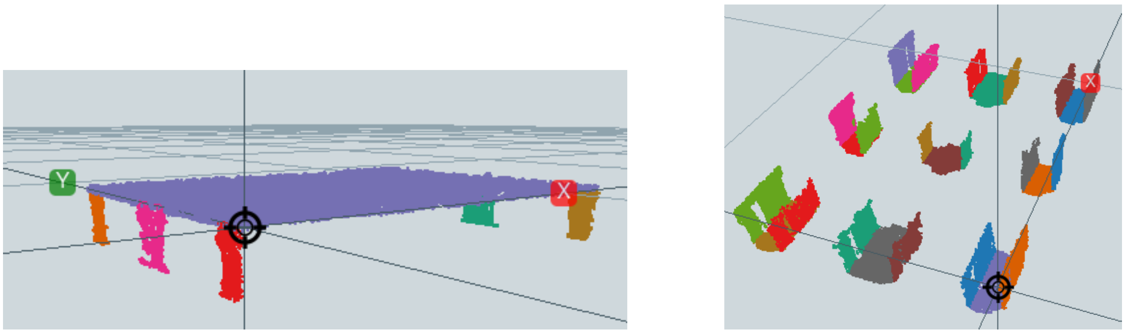


Figure 23: NLP pallet [3]



(a) Top view point cloud of an NLP pallet

(b) Side view point cloud of an NLP pallet

Figure 24: Point cloud image of a NLP pallet using Intel® RealSense™ L515, provided by Solwr.

The current cameras each take a single picture and then combine them with LiDAR measurements to generate 3D data for various points in the image. This is represented as a point cloud that combines the photo with 3D world coordinates, as seen in figure ??. Since the cameras are at fixed positions relative to each other, it is possible to stitch together the different point clouds into one piece of 3D information. This shows all the surfaces of the pallets that the six different cameras can see. The combined 3D model is then used to classify the type of pallet, and analyzed for potential damages.

The LiDAR camera is a type of ToF camera, a time of flight camera. It generates the 3D image by combining a time of flight sensor with a camera. In the Intel® RealSense™ L515 camera the ToF sensor is a laser that sends out a beam, and then measures the time it takes for it to reflect off something and return. This data gives the depth of any given point in the image taken by the camera component.

3.1.10 Economy

In the economy section the group considers components and components that is likely to be upgraded, removed or replaced in a potential new solution.

Intel® RealSense™ Camera & robot gripper
The current pricing for the Intel® RealSense™ L515 is 589$ per camera[23]. Since the current camera system is a static system, there are no moving mechanical components changing the cameras position. This means that the cameras needs close to zero maintenance, and therefore adds little to nothing maintains cost. However, in order to capture a complete coverage 3D point cloud in the current image acquisition, the robot gripper must lift each pallet. The lifting mechanism will almost certainly be included in future solutions, and therefore can be excluded from the economy calculations.

Raspberry Pi
In the current solution each camera requires one Raspberry Pi for each camera, since the cameras uses the USB protocol. The USB protocol has limitations in data-flow, and by using a Raspberry PI the data can be transferred to the industrial computer, using the Ethernet protocol.

Camera portal
The camera portal will most likely be altered in some way, in the new camera acquisition. However, this does not cause any problems, since the camera portal in designed by needs of the cameras localization. A cost in altering the camera portal design is highly likely, but impossible to calculate. The group has therefore evaluated that it's important to factor in, but as a unknown cost this early in the process.

components that is likely to be upgraded

Hardware	Quantity	Price per unit	Total price
Intel® RealSense™ L515	6	589$ [23] excluded tax and shipping	3 534$
Raspberry Pi	6	145$ [24] bought locally in Norway	870$
Ethernet cables	6	unknown	unknown
Camera portal	1	unknown	unknown

Table 2: The current camera systems hardware pricing

3.1.11 Statistics

The Sort™ system at H.I Giørtz is linked to a data collection server named Datadog. Datadog allows Solwr to monitor the system's performance and accuracy in real time. Table 3 was created using the data collected over the last month, as of the 5th of May, 2022.

Measurement	Value
Active, per day	6.0h
Pallets processed	25054
Average cycle time	24s
Pallets per hour	150

Table 3: Sort™'s statistics gathered the 5th of May 2022

3.2 Self-developed FOV software

In order to visualize and demonstrate different solutions, as well as test the viability of certain cameras, the group has created software that presents the results visually. The script, found in appendix [G], was used as a quick way to get rough results without needing to program a simulator or build a prototype.

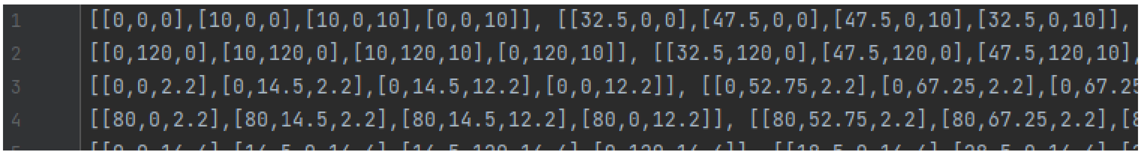
The script is written in Python, can visualize a camera's field of view and determine which points of a 3D-object that can be seen from a camera's position. The results are based on the camera specifications the user specifies within the script, and the placement and orientation of the camera. If needed, the script handles several cameras and outputs the combined results of all cameras.

This section details the creation of this script, as well as a detailed explanation of how each mode works.

A link to the most recent version of the software can be found at: Website link.

3.2.1 Object representation

The group has developed a script in order to calculate the field of view (FOV) of different cameras. Any 3D-object can be used as a reference. The script requires a file containing details about the 3D-object. More specifically, it needs coordinates in 3D-space of all corners of the object.



```
[[0,0,0],[10,0,0],[10,0,10],[0,0,10]], [[32.5,0,0],[47.5,0,0],[47.5,0,10],[32.5,0,10]],
[[0,120,0],[10,120,0],[10,120,10],[0,120,10]], [[32.5,120,0],[47.5,120,0],[47.5,120,10],
[[0,0,2.2],[0,14.5,2.2],[0,14.5,12.2],[0,0,12.2]], [[0,52.75,2.2],[0,67.25,2.2],[0,67.25
[[80,0,2.2],[80,14.5,2.2],[80,14.5,12.2],[80,0,12.2]], [[80,52.75,2.2],[80,67.25,2.2],[8
```

Figure 25: Text representing surfaces and corners of a euro pallet in 3D-space.

The algorithm first off imports a text file containing the corners and surfaces of a pallet, seen in figure 25. It generates a list where each element of the list contains four points in 3D-space. These four points make up a surface. An example of how these points represent a pallet can be seen in figure 26. The blue points are written as coordinates in 3D-space and the red area is the surface generated by them.

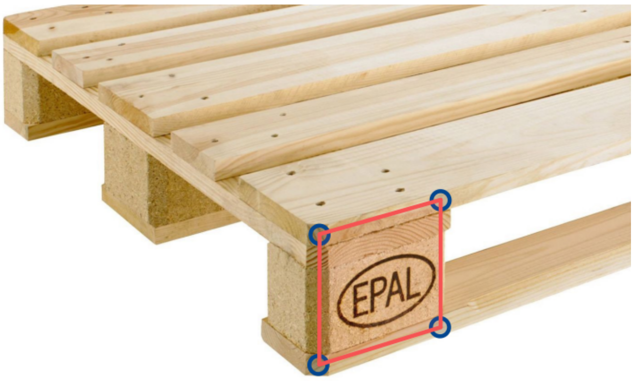


Figure 26: Method of representing pallet as a text file

Secondly, it uses the initial points to generate additional points along the sides of each surface. This is done by mathematically calculating which direction the side is pointing and then generating a specified amount of points along it, starting at one corner and ending at another. This is repeated for each side, and then for each surface. This is done in order to make the results more detailed.

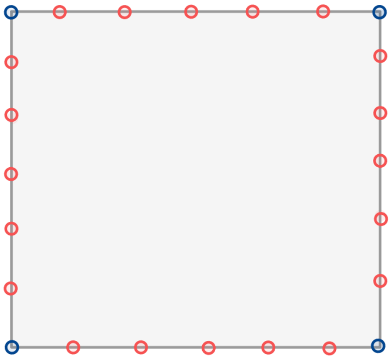
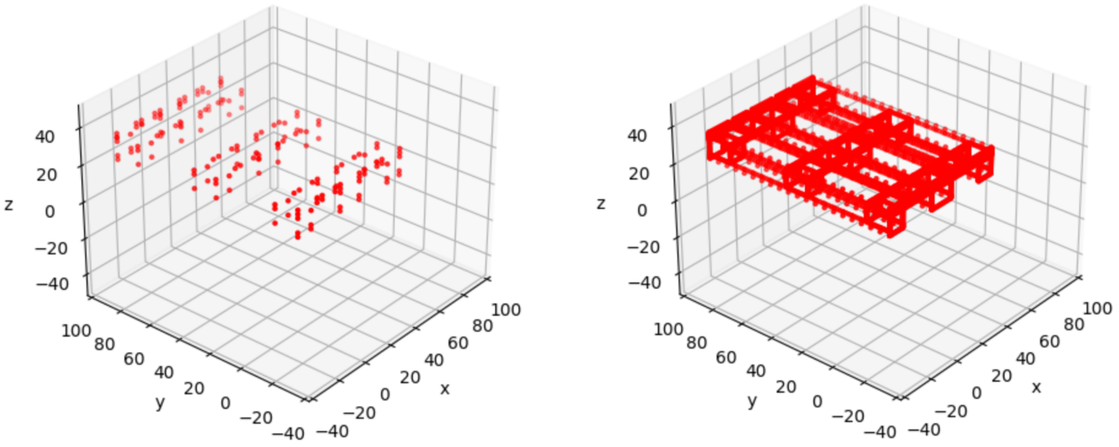


Figure 27: Additional points generated for a higher resolution

The generation of additional points is exemplified in figure 27. The points in blue are initial ones, retrieved from the text file. The red points mark the additional ones generated by the algorithm. This process gives the user a better understanding of which areas are seen by the camera and which are not. Before the additional points, the algorithm would not be able to tell how much of the sides could be seen. The increased resolution is shown in figure 28.



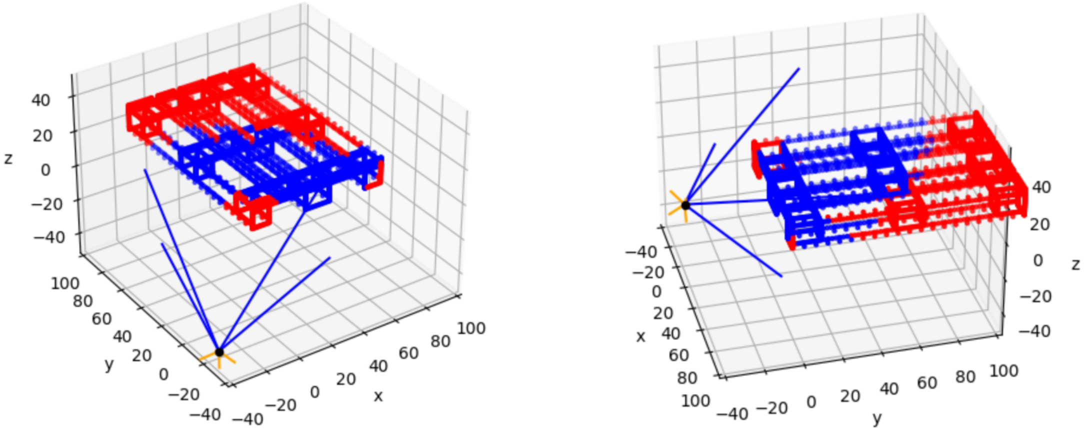
(a) Before adding points

(b) After adding points

Figure 28: Increasing resolution of 3D-model

3.2.2 Checking field of view

An example of this script in use can be seen in figure 29. Blue points are points inside the FOV, red points are not within the FOV. It is important to note that this mode does not take into account obstructed points and is only used to verify the viability of cameras and camera placements. If the results of the FOV test are promising the camera setup can be tested using another mode, where surfaces are used to detect line-of-sight obstructions of points.



(a) FOV, angle 1

(b) FOV, angle 2

Figure 29: Testing of FOV

This mode requires the camera placement, its FOV represented in horizontal and vertical angles, its focus point and it's range span. The camera placement is the origin of the FOV. The FOV angles determine the width and height of the FOV. The focus point is a point in 3D-space where the camera is pointed. Finally, the range is the cameras minimum and maximum viewing distance.

In order to represent the FOV in vectors and planes, the script uses methods and formulas described in section 2.2.4. A unit vector is constructed between the camera and the focus point, which is used to make two perpendicular vectors that define a new coordinate system, seen in figure 30.

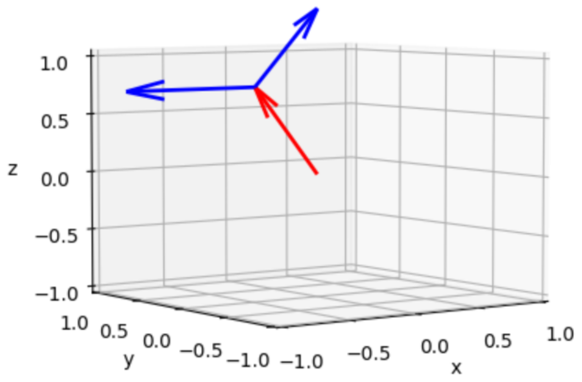


Figure 30: New coordinate system based on the blue vectors

The first of the two vectors is created using vector rotation. By using a rotation matrix that rotates the focus vector 90 degrees around the Z-axis, explained in section 2.2.4, seen in figure 31, a perpendicular vector to the focus vector is created. Due to the Z-component of the matrix being set to 0, the vector only spans along the XY-plane. After it has been rotated, it is scaled to be a unit vector. The resulting vector is placed at the end of the focus vector.

$$\begin{bmatrix} \mathrm{Cos}(90) & -\mathrm{Sin}(90) & 0 \\ \mathrm{Sin}(90) & \mathrm{Cos}(90) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figure 31: Rotation matrix used for first axis.

With two vectors defined, a third one can be created perpendicular to both of them using the cross product. This vector is the second axis of the new coordinate system. Seeing that the third vector is a cross product of two unit vectors, no additional scaling is necessary for it to also be a unit vector.

The script then uses the angles of the camera FOV to determine the direction of the FOV vectors. Using trigonometry, the angles are converted to vectors. Since all vectors are made to be unit vectors, the movement along the newly defined axes is the tangent of the FOV angle.

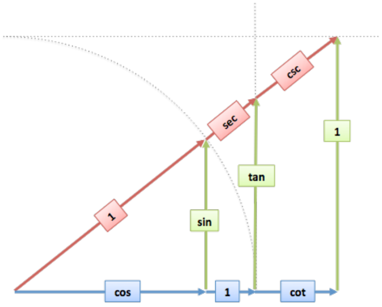
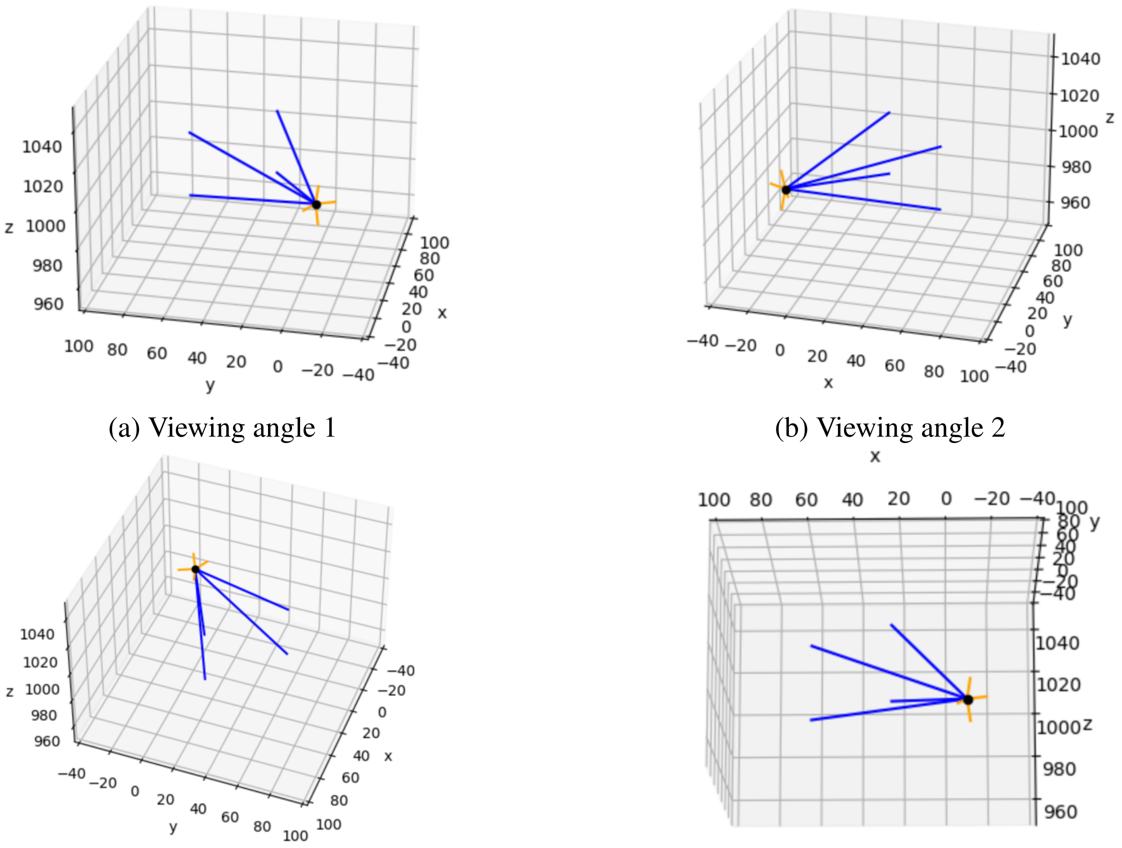


Figure 32: Method for converting angle information to vector information

The method for converting angle information to vector information is done for both axes. The movement along the axes are the same, but negative, for the mirrored sides. This means four vectors can be made by combining positive and negative movement along both axes. These four vectors make up the corners of the FOV.



(a) Viewing angle 1

(b) Viewing angle 2

(c) Viewing angle 3

(d) Viewing angle 4

Figure 33: Field of view

After the FOV vectors have been constructed, the sides of the FOV ”pyramid” are defined as planes. This is done by using two of the FOV vectors and the camera’s placement with the formula described in section 2.2.4. This is repeated four times. These planes define a four-sided pyramid in 3D-space, excluding the bottom, as seen in figure 34.

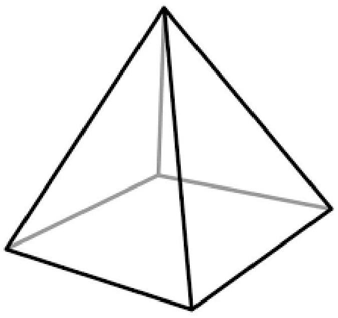


Figure 34: Shape of FOV pyramid

Next, the script constructs normal vectors out of these planes. This is done in an order which makes the normal vectors point outwards from the center. The normal vectors are shown in figure 35, in yellow.

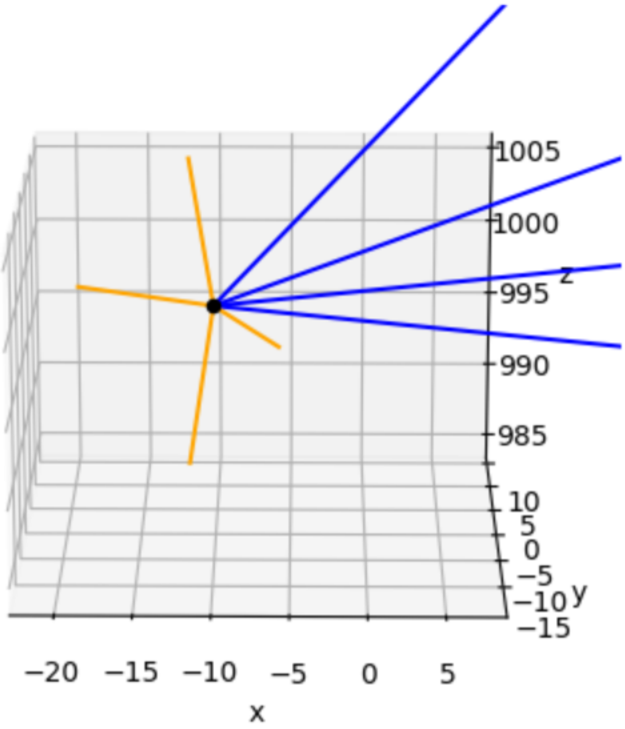
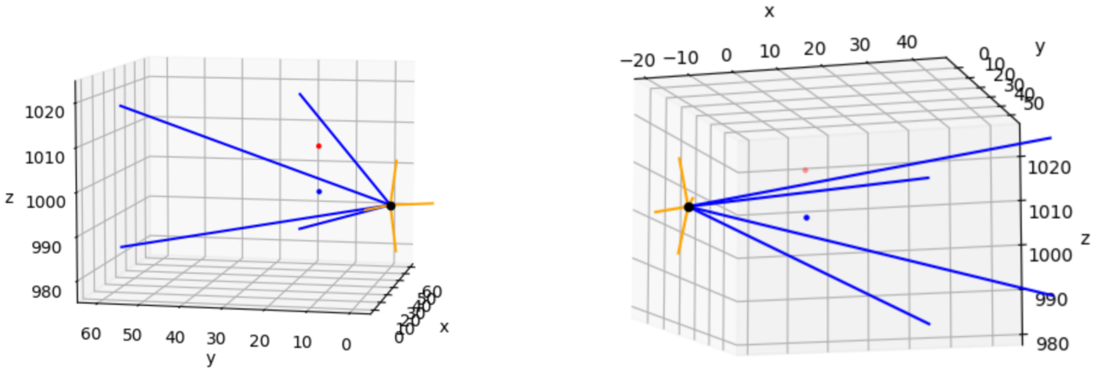


Figure 35: Normal vectors out of planes

Finally, all the points representing the pallet are run through a calculation of whether or not they are within the FOV. This is done by using the method described in section 2.2.4. By comparing each surface of the FOV and the amount of their corresponding normal vector needed to reach each point of the pallet, the script determines if they are within the FOV of the camera or not. Because of the normal vectors all pointing outward from the FOV pyramid, the movement should require a negative amount of all the normal vectors if it is within the FOV. The script then displays all points, where their color represents if they are seen or not, as seen in figure 36. Blue means they are seen, red means they are not seen.

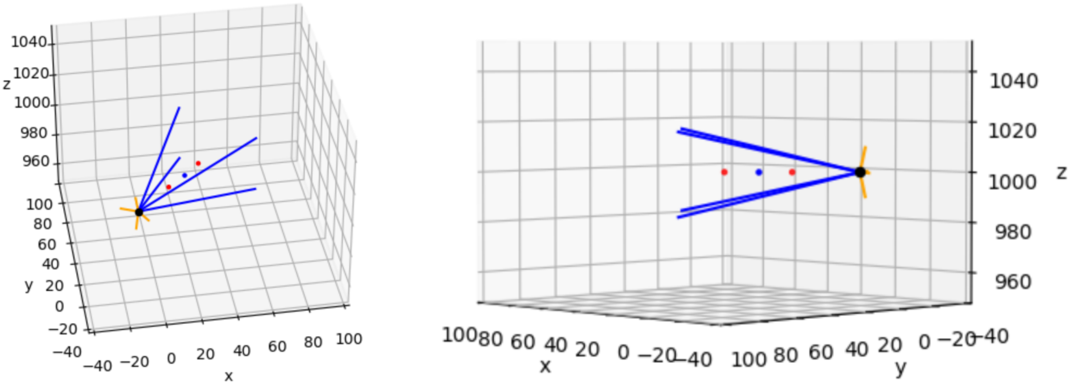


(a) Viewing angle 1

(b) Viewing angle 2

Figure 36: Field of view test

In addition to this, the camera can be set to have a minimum and maximum range. In order to verify that the points are also within these limits, the script uses the formula described in 2.2.4. The points' distance to the camera are compared to be between the lower and higher range. In figure 37 this is exemplified with three points that are all within the FOV, but only one is within the camera's operating range.



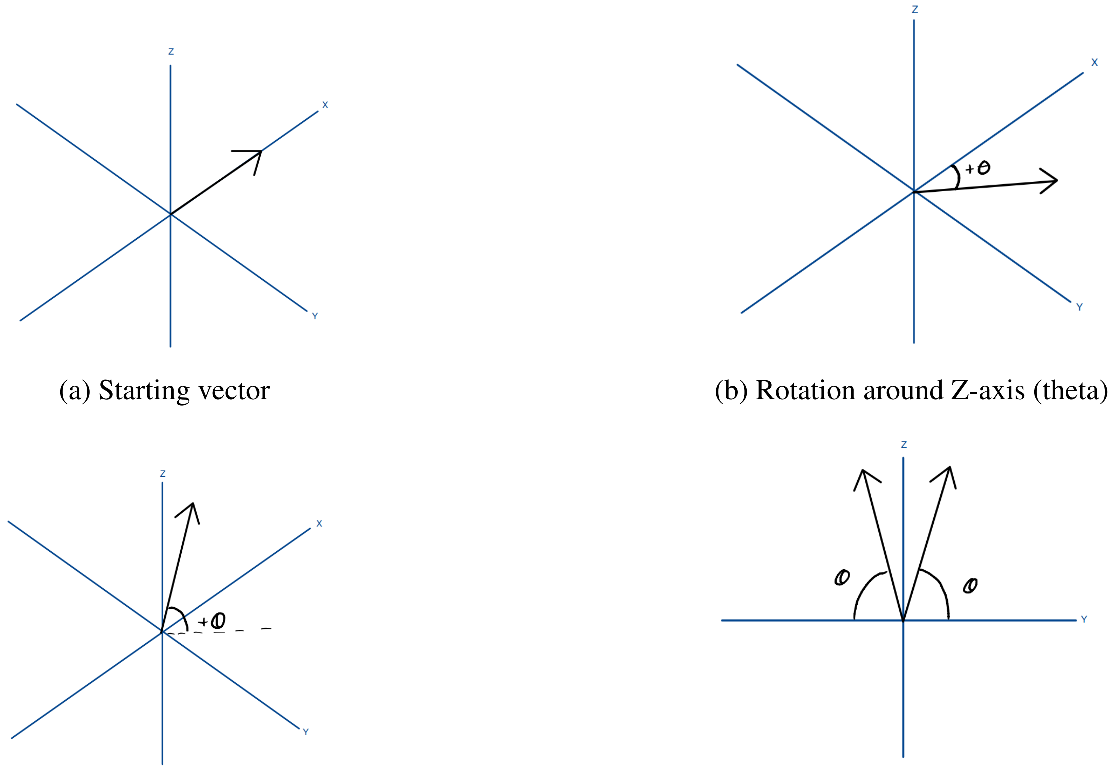
(a) Viewing angle 1

(b) Viewing angle 2

Figure 37: Field of view range

Software that did not work

Before the idea of creating their own coordinate system, the group also tried creating the FOV pyramid using spherical coordinates. This did not work because of how vectors converge when applying vertical rotation using these methods. The problem is visualized in figure 38. The rotation around the Z-axis is correct, but all vectors converge to point directly upwards when applying rotation around the XY-plane.



(a) Starting vector

(b) Rotation around Z-axis (theta)

(c) Rotation upwards (phi)

(d) Vectors converging at 90° rotation along the XY-plane

Figure 38: Rotations using spherical coordinates

3.2.3 Checking seen points

In order to verify the viability of the possible solutions, the group uses the same script as to check the FOV, but in a different mode. The script uses the same text file containing all corners of a pallet in 3D-space. This file is run through an algorithm to check which points are seen from a certain point, and which are obstructed by various surfaces. In addition to this, the algorithm takes into account the specified FOV of the camera, and it's operating range. The algorithm can be configured to include as many cameras as needed. The resulting plots are their collectively seen and not seen points.

The script first off increases the resolution of the pallet using the method described in section 3.2.1. Then, it checks to see which points are within the first camera's FOV, same as in section 3.2.2. Only the points that are within the camera's FOV are run through the rest of the algorithm. The points outside of the FOV are stored in a cache. These are either checked by the next camera or marked as not seen if the current camera is the last.

While adding more points, the script also defines planar equations for all the surfaces formed by the initial points. The planar equations, described in section 2.2.4, are used later in order to check whether or not the line-of-sight to a point is obstructed.

The points within the FOV are assessed using a method similar to ray casting, described in section 2.2.2. The ray is a vector between the camera origin and each individual point. Based on the settings of the script, this ray is discretized a certain amount of times. Discretization is explained in section 2.2.4. By comparing all the discrete values of the ray to all the surfaces of the pallet, the script can determine when the ray approaches and crosses a surface. The ray crosses a plane when the planar equation equals zero.

Given that it is not a continuous ray, some margin must be accounted for. An illustration of why can be seen in figure 39. The discrete values of the ray might not land exactly in the plane, but rather have two values close on either side. In figure 39 the point marked as a red cross would have been the crossing point, if it were continuous.

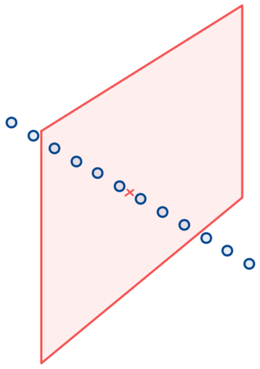


Figure 39: Discrete ray through surface

Should the result of the planar equation equal zero ± the specified margin, the script checks to see whether or not the ray is within the four corners of the surface. The reason for this is because of how a planar equation is not confined. It is defined without limits from negative infinity and to positive infinity along all axes. The planar equation using the two rays' points shown in figure 40 will both equal zero where the red crosses are, even though one of them does not go through the surface.

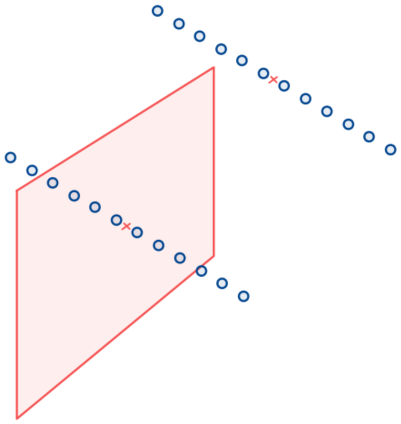


Figure 40: Two discrete rays crossing a plane

Due to the same reason as before, there also needs to be some margin accounted for in this step. When checking if the ray is within the confined surface, a margin is added to the surfaces' position. This creates a box in 3D-space, shown in figure 41. If the ray is within this box, the current point is labeled as not seen by this camera. In other words; if the ray crosses a surface on it's way to a point, then that point is obstructed. Then, that point is sent to the cache for the next camera to check it.

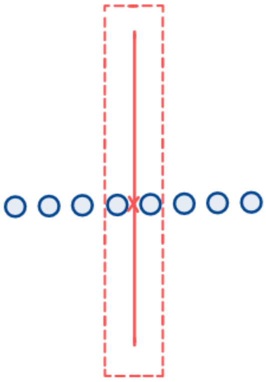
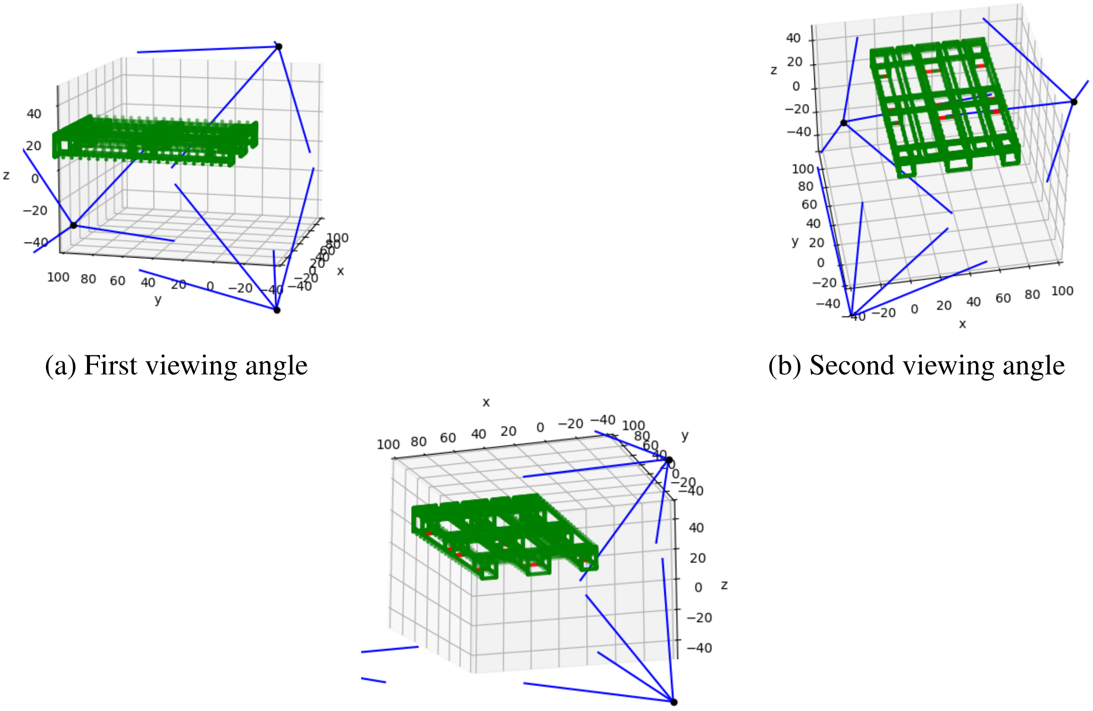


Figure 41: Box created in order to capture discrete values not exactly in the plane



(a) First viewing angle

(b) Second viewing angle

(c) Third viewing angle

Figure 42: 3D-plots showing results of algorithm. Cameras are shown as black dots.

After all cameras are finished, the points are shown in a 3D-plot. Their color tells the user if they are seen or not. Green is seen, and red is not seen. An example of this can be seen in figure 42. In that example the script used three cameras, one above and two below to the side.

Asynchronous programming

In order to optimize the script, asynchronous programming was used, described in section 2.2.3. This a more resource-intensive method, but it is faster. Instead of running the algorithm one camera at a time, all cameras are run simultaneously. This is configurable within the script. If the user for some reason does not want to tie up the computer's resources, the algorithm can be run synchronously at the expense of the total time used.

Using this method, the cameras do not initially know which points are seen by the others. Therefore, a queue had to be implemented. The algorithms for each camera pull out the element, or point, at the front of the queue and checks whether or not it can see it. The next camera pulls out the next element. If the point can be seen, this point is added to a collection of seen points. If it cannot be seen, it is added to the back of the queue for it to be checked by a different camera. All cameras keep track of which points it cannot see, so that when the points is at the front of the queue once more, it does not bother to check it again. When all points remaining in the queue are checked by each camera, the algorithm finishes and the remaining points are labeled as not seen.

The workflow of the algorithm using asynchronous programming can be seen in figure 43.

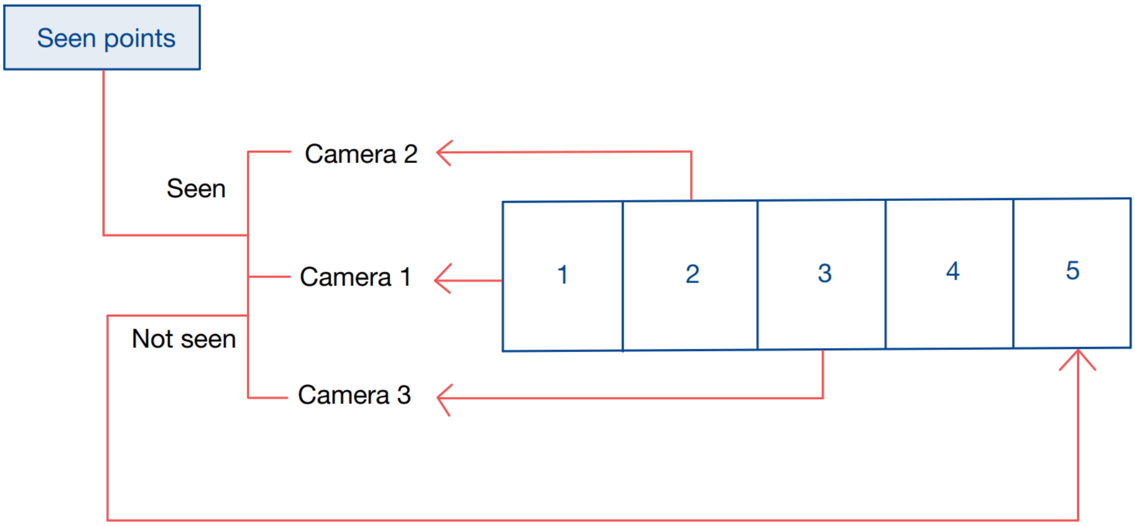


Figure 43: Asynchronous workflow

3.3 Possible designs

This section contains information about how the group used Onshape in order to model possible designs. The first chapter details the models the group received from Solwr and how these models uncovered important limitations.

In addition to this, it details the designs the group has come up with. Some of the designs are grouped together with similar ones. Each design is numerated by the chronological order in which they were created. On every design, the cameras are illustrated with a black box. All 3D-models are made using Onshape, an online 3D CAD software.

This section is meant to convey the different designs the group has looked at and detail their positive and negative aspects.

3.3.1 Using CAD as inspiration and visualization

Since the majority of the hardware development for Sort™ was conducted using CAD, Solwr now has a comprehensive and detailed 3D CAD of the current version of Sort™. The group was granted access to Sort™ in 3D, which was utilized to visualize ideas throughout the brainstorming session.

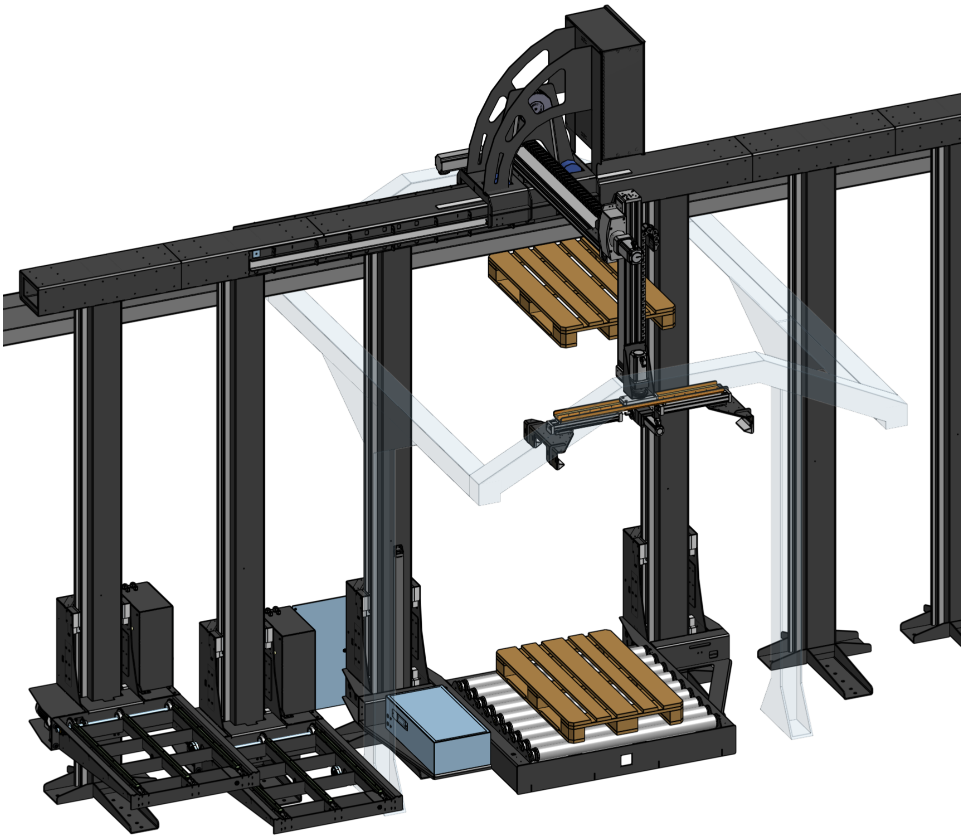
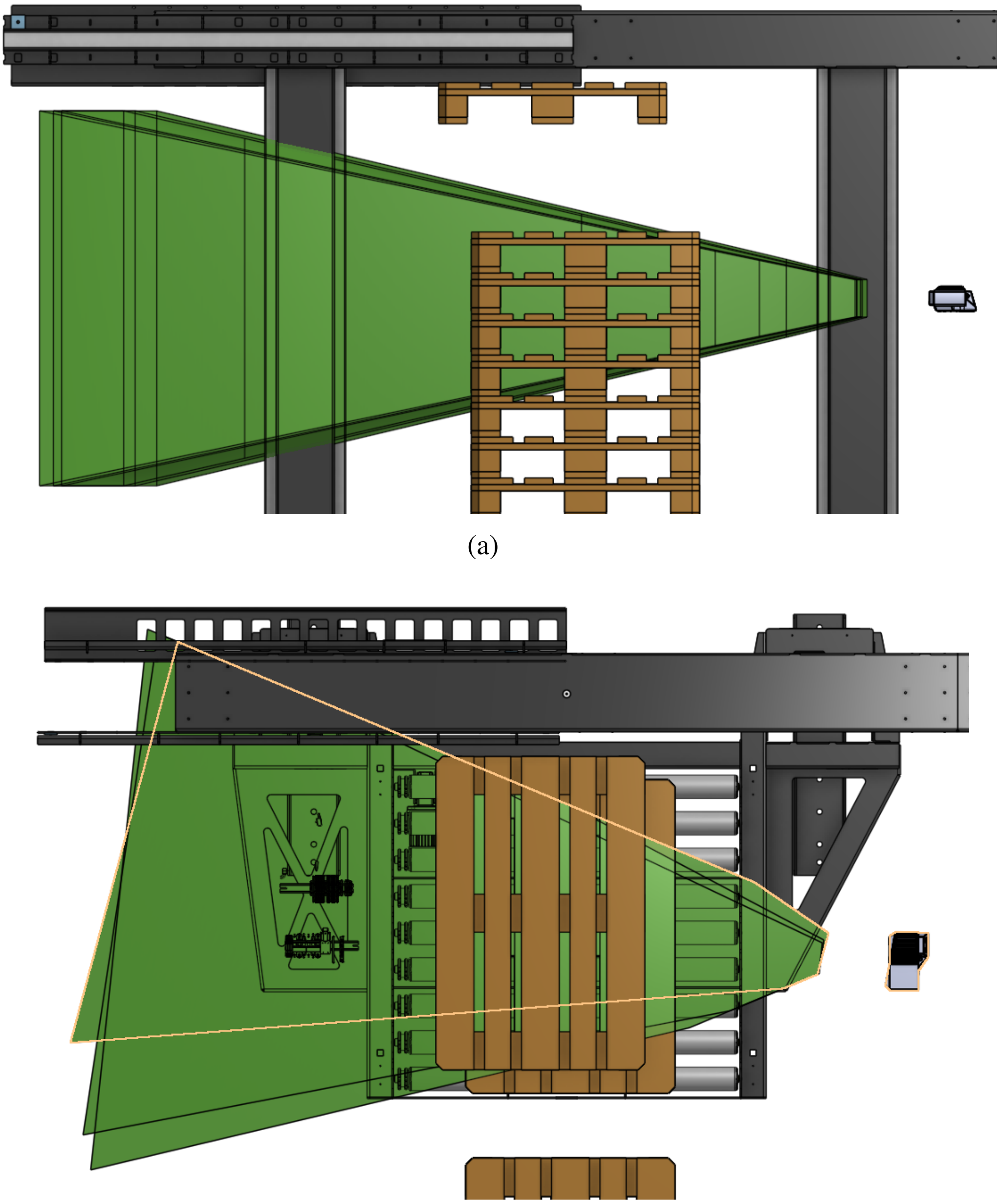


Figure 44: 3D CAD of sort

3.3.2 Imported Zivid FOV


(a)

(b)

Figure 45: Difference between small, medium and large FOV, with the small FOV outlined in yellow.

One benefit was the possibility to import a 3D CAD and FOV from Zivid One$^+$ cameras. Zivid provides FOV for all three cameras in their line up: Small, medium and large. However, the CAD representing Zivid's FOV does project the correct FOV, but does not factor in each cameras operating distance. In figure 45 are all three models located in parallel, to illustrate the difference in angle and size. See appendix [C] for each cameras operating distance.

3.3.3 Obstacle handling

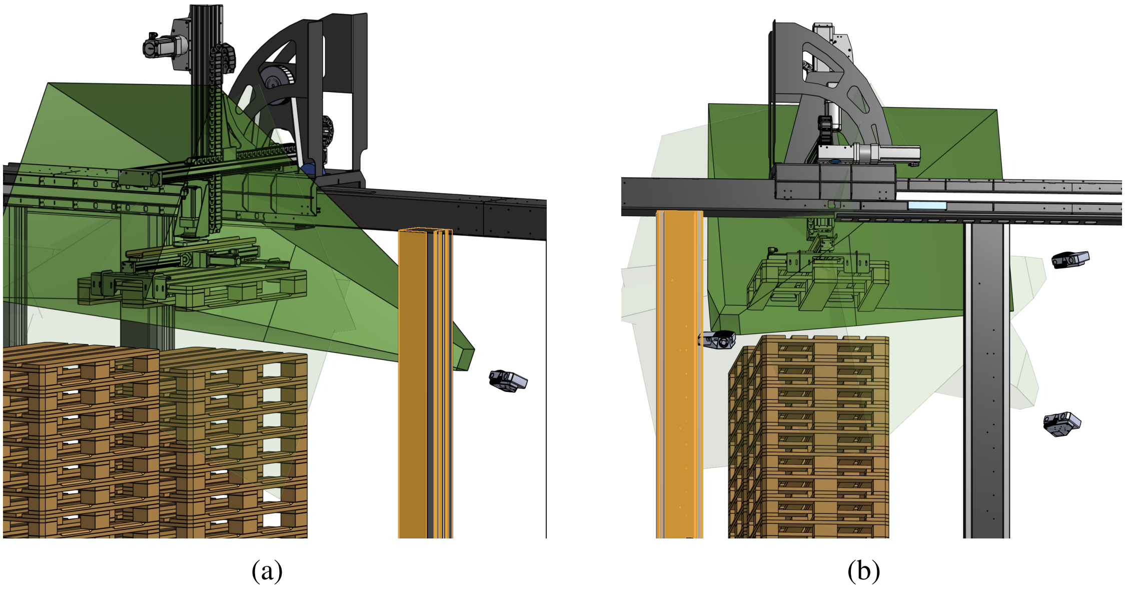


Figure 46: Displaying the limitation due to the pole, marked in yellow

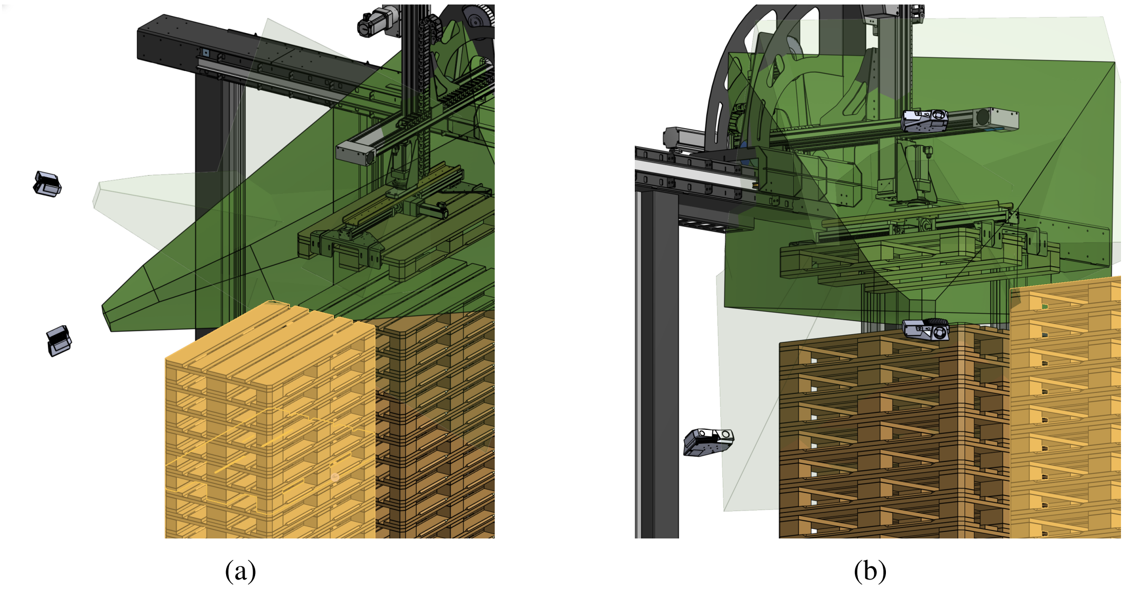


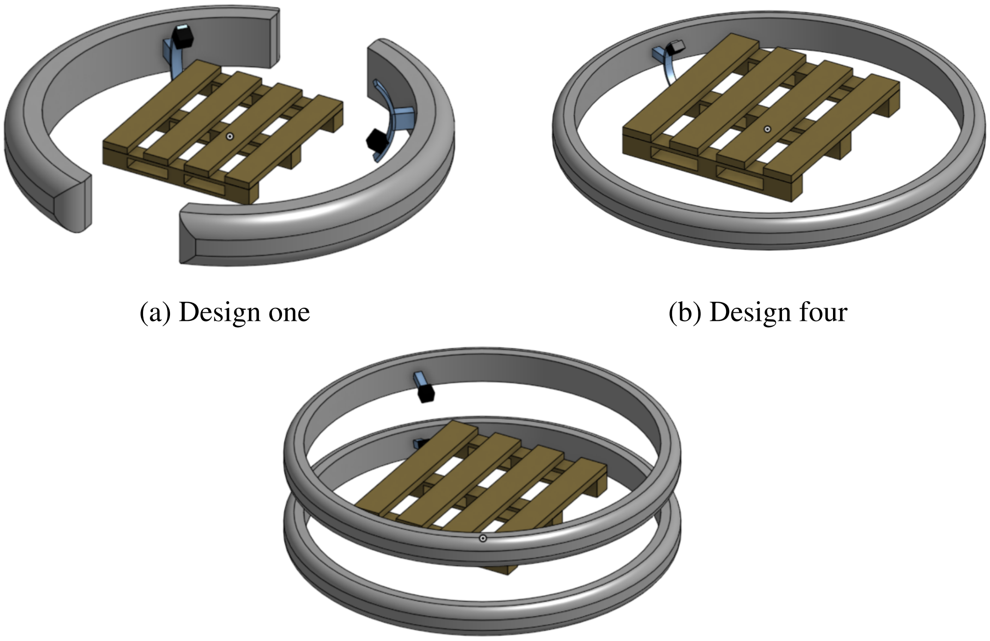
Figure 47: Displaying the limitation due to the following stack, marked in yellow

The 3D CAD was not only useful for the brainstorm process, but also to be alert of possible obstacles to be aware of. One obstacle was the pole illustrated in figure 46, where the pole highlighted in yellow had to be considered when designing the solution. Another obstacle to be aware of was the following pallet stack illustrated in figure 47, where the pallet stack highlighted in yellow could block some of the camera's FOV. However, the distance between stacks, is highly customizable if being a limiting factor.

3.3.4 Design one, four & five

Characteristics:

- one & two cameras
- movement of cameras, horizontal
- movement of cameras, vertical
- circular axis



(a) Design one

(b) Design four

(c) Design five

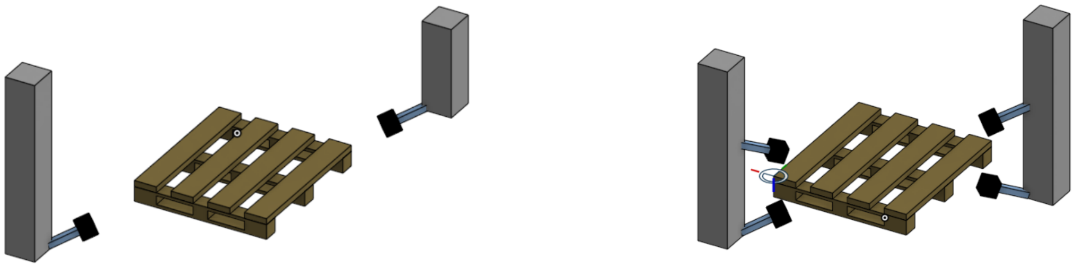
Figure 48: Illustration of design 1, 4 and 5

These designs only use two cameras, but they are unnecessary complex because of the circular axis. If the cameras only take photos at the end of every bar, it would make just as much sense to use a linear axis. The group also learned that a circular axis is not a standard, in comparison with a linear axis, which is widely used in the industry. Previous experiences from engineers at Solwr informed that with custom parts require longer delivery times, and additional cost. Due to these reasons, design one, four and five was not pursued further.

3.3.5 Design two & three

Characteristics:

- two & four cameras
- stationary cameras
- pillars mounted to frame



(a) Design two

(b) Design three

Figure 49: Illustration of designs 2 and 3

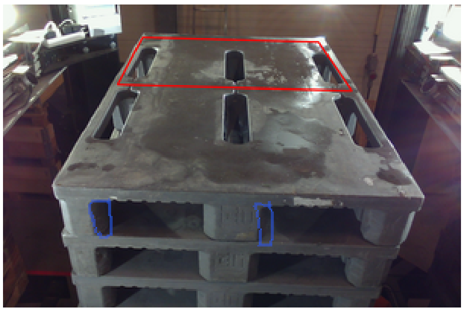


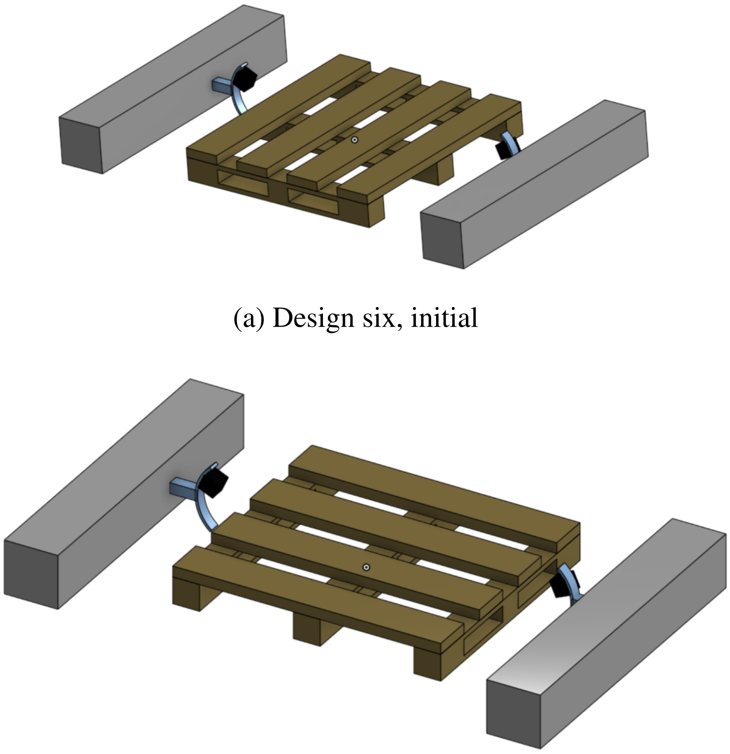
Figure 50: Dead zone using two static cameras, marked in blue.

Two cameras, as seen in figure 49a are not sufficient when both cameras and the pallet are stationary. There would be blind spots in between the legs of the pallet, marked in blue in figure 50. Four cameras, seen in figure 49b, would be too expensive. A possible way to fix the issue with design two, would be to add a third camera. The third camera would be used to map the top of the pallet. Due to these reasons, design two, and three were not pursued further.

3.3.6 Design six

Characteristics:

- two cameras
- moving cameras
- pillars mounted to frame
- rotating pallet



(a) Design six, initial

(b) Design six, turned

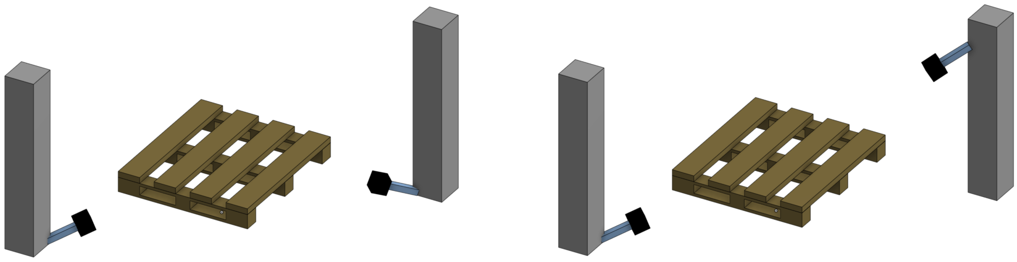
Figure 51: Illustration of design 6

Using this design, the total cost of the cameras would be low, since its only using two cameras. However, the group evaluated it to be hard to cover the inner center of the pallet, and therefor not a optimal solution since there might be damages in that area. It also requires movement along non-linear axes when moving vertically, making it unnecessarily complex. In addition to this, the rotation of the pallet would add significant time to the total time needed per pallet. Due to these reasons, design six was not pursued further.

3.3.7 Design seven

Characteristics:

- two cameras
- moving cameras
- pillars mounted to frame



(a) Design seven, initial (b) Design seven, transformed

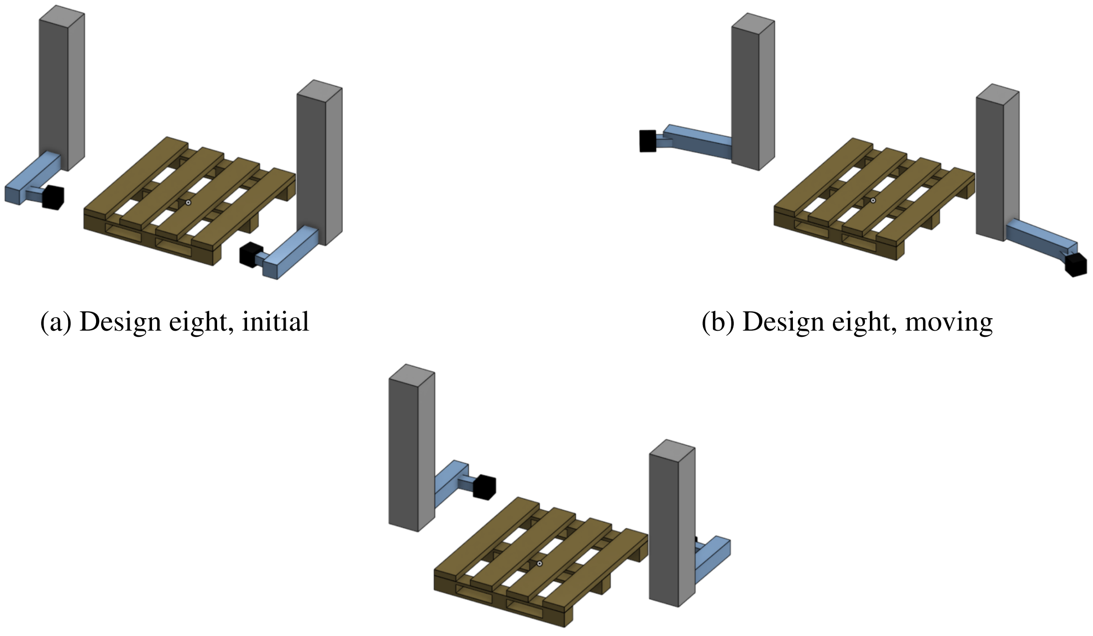
Figure 52: Illustration of design seven

This design improves upon design two, three and six. One of the cameras are mounted on a linear axis, making it function as a hybrid between design two and four, while including the movement from design six. This looks to be a cost-effective solution. The design will with high probability be a cheaper design utilizing an axis, a motor and a control system, than to purchase a third camera. Since the group evaluated this as a possible viable solution, this was one of the pursued designs.

3.3.8 Design eight

Characteristics:

- two cameras
- moving cameras
- moving camera mounts
- pillars mounted to frame



(a) Design eight, initial

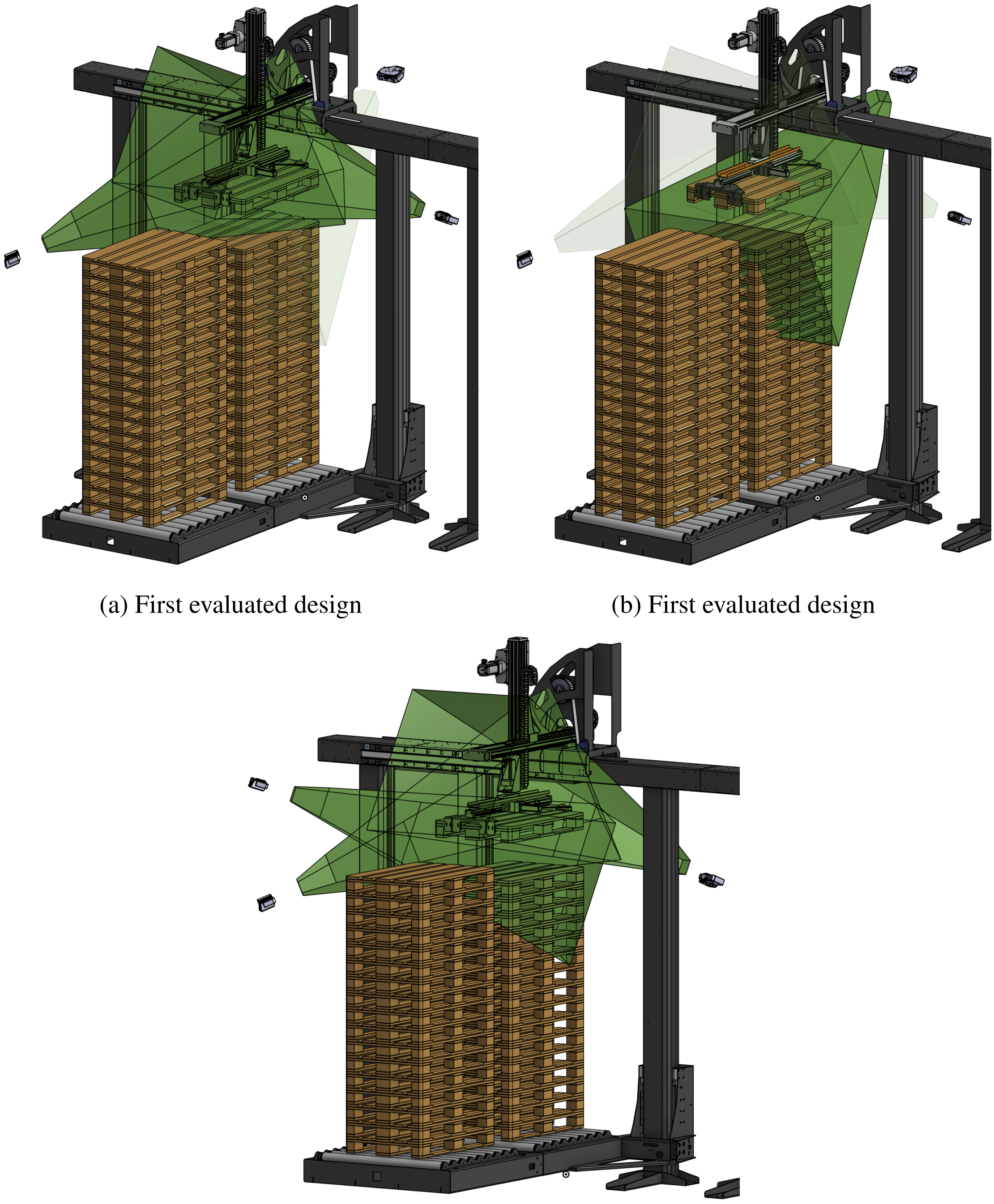
(b) Design eight, moving

(c) Design eight, moved

Figure 53: Illustration of design eight

Design eight involves moving both cameras and the mount they are attached to. It would be quite complex and heavily dependent upon accurate and fast movement. Design eight was therefore not pursued further.

3.3.9 Chosen design



(a) First evaluated design

(b) First evaluated design

(c) Final design using linear axis

Figure 54: Presenting two viable solutions in CAD through Onshape

After several concepts during the brainstorm session, the group preferred the solution in figure 54a and 54b using three cameras. Placing two cameras diagonally to each other, and the third camera as a top camera. The top camera could either be in front or behind the camera portal on Sort™. The top camera's sole job was to cover the pallet's top surface, while the two diagonal cameras covered the sides and bottom. However, in order to keep the number of cameras as low as feasible, the group decided that the top camera should be located directly above one of the diagonal cameras, rather than in front or behind Sort™.

This solution is illustrated in figure 54c, and allows the group to use the same camera in both locations by implementing a linear axis as illustrated in figure 52.

After considering all the designs, both Solwr and the bachelor group decided that design seven, or a variant thereof, would be the most promising one. It keeps the total cost relatively low compared to the other ones, while not adding a lot of complexity to the system.

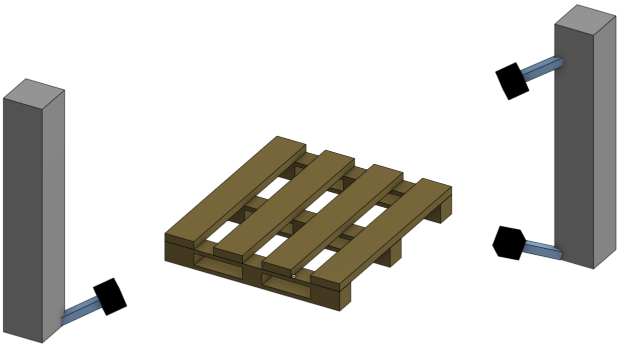


Figure 55: Variant of design seven

If the linear axis and the servo implementation for adjusting height and angle is too complex, a possible variant would be to use a design with three cameras, seen in figure 55. This would reduce the complexity whilst maintaining the area covered by design seven.

3.4 Possible cameras

The process for finding the optimal camera for this project solution had several steps:

- Evaluation of each camera's specifications
- Evaluation of FOVs
- Evaluation of delivery times & prices
- Consider Norwegian tax laws, how the import from other countries would effect the total price.

Camera specification evaluation
Research was done, documenting the different manufacturers' datasheets for each individual camera. The gathered results are displayed in table 4.

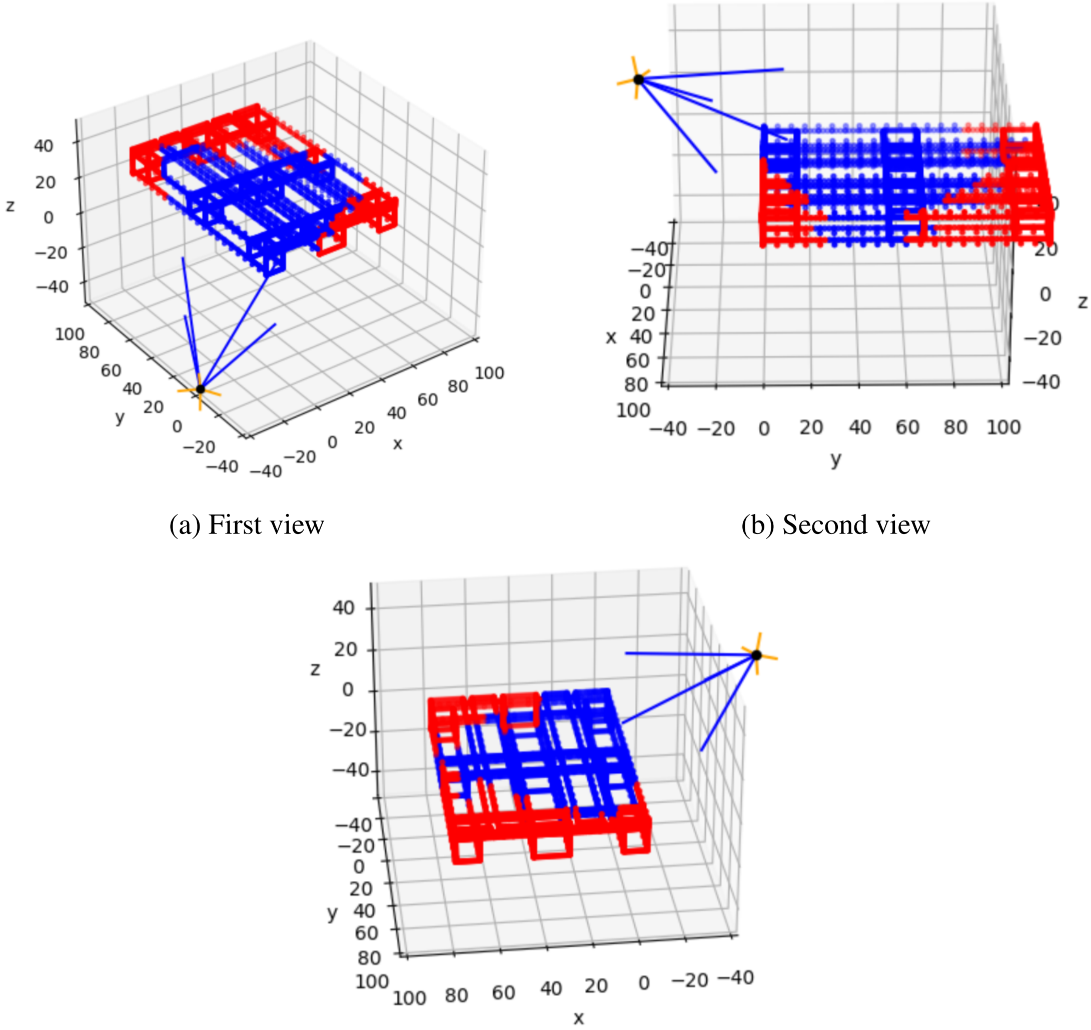
Specifications for possible new cameras					
Name	Resolution	FOV size at (mm)	Depth accuracy	Range (mm)	Link
Intel - RealSense L515	1024×768	70°horizontal 55°vertical (±3°)	$\tilde{5}$ mm to $\tilde{1}$4 mm thru 9 m^2	250 to 9000	1[B]
Zivid - One$^+$ Medium	1920x1200	33°horizontal 25°vertical	<100μm (1800) euclidean distance	500 to 2000	[1]
Zivid - One$^+$ Large	1920x1200	39°horizontal 25°vertical	<350μm (1800) euclidean distance	1200 to 3000	1[C]
Zivid - Two	1944x1200	50°horizontal 36°vertical	55μm(700) euclidean distance	300 to 1500	1
Nerian - Scarlet 3D Depth Camera	1921x2048	38.5°horizontal 32.8°vertical	2.7mm at 3m	1700 to 5000	1, 2
Mech-mind - Mech-Eye laser L	2048x1536	50°horizontal 45°vertical	1mm at 3m	1500 to 3000	1
Cognex - 3D-A5120 Extended working volume	n/a	53°horizontal 39°vertical	n/a	800 to 2800	1

Table 4: Specifications for possible new cameras

FOV evaluation

The FOV evaluation is carried out with the use of the group's own software, which is detailed in chapter 3.2.2. The software verifies that one Euro pallet fits within the field of view of the cameras, which is essential to achieve sufficient reductions in the number of cameras used. The pallet is made up of blue points, which turn red if one of them is outside the FOV. Three photos of the same camera are given from different perspectives for each camera to demonstrate that there are no red dots.

3.4.1 Zivid - One$^+$ Medium



(a) First view

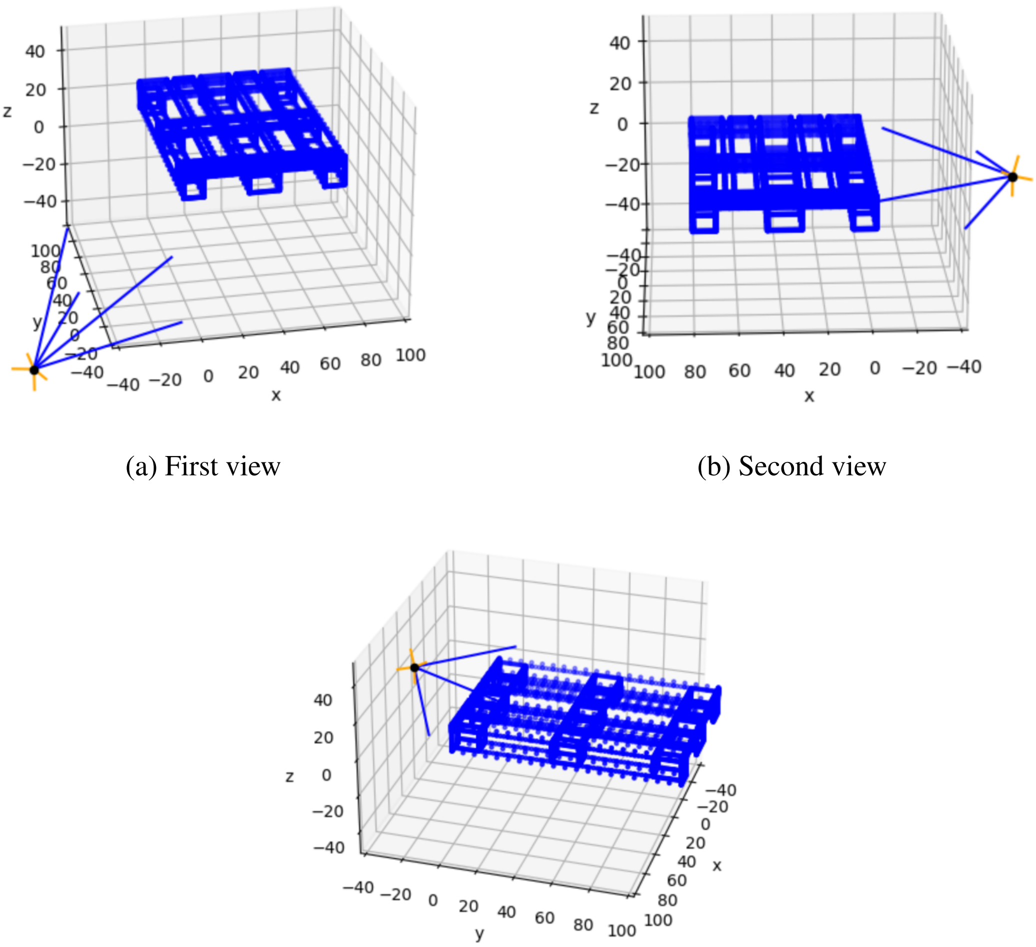
(b) Second view

(c) Third view

Figure 56: Checks if a EURO pallet fits within the Zivid One$^+$ Medium FOV

Due to the low FOV combined with the short operating range, Zivid One$^+$ Medium is not a viable camera. It would require too many cameras in order to cover every corner of the pallet. In addition to this, the camera uses USB for data transfer and not Ethernet, which is preferred.

3.4.2 Zivid - One$^+$ Large



(a) First view

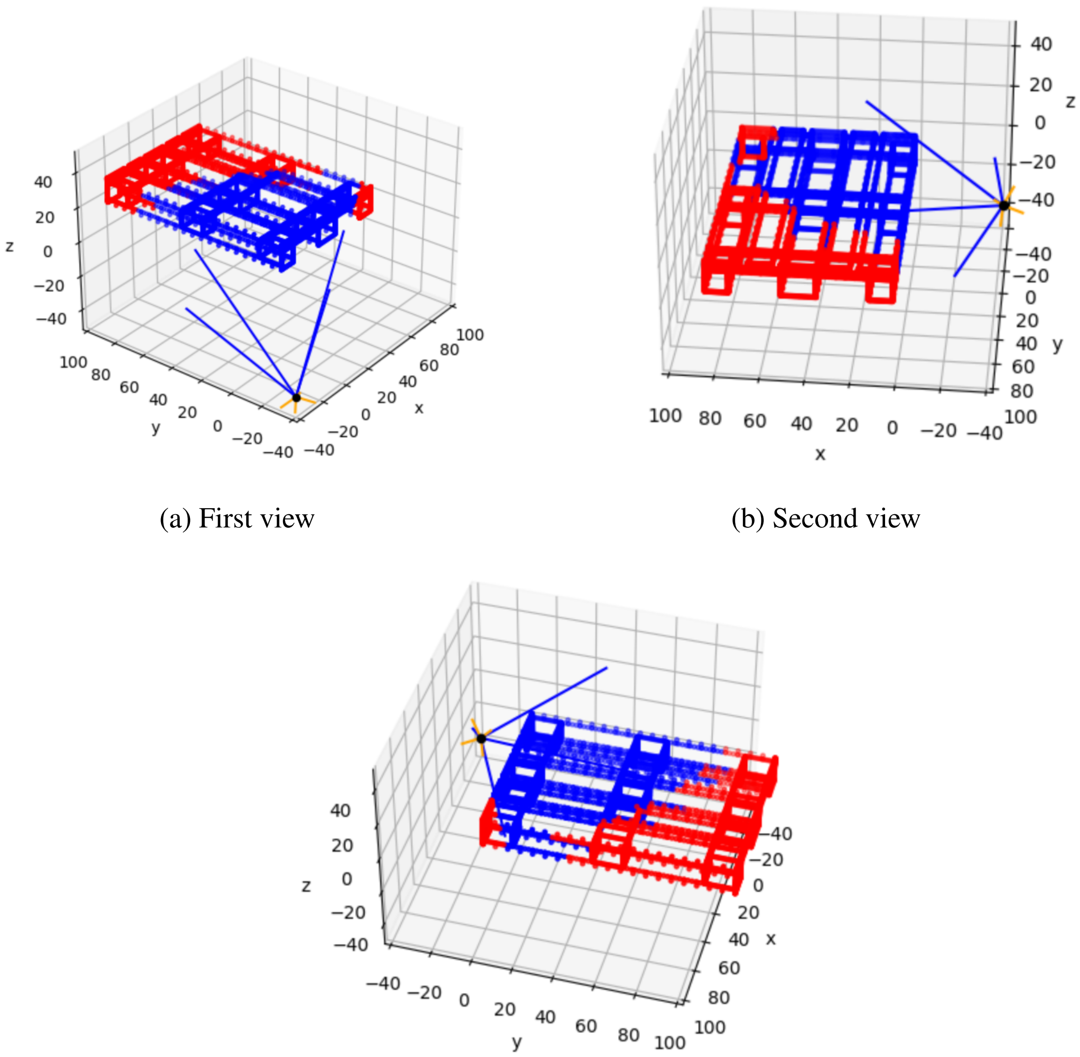
(b) Second view

(c) Third view

Figure 57: Checks if a EURO pallet fits within the Zivid One$^+$ Large FOV

Zivid One$^+$ Large looks to be a viable option due to its accuracy and FOV. This camera has been considered by Solwr previously. Zivid is an Norwegian vendor, which is beneficial in regards to delivery time and price. One downside with Zivid is the USB communication. However, Zivid provides optical USB cables with 25M length for 420 Euros [25]. Using an optical USB cable would remove the need to have Raspberry Pis mounted on the camera portal.

3.4.3 Zivid - Two



(a) First view

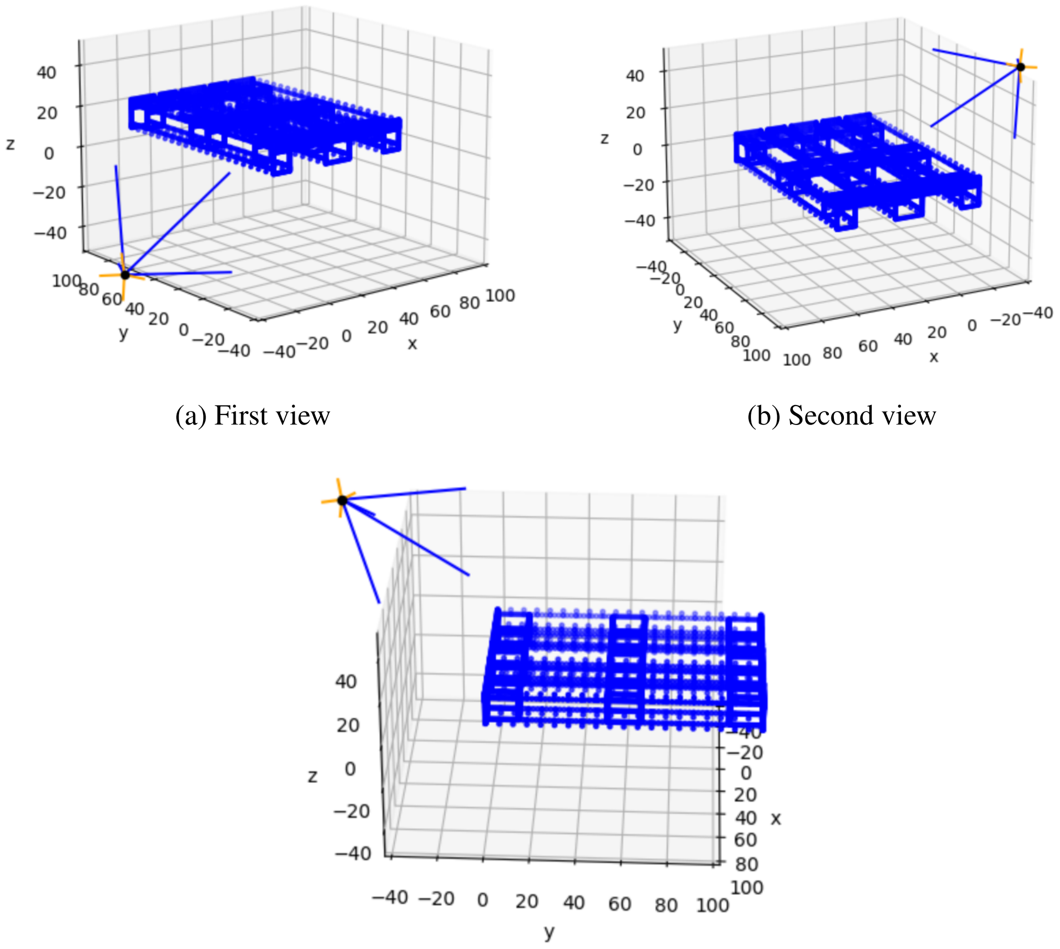
(b) Second view

(c) Third view

Figure 58: Checks if a EURO pallet fits within the Zivid Two FOV

The Zivid Two camera has high accuracy and Ethernet communication, but due to a short FOV this camera is not a viable option for Sort™. It would require too many cameras in order to cover the entire pallet.

3.4.4 Nerian - Scarlet 3D depth camera



(a) First view

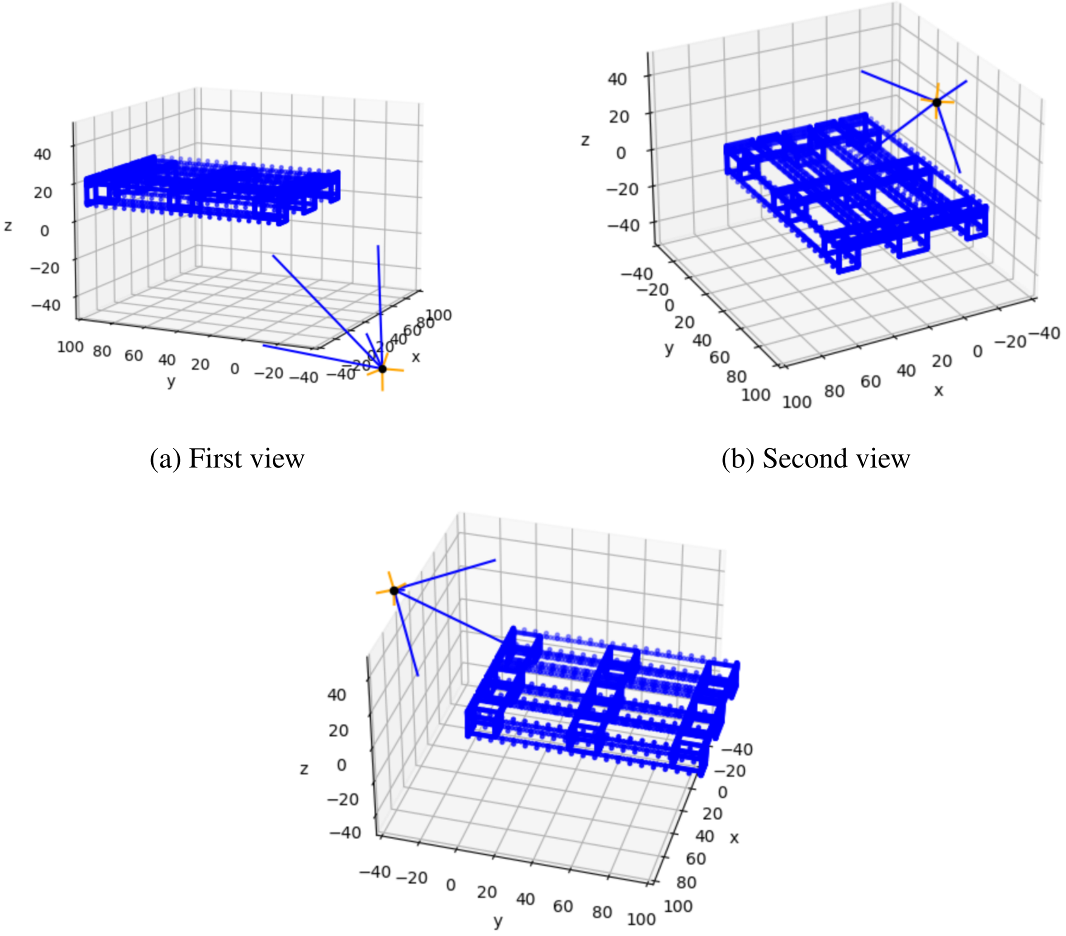
(b) Second view

(c) Third view

Figure 59: Checks if a EURO pallet fits within the Nerian, Scarlet 3D depth camera FOV

The Nerian Scarlet 3D camera is promising due to its high accuracy, FOV and range. In addition to this, it uses Ethernet communication.

3.4.5 Mech-mind - Mech-Eye laser L



(a) First view

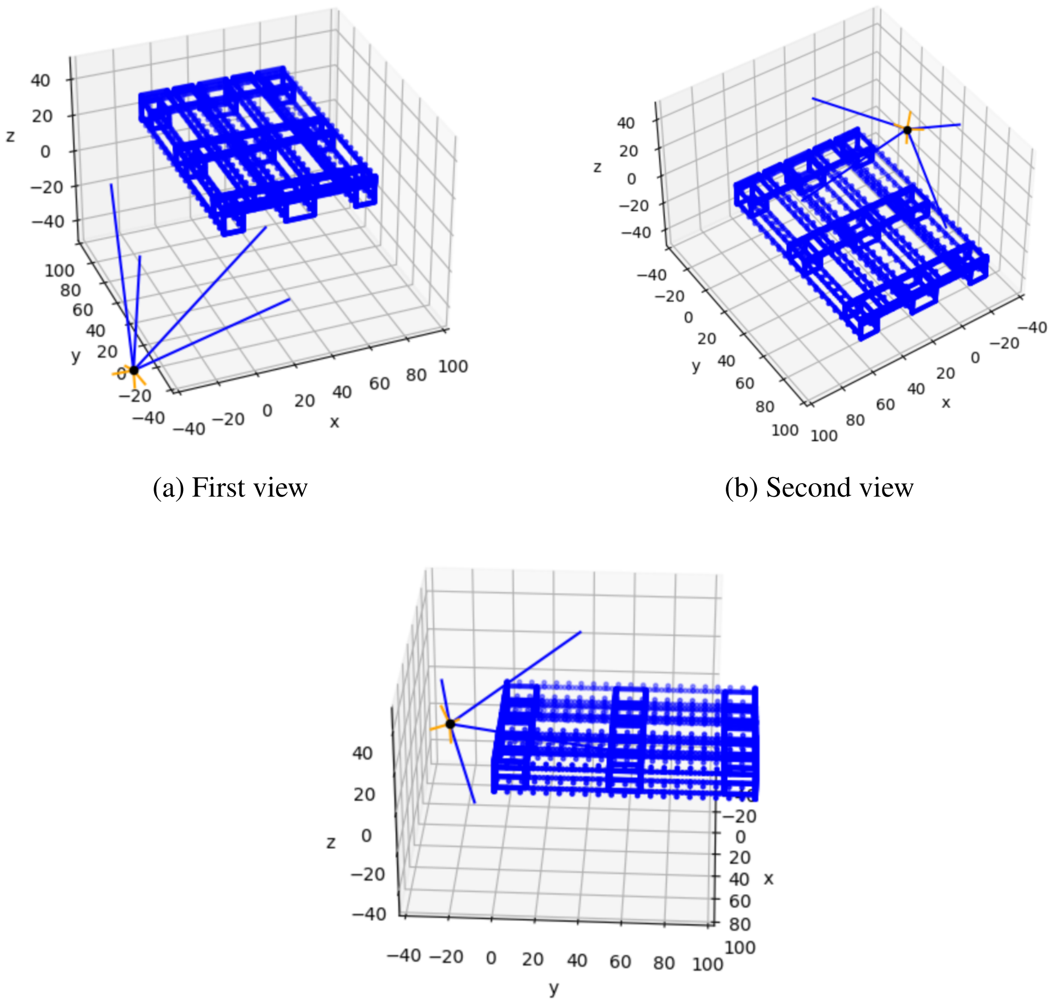
(b) Second view

(c) Third view

Figure 60: Checks if a EURO pallet fits within the Mech-mind, Mech-eye Laser L FOV

The Mech-Eye camera range from Mech-Mind has two models that look promising for this application. The Mech-Eye laser L and the Mech-Eye Deep both offer a large FoV suitable for pallet detection. The Laser L looks like the better choice due to it's quicker capture time and better accuracy. The cameras also offer Ethernet communication, Solwr has expressed a desire to move away from the current USB solution.

3.4.6 Cognex - 3D-A5120, extended working volume



(a) First view

(b) Second view

(c) Third view

Figure 61: Checks if a EURO pallet fits within the Cognex extended FOV

The Cognex camera with extended working volume has more then 1.5 million 3D data points detected, due to its patented 3D LightBurst technology. The camera also uses 10Gb Ethernet for communication. The software used is VisionPro which is the industry-leading PC-based vision software for the Microsoft® Visual Studio® .NET programming environment [26].

Price and delivery time, according to each manufacturer
Each company was contacted per mail, and all companies with an exception of Cognex responded. Each sales agent operated with different currencies, which the group decided to not alter.

Pricing & delivery time				
Name	Model	Price FOB	Delivery time	Located
Zivid	One$^+$ Medium	10 500 EURO	4 weeks	NORWAY
Zivid	One$^+$ Large	10 500 EURO	4 weeks	NORWAY
Mech-mind - Mech-Eye	Laser L	8 500 USD + tax	2 weeks	EUROPE
Mech-mind - Mech-Eye	Deep	8 000 USD + tax	2 weeks	EUROPE
Cognex - 3D-A5120	Extended Working Volume	n/a	n/a	EUROPE
Nerian	Scarlet	11 000 EURO	7 months	SWEDEN

Table 5: Possible cameras' pricing and delivery times

Norwegian VAT

By buying components locally within Norwegian borders, the VAT will be paid in full. However, the VAT will then be deducted at a later time [27]. This results in a 20% reduction in the total price buying from the Norwegian supplier Zivid. See the calculation in equation 20.

$$Price\ \ included\ \ VAT = 10\ \ 500\ \ Euro \tag{18}$$

$$Total\ \ VAT = 25\% \tag{19}$$

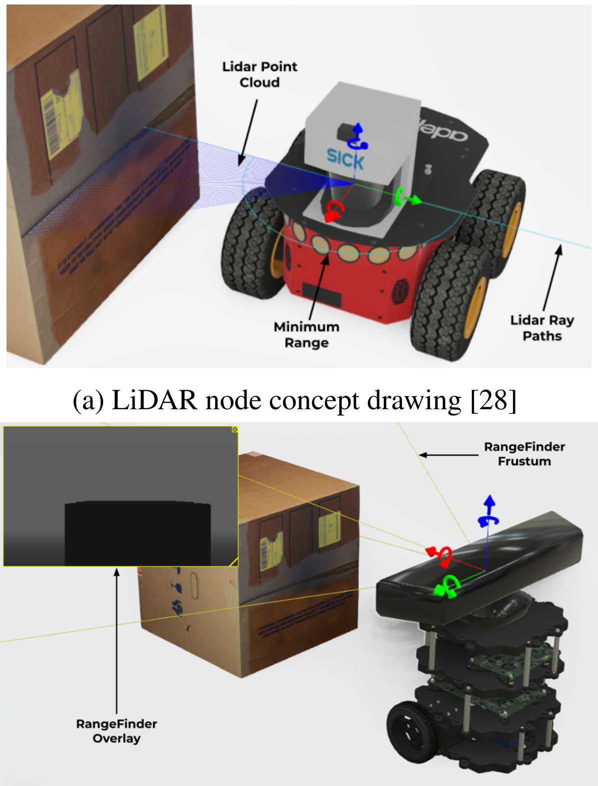
$$Price\ \ excluded\ \ VAT = \frac{10\ \ 500\ \ Euro}{125\%} = 8\ \ 400\ \ Euro \tag{20}$$

3.4.7 Chosen camera

After comparing the cameras' specifications, price and delivery time, the group has concluded that the most suitable camera is the Zivid One$^+$ Large. The company is Norwegian which gives Solwr the advantage of deducting the VAT of the camera pricing, due to the camera not needing to be imported. This makes the Zivid the cheapest camera relative to the resolution it has, by a good margin. Therefore, the group is confident that Solwr should order these cameras for testing, since real tests are a necessary to be certain that camera meets all the requirement when in the given environment.

3.5 The Webots simulator

Webots' software is used for creating advanced simulations. The group's goal was to create a simulator with the ability to collect realistic data. Afterwards, they would use this data to project a point cloud and determine the optimal camera solution. The optimal solution was derived from how the simulator generated the best point cloud.



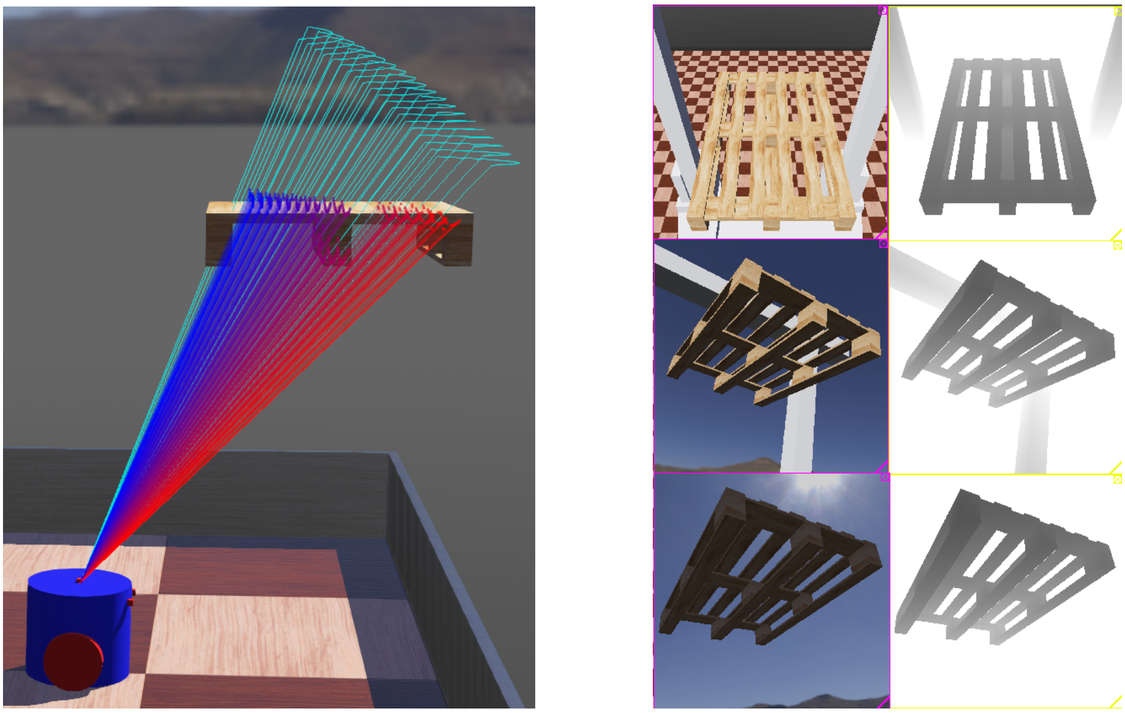
(a) LiDAR node concept drawing [28]

(b) Range finder concept drawing [29]

Figure 62: Concept drawing of Lidar and RangeFinder

After researching the Webots library, the group discovered two possible ways to sample 3D data, seen in figure 62. The group's first attempt was to use a LiDAR node on top of a moving robot. This was done in order to model and simulate a laser scan, as seen in figure 63a [28].

The other solution was to use a range finder that models a depth camera [29]. Along with the range finder, the group attached a camera module with the same specifications. This made it possible to display both the camera view and the depth in separate windows, seen in figure 63b.



(a) LiDAR node implemented

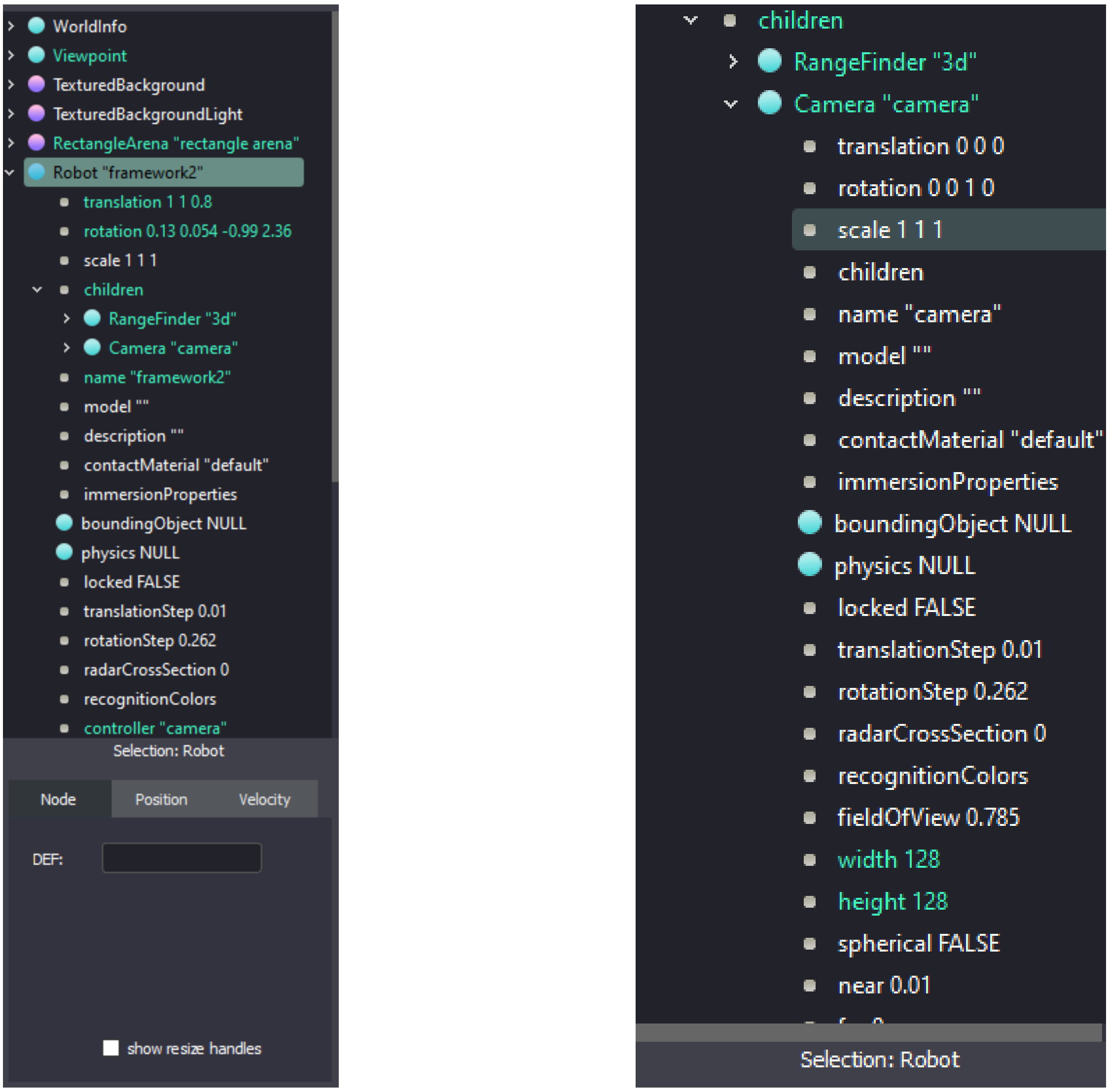
(b) Camera range finder implemented

Figure 63: Two different methods to sample data for a point cloud

The key distinction is that the vertical field of view of the range finder node is enforced by the image size (width and height), whereas the LiDAR node's vertical field of view is imposed by the depth buffer lines of pixels (laser scan). It will therefore be easier to model various 3D cameras through the rangefinder and camera combination [29].

The group found that a range finder combined with a camera would be the best way to gather data. By combining the two, the group would, in addition to displaying a point cloud, be able to display color of each point. This made for more intuitive results as they could now fully reconstruct an object.

With the method of data gathering selected, next came the building of the simulation environment. For each camera, a robot node with a range finder and camera was created. These child nodes can be configured to have the specifications of the cameras the group wanted to simulate. Child nodes inherit the position of their parent, so moving the camera is simply moving the robot node and both children move with it.



```
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
Robot "framework2"
    translation 1 1 0.8
    rotation 0.13 0.054 -0.99 2.36
    scale 1 1 1
    children
        RangeFinder "3d"
        Camera "camera"
    name "framework2"
    model ""
    description ""
    contactMaterial "default"
    immersionProperties
    boundingObject NULL
    physics NULL
    locked FALSE
    translationStep 0.01
    rotationStep 0.262
    radarCrossSection 0
    recognitionColors
    controller "camera"
Selection: Robot
Node    Position    Velocity
DEF:
□ show resize handles
```

(a) Robot in the scene tree

```
children
    RangeFinder "3d"
    Camera "camera"
        translation 0 0 0
        rotation 0 0 1 0
        scale 1 1 1
        children
        name "camera"
        model ""
        description ""
        contactMaterial "default"
        immersionProperties
        boundingObject NULL
        physics NULL
        locked FALSE
        translationStep 0.01
        rotationStep 0.262
        radarCrossSection 0
        recognitionColors
        fieldOfView 0.785
        width 128
        height 128
        spherical FALSE
        near 0.01
Selection: Robot
```

(b) Camera in the scene tree

Figure 64: The scene tree showing the robot and camera nodes

These robots can then be moved into position. The groups first simulator environment was two robots facing a simple box. In order to capture the images and analyze them further, the robot controllers need to be programmed. The controllers can be programmed in several languages, Python, Matlab, C, C++, java and ROS. The group was most familiar with python, so that was the language selected for the controller program [H].

The controller has to initialize both the camera and rangefinder nodes, and export the captured data in a useful format for the point cloud generation. The data from each of the sensor elements is an image matrix. The camera contains the RGB color values for each pixel and the range finder matrix contains the distance to any detected surface for each pixel. Using the numpy library, these matrices are written to text files and can then be accessed by other programs for processing.

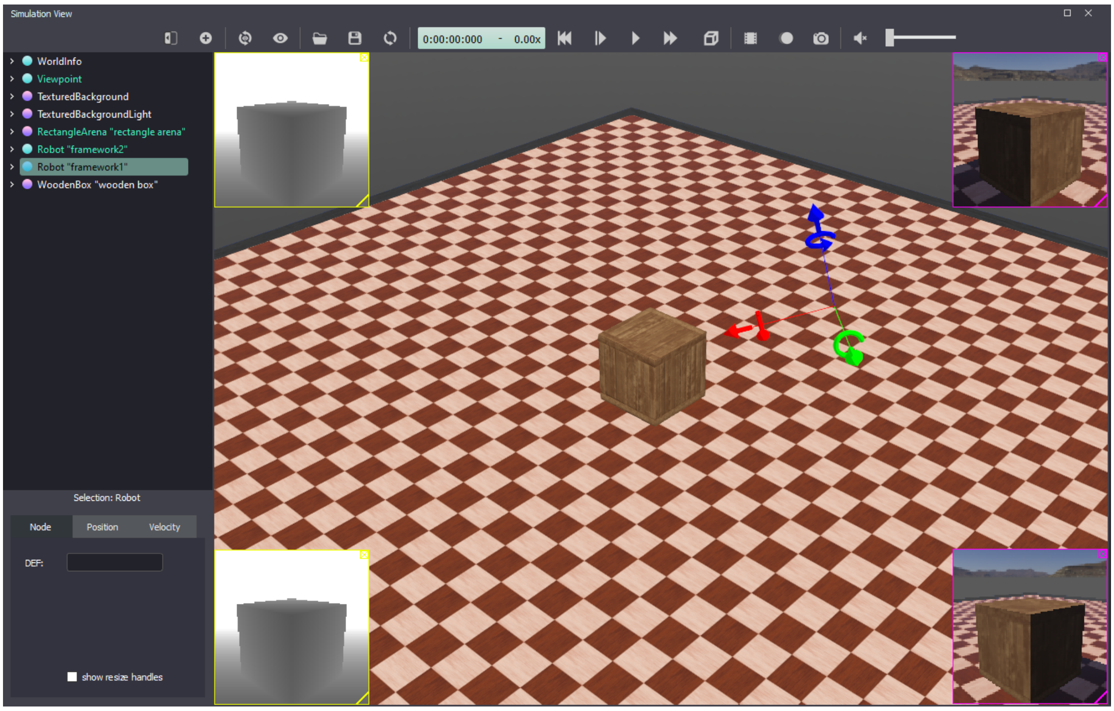


Figure 65: The webots simulation window of the simple simulation environment

With the simple simulation environment and robot controllers complete, the next step was to process the data into point clouds and verify that it was working as intended. In order to display and manipulate the point clouds the group needed to find a new toolkit. In previous 3D geometry applications the group had done, the python module ”matplotlib” was used. For these point clouds with several thousand points, this module is not well suited. After looking into what tools existed, the group decided to use the python module ”open3D”. This was because it is well documented with several examples available. As well as having a set of point cloud specific functions that fit the group's needs. Using the open3D module, the matrices are imported by another python script [I]. It also uses the camera parameters converted to coordinates rather than pixel values. With the rotation and translation from the Webots environment, a transformation matrix is created, and the point clouds are oriented such that they fit together.

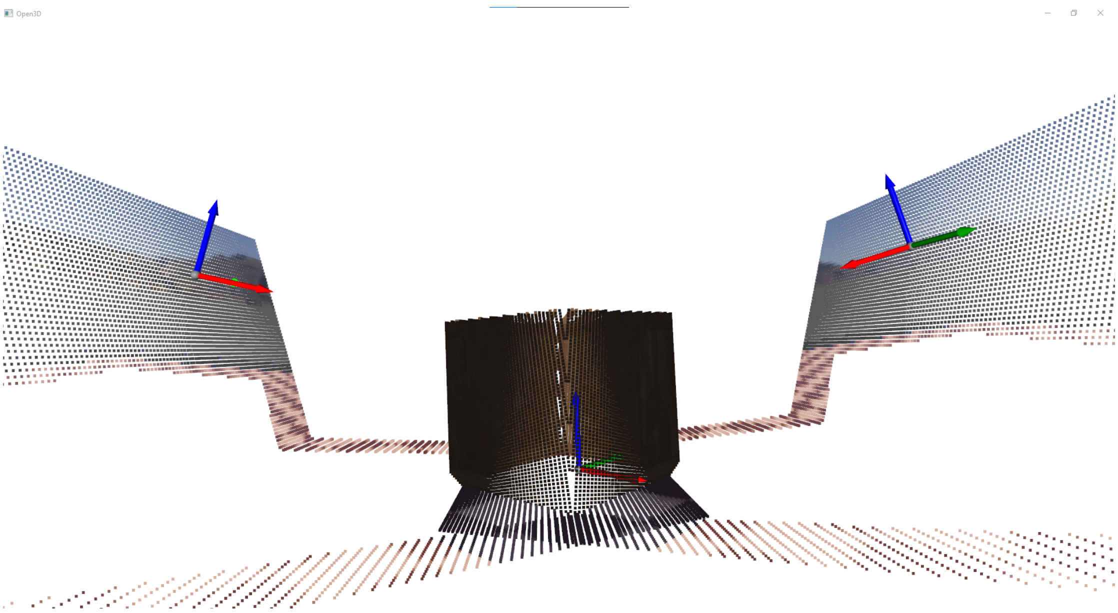


Figure 66: The combined point clouds from the simple simulation, the axes represent the camera positions.

Satisfied that the method for extracting and processing data works, the group then started modifying the simulator environment to fit the Sort™ parameters. A simple model of the main tower was created in Onshape and imported into Webots. A simplified model of Sort™ was used rather than a more complex one to lessen the computing power required. A third camera was added and the cameras were moved into position. This simulation environment then allowed the group to take the three pictures, and verify that they detect enough of the pallet to make a successful identification.

To model the possibility of using two cameras to capture three images, a movable camera was added to a linear axis. To simulate the linear axis, several movable objects had to be added. A static beam would serve as the base. A movable box was added to the beam, and another box that could be tilted was added to the movable box. The camera was then added as a child of the tiltable box. The camera could now be moved up and down the beam, as well as being tilted up and down.

With the environment set up and the data processing script done, the simulation was complete and the camera data can be examined.

Chapter 4

Physical prototyping

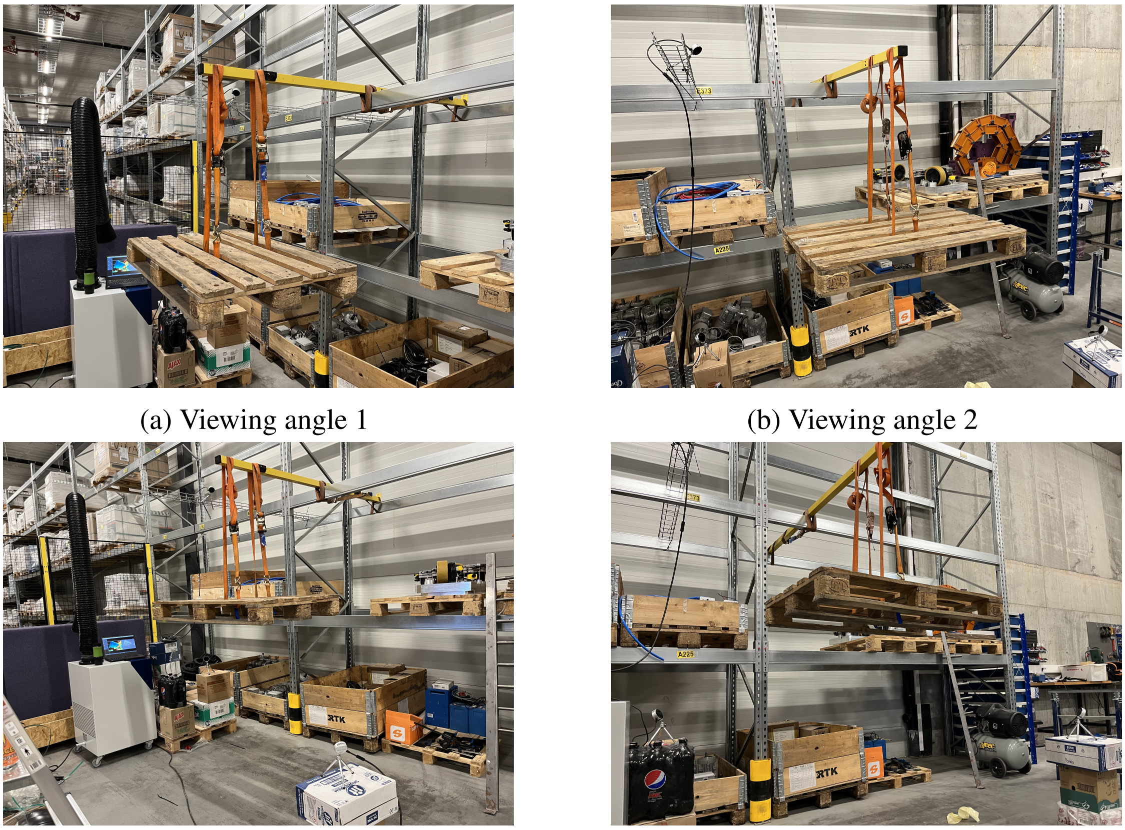
This chapter details the group's efforts to create physical prototypes of the suggested solution. The prototypes was developed at Solwr's warehouse with equipment that had previously been used to build prior versions of Sort™.

The prototypes used the Intel cameras from the current version of Sort™. Therefore, the prototypes are based on their FOV and range. Subsection 4.4, however, shows a conversion for the setup needed for the new Zivid cameras.

4.1 The first prototype

Boxes, straps, pallets, and the current cameras were used to construct the initial prototype. The scanned pallet was held in the air by straps, and the cameras were mounted on stacked boxes of everyday items found throughout the warehouse. This setup can be seen in figure 67.

This prototype was meant to verify the field of view using two cameras directly above one another. Because of how time consuming setting up a motor-controlled linear axis is, the group wanted to make sure that the solution was viable before proceeding.



(a) Viewing angle 1

(b) Viewing angle 2

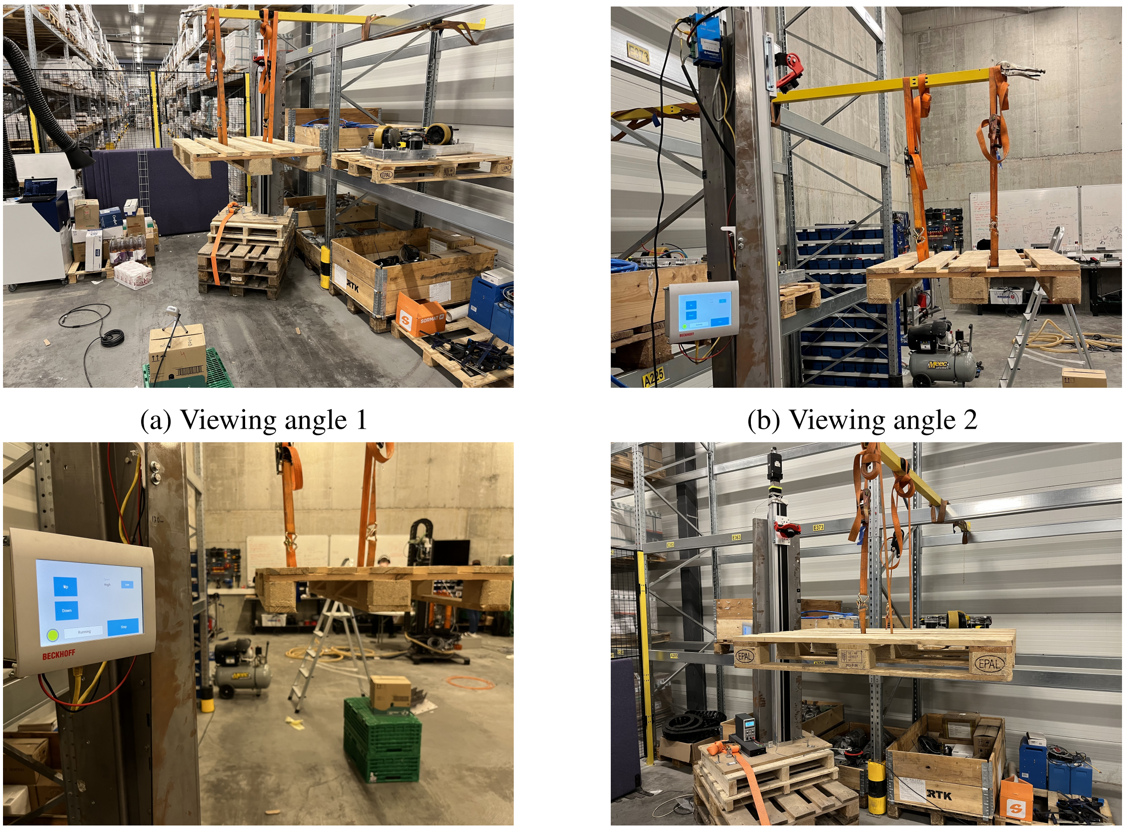
(c) Viewing angle 3

(d) Viewing angle 4

Figure 67: Prototype 1

4.2 The second prototype

The second prototype introduced a linear axis. The first prototype showed that the cameras were able to map a sufficient area of the pallet with two of the cameras placed directly vertical of one another. Therefore, the group proceeded by adding a motor, a linear axis, a PLC and an HMI. The two vertical cameras were now replaced by one camera being moved up and down the axis. The setup can be seen in figure 68.



(a) Viewing angle 1

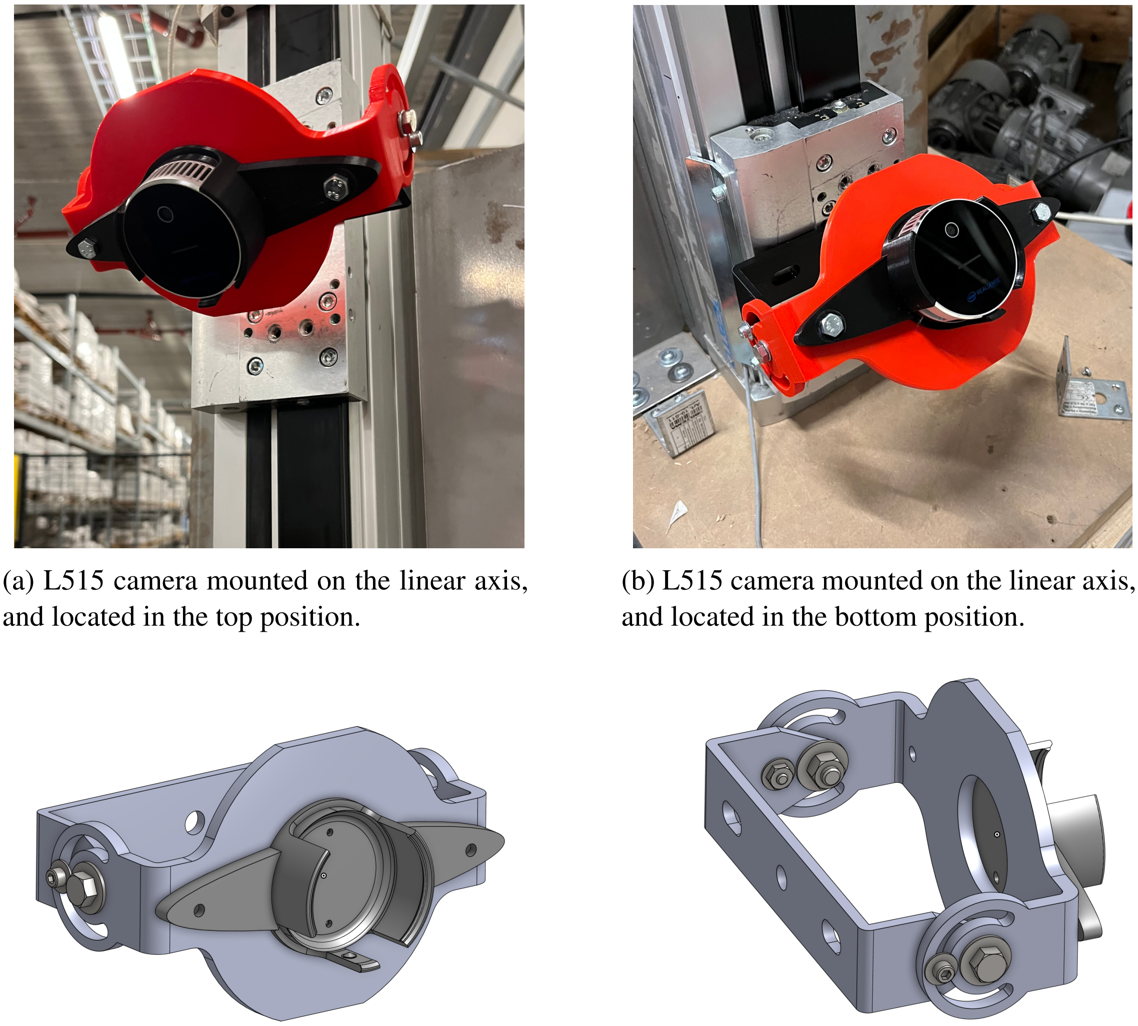
(b) Viewing angle 2

(c) Viewing angle 3

(d) Viewing angle 4

Figure 68: Prototype 2

4.2.1 Camera mounting on the linear axis



(a) L515 camera mounted on the linear axis, and located in the top position.

(b) L515 camera mounted on the linear axis, and located in the bottom position.

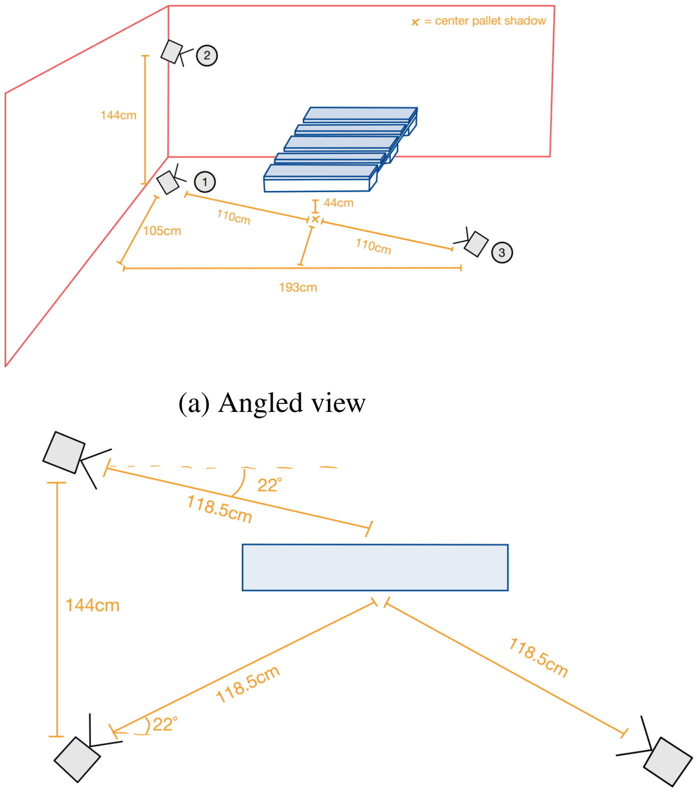
(c) The L515 camera mount CAD, front view

(d) The L515 camera mount CAD, side view

Figure 69: 3D-printed camera mount

4.3 Camera placement

Given that the group was using Intel's RealSense cameras, the camera placement will vary from what is intended with the new cameras. This setup should still work as a proof-of-concept, seeing that the new cameras have better specifications. The camera setup can be seen in figure 70.



(a) Angled view

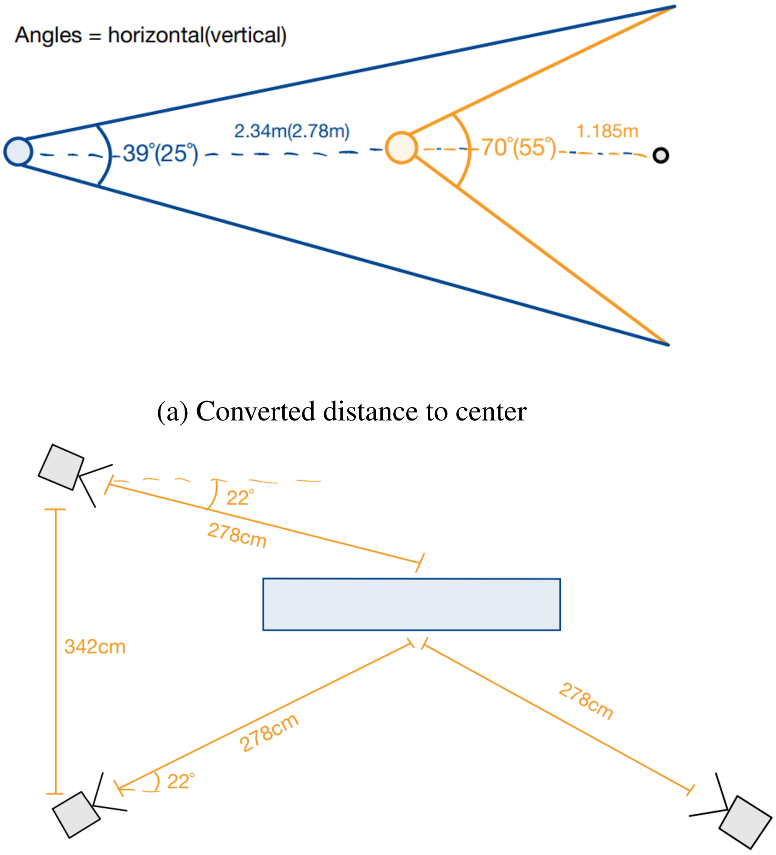
(b) Side view

Figure 70: Camera placement

Two of the cameras are placed directly vertical to one another, while the third is placed diagonally from the bottom one of the first two.

4.4 Using the Zivid One$^+$ Large in the prototype

In order to use this setup with the new cameras, the distances would have to be increased. This is because of the higher minimum operating range of the Zivid One+ Large cameras, as well as its smaller FOV. To determine the limiting factor, the group has defined a formula described in 2.2.4. The limiting factor was discovered to be the vertical FOV angle. In order to compensate, the distance from the camera to the center of the pallet would need to be 2.78m instead of the 1.185m when using the Intel cameras. The conversion can be seen in figure 71.



(a) Converted distance to center

(b) Converted distances, side view

Figure 71: Calculation of new distances

4.5 Prototype hardware and software

To achieve building a prototype, the group needed several components. Fortunately Solwr had components from previous installations and prototypes, and therefore there was no need to buy additional hardware.

Hardware that Solwr provided:

- 3 x RealSense L515 cameras
- Nanotec brushless DC motor, DB87L01-S
- Nanotec motor controller/ Drive, N5-2-1
- Incremental encoder, WEDS5541
- A linear axis
- Beckhoff PLC with touch screen, CP6606
- 230V AC / 24V DC converter
- 230V AC / 48V DC converter

4.5.1 Electrical wiring - Motor, Drive, Encoder, and Power supply

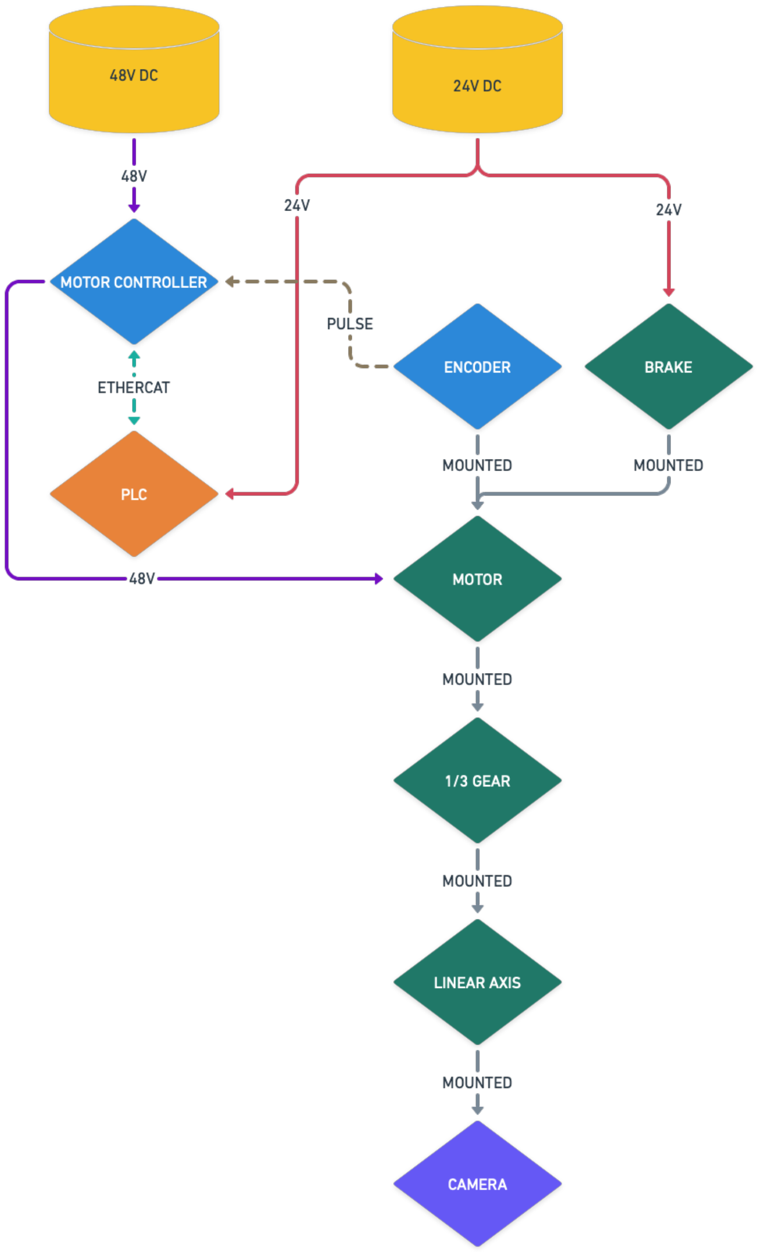


Figure 72: Component hierarchy

The group needed to establish connection between the PLC, motor controller, motor, and encoder in order for the linear axis to function. Furthermore, the group needed to learn Nanotec's software in order to set up and sync all of the hardware. Finally, the group needed to write a PLC program for controlling the axis, using an HMI.

The group used Ethernet and EtherCAT communication, as well as Nanotec's Plug&Drive Studio 1 software, to program and set up the controller. The group used EtherCAT to manually set the IP, and Ethernet to setup the motor controller. They were recommended to do this by Solwr. EtherCAT was also the protocol chosen for operations after the setup, and the system architecture is illustrated in figure 72.

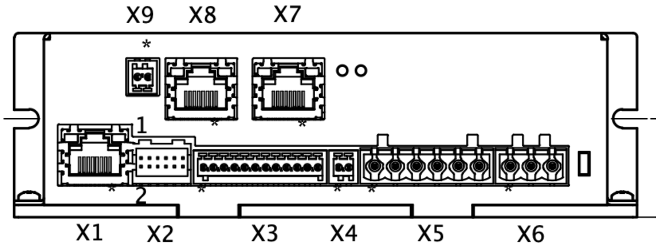


Figure 73: Nanotec N5 2-1 motor controller I/O drawing

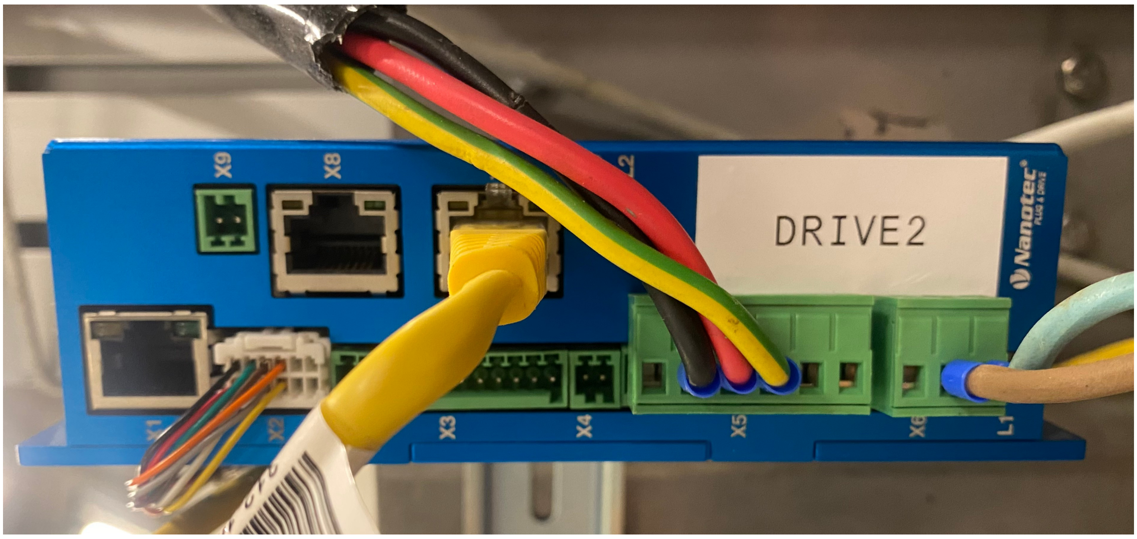


Figure 74: Nanotec N5-2-1 motor controller electrical wiring

A three-phase 48V DC power source was required to power the drive and motor. A 230V AC / 48V DC converter was utilized and wired to port X6. The three phases of the motor was wired to X5. Finally, the encoder was linked to X2, and wired according to the encoder datasheet and the drive quick guide in Appendix [E] and [F]. To terminate the encoder cables, the tool WC-244 **??** were used, together with sph-001t-p0.5 sockets.

The motor has a brake mounted on it's rotor which is always applied, unless the brake is provided with 24V DC. Therefore a 230v AC / DC regulator was utilized.

4.5.2 Static IPv4 controller and PC

Neither the group or Solwr knew the static IPv4 address for the motor controller. The group was not able to connect to it. Therefore, the group needed to read or set a new static IPv4 address for the drive. The group achieved this with a combination of research and trial and error. There are several ways for the drive to obtain an IP, according to the motor controllers quick guide in appendix [F]. A DHCP server, for example, can automatically assign an IP address. Another option is to use EtherCAT to manually set it, which was the technique used using Plug&Drive Studio 1. The group made a how-to-guide located in appendix [A.1]. The static IPv4 was read to be 10.0.0.6.

The static IPv4 address of the computer in use was set to 10.0.0.3. Any address that has the same three first numbers, but a different fourth one, can be used.. The group made a how-to-guide on setting a static Ethernet IP on a PC, and is located in appendix [A.2].

4.5.3 Motor setup using Ethernet in Plug&Drive Studio 1

Since the IP had been manually set, the Ethernet Port X1 could now be utilized to establish a link between the PC and the drive. The setup required some motor specification, which is located in the motor datasheet in appendix [D]. During the automatic setup, the electrical peak and rated current restrictions, as well as the number of poles, were required specifications to avoid damaging the motor controller. The group made a guide for connecting to and setting up the motor controller using the Plug&Drive Studio 1. This guide is in appendix [A.3].

The group left the motor unloaded on a table during the automatic setup. The rotor started turning as soon as the auto setup started. All other essential parameters, including encoder requirements, were determined by the auto setup.

The group went on to designing and building the PLC software for moving the axis now that all of the components were properly set up.

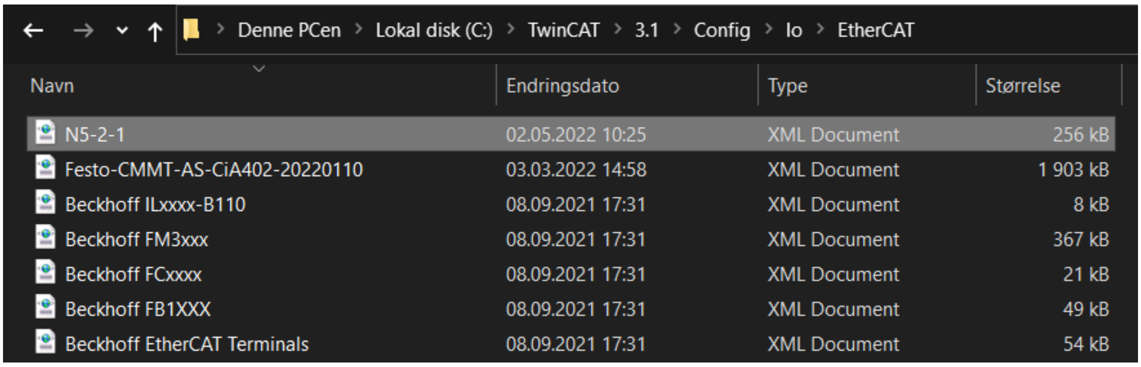
4.6 Beckhoff and TwinCAT 3 PLCs

Solwr uses Beckhoff PLCs to control the movement of their robots. Their PLCs are programmed in TwinCAT 3, and this is what the group will be doing as well. TwinCAT 3 runs on Microsoft Visual Studio, with Structured Text as the group's programming language of choice.

In this project, the group used a CP6606 PLC with a built-in touchscreen as the HMI. The group created a program that moves a linear axis between a minimum and maximum position at different speeds. Between the PLC and the drive, the EtherCAT communication is wired through port X7.

4.6.1 Adding and installing the drive's ESI-file to TwinCAT

The PLC must be able to communicate with the motor drive in order to operate the motor. TwinCAT 3 has an NC-Axis capability that allows it to create a link between the PLC and the drive. Since the drive was manufactured by Nanotec, an EtherCAT ESI-file must be added before TwinCAT 3 can read it. The file required in this case is N5-2-1, which may be downloaded from Nanotec's website. Afterwards, the file must be copied into the EtherCAT folder, found within the TwinCAT 3 installation folder. Figure 75 illustrates the file name and the folder pathway. A PC restart is recommended after adding the file.



Denne PCen › Lokal disk (C:) › TwinCAT › 3.1 › Config › Io › EtherCAT

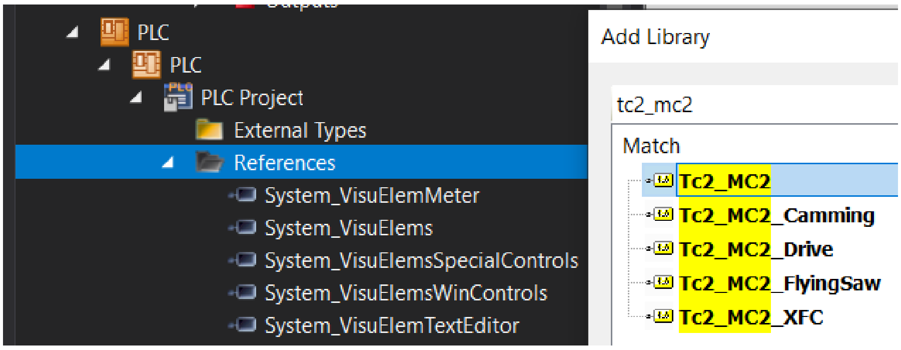
Navn	Endringsdato	Type	Størrelse
N5-2-1	02.05.2022 10:25	XML Document	256 kB
Festo-CMMT-AS-CiA402-20220110	03.03.2022 14:58	XML Document	1 903 kB
Beckhoff ILxxxx-B110	08.09.2021 17:31	XML Document	8 kB
Beckhoff FM3xxx	08.09.2021 17:31	XML Document	367 kB
Beckhoff FCxxxx	08.09.2021 17:31	XML Document	21 kB
Beckhoff FB1XXX	08.09.2021 17:31	XML Document	49 kB
Beckhoff EtherCAT Terminals	08.09.2021 17:31	XML Document	54 kB

Figure 75: The file pathway for inserting the N5-2-1 ESI file

4.6.2 Linking motor controller and axis in TwinCAT 3

Within Visual Studio the group was able to locate the motor drive as a device, within ”I/O” in the side menu named the Solution Explorer. This was accomplished by right-clicking ”I/O” and selecting ”search”. TwinCAT 3 establishes an axis automatically, for the group to link with the drive. TwinCAT 3 can after pairing read all real-time values directly from the motor drive, such as values for velocity, position, torque, etc. The groups how-to-guide is listed in appendix [A.4].

Installing Tc2 Mc2 library


PLC
PLC
PLC Project
External Types
References
System_VisuElemMeter
System_VisuElems
System_VisuElemsSpecialControls
System_VisuElemsWinControls
System_VisuElemTextEditor

Add Library

tc2_mc2

Match

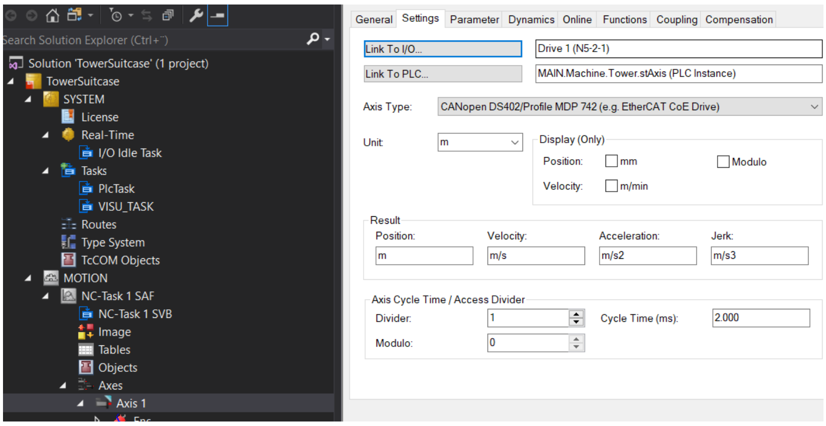
- Tc2_MC2
- Tc2_MC2_Camming
- Tc2_MC2_Drive
- Tc2_MC2_FlyingSaw
- Tc2_MC2_XFC

Figure 76: Installment of the Tc2 Mc2 library through references within the solution explorer.

The Tc2 MC2 library from TwinCAT 3 Motion Control offers function blocks for programming machine applications. It is based on the Motion Control function blocks V2.0 PLCopen specification [30].

To establish a link between the axis and PLC, an axis reference variable ”AXIS_REF” needs to be created. All manufacturers who use PLC Open offer this axis reference as a standard variable type. The Tc2 MC2 library provides ”AXIS_REF” which must be included in the PLC’s ”References” section. To install a new library simply right-click ”references” in the solution explorer seen in figure 76, and select ”Add new library”.

The AXIS_REF was placed in a function block in the PLC program under the variable name: ”stAxis” [31].



Link To I/O... Drive 1 (N5-2-1)

Link To PLC... MAIN.Machine.Tower.stAxis (PLC Instance)

Axis Type: CANopen DS402/Profile MDP 742 (e.g. EtherCAT CoE Drive)

Unit: m

Display (Only)
Position: □ mm □ Modulo
Velocity: □ m/min

Result

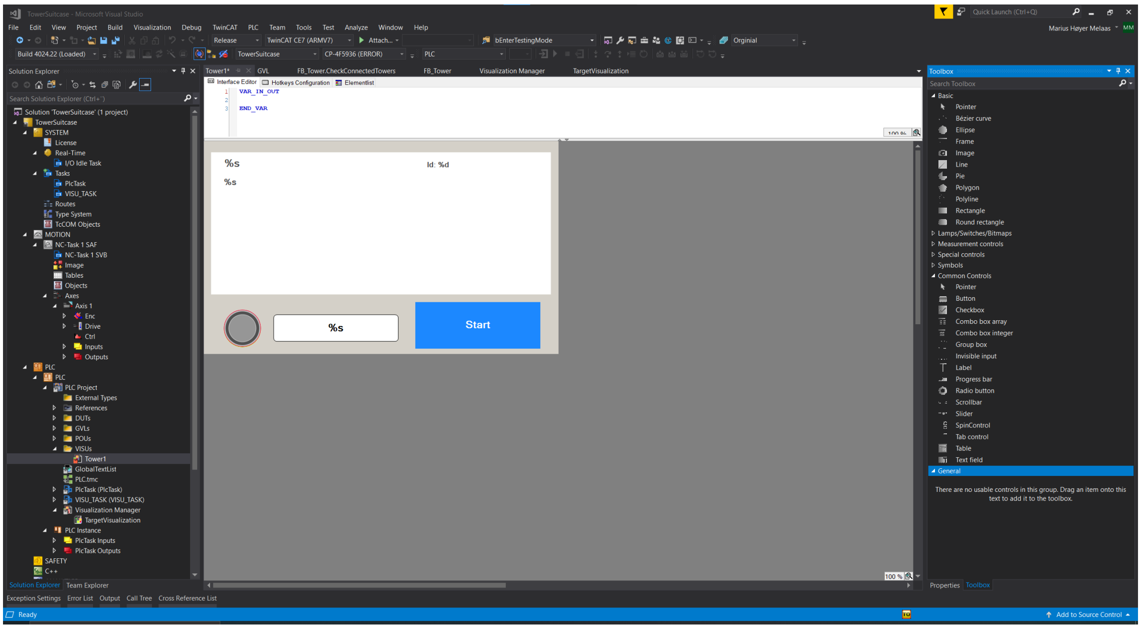
Position:	Velocity:	Acceleration:	Jerk:
m	m/s	m/s2	m/s3

Axis Cycle Time / Access Divider
Divider: 1 Cycle Time (ms): 2.000
Modulo: 0

Figure 77: Navigate to ”Axis” in the solution explorer, and then into the ”settings” tab. Make a link for both PLC and I/O.

When the Tc2 MC2 library is installed and the ”AXIS_REF” is referenced, a link between the PLC and the drive can be established. As illustrated in Figure 77 , create a ”Link to I/O” and ”Link to PLC.”

4.6.3 Building the HMI



```
VAR_IN_OUT

END_VAR
```

Figure 78: HMI building environment in Visual Studio

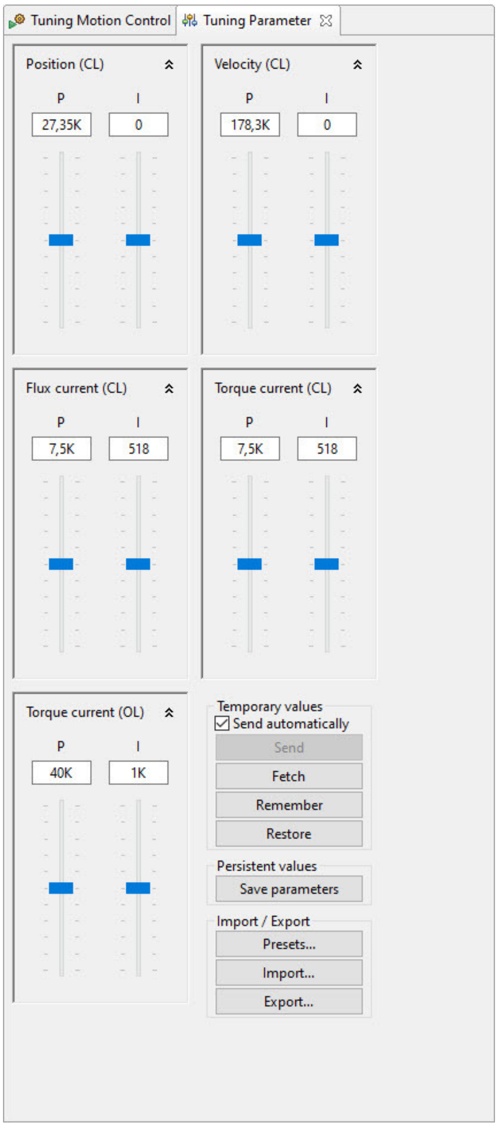
The HMI is created in Visual studio and is linked to variables in the PLC program. All buttons and lamps are picked from the ”Toolbox” located to the right in figure 78. Afterwards, each button is linked to several variables or states that decides if the button should execute an action or be hidden. This makes it seem like the HMI switches between several screens. While in reality, the buttons that are not needed, are hidden.

If the prototype was more complex, the group would have made multi-screen HMI. This makes the HMI easier to edit or change in future updates. However, for this simple prototype, this was not needed.

4.6.4 Axis parameter settings

Several settings regarding the axis and encoder have to be specified in order to ensure that the linear axis travels as predicted. The group found inspiration in a Beckhoff guide [32], and discovered that the majority of parameters were set correctly by default. The calculations for maximum velocity and scaling factor, however, did not match the needs of the group. The group found the needed formulas in Beckhoff’s guide, and calculated new maximum velocity and scaling factor values based on those. To see the group’s calculations, and the group’s how-to-guide, see appendix [A.6]

4.7 Motor controller tuning using Plug&Drive Studio 1



Position (CL): P 27,35K, I 0

Velocity (CL): P 178,3K, I 0

Flux current (CL): P 7,5K, I 518

Torque current (CL): P 7,5K, I 518

Torque current (OL): P 40K, I 1K

Temporary values

☑ Send automatically

Figure 79: The motor controller tuning parameters in the Plug&Drive Studio 1 software

Since the motor, motor controller, encoder and PLC are all set up and connected using EtherCAT, the remaining task is to tune the motor controller output. The parameters apply to the position, velocity, current and torque. The tuning is done through adjusting the P (proportional) and I (integral) values in the Plug&Drive Studio 1 software. Before tuning, the motor and linear axis must be connected, the correct weight must be attached to the linear axis.

The tuning was mostly done through trial and error, and the group noticed that with improper tuning the motor had a high pitched sound and a rough jerk while starting and stopping the movement. After tuning, the motor and axis are moving silently and smooth. The final PI values are displayed in figure 79.

4.8 Processing the prototype data

The data acquired from the prototype is in the form of three separate point clouds generated by the intel cameras. These need to be transformed such that they fit together. The exact position and rotation of the cameras is not known so the group sought to use another method to fit the point clouds together. The group landed on the open3d multiway registration method[33]. This method uses as inputs the three point clouds and finds the transformation required for the surfaces to overlap.

For the first attempt the method failed. Due to the large amount of extra points from the walls and floor there were a lot of extra flat surfaces that interfere. In order to solve this problem the group wanted to filter out the unnecessary data from the point clouds. The first attempts at this were made using the built in filters that remove outliers. This did not remove the large flat surface from the floor and wall. So another filter that removes any points further away from an arbitrary point was used. In our case the camera was selected as the filter point and any point more than 2 meters from the camera were removed.

With the filtered point clouds the multiway registration succeeded in creating a transformation for each of the three point clouds and placed them on top of each other.

Chapter 5

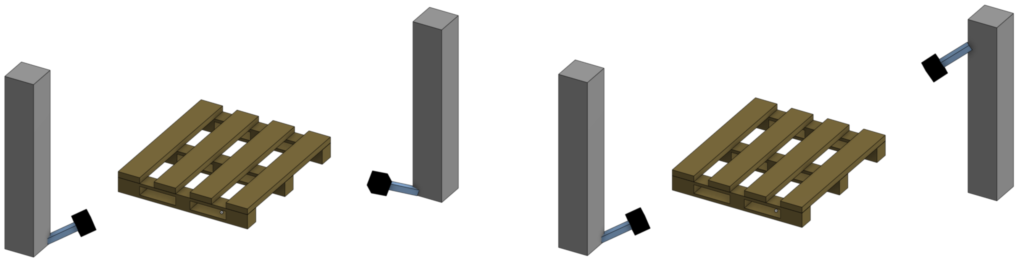
Results

This chapter contains 3D-models, 3D-plots and a description of what the group has accomplished during the project period. It will also contain the group's recommendation on how the employer should proceed in regards to redesigning the current version of Sort™.

5.1 Design

The group chose design seven in chapter 3.3.7 as the most suitable solution. This solution included the fewest amount of cameras, while still maintaining the resolution and speed the system requires. The group decided that a linear axis would be a viable solution for moving the camera between two vertical positions. The moving camera would first take a photo from the top view, and then move to the lower position. After reaching the set position and the grabber having picked up the pallet, the camera would capture a photo of the bottom and sides.

Several components are required. To construct the linear axis, the Zivid One$^+$ Large camera and a frame, as well as a tiny servo to alter the camera's angle, will be needed. The linear axis will need to be highly accurate. A larger, powerful servo or stepper motor will be required to move that axis. A encoder is required for the motor to rotate the desired distance. For emergency stops, a brake is preferred. A PLC is required to run the program and set up a motor controller in TwinCAT. The last is a power supply for the PLC and motor/motor controller. When the linear axis is built, the remaining component is the diagonally camera.



(a) Camera in bottom position

(b) Camera in top position

Figure 80: Design seven

5.2 Camera

The chosen camera was Zivid's One$^+$ Large. The decision was made by comparing prices, resolutions, fields of view and delivery times of several different cameras. The One$^+$ Large has specification suitable for the needs of Solwr and their Sort™. Zivid is also Norwegian, meaning delivery time is short and Solwr can deduct the VAT from the purchase.

Figure 81: Zivid camera

The camera specifications and price can be seen in table 6.

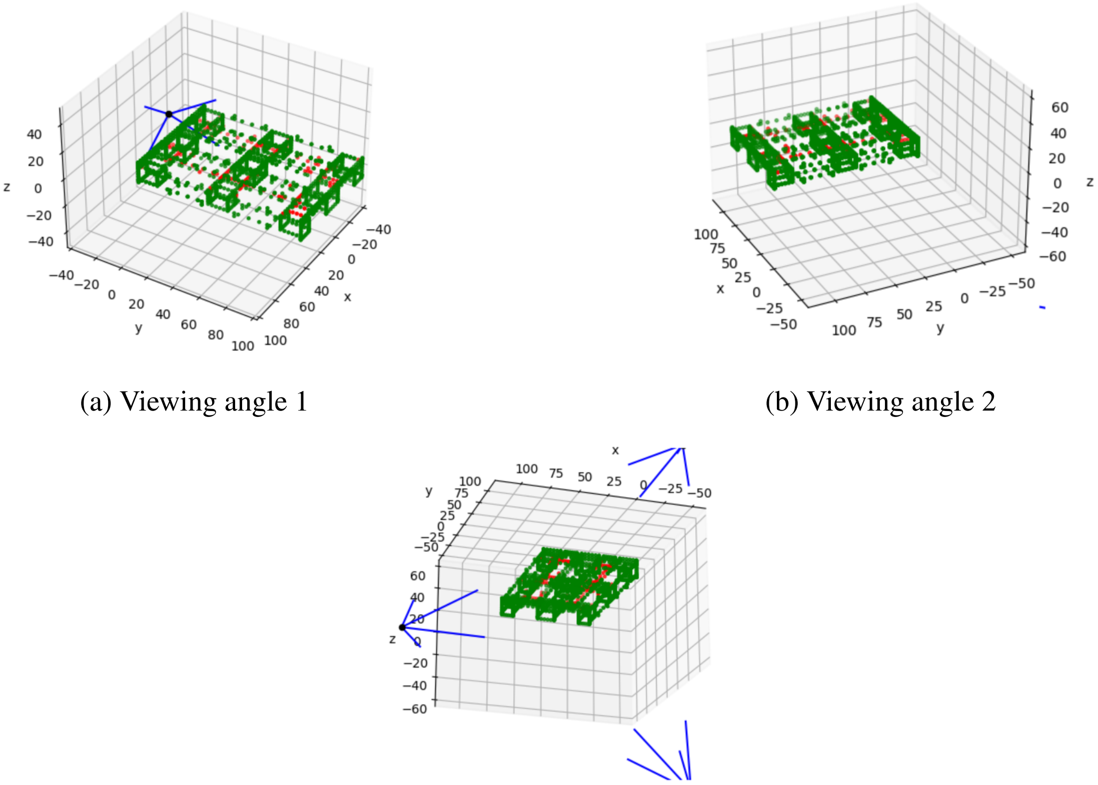
Specifications for Zivid One$^+$ Large					
Resolution	FOV	Depth accuracy (distance in mm)	Range (mm)	Delivery time	Price (ex. VAT)
1920x1200	39°horizontal 25°vertical	$<350\mu$m (1800)	1200 to 3000	4 WEEKS	8400 EURO

Table 6: Zivid specifications and price

5.3 Self-developed FOV software

The self developed software proved to be a important tool for the groups research. Both while testing out different designs, and in verification in the groups hypothesis regarding camera, angle and acquisition. Once the group had received feedback from Solwr regarding one of their designs, the camera specification and location were run through the script, with promising results displayed in figure 82.

The figure 82 shows that there are some blind spots. However, these blind spots are in the current acquisition as well. After conferring with Solwr they concluded that it will have little to no impact on Sort™'s sorting results.



(a) Viewing angle 1

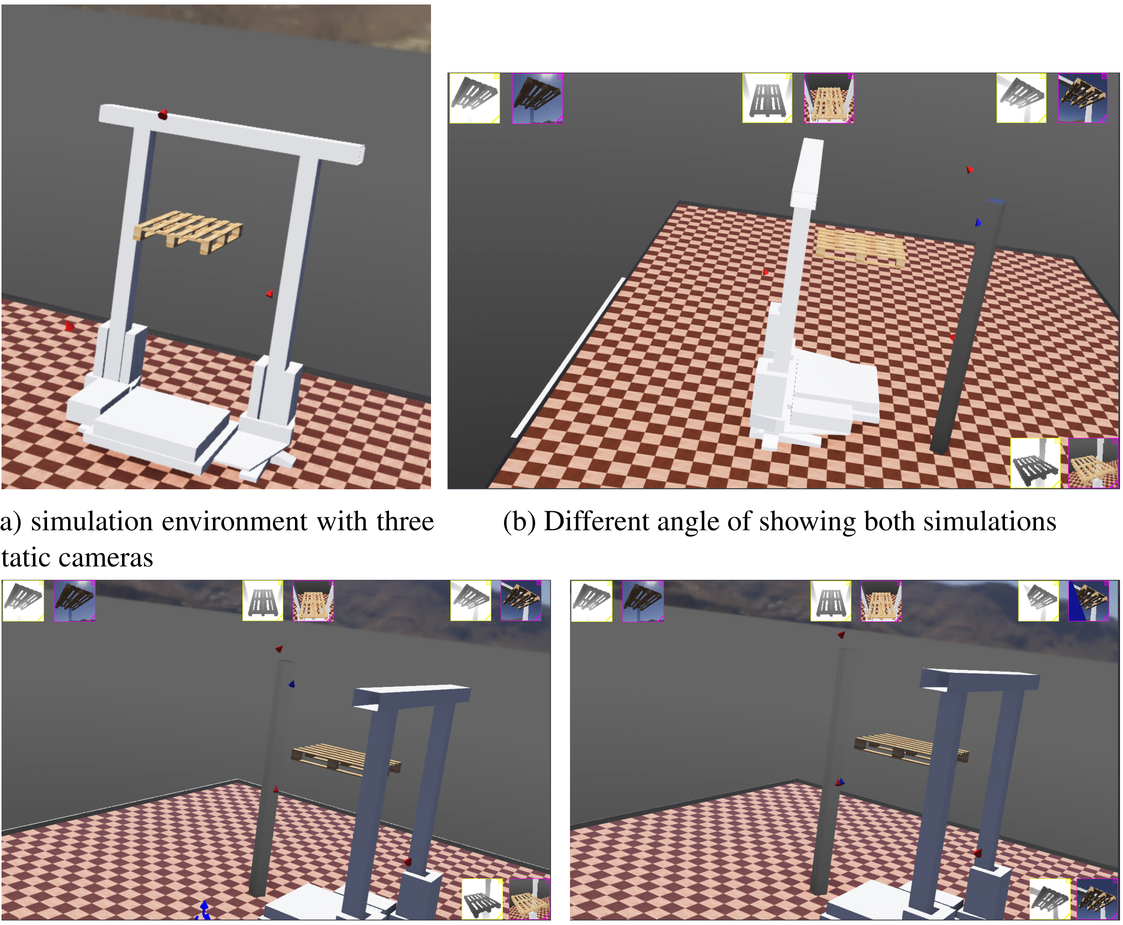
(b) Viewing angle 2

(c) Viewing angle 3

Figure 82: Results from FOV software

5.4 Simulation

5.4.1 Webots simulation of two solutions



(a) simulation environment with three static cameras

(b) Different angle of showing both simulations

(c) Linear axis camera(blue object) in the top position

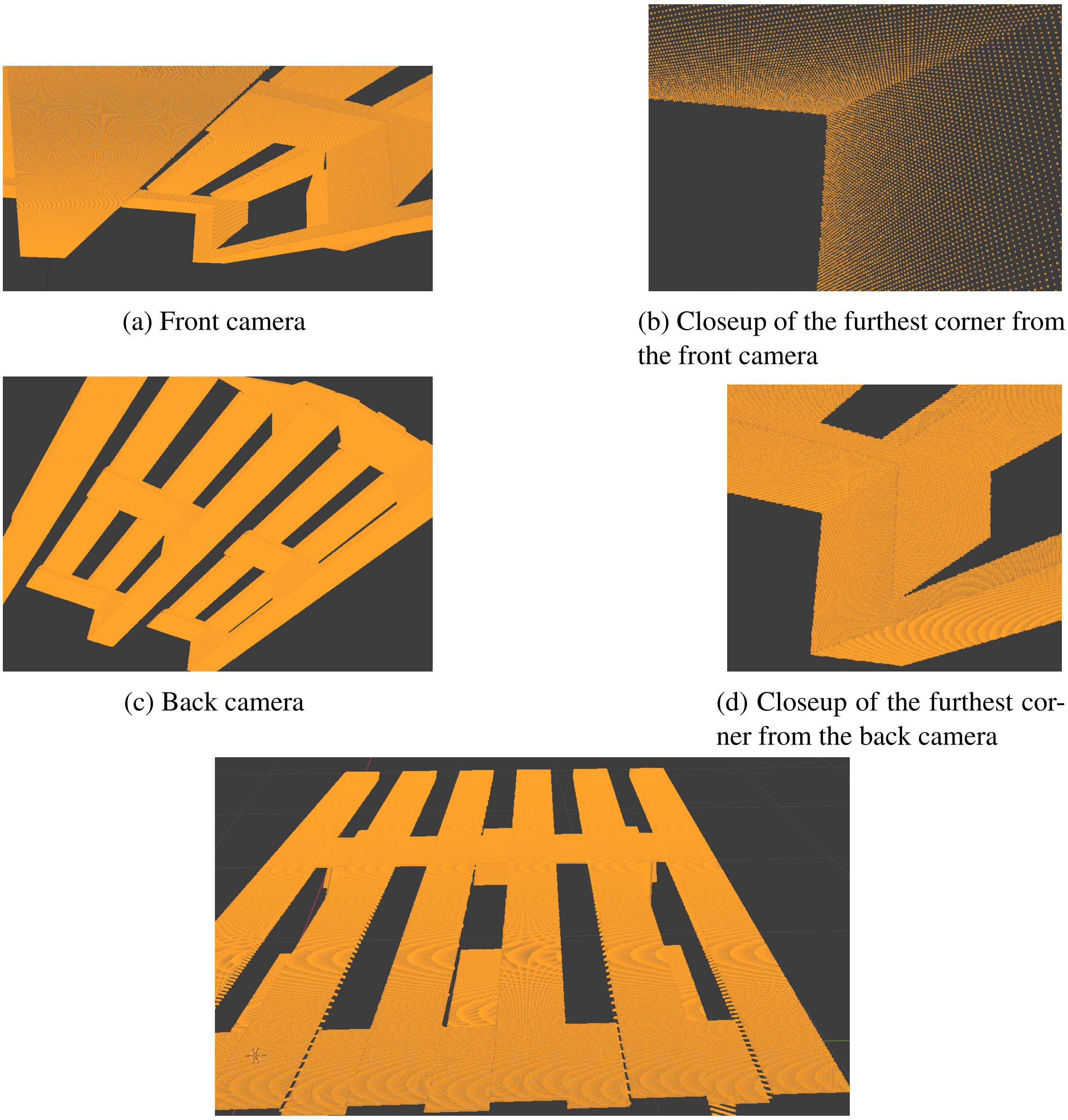
(d) Linear axis camera(blue object) in the bottom position

Figure 83: Two different solution simulated in the simulator environment Webots

In Webots the group made a simpler version of Sort™, to make simulation less computational demanding. However, the dimensions are still intact to simulate a realistic scenario. Two solution were simulated, one being a 3 camera setup illustrated in figure 83a using the three red static objects. The other simulation utilizes both a pole and joints to behave as a linear axis. This way the simulation only needs the blue moving object, and the red objects on is diagonal, illustrated in figure 83c and 83d.

The simulated version gave the group the ability to test out different positions and angles. In addition to this, the group collected data for the three scan points used to generate the 3D data. By simulating the linear axis, the group could find the length needed for the linear axis used in the prototype.

5.4.2 Zivid One$^+$ Large simulated pallet point cloud



(a) Front camera

(b) Closeup of the furthest corner from the front camera

(c) Back camera

(d) Closeup of the furthest corner from the back camera

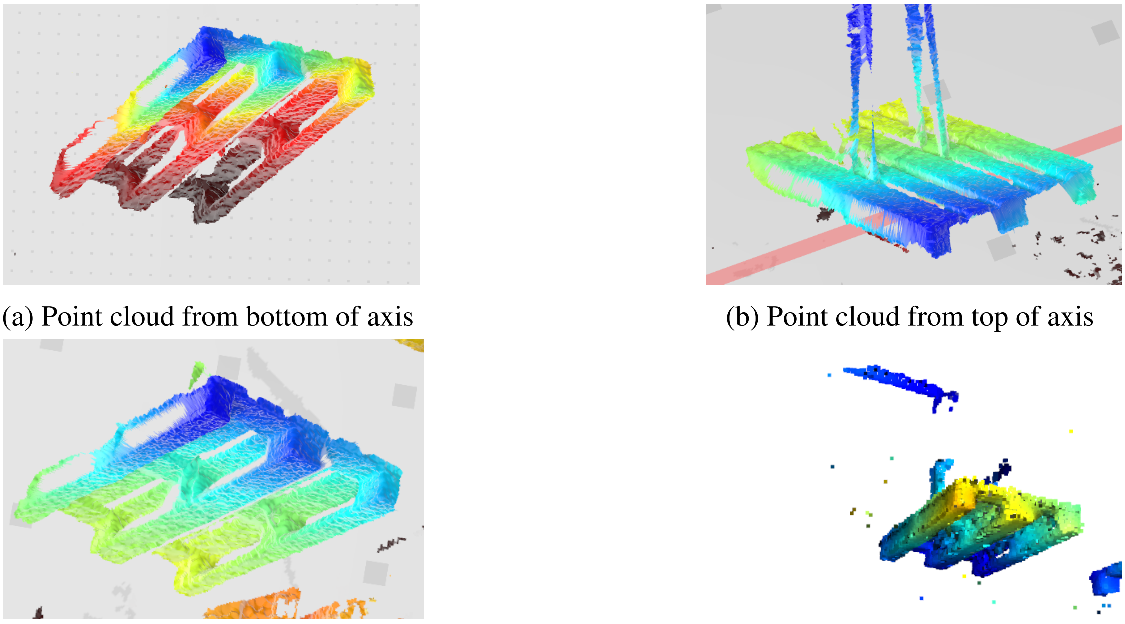
(e) Top camera

Figure 84: Point clouds from the simulated pallet taken with the Zivid one$^+$ Large camera specifications

The simulation cameras were set up with the Zivid one$^+$ Large specifications. The data was then fed trough the python point cloud generation script [I] to generate the point clouds.

5.5 Prototype

5.5.1 Camera setup



(a) Point cloud from bottom of axis

(b) Point cloud from top of axis

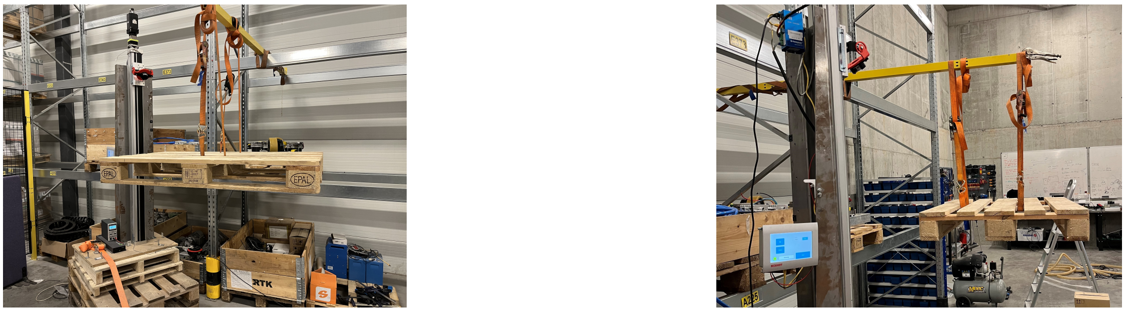
(c) Point cloud from diagonal camera

(d) Combined point clouds

Figure 85: Prototype, point clouds

The working prototype uses two cameras as planned and simulated, and in line with the motivation of this thesis. The motivation to limit the number of cameras as much as feasible, and make Sort™ economical competitive. One camera was mounted on a linear axis, moving vertically between a upper and a lower position. Each camera position generates a point cloud of the pallet from that perspective, shown in figure 85. Individually, these are deficient in regards to detail. They can, however, be meshed together, creating a 3D-object representing the pallet, shown in figure 85d.

5.5.2 Mechanical setup



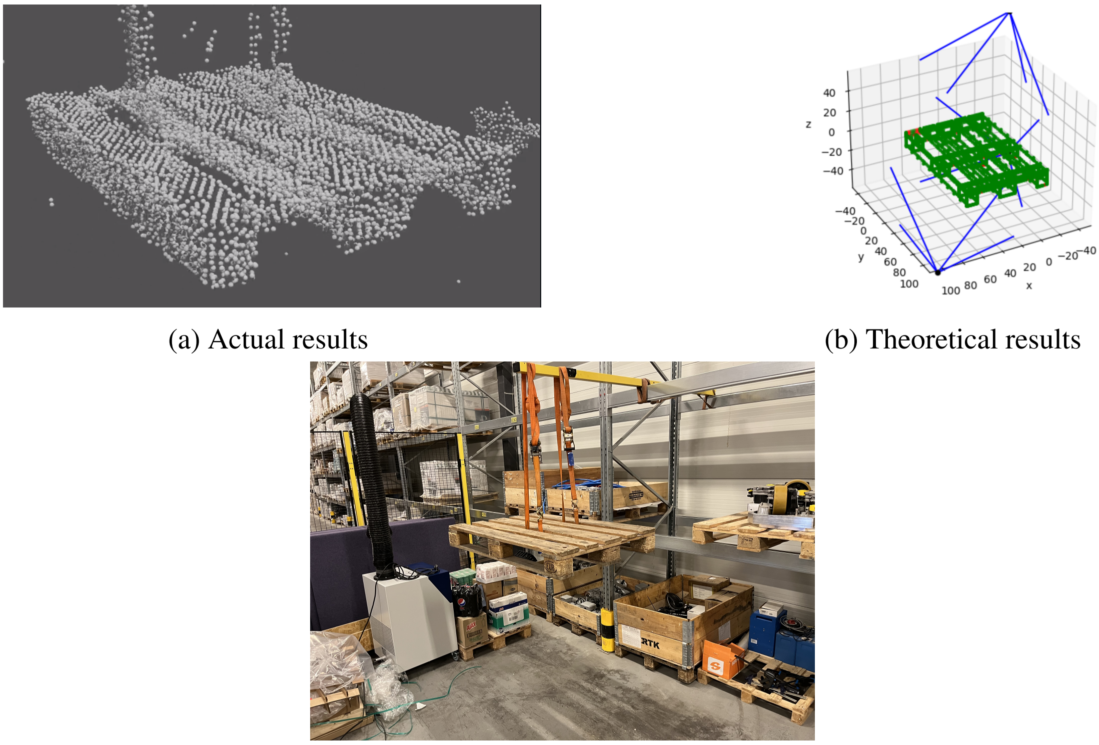
(a) Prototype, viewing angle 1

(b) Prototype, viewing angle 2

Figure 86: Finished prototype

5.5.3 Comparison of detected points in the point cloud

The group compared the expected and actual outcomes regarding detected points in the point cloud, in order to verify the results. They were able to assess whether or not the final output matched what they initially thought. This was done using the software's 3D plots and a merged point cloud, from the cameras used in the prototype.



(a) Actual results

(b) Theoretical results

(c) Actual image

Figure 87: Compared point clouds

The expected results through software and the results from the prototype were very similar, as seen in figure 87. Because the prototype was built with the Intel RealSense L515, the software is configured to have the same specifications. As mentioned previously in chapter 5.3, there are certain blind spots, but these are non-critical. These are findings that supports the group hypothesis to be a viable option for the next generation of Sort™.

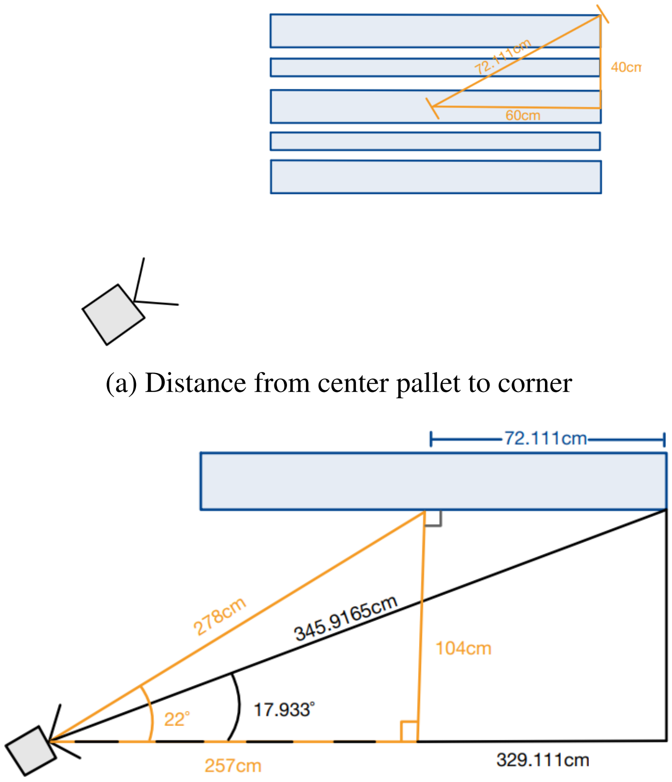
5.5.4 Distance between points

Before confirming the viability of the decided upon camera setup, the group had to make sure the resolution would be high enough to meet Solwr's criteria of a maximum of 5mm between points. In order to do this they used trigonometry in order to calculate the distance between two adjacent point along the limiting factor. Same as before, this would be the horizontal angle and horizontal resolution of the camera. Reading the camera specifications, the camera produced 1200 points along a 25° FOV.

First, they calculated the angle difference between two consecutive points.

$$Angle \quad between \quad points = \frac{25}{1200} = 0.0208^\circ \tag{21}$$

Then, they calculated the angle from the camera to the farthermost point of the pallet, relative to the ground. The distance is simplified to only consider two dimensions. In the prototype, the total distance consists of movement along two axes, but these calculations treat the total distance as an axis of it's own. This makes the next calculations easier.



(a) Distance from center pallet to corner

(b) Calculating angle

Figure 88: Calculating angle between the ground and the farthest point

$$Angle = \arccos(\frac{329.111}{345.9165}) = 17.933^\circ \tag{22}$$

Using that angle, they constructed unit vectors between the camera and the two farthest points using spherical coordinates converted to rectangular coordinates.

$$\begin{gathered}\rho = 1\\ \Delta\phi = \tfrac{25}{1200} = 0.0208^\circ\\ \phi_1 = 72.067^\circ\\ \phi_2 = 72.0462^\circ\\ \theta = 0^\circ\end{gathered}$$

$$\vec{v_1} = \begin{bmatrix}\sin(72.067)\cdot\cos(0) & 0 & \cos(72.067)\end{bmatrix}$$

$$\vec{v_2} = \begin{bmatrix}\sin(72.0462)\cdot\cos(0) & 0 & \cos(72.0462)\end{bmatrix}$$

In order to determine where these vectors reach the surface of the pallet, the group needed to define a planar equation 2.2.4, same as in the FOV software 3.2.3.

$$\hat{N} = \begin{bmatrix}0 & 0 & -1\end{bmatrix}$$

$$Planar \quad equation = -1\cdot(z-104) \rightarrow z = 104 \tag{23}$$

Because the bottom of the pallet is a flat surface in the xy-plane, crossing it only depends on the z-component. The d-component of the planar equation is based on the height difference of the camera and the bottom of the pallet, 104cm.

The next step is to find the ρ-value where the z-component of each vector is 104.

$$\rho_1 = \frac{104}{\cos(72.067)} = 337.76691 \tag{24}$$

$$\rho_2 = \frac{104}{\cos(72.0462)} = 337.38846 \tag{25}$$

Using these values of ρ, the group can calculate where in 3D-space the two neighboring points are. This, in turn, can be used to calculate their distance.

$$p_1 = 337.76691 \cdot \vec{v_1} = [321.35726, 0, 104] \tag{26}$$

$$p_2 = 337.38846 \cdot \vec{v_2} = [320.95946, 0, 104] \tag{27}$$

$$distance = p_1 - p_2 = 0.3978cm \approx 4mm \tag{28}$$

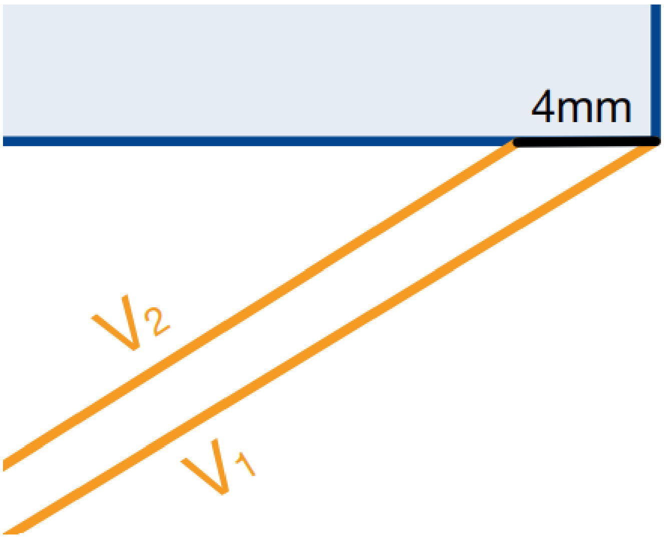


Figure 89: Distance between points

From these calculations the group concluded that the proposed camera setup is viable using Zivid One$^+$ Large.

The results of the prototype shows that the group's camera setup is viable. Seeing that the prototype pallet was suspended by straps, some parts of the pallet was missing, but that would not be the case in the real Sort™. With the distances converted to the distances needed with the Zivid cameras, the group calculated that the maximum distance between each point would be 4mm. This is sufficiently below the maximum Solwr had set for the group, which was 5mm.

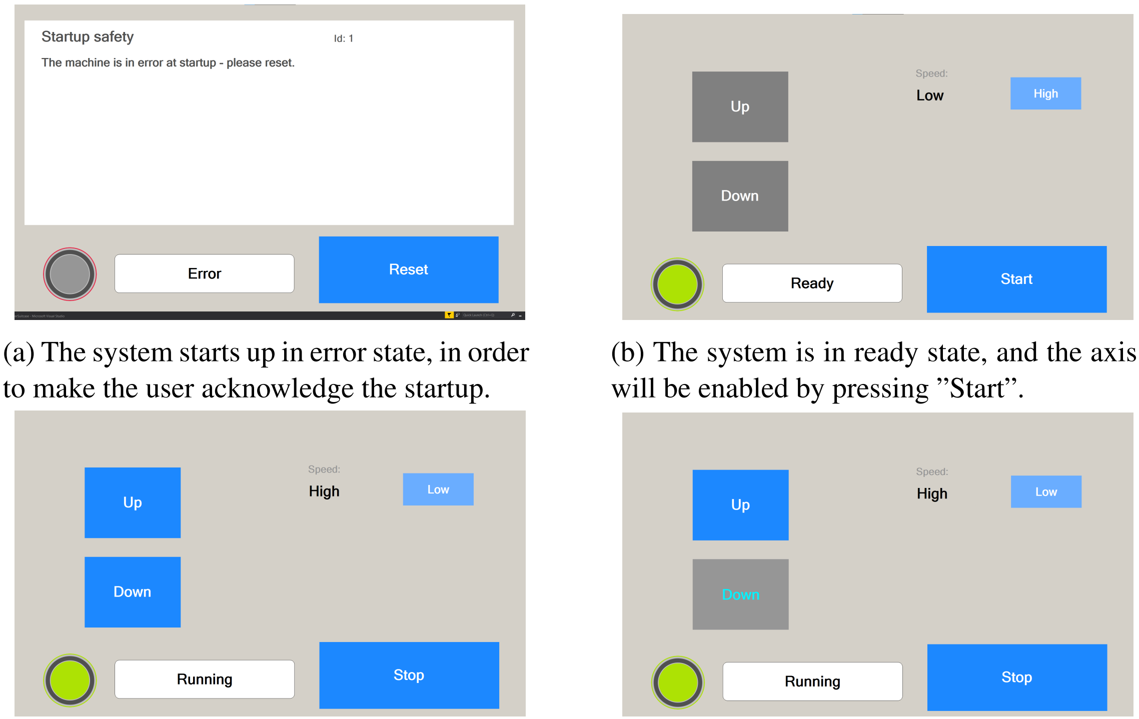
5.5.5 Speed of the prototype

By using Beckhoff's guide, the group calculated the maximum velocity the camera could reach with the prototype. A detailed calculation can be seen in [A.6]. The speed was found to be 0.416m/s. This would not negatively impact the time Sort™ uses per pallet. The current average cycle time is 24 seconds 3, this is enough time for the camera to take a photo of the top, then move to the bottom of the axis.

↩

5.5.6 HMI

While building and programming the prototype the group created a HMI (Human Machine Interface) in order to move the camera along the linear axis. Solwr will be using the prototype for several tests using the actual Zivid camera, therefore the group prioritized to create a easy, intuitively user experience. The group considered the design to be informative, not overly advanced, and having two speed options as useful when experimenting with different positions and angels.



Startup safety

Id: 1

The machine is in error at startup - please reset.

(a) The system starts up in error state, in order to make the user acknowledge the startup.

(b) The system is in ready state, and the axis will be enabled by pressing ”Start”.

(c) The state is in running state, and the axis can now be moved using the up/down buttons.

(d) Illustrates the using pressing the ”Down” button using high speed.

Figure 90: The HMI explanation

HMI user explanation:
As seen in figure 90a the systems starts up in error state, with a explanatory message to the user. This will require the user to reset the error, to proceed towards enabling the axis. During error state the led is blinking red.

In figure 90b the system has reached ready state, and the axis can be enabled by pressing ”Start”. The speed can at all times be adjusted between two levels high and low. The values for high and low speed is permanently set in the PLC software. During ready state the led is blinking green, and both buttons has no function, and therefore displayed with a gray color.

In figure 90c the axis is enabled, and the linear axis can be moved, using the up/down button. During running state the led is constant green.

In figure 90d is the ”Down” button pressed and is indicated by the gray color. The axis will automatically stop when reached the max or min position.

5.6 Economy for the chosen solution

Hardware	Quantity	Price	Total price
Zivid One$^+$ Large	2	10 500 €	21 000$
Intel RealSense L515	-6	589$	-3534$
Raspberry Pi	-4	145$	-580$
Optical cables	2	420 €	840 €
Linear axis	1x3m	unknown	unknown
Actuator	1	unknown	unknown
Camera portal	-1	unknown	unknown
Motor	1	unknown	unknown
Servo	1	unknown	unknown
Servo drive	1	unknown	unknown
Absolute encoder	1	unknown	unknown
Maintenance	1	unknown	unknown
Further research	1	unknown	unknown

Table 7: The new camera system's pricing

In table 7 the group has compiled a list of expenses and savings expected with the proposed solution. There are some items that need to be added once Solwr has made further progress. The group has not been able to get pricing for the axis, servo and motor. Once Solwr has done more research about the solution, they would be able to fill out the rest of the table. The savings are noted as a negative quantity, making the total sum of the table the price difference of the new versus the old system.

Chapter 6

Discussion

This chapter includes discussion about the results, uncertain elements within the thesis, possibilities the group did not have time to implement and possible shortcomings of the work done by the group.

6.1 Self-developed FOV software

6.1.1 Step count

The software currently has no recommended step count. Meaning the ray is discretized at a seemingly random, but high, number of intervals. The uncertainty of how the result of the planar equation varies as the ray approaches the plane is the reason for this. Currently, the software uses a very high step count, resulting in high margins at the expense of efficiency. The efficiency can be improved by calculating the smallest number of steps required for at least one sample of the ray to be within the ”box” around the surface, as illustrated in figure 41.

The current method of determining whether or not the ray travels through a surface is, in the groups opinion, satisfactory. The only consequence being an inefficient use of computational power. The most accurate result of the software would be when using a continuous ray, but the current version of the software does not support that.

6.1.2 Importing 3D-models

An improvement to the current software would be to implement a function to import .stl-files directly. The current software uses a .txt-file with hard coded points that represent the corners of a euro pallet, which limits the functionality of the software. Using .stl-files, the script would be able to handle all 3D-object. Since Solwr’s customers handles a variety of pallets, this feature is recommended for future work. However, the group is very satisfied being able to write this code from the ground up, since it has contributed heavily the group’s research.

6.1.3 Adding points of angled surfaces

The current software does not handle angled surfaces when adding additional points. Adding support for this would be relatively simple and make the software more versatile. The reason for not implementing this functionality in the current version of the software, is due to lack of angled surfaces on today’s pallets. Therefore, this feature is not necessary in the current usage of the software, and the group decided to prioritize other tasks.

6.1.4 Expected results proved to be trustworthy

In the comparison of the expected results from the software, and the actual results from the camera placed on the prototype, were promising. the data was very similar, which indicates that the calculation done in software were valid. By adding other 3D objects, the software can now be more trusted to be correct. However, the software has not been tested using other 3D objects, and therefore needs more testings before used commercially.

6.2 Camera

6.2.1 Cognex

The group reached out to several manufactures and discussed price, delivery, and specific camera features for the intended use case. However, some did not respond. One of which was Cognex. Cognex advertised a promising camera, but the group was not able to get in contact with them, nor did they have necessary details on their website to form a decision. The advertised field of view was promising, and given limited time, the group recommends Solwr to again reach out to Cognex before making a final decision on Zivid One^{+} Large.

6.2.2 USB vs. Ethernet

Initially, Solwr wanted the new cameras to utilize Ethernet for the camera data transfer. The new camera does not. The resolution and price of the Zivid One^{+} Large took precedence. Going forward, Solwr should stay up-to-date on developments from Zivid in case of new releases with all the wanted specifications. This is probable since the Zivid two camera does use Ethernet, but does not have the required FOV.

6.2.3 How the discontinuation of the Intel® RealSense™ L515 might impact the industry

Previously, Solwr had little incentive to consider any other cameras than the L515's. The lower resolution could be compensated for by having more cameras. Now that the L515's are discontinued, it removes the supply of low-cost cameras and generate a higher demand for high-cost industrial cameras. As a consequence, this might accelerate the development of the industrial cameras.

6.2.4 Reduction of cameras and its challenges

In the beginning of the process of finding a viable acquisition, the three static camera solution was considered. The group evaluated if the complexity would be an issue by having movable cameras. Both due to maintenance, added cost, and calibration issues. However, the group learned that a linear axis can be moved with high accuracy, which will lead to the camera not needing to be recalibrated during normal use. The camera itself is built to handle rough environments [34], hence does not need service often, nor will the servo require more service then the other servos that are installed on Sort™. According to Solwr the motor that will be used will require very little maintenance. The saved cost by limiting the solution to two cameras, will reduce the total cost greatly, and it is considered cheaper to implement the linear axis then buying three cameras. By adding movable cameras, distortion of images can be a challenge. In related work the group learned that there are several ways, both in software and hardware to make up for or reduce vibration.

Vibration in the new solution

The linear axis movement looks solid, but the group have not tested the prototype traveling at full speed. With high speed the group estimates vibration that needs to be assessed. The camera and all other components must be rigid enough to dissipate any vibration before taking a photo. Otherwise the point cloud may be distorted. This is explained in more detail in chapter 1.6.1.

Complexity by adding more movement in the new solution

The group did consider including more camera motion to capture even more images using only the two cameras. This was turned down in order to reduce the number of moving parts required. Since identification could be done with the three images it would add more complexity to the system that would require more parts, more controllers and would introduce more points of failure. Due to the robot grippers path it was also decided that the back camera should be stationary in the lower position to remove the possibility of collisions.

6.2.5 Distance between points in the point cloud

The group had a requirement that the point cloud in the new acquisition could not exceed 5mm between two point's, and in the groups calculations the maximum distance between points was 4mm. This strongly suggests that even with only using two Zivid cameras the groups solution will be viable. However, real tests using the Zivid cameras will be the deciding factor.

6.3 Simulator

The point cloud from the simulator shows very clear edges and clearly defined planks for the pallet. This will make identification of the pallets possible with the proposed camera setup. The important step past this is catching faults. Rotation in the wooden blocks, tears or breaks. With the high resolution from the cameras these faults should be just as visible, if not clearer than the current system.
The issues with the simulator is that in it, everything is assumed to be ideal. There is no variation in lighting, the pallets are all uniformly placed and there is no noise in the signal. In the real world, this would not be the case. The pallets are stacked and placed on the in feed by human operators so there will be a slight variation in position across the pallets. Different warehouses will have different lighting conditions and pallets might be wet and cause glare or reflections not picked up in the simulator. In order to determine that the system is robust enough to handle these conditions, more physical testing is required. Webots does offer the option for introducing noise to the signal, but from Solwr's experience it does not translate well into real world applications.

6.4 Prototype

6.4.1 Prototyping using the Intel® RealSense™ L515 camera

The group had to prototype using the Intel® RealSense™ L515 camera, since the expected delivery for the Zivid One$^+$ Large camera was set to 8 weeks if rented, and four weeks if purchased. When the group, together with Solwr, found Zivid to be the optimal supplier, Solwr wanted to rent the cameras. Even with all of the calculations, actual testing was required before investing 10 500€ per camera. Since Zivid's rental cost was 250€, it was not ordered until the group had documented satisfactory result. This is the reason the cameras have yet to arrive at Solwr's warehouse as of the end of the project period. Because the group deemed a prototype to be a high-value asset to the research, it was carried out with the available components, and theoretical calculations were made based on the results.

6.4.2 The height of the linear axis

Solwr provided a linear axis, that fortunately met the projects requirements, while using the Intel® RealSense™ L515 camera. However, if the group were to implement the Zivid One$^+$ Large camera in the second prototype, the group would need a longer linear axis illustrated in figure 71. Today's linear axis enables for 144cm of movement, and the calculated required length using the Zivid cameras is 342cm. The conversion is explained in chapter 2.2.4.

If the proposed solution were to be implemented into Sort™, some of the needed length will be gained from the movement of the pallet. The grabber lifts the pallet from the rest of the stack in order to take photos of the bottom. The amount it is lifted can be adjusted so that the total length of the axis does not have to be as long as in the prototype.

6.4.3 Incremental encoder

The prototype in chapter 4.2 is using a incremental encoder, since this was the components Solwr had at hand. One issue is that when enabling the axis, the encoder sets the current position as the reference also called position zero. Therefore the axis needs to be in the correct position when the axis is enabled, for the axis to have correct top and bottom position.

This issue can be resolved in several ways. One way is to implement a homing sequence, which is explained in 2.3.3. The group would also have to program a homing within in the PLC software, by utilizing the Tc2 MC2 library explained in chapter 4.6.2. The function to utilize is called MC_Home [35]. Another solution would be using a absolute encoder since the absolute encoder knows its position at all times, regardless of the axis status. The absolute encoder is explained in chapter 2.3.3.

6.4.4 Adding a light source

In most point cloud acquisitions, the cameras output is highly dependent on the ambient light. Adding a light source would be preferred to better ensure correct and consistent results. This is especially important since Sort™ will be placed in different warehouses with different light conditions. The amount of light reflected off a surface might vary depending on the ambient light or its dryness and cleanliness. Having lights mounted on or around the camera can partly negate the impact of these factors. An evenly diffused light would be the optimal solution, due to the fact that light can directly impact the robot's ability to sort pallets correctly.

6.4.5 Uncertainty in the new solution

When adding complexity to a system it usually results in a higher system uncertainty. However, most components have defined uncertainty from industrial factory testing. Then, the components purchased can be selected with an accuracy requirement, which can predict the system's uncertainty. The linear axis if used in Sort™ need to be accurate and consistent. A small error over a longer period of time could eventually become significant, and make the image processing unreliable. This can be addressed by having a higher gear or more accurate encoder in order to have a smaller distance traveled per encoder pulse.

6.4.6 Added cost with the new solution

Due to the old cameras' price being considerably lower than the others on the market there will be a price increase with the new solution. The groups goal has therefore been to minimize this increase. An important factor has been to make sure as few as possible cameras are used without adding too much maintenance costs in movable parts. The group is satisfied that the removal of one camera will reduce the cost even with the added axis and rotating motor.

6.4.7 Speed of the new solution

The current system uses ≈24 seconds per pallet, including moving the pallet to the correct stack 3. The current image acquisition uses ≈6 seconds per pallet. The new solution would have to be configured with a high enough motor speed in order to not negatively impact how fast the system is. This can be done when configuring the motor controller 4.5.3. There is still a difference between the motion in the prototype and Sort™. In the groups prototype the linear axis provides all the movement, but in the Sort™ the pallet will be lifted by the robot gripper simultaneously as the camera moves along the linear axis. The lifting height can also be adjusted, and the camera will therefore travel less than the static prototype. This means that the camera get into position faster, making it less likely to negatively impact the sorting time.

The calculated max velocity for the linear axis prototype is 0.4m/s, which is sufficient considering the distance the camera needs to travel can be estimated to be 342cm - the robot gripper movement and approximated to be 70cm. i.e approximated distance if installed on Sort™ will be 272cm traveling distance. The linear axis used in the prototype will then use about 4,6 seconds to change positions. This time can be reduced with more powerful hardware.

6.4.8 Data from the prototype

The data taken from the prototype was stitched together using an existing algorithm. The algorithm was very susceptible to noise and random points in the different point clouds. In the first few test the presence of large surfaces such as the floor or wall, it struggled to orient the pallets due to the other surfaces dominating the data. After the unnecessary data was removed the matching improved significantly. In further projects either accurately measuring each camera position or including a common reference would improve the process of combining the different views.

6.5 Workflow

The group did not deviate much from the original Gantt diagram [N]. The updated Gantt diagram shows that the group completed the prototype faster, and spent more time writing the thesis than expected [M]. The group also added the FOV software some time into the bachelor period. This proved to be a valuable addition seeing that the FOV software has contributed to many of the group's results.

Chapter 7

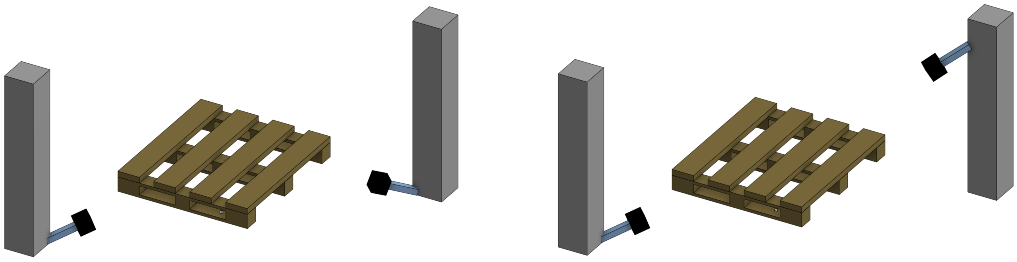
Conclusion and future work

This chapter contains a summary of the group's results and the work needed in order to implement the proposed solution in the next version of Sort™.

7.1 Conclusion

As stated in the project preliminary report, the group's objective was as follows: ”The project is to develop, test and evaluate enhancements to a pallet sorting machine produced for logistics centers across Norway by Currence Robotics. The group will primarily work with designing a new image acquisition solution. This is due to the current cameras being discontinued by the manufacturer. The current setup uses 6 cameras to map each pallet from multiple angles. The camera that is considered to be the replacement is too expensive for this setup, due to competing systems' pricing. The group will therefore attempt to minimize the amount of cameras needed, without compromising the system's accuracy” [R]. Note that the name of the employer has changed since the start of the project, from ”Currence Robotics” to ”Solwr”. The scope has also been redefined to ”redesign the point cloud acquisition for Sort™”.

Given the results they have produced, the bachelor group has concluded that the most promising design is design seven with the Zivid One^{+} Large. The two bottom cameras combine to cover the entire bottom of the pallet. With the Zivid One^{+} Large's resolution, the corners and edges appear sharp and clear. The design includes a linear axis, which moves one of the cameras up and down vertically. The camera would be mounted on a tiltable mount, to be able to take images in both top and bottom position. Since the position changes, the angle would also have to be adjusted. This would be done by a mechanism that tilts the camera. The mechanism is not detailed in this thesis. Due to it only being a prototype, the group prioritized other tasks, since they did not deem it a critical feature. The tilt was adjusted manually using marked positions on the camera mount. One option would be to have a servo motor tilting the camera while moving along the linear axis. Another would be to have a spring loaded mechanism, which would be a cheaper option.



(a) Design seven, initial

(b) Design seven, transformed

Figure 91: Illustration of design seven

The choice between different designs and different cameras is based on several factors. The design is chosen based on its complexity, number of required cameras and its mechanical feasibility. The camera is chosen based on its FOV, resolution, price and delivery time.

The proposed solution complies with all of Solwr's criteria and would not add too much complexity to Sort™. This includes generating a point cloud dense enough so that points within it are no more than 5mm apart. The calculated distance using the chosen design and camera was 4mm between points at the surface farthest away from the camera.

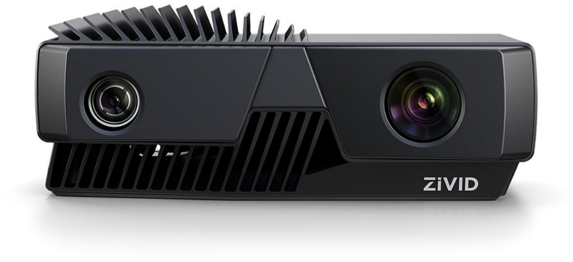


Figure 92: Zivid camera

The prototype data introduced more noise and uncertainty compared to the simulated and calculated data. The edges and corners were still clear. However, the group now had to filter out the background. This proved to be quite hard, and some noise still remained. Solwr's engineers have already made software that handles this issue. However, the group wanted to produce their own software, to learn and to provide their own solution.

Based on the results from the FOV software, the simulation and the physical prototype, the group concluded that their results are satisfactory, and that they have completed their objective.

Before implementing the proposed camera setup, it should be tested using the new Zivid cameras. The cameras have yet to be delivered, due to a long delivery time frame. However, testing the new cameras is necessary in order to verify the group's results. In order to do this, Solwr can adjust the prototype as described in section 4.4. Some adjustments to the measurements are to be expected.

7.2 Future work

A light source should be added to the camera setup. As explained in section 6.4.4, this would result in more consistent point clouds. The project preliminary report did mention this was one of the things the group would look at [R], but the group did not have sufficient time to include it in the prototyping.

The linear axis of the prototype would need to be replaced. Given the longer distances required when using the new cameras, the linear axis needs to be longer in order for the prototype to work. The chosen cameras have a narrower field of view. This means they need to be farther from the pallet. Seeing that the cameras are angled, this means they also need to be placed higher and lower relative to the pallet.

Another important step that remains is to design the mechanism that will change the angle of the camera. The simplest one would be a servo that rotates the camera mount while the linear axis is moving the camera, but mechanical solutions using springs were discussed with Solwr during one of the progress meetings.

It will also be important to test different types of- and damaged pallets with the physical prototype to be certain that the pallets can be correctly classified.

In order to verify the time it takes for the proposed solution to move the camera, Solwr would need to adjust the speed of the motor. As it stands, it travels at a speed of 0.1m/s. This is too slow for the actual Sort™. When increasing the velocity, a problem to consider is distortion caused by vibration. Chapter 1.6.1 includes a thesis on several methods to counteract this. Too much residual vibration could make the point cloud acquisition unreliable.

Should the linear axis prove to be too complex, Solwr should consider using the variant of design seven 55. It removes the linear axis, but adds a third camera.

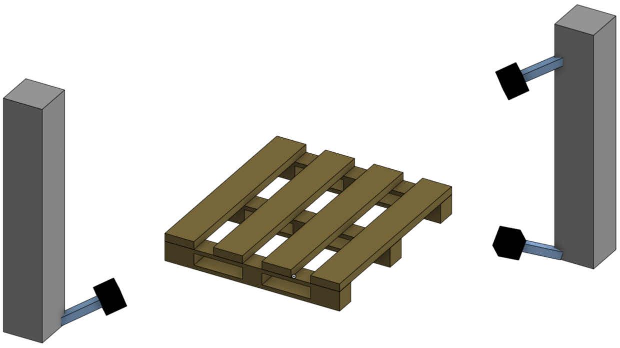


Figure 93: Illustration of variant

There is also the possibility of moving forward with more research independently from Solwr in the point cloud analysis. The group's scope did not cover the pallet identification and classification, but future thesis' could. Data from the simulation and software could provide useful insight, and can be a useful resource in other thesis'. A possible topic would be to use the data in order to train a neural net to differentiate between pallets.

The software can be upgraded with the functionality of importing .stl-files. This would make it more versatile and more useful in different applications.

References

[1] Dag Håkon Hanssen, Programmerbare logiske styringer. Bergen: Fagbokforlaget, 2015.

[2] Currence Robotics, "Sort™," 2021. Website link, date accessed 25-January-2022.

[3] nlpool, "Nlp plastic full pallet." Website link, date accessed 02-February-2022.

[4] Allen, The British Industrial Revolution in Global Perspective: How Commerce Rather than Science Caused The Industrial Revolution and Modern Economic Growth. Bloomington: Indiana University Press, 2006.

[5] Rosheim, Robot Evolution: The Development of Anthrobotics. Hoboken: Wiley-Interscience, 1994.

[6] Solwr.com, "Solwr delivers tomorrow's logtech solutions." Website link, date accessed 17-May-2022.

[7] I. Norge, "Finansiering av oppstart." Website link, date accessed 19-May-2022.

[8] Intel.com, "Are the intel® realsense™ lidar, facial authentication and tracking product families being discontinued?." Website link, date accessed 17-May-2022.

[9] Framos.com, "Intel® realsense™ discontinues lidar, fa and tracking product lines, focuses on stereo vision." Website link, date accessed 17-May-2022.

[10] C. D. Rodin, "Applications of high-precision optical imaging systems for small unmanned aerial systems in maritime environments," 2019.

[11] E. S. Eissa, "Image space coverage model for deployment of multi-camera networks." Website link, date accessed 5-May-2022.

[12] Haines, Hanrahan, Cook, Arvo, Kirk, Heckbert, An Introduction to Ray Tracing (The Morgan Kaufmann Series in Computer Graphics) 1st Edition. Burlington: Morgan Kaufmann Publishers, 1989.

[13] BMC.com, "Asynchronous programming: A beginner's guide." Website link, date accessed 9-May-2022.

[14] Adams, Essex, Calculus - A Complete Course 7th ed. Toronto: Pearson Canada, 2009.

[15] Hartshorne, Euclid and Beyond. New York: Springer New York, 2005.

[16] Kuipers, Quaternions and rotation sequences : a primer with applications to orbits, aerospace, and virtual reality. New Jersey: Princeton University Press, 2002.

[17] ZIVID, "Basic 3d machine vision techniques and principles." Website link, date accessed 12-May-2022.

[18] Ma, Soatto, Košecká, Sastry, An Invitaion to 3-D Vision. New York: Springer, 2004.

[19] plcopen.org, "What is plcopen." Website link, date accessed 8-May-2022.

[20] Y. Chen, H. Chen, M. Zhang, and Y. Li, "The relevant research of coe protocol in ethercat industrial ethernet," 10 2010.

[21] AP associated pallets, "Protecting pallets from forklift damage," 2018. Website link, date accessed 25-January-2022.

[22] Advice monkey, Sirdion, "Choosing between wood and plastic pallets," 2014. Website link, date accessed 25-January-2022.

[23] Intel, "Intel realsense l515." Website link, date accessed 02-February-2022.

[24] Power.no, "Raspberry pi 4 model b 8gb." Website link, date accessed 12-May-2022.

[25] Zivid.com, "Data cables - zivid one^{+}." Website link, date accessed 13-May-2022.

[26] cognex.com, "3d-a5000 series area scan 3d camera." Website link, date accessed 03-April-2022.

[27] Skatteetaten, "Slik fungerer mva." Website link, date accessed 13-May-2022.

[28] Webots, "Lidar." Website link, date accessed 19-May-2022.

[29] Webots, "Rangefinder." Website link, date accessed 19-May-2022.

[30] Beckhoff.com, "Plc library: Tc2_mc2 overview." Website link, date accessed 14-May-2022.

[31] Beckhoff, "Axis_ref." Website link, date accessed 11-May-2022.

[32] Beckhoff, "Nc settings." Website link, date accessed 11-May-2022.

[33] open3D, "Multiway registration." Website link, date accessed 19-April-2022.

[34] Zivid, "Zivid one+ main page." Website link, date accessed 19-January-2022.

[35] Beckhoff.com, "Plc library tc2_mc2 mc_home." Website link, date accessed 14-May-2022.

Chapter 8

Appendix

Appendix A

The group's how-to-guide for programming drives, PLC and HMI

A.1 Setting static IPv4 for motor controller

Setting static IPv4 for motor controller

A how-to-guide to set a static IPv4 Ethernet address on motor controller.

There are several ways for the drive to obtain an IP, according to the motor controllers quick guide in appendix [F]. A DHCP server, for example, can automatically assign an IP address. Another option is to use EtherCAT to manually set it, which was the technique used using PlugDrive Studio 1. When the guide is completed, keep following instructions in chapter [4.5.2].

To manually set an IP address connect a PC to port X7, which is the EtherCAT input, and follow the steps below:

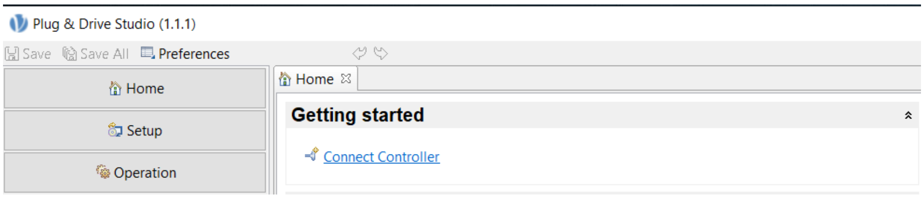
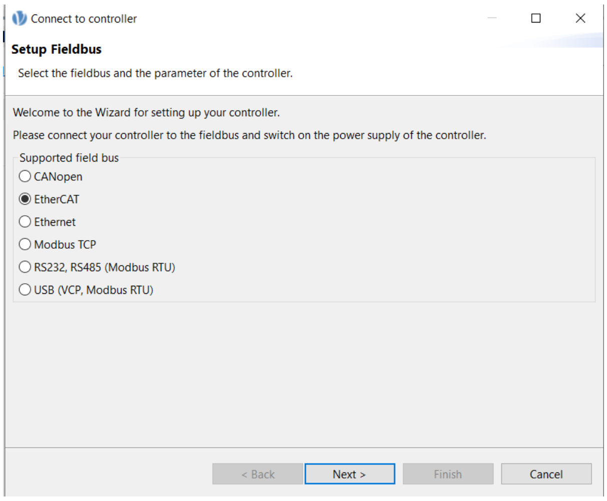


Figure 94: Connect to drive, by pressing ”Connect controller”



Setup Fieldbus

Select the fieldbus and the parameter of the controller.

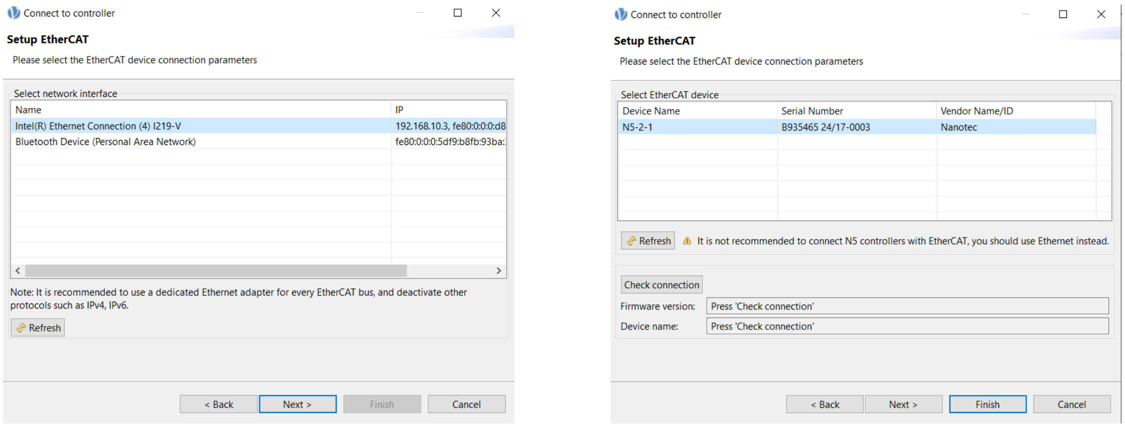
Welcome to the Wizard for setting up your controller.

Please connect your controller to the fieldbus and switch on the power supply of the controller.

Supported field bus

○ CANopen
● EtherCAT
○ Ethernet
○ Modbus TCP
○ RS232, RS485 (Modbus RTU)
○ USB (VCP, Modbus RTU)

Figure 95: Chose the communication protocol ”EtherCAT” and continue by pressing ”Next”



Connect to controller

Setup EtherCAT

Please select the EtherCAT device connection parameters

Select network interface

Name	IP
Intel(R) Ethernet Connection (4) I219-V	192.168.10.3, fe80:0:0:0:d8
Bluetooth Device (Personal Area Network)	fe80:0:0:0:5df9:b8fb:93ba:

Note: It is recommended to use a dedicated Ethernet adapter for every EtherCAT bus, and deactivate other protocols such as IPv4, IPv6.

(a) Select the network interface with the corresponding IP of the computer being used.

Connect to controller

Setup EtherCAT

Please select the EtherCAT device connection parameters

Select EtherCAT device

Device Name	Serial Number	Vendor Name/ID
N5-2-1	B935465 24/17-0003	Nanotec

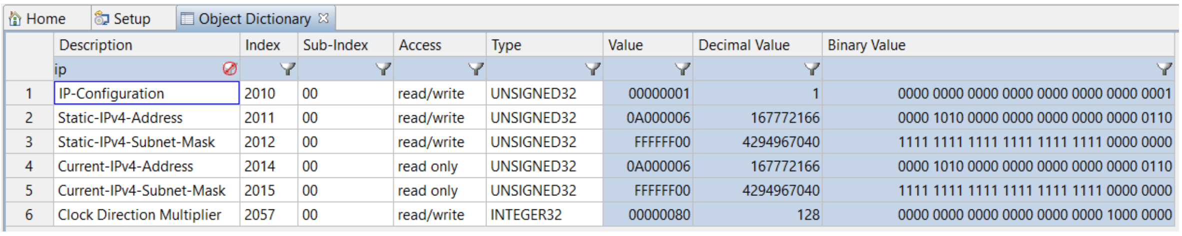
⚠ It is not recommended to connect N5 controllers with EtherCAT, you should use Ethernet instead.

Firmware version: Press 'Check connection'

Device name: Press 'Check connection'

(b) Select the drive in use, and check connection.

Figure 96: Selecting of network interface, and EtherCAT device



	Description	Index	Sub-Index	Access	Type	Value	Decimal Value	Binary Value
	ip							
1	IP-Configuration	2010	00	read/write	UNSIGNED32	00000001	1	0000 0000 0000 0000 0000 0000 0000 0001
2	Static-IPv4-Address	2011	00	read/write	UNSIGNED32	0A000006	167772166	0000 1010 0000 0000 0000 0000 0000 0110
3	Static-IPv4-Subnet-Mask	2012	00	read/write	UNSIGNED32	FFFFFF00	4294967040	1111 1111 1111 1111 1111 1111 0000 0000
4	Current-IPv4-Address	2014	00	read only	UNSIGNED32	0A000006	167772166	0000 1010 0000 0000 0000 0000 0000 0110
5	Current-IPv4-Subnet-Mask	2015	00	read only	UNSIGNED32	FFFFFF00	4294967040	1111 1111 1111 1111 1111 1111 0000 0000
6	Clock Direction Multiplier	2057	00	read/write	INTEGER32	00000080	128	0000 0000 0000 0000 0000 0000 1000 0000

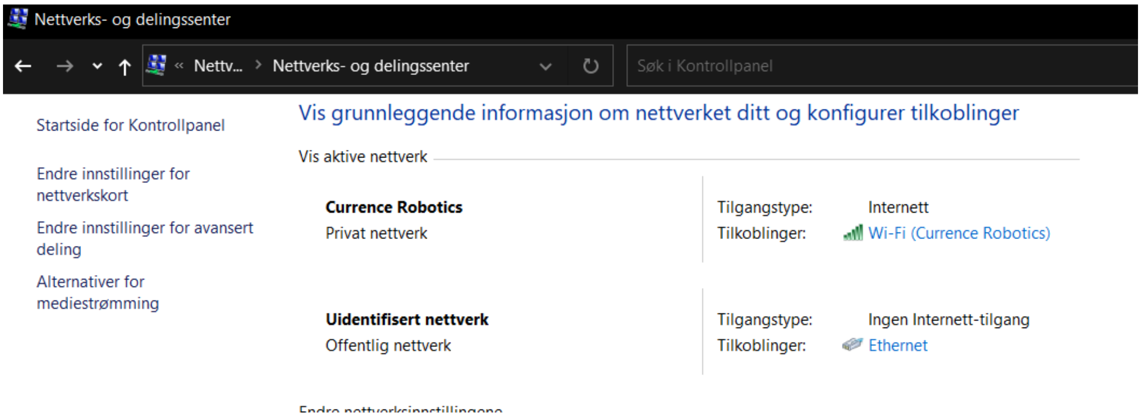
Figure 97: Changing or reading the ”Static-IPv4-Address” from the ”value” column. The IP is displayed in HEX decimal.

A large list will appear after selecting ”Object Dictionary” from the main menu in Plug-Drive Studio 1. Then, in the search area, type ”ip,” and the list will be reduced down to the list shown in Figure 96. The second line's ”Static-IPv4-Address” can now be changed/read. The IP address is displayed in HEX decimal in the ”value” column and must be converted bit by bit. The static IP address is 10.0.0.6 since 0A=10, 00=0, 00=0, 06=6.

A.2 Setting static IPv4 for a computer

Setting static IPv4 for a computer

A how-to-guide to set a static IPv4 Ethernet address on a PC. In this example it is sat to 10.0.0.3. When the guide is completed, keep following instructions in chapter [4.5.3].



Nettverks- og delingssenter

Nettv... › Nettverks- og delingssenter

Søk i Kontrollpanel

Startside for Kontrollpanel

Endre innstillinger for nettverkskort

Endre innstillinger for avansert deling

Alternativer for mediestrømming

Vis grunnleggende informasjon om nettverket ditt og konfigurer tilkoblinger

Vis aktive nettverk

Currence Robotics
Privat nettverk

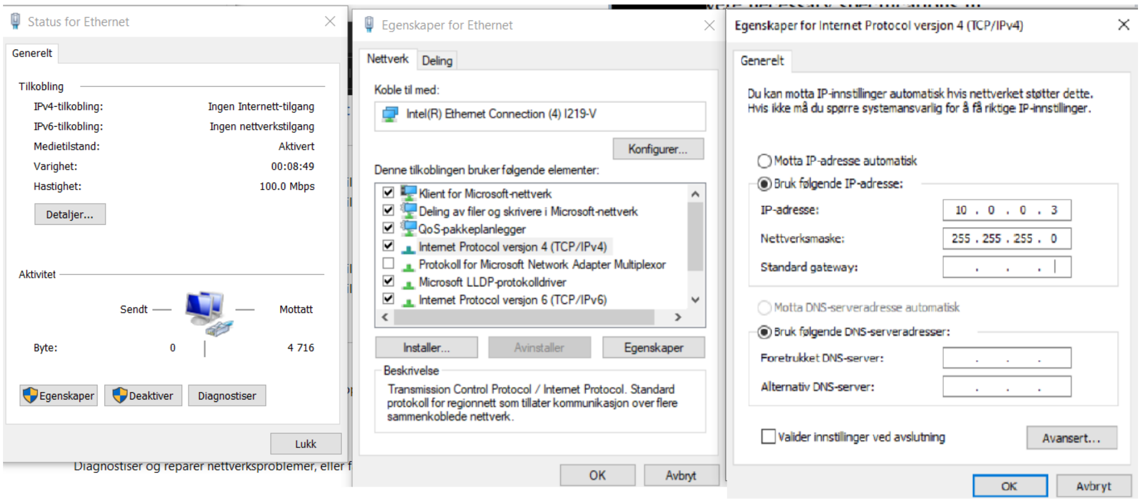
Tilgangstype: Internett
Tilkoblinger: Wi-Fi (Currence Robotics)

Uidentifisert nettverk
Offentlig nettverk

Tilgangstype: Ingen Internett-tilgang
Tilkoblinger: Ethernet

Endre nettverksinnstillingene

(a) Navigate to ”Nettverks- of delingssenter”, and select ”Tilkoblinger: Ethernet”



Status for Ethernet

Generelt

Tilkobling

IPv4-tilkobling:	Ingen Internett-tilgang
IPv6-tilkobling:	Ingen nettverkstilgang
Medietilstand:	Aktivert
Varighet:	00:08:49
Hastighet:	100.0 Mbps

Detaljer...

Aktivitet

	Sendt	Mottatt
Byte:	0	4 716

Egenskaper Deaktiver Diagnostiser

Lukk

Diagnostiser og reparer nettverksproblemer, eller f

Egenskaper for Ethernet

Nettverk Deling

Koble til med:

Intel(R) Ethernet Connection (4) I219-V

Konfigurer...

Denne tilkoblingen bruker følgende elementer:

- ☑ Klient for Microsoft-nettverk
- ☑ Deling av filer og skrivere i Microsoft-nettverk
- ☑ QoS-pakkeplanlegger
- ☑ Internet Protocol versjon 4 (TCP/IPv4)
- ☐ Protokoll for Microsoft Network Adapter Multiplexor
- ☑ Microsoft LLDP-protokolldriver
- ☑ Internet Protocol versjon 6 (TCP/IPv6)

Installer... Avinstaller Egenskaper

Beskrivelse

Transmission Control Protocol / Internet Protocol. Standard protokoll for regionnett som tillater kommunikasjon over flere sammenkoblede nettverk.

OK Avbryt

Egenskaper for Internet Protocol versjon 4 (TCP/IPv4)

Generelt

Du kan motta IP-innstillinger automatisk hvis nettverket støtter dette. Hvis ikke må du spørre systemansvarlig for å få riktige IP-innstillinger.

○ Motta IP-adresse automatisk

● Bruk følgende IP-adresse:

IP-adresse:	10 . 0 . 0 . 3
Nettverksmaske:	255 . 255 . 255 . 0
Standard gateway:	. . .

○ Motta DNS-serveradresse automatisk

● Bruk følgende DNS-serveradresser:

Foretrukket DNS-server:	. . .
Alternativ DNS-server:	. . .

☐ Valider innstillinger ved avslutning

Avansert...

OK Avbryt

(b) In ”status for Ethernet”, double-click ”Egenskaper”. In the next sub-menu select ”Internet Protocol Version 4(TCP/IPv4)”, and finally change the IP address.

Figure 98: Changing the computers internal Ethernet IP to 10.0.0.3, through ”Nettverks- og delingssenter”.

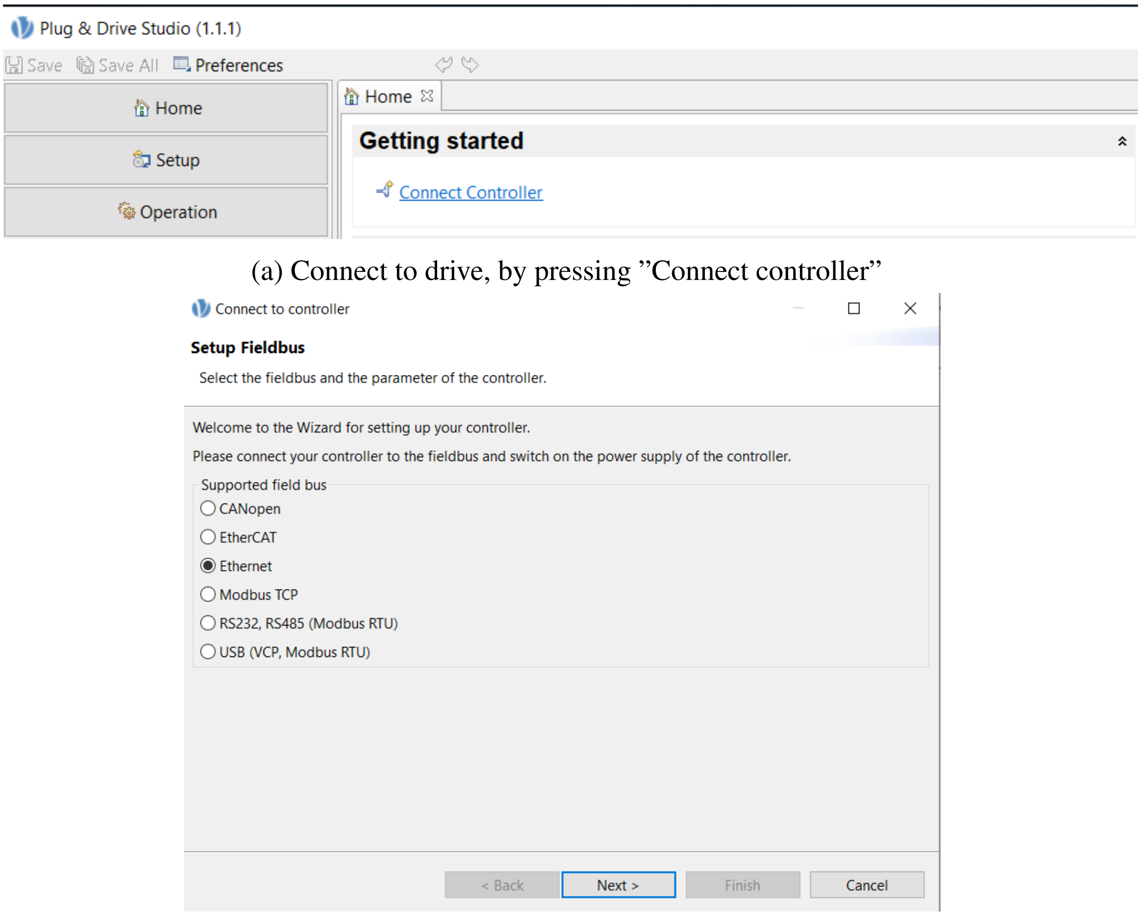
Navigate to ”Nettverks- og delingssenter” and follow the submenus in figure 98 to alter the computer’s static IPv4 address. Since the drive’s static IP is now 10.0.0.6, the computer’s IP must be 10.0.0.X, with the final digit in the range 0-5 or 7-255. Figure 98b’s IP address is an compatible example: 10.0.0.3.

A.3 Motor setup though Ethernet in PlugDrive Studio 1

Motor setup though Ethernet in PlugDrive Studio 1

A how-to-guide to setup the motor and motor controller using the Nanotec's software PlugDrive Studio 1.

The Ethernet Port X1 could now be used to establish a link between the PC and the drive because the IP had been manually specified. Some motor requirements were required for the setup, which were found in the motor datasheet in appendix [D]. The electrical peak and rated current limits, as well as the number of poles, were necessary specifications in order to avoid damaging or breaking the motor controller, during the automatic setup. Follow the guide below to connect, and preform the setup in PlugDrive Studio 1. When the guide is completed, keep following instructions in chapter [4.6].



(a) Connect to drive, by pressing ”Connect controller”

Connect to controller

Setup Fieldbus

Select the fieldbus and the parameter of the controller.

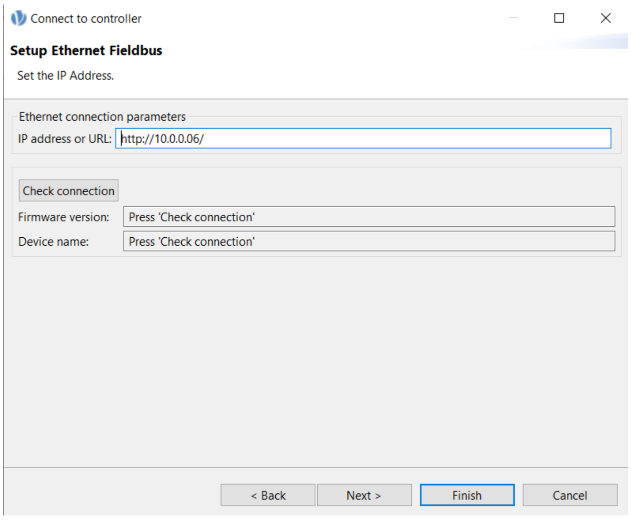
Welcome to the Wizard for setting up your controller.

Please connect your controller to the fieldbus and switch on the power supply of the controller.

Supported field bus
- ○ CANopen
- ○ EtherCAT
- ● Ethernet
- ○ Modbus TCP
- ○ RS232, RS485 (Modbus RTU)
- ○ USB (VCP, Modbus RTU)

(b) Chose the communication protocol ”Ethernet”

Figure 99: Connect and select ”Ethernet” as the communication protocol of choice, and continue by pressing ”Next”



Connect to controller

Setup Ethernet Fieldbus

Set the IP Address.

Ethernet connection parameters

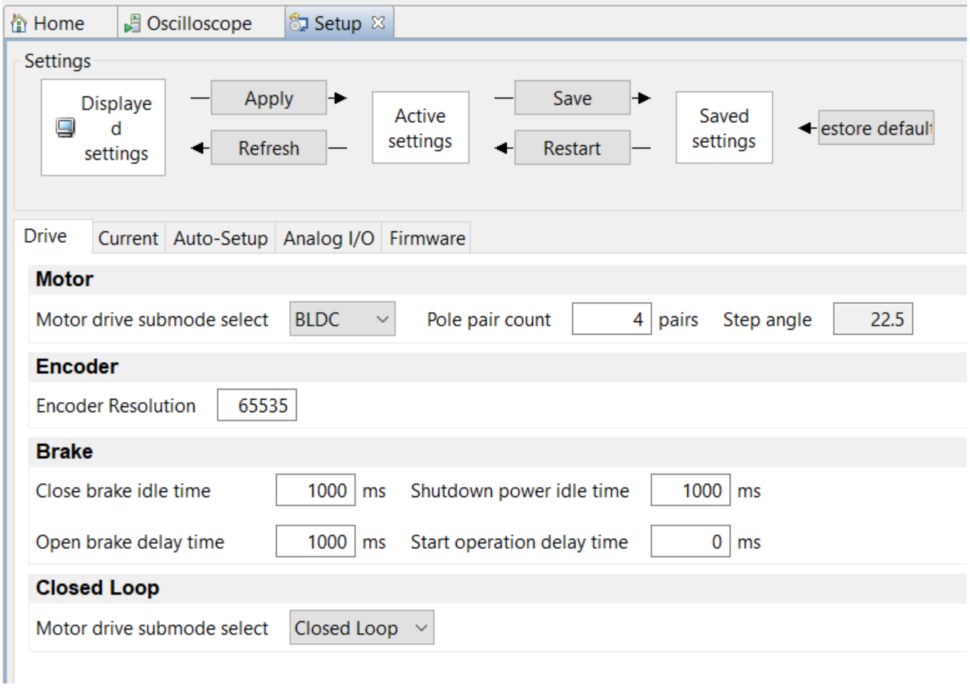
IP address or URL: http://10.0.0.06/

Check connection

Firmware version: Press 'Check connection'

Device name: Press 'Check connection'

Figure 100: Type in the drives ”Static-IPv4-Address” and check connection



Motor

Motor drive submode select: BLDC   Pole pair count: 4 pairs   Step angle: 22.5

Encoder

Encoder Resolution: 65535

Brake

Close brake idle time: 1000 ms   Shutdown power idle time: 1000 ms

Open brake delay time: 1000 ms   Start operation delay time: 0 ms

Closed Loop

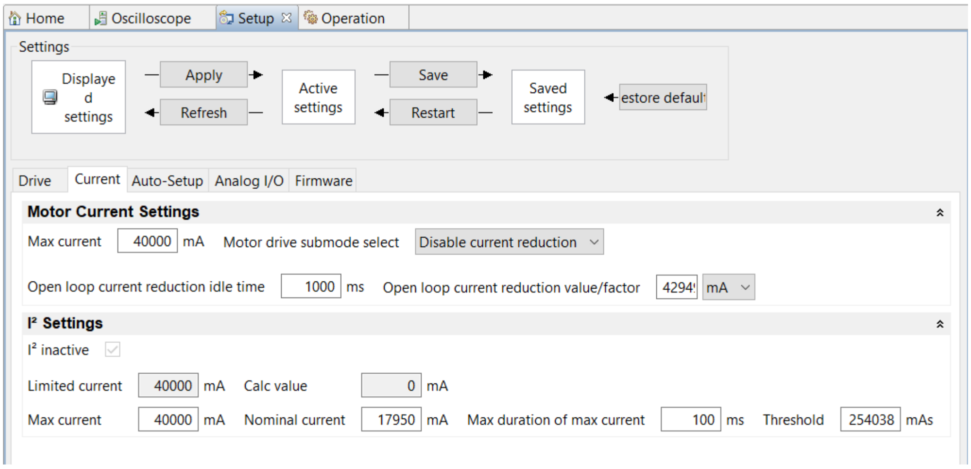
Motor drive submode select: Closed Loop

Figure 101: Set the ”motor drive submode select” to ”BLDC”, and ”pole pair count” to ”4”.

In this set up process the first parameters to change are within ”setup” and then within the tab called ”Drive” like in figure 101. Parameters for Encoder and Brake will be filled in during auto setup. However, choose to use a closed loop, since the motor is getting a feedback on it’s position. Thereafter fill in the motor parameters found in appendix [D]:

Motor

- Motor drive submode select = BLDC
- Pole pair count = 4
  (The step angle will adjust depending on the number of pole pair)



Motor Current Settings

Max current 40000 mA Motor drive submode select Disable current reduction

Open loop current reduction idle time 1000 ms Open loop current reduction value/factor 4294 mA

I² Settings

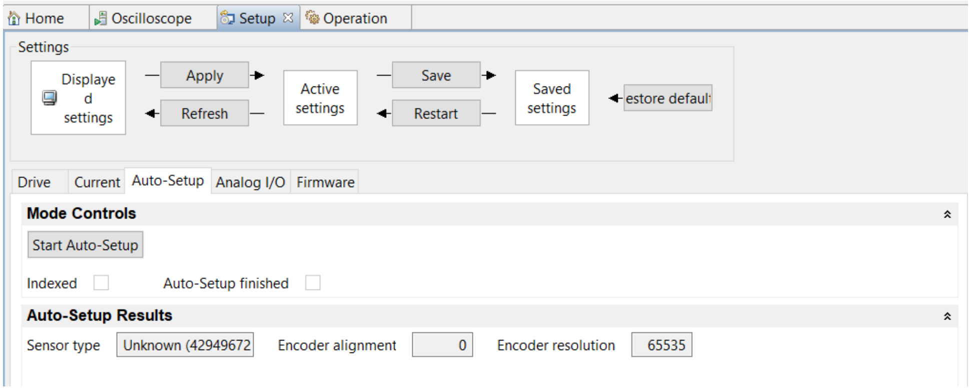
I² inactive ☐

Limited current 40000 mA Calc value 0 mA

Max current 40000 mA Nominal current 17950 mA Max duration of max current 100 ms Threshold 254038 mAs

Figure 102: Change the parameters for ”Max current” and ”Nominal current”

The second parameters to is located in the tab ”Current”, seen in figure 102. Parameters to be filled in are ”Max current” and ”Nominal current”, which is found in appendix [D]. However, the the ”Max current” has a real value of 53.85A, but the drive can max output 40A.



Mode Controls

Indexed ☐ Auto-Setup finished ☐

Auto-Setup Results

Sensor type Unknown (42949672 Encoder alignment 0 Encoder resolution 65535

Figure 103: Press ”Start Auto Setup”, while the motor is lying still and unloaded

The next step is within the tab ”Auto setup”, seen in figure 103. Leave the motor without load, and for instance laying on a table while the auto setup is running. Now the auto setup will start rotating the motor, and collecting the remaining parameters.

A.4 Linking motor controller and axis in TwinCAT

A.5 Connecting and linking a drive to an axis in TwinCAT

A how-to-guide to connect to the Nanotec's N5-2-1 drive from the PLC, and added the drive as an device. Thereafter a axis will be added automatically, but not linked. When the guide is completed, keep following instructions in chapter [4.6.2].

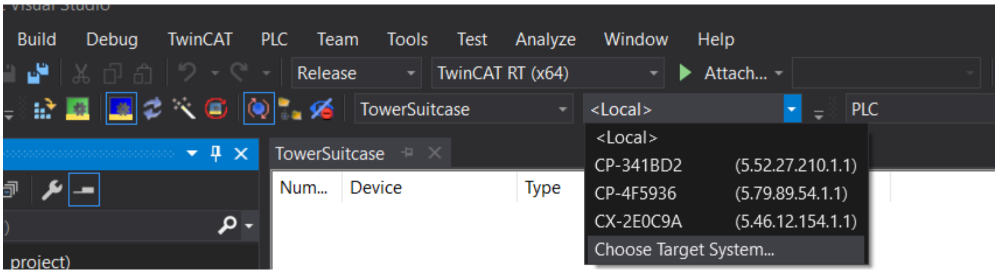
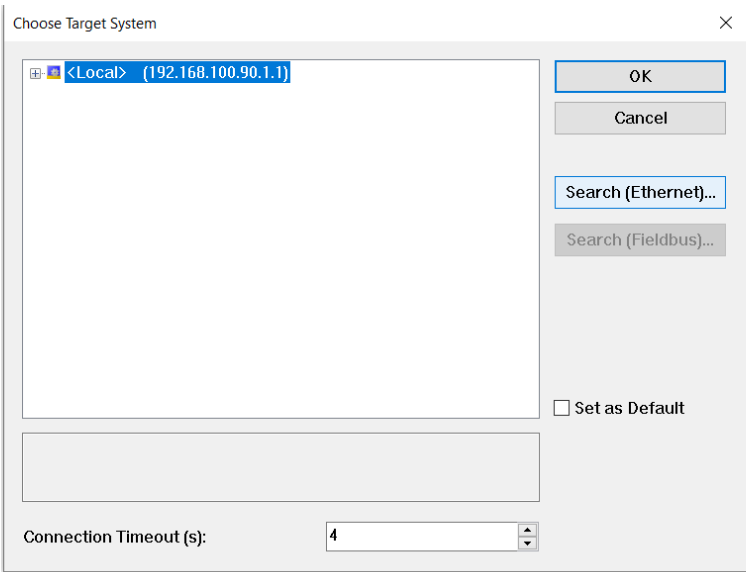


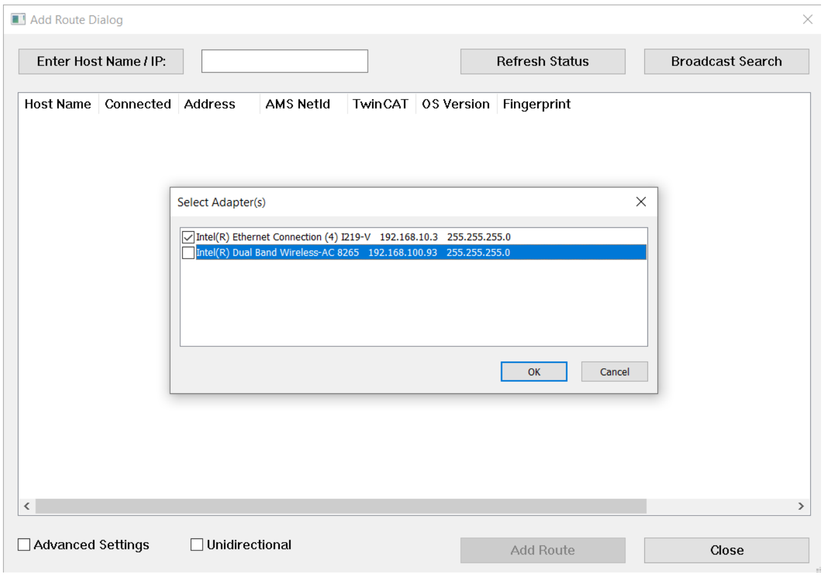
Figure 104: Press "Choose Target System"



<Local> (192.168.100.90.1.1)

Connection Timeout (s): 4

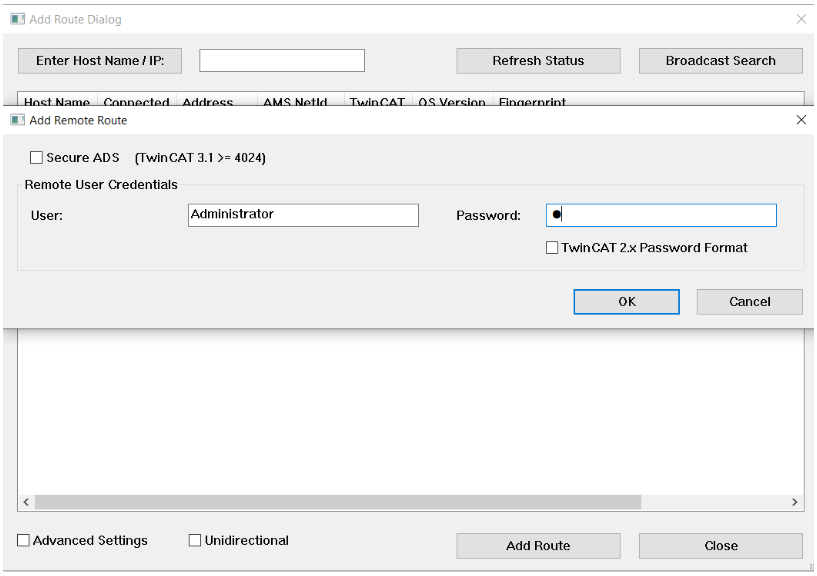
Figure 105: Press "Search (Ethernet)..."



☑ Intel(R) Ethernet Connection (4) I219-V 192.168.10.3 255.255.255.0
□ Intel(R) Dual Band Wireless-AC 8265 192.168.100.93 255.255.255.0

□ Advanced Settings
□ Unidirectional

Figure 106: First press ”Broadcast Search”, and in ”Select Adapters” select the adapter with the computers IPv4 address.



□ Secure ADS (TwinCAT 3.1 >= 4024)

User: Administrator

Password: ●

□ TwinCAT 2.x Password Format

□ Advanced Settings
□ Unidirectional

Figure 107: Nanotec’s drive default password = 1

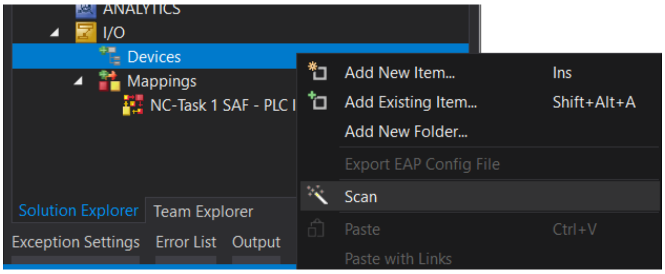


Figure 108: Right click on devices in solution explorer, and press ”Scan”

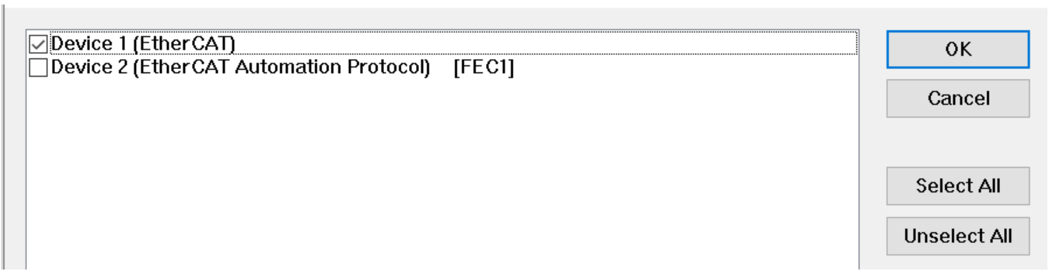
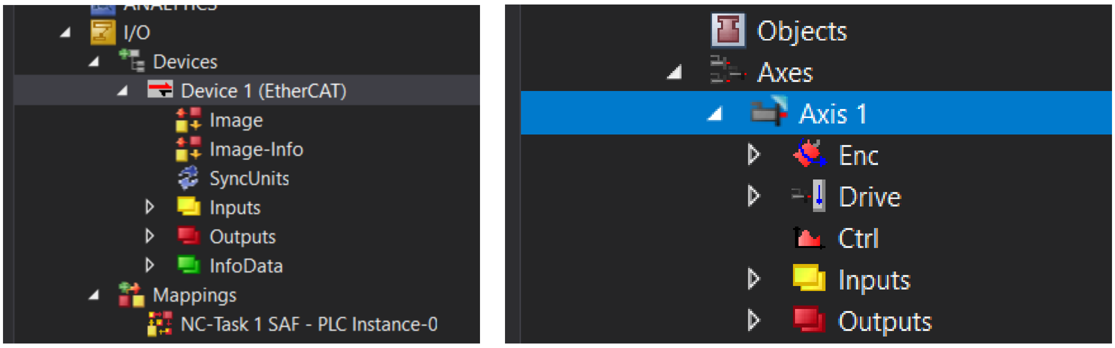

☑ Device 1 (EtherCAT)
□ Device 2 (EtherCAT Automation Protocol) [FEC1]

Figure 109: Select ”device 1 (EtherCAT)” and press ”OK”



(a) The N5-2-1 drive has been added as Device 1(EtherCAT)

(b) A axis will automatically be added

Figure 110: Both the devices and a axis are now added

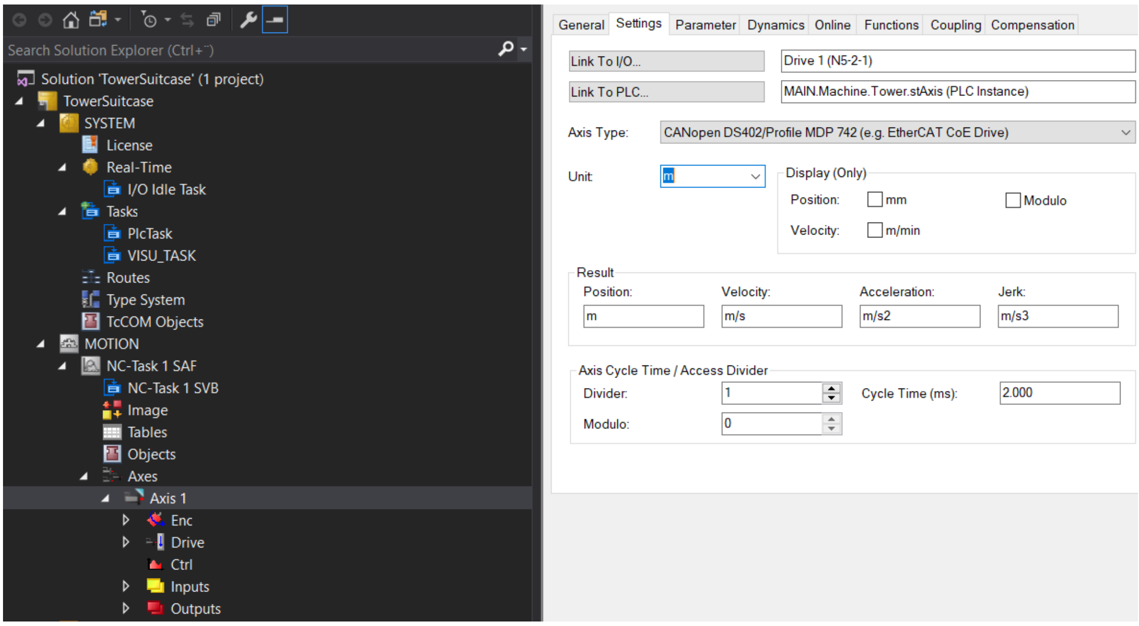
A.6 Axis parameter settings

Axis parameter settings

How-to-guide to fill in paramters within the axis and encoder in the soultion explorer in TwinCAT.

Several settings in the axis and encoder have to be filled in to ensure that the linear axis travels as predicted. The group found inspiration in a Backhoff guide [32], and discovered that majority of the parameters were set correctly by default. The calculations for maximum velocity and scaling factor, however, did not match the needs of the group. The team derived new formulas and calculated new maximum velocity and scaling factor values. When the guide is completed, keep following instructions in chapter [4.7].

Select the unit for the axis distance traveled



Link To I/O... Drive 1 (N5-2-1)

Link To PLC... MAIN.Machine.Tower.stAxis (PLC Instance)

Axis Type: CANopen DS402/Profile MDP 742 (e.g. EtherCAT CoE Drive)

Unit: m

Display (Only)

Position: □ mm □ Modulo

Velocity: □ m/min

Result

Position:	Velocity:	Acceleration:	Jerk:
m	m/s	m/s2	m/s3

Axis Cycle Time / Access Divider

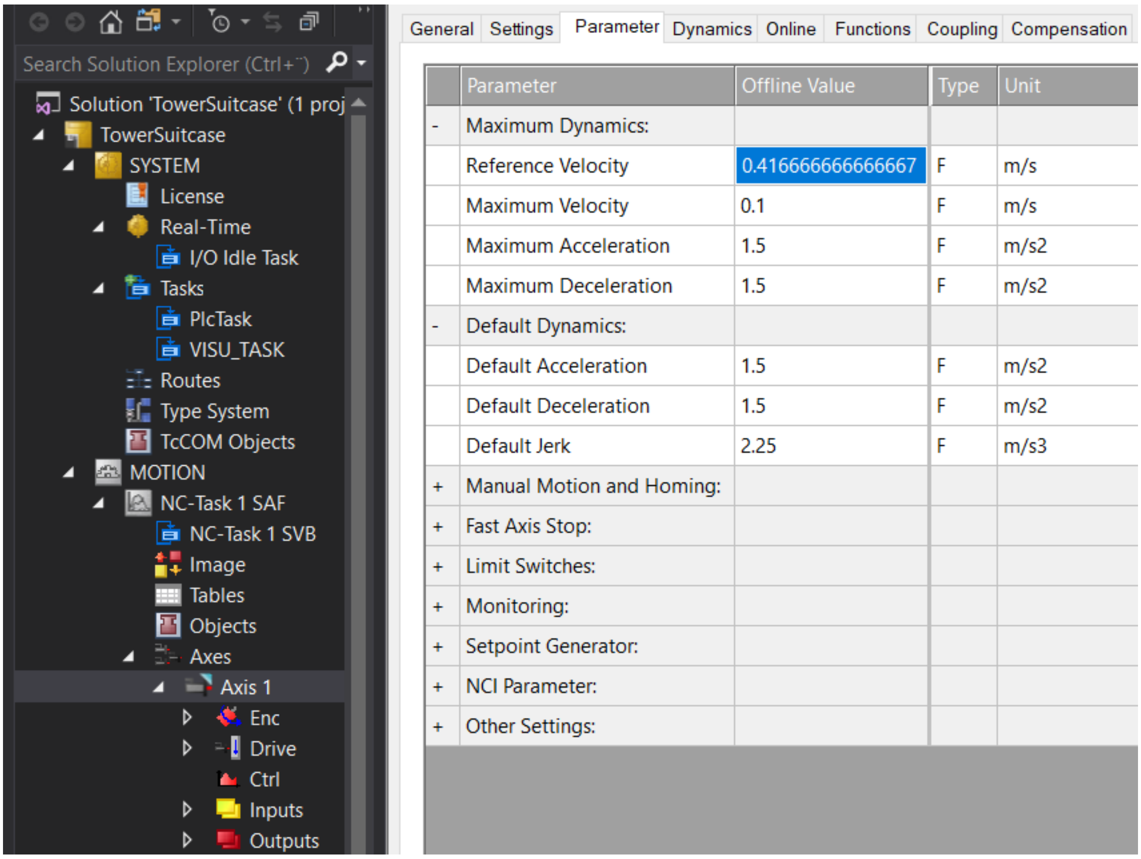
Divider: 1

Cycle Time (ms): 2.000

Modulo: 0

Figure 111: Fill in the wanted unit for the axis movement. The group choose the unit ”m” for meter.

Theoretical Max Velocity calculations



	Parameter	Offline Value	Type	Unit
-	Maximum Dynamics:			
	Reference Velocity	0.416666666666667	F	m/s
	Maximum Velocity	0.1	F	m/s
	Maximum Acceleration	1.5	F	m/s2
	Maximum Deceleration	1.5	F	m/s2
-	Default Dynamics:			
	Default Acceleration	1.5	F	m/s2
	Default Deceleration	1.5	F	m/s2
	Default Jerk	2.25	F	m/s3
+	Manual Motion and Homing:			
+	Fast Axis Stop:			
+	Limit Switches:			
+	Monitoring:			
+	Setpoint Generator:			
+	NCI Parameter:			
+	Other Settings:			

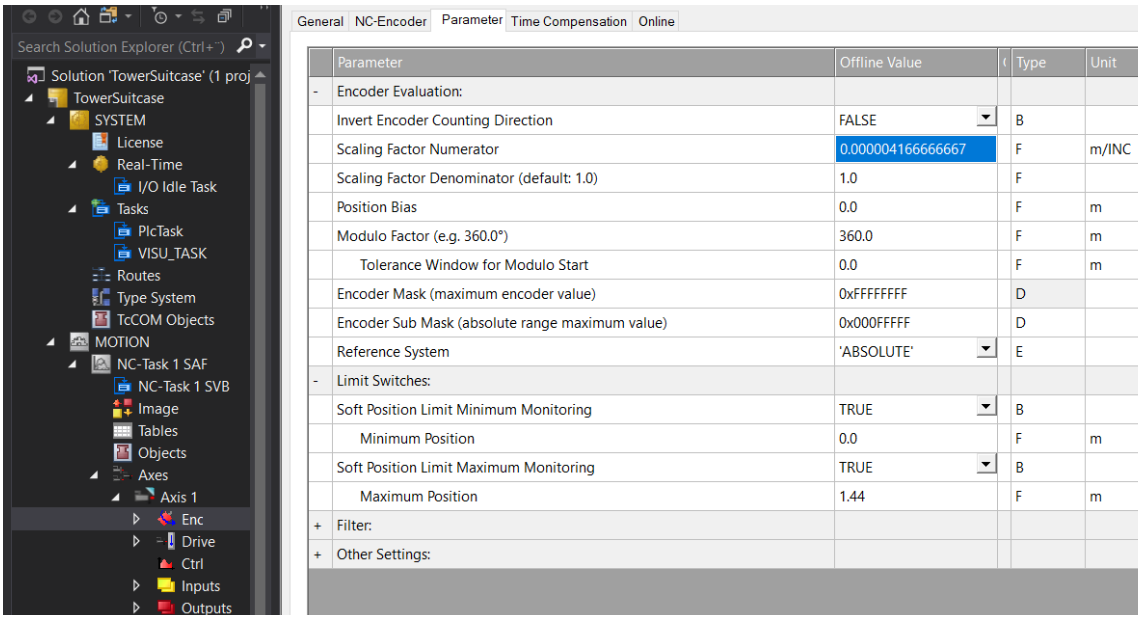
Figure 112: Calculate the theoretical max velocity of the motor, and fill it in under ”Reference velocity”. Fill in the actual wanted max velocity in under ”Maximum velocity”

”Maximum velocity” is the maximum allowed speed for the axis, and the group choose 0.1 m/s for this prototype.

$$Distance\ \ motor\ \ rotation = \frac{Distance\ \ axis\ \ rotation}{Motor\ \ gear\ \ ratio} = \frac{0.025m}{3} = 8.333 \cdot 10^{-3}\ \ [m] \quad (29)$$

$$Max\ \ motor\ \ velocity = \frac{Motor_{RPM} \cdot Distance\ \ motor\ \ rotation}{Seconds\ \ in\ \ one\ \ minute} = \frac{3000 \cdot 8.33 \cdot 10^{-3}}{60s} = 0.416\ [m/s] \quad (30)$$

Scaling factor calculations



- Solution 'TowerSuitcase' (1 proj
  - TowerSuitcase
    - SYSTEM
      - License
      - Real-Time
        - I/O Idle Task
      - Tasks
        - PlcTask
        - VISU_TASK
      - Routes
      - Type System
      - TcCOM Objects
    - MOTION
      - NC-Task 1 SAF
        - NC-Task 1 SVB
        - Image
        - Tables
        - Objects
        - Axes
          - Axis 1
            - Enc
            - Drive
            - Ctrl
            - Inputs
            - Outputs

	Parameter	Offline Value	(	Type	Unit
-	Encoder Evaluation:				
	Invert Encoder Counting Direction	FALSE		B	
	Scaling Factor Numerator	0.000004166666667		F	m/INC
	Scaling Factor Denominator (default: 1.0)	1.0		F	
	Position Bias	0.0		F	m
	Modulo Factor (e.g. 360.0°)	360.0		F	m
	Tolerance Window for Modulo Start	0.0		F	m
	Encoder Mask (maximum encoder value)	0xFFFFFFFF		D	
	Encoder Sub Mask (absolute range maximum value)	0x000FFFFF		D	
	Reference System	'ABSOLUTE'		E	
-	Limit Switches:				
	Soft Position Limit Minimum Monitoring	TRUE		B	
	Minimum Position	0.0		F	m
	Soft Position Limit Maximum Monitoring	TRUE		B	
	Maximum Position	1.44		F	m
+	Filter:				
+	Other Settings:				

Figure 113: Calculate the scaling factor and fill it in under "Scaling Factor Numerator"

$$Scaling\ \ factor = \frac{Distance\ \ motor\ \ rotation}{Pulses\ \ per\ \ rotation} = \frac{8.33 \cdot 10^{-3}}{2000} = 4.166 \cdot 10^{-6}\ \ [m/pulse] \tag{31}$$

Appendix B
Intel RealSense L515 Datasheet

Intel® RealSense™ LiDAR Camera L515

Datasheet

Intel® RealSense™ LiDAR Camera L515

Revision 003

January 2021

Revision: 003

Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

Contents

Figures

Tables

Revision History

Revision Number	Description	Revision Date
001	Initial release	December 2019
002	•Section 1. Description and Features •Section 3.1.1. Camera Accuracy Health •Section 3.4. Depth Start Point (Ground Zero Reference) •Section 7.1.4. Embedded Laser Information •Table 3-2. Depth Camera Controls •Table 3-4. Depth Quality Metrics •Table 3-5. Depth Quality Specification •Table 3-6. Depth Start Point •Table 3-7. Image Formats •Table 4-2. Color Camera Properties •Table 4-5. Storage and Operating Conditions •Figure 3-2. LiDAR Camera Depth Start Point Reference •Figure 6-1. Intel® RealSense™ LiDAR Camera L515	June 2020
003	•Table 3-1. Depth Specification •Table 3-2. Depth Camera Controls •Table 3-7. Image Formats •Figure 3-3. LiDAR Camera X-Y Depth Origin Reference	January 2021

§ §

1 Description and Features

Description

The Intel® RealSense™ LiDAR Camera L515 is Intel's first release of a LiDAR camera enabling highly accurate depth sensing in a small form factor.

Small enough to fit in the palm of your hand, the L515 is 61mm in diameter and 26mm in height. At approximately 100g, it's designed to be easily situated on any system, or attached to a tablet or phone. It also runs at less than 3.5W, considerably lower than competing time-of-flight (TOF) solutions. All depth calculations run on the device resulting in true platform independence.

With a short exposure time of <100ns per depth point, even rapidly moving objects can be captured with minimal motion blur. Optimized for indoor lighting, the L515 processes over 23 million depth points per second via a custom made ASIC. The product has been designed for use case flexibility with the inclusion of an RGB camera and an inertial measurement unit.

Features

- Depth Capture from 0.25 to 9m[1]
- 2MP RGB Camera[2]
- Inertial Measurement Unit (IMU)
- Up to 30FPS Depth at 1024x768 (XGA)
- Up to 30FPS Color at 1920x1080 (FHD)
- Class 1 Laser Compliant
- Device Accuracy Health[2]

(1) Tested at 95% reflectivity.

(2) RGB camera always on.

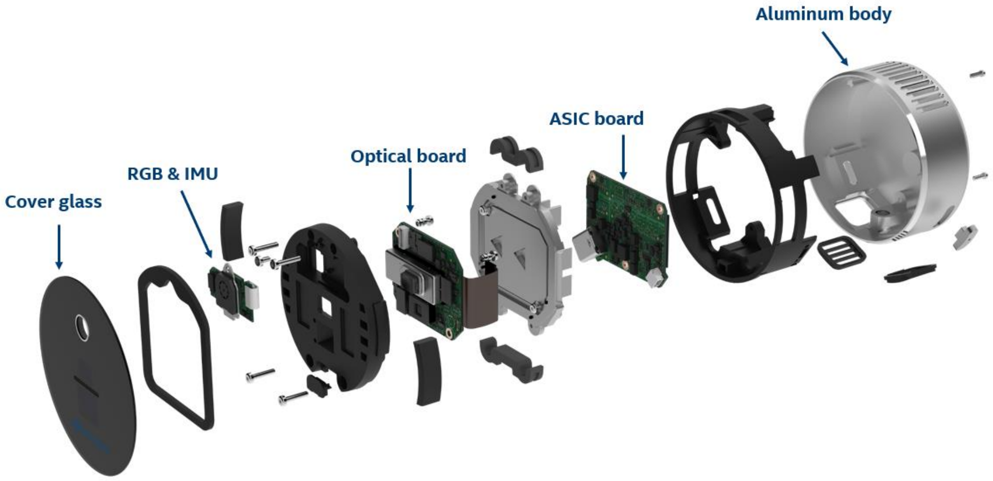
Minimum System Requirements

USB 3.1 Gen1

Ubuntu*16.xx/18.04 LTS

Windows*10 (build 15063 or later)

Figure 1-1. Intel® RealSense™ LiDAR Camera L515 Exploded View



§ §

2 Introduction

2.1 Purpose and Scope of this Document

This document captures the specifications for the Intel® RealSense™ LiDAR Camera L515.

2.2 Terminology

Table 2-1. Terminology Table

Term	Description
Depth	Depth video streams are like color video streams except each pixel has a value representing the distance away from the camera instead of color information
FOV	Field Of View (FOV) describes the angular extent of a given scene that is imaged by a camera. A camera's FOV can be measured horizontally, vertically, or diagonally
Host System	Computer or SOC connected to depth camera
IR Laser	This refers to the source of infrared (IR) light used for illuminating a scene, object, or person to collect depth data.
IMU	Inertial Measurement Unit is a system-in-package for the detection of acceleration in 3 dimensions and rotations in 3 dimensions.
LiDAR	Light Detection and Ranging is a remote sensing technology that measures the distance to objects and targets using a combination of laser light and receivers.
MEMS	Micro-Electro-Mechanical System
RH	Relative humidity
TBD	To Be Determined. In the context of this document, information will be available in a later revision.

2.3 LiDAR Technology Overview

The Intel® RealSense™ LiDAR Camera L515 uses an IR laser, a MEMS, an IR photodiode, an RGB imager, a MEMS controller, and a vison ASIC. The MEMS is used to scan the IR laser beam over the entire field-of-view (FOV). The L515 vision ASIC will process the data from the reflected beam captured by the photodiode and will output a depth point representing the accurate distance of a specific point in the scene from the camera. Aggregation of the depth points will generate a point cloud depth data representing the full scene.

§ §

3 Functional Specification

3.1 Depth Camera Specification

Table 3-1. Depth Specification

Depth Resolution	Number of depth points per second	FOV[1]	Range @ 15% reflectivity[2]	Range @ 95% reflectivity[2]
QVGA (320x240)	2.3M	70º x 55º	0.25 - 3.9m	0.25 - 9m
VGA (640x480)	9.2M	70º x 55º	0.25 - 3.9m	0.25 - 9m
XGA (1024x768)	23.6M	70º x 55º	0.25 - 2.6m	0.25 - 6.5m

[1] Due to mechanical tolerances, FOV can vary +/- 2 degrees.

[2] Max range is specified for the center 10% ROI of the image, as long as the operating conditions are met.

3.1.1 Camera Accuracy Health

In order to ensure the long-term optimal accuracy of the L515's cutting edge depth technology, Intel® has implemented an additional accuracy assurance method utilizing the RGB camera. The feature runs on the host as part of the Intel® RealSense™ SDK 2.0 and will require a few RGB frames to be sent to the host. These RGB frames are used to analyze the scene and compared with the depth camera to verify alignment between both cameras.

The accuracy health-test and maintenance feature is automatically enabled and all customers gain this feature without any user interaction.

To ensure complete transparency, this functionality is in the Intel® RealSense™ SDK 2.0 open source SDK (LibRealSense).

3.2 Depth Camera Controls and Data Format

In order to achieve optimal performance of the camera, three presets are offered based on the desired range of the application.

Table 3-2. Depth Camera Controls

Preset	Description
Max Range	This preset is useful when there is no ambient light in the scene (fully indoors use case, with no light coming through windows). With this preset the laser power is set to maximum as well as the receiver gain which optimize the depth quality in indoor conditions.

Preset	Description
Short Range	This preset lowers the laser power and gain so that close objects do not oversaturate the receiver. This allows operation at a close distance to objects. This setting may not be good if objects further away in the scene also need to perform well.
No Ambient Light	Same as Max Range preset, this preset is useful when there is no ambient sunlight in the scene. The main difference between the presets is the laser power which is lower on this preset to avoid false depth on objects that are on longer distances than the ambiguity range (10m-VGA, 6.5m-XGA).
Low Ambient Light	This preset is recommended for environments where there may be a low amount of ambient sunlight present. Similar to Max Range preset the laser power is set to maximum but the receiver gain is reduced to avoid saturation of the camera due to ambient sunlight. The preset is also recommended for cases that the user wants to detect close objects (<50cm).

Table 3-3: Depth and Infrared Data Formats

FORMAT	KEY	TYPE	DESCRIPTION
Depth	Z	16b UINT	Depth format equating to distance from the device subassembly planar surface to the object.
Infrared	Y8	8b UINT	IR image representing the intensity of the reflected IR laser reflected off the objects in the scene.
Confidence	C	4b UINT	Provides a per pixel confidence value, 0xF equals high confidence and 0x0 represents low confidence.

3.3 Depth Quality Metrics

Table 3-4: Depth Quality Metrics

METRIC	DEFINITION
Depth Accuracy	Represents the average difference of valid pixels relative to ground truth.
Depth Standard Deviation	Represents the total spread (noise) of the depth values relative to ground truth.

Figure 3-1. Depth Quality Metric Illustration

DEPTH ACCURACY AND DEPTH RMS ERROR

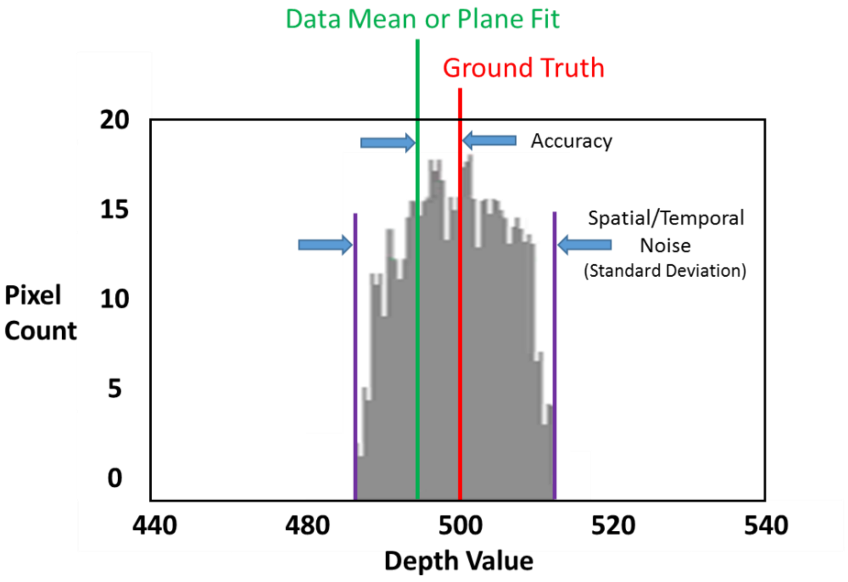


Table 3-5. Depth Quality Specification

Metric	Value	Notes
Depth Accuracy – Avg	< 5mm @ 1m < 14mm @ 9m	VGA resolution, 95% reflectivity
Depth – Std Dev	2.5mm @ 1m 15.5mm @ 9m	VGA resolution, 95% reflectivity
Exposure Time	< 100ns per depth point	Robust against motion blur
Lighting Condition	< 500 lux sunlight (0.4uW/cm2/nm)	

3.4 Depth Start Point (Ground Zero Reference)

The depth start point or the ground zero reference can be described as the starting point or plane where depth = 0. For LiDAR camera (L515), this point is referenced from front of camera cover glass

Figure 3-2. LiDAR Camera Depth Start Point Reference

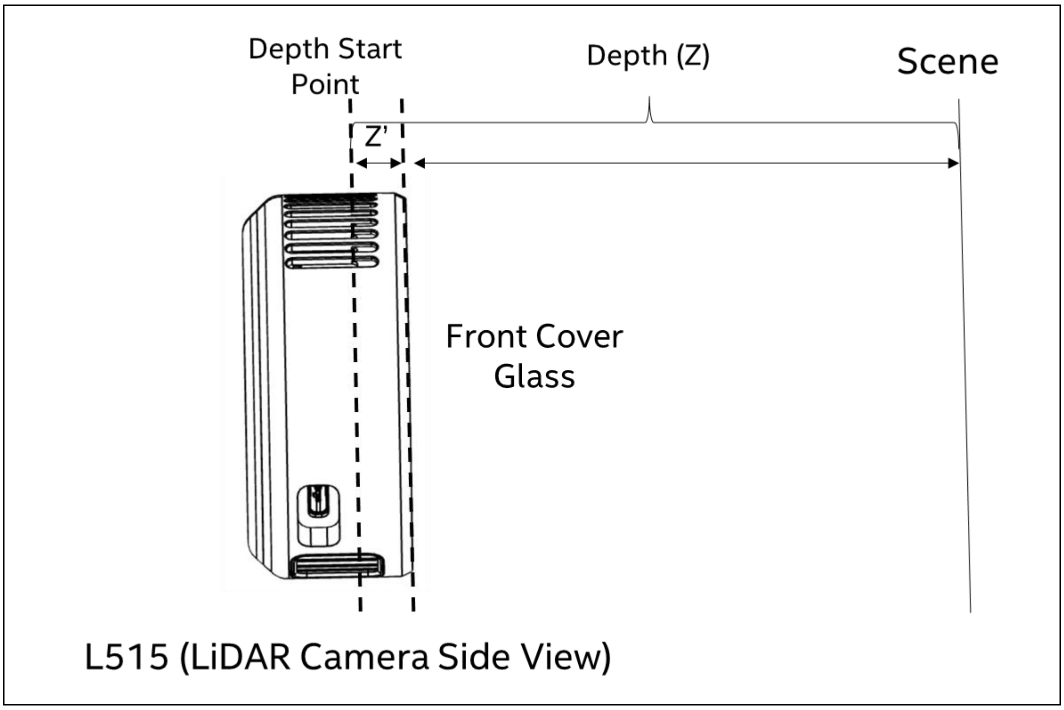


Table 3-6. LiDAR Depth Start Point

LiDAR Camera	Camera Front Glass (Z')
L515	-4.5mm

NOTES:

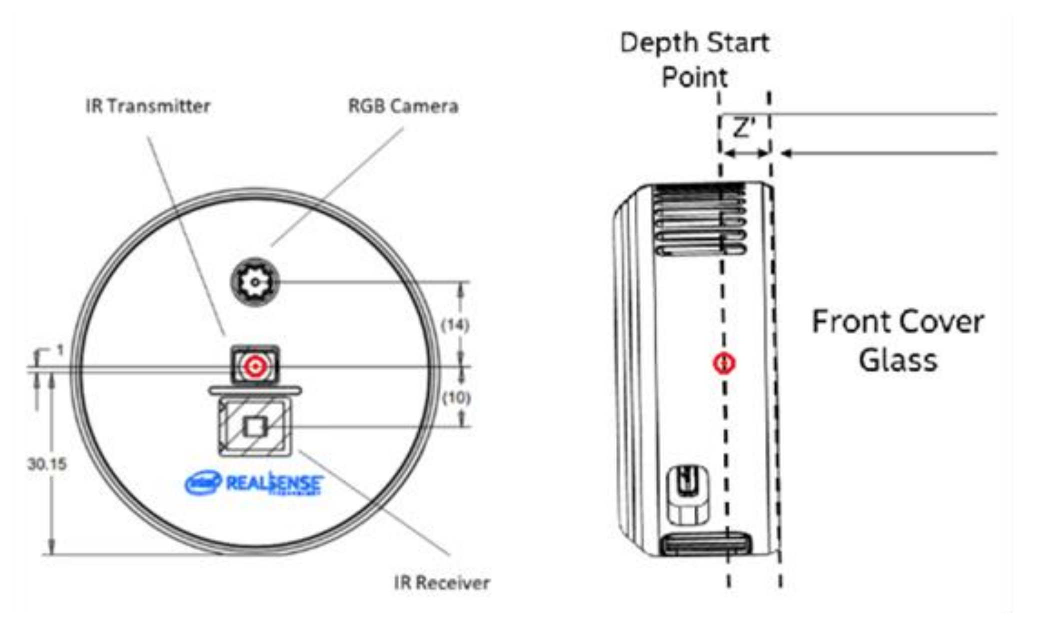
If depth measurement reference is front cover glass, then |Z'| should be added to measured value to determine Ground Truth.

This value can be read via Intel® RealSense™ SDK 2.0 APIs. Please see the latest SDK for reference.

3.4.1 Depth Origin X-Y Coordinates

The depth origin X-Y coordinates is the X-Y center of the IR Transmitter.

Figure 3-3. LiDAR Camera X-Y Depth Origin Reference



3.5 Image Formats and Color Camera Functions

Table 3-7. Image Formats

Format	Resolution	Frame Rate (FPS)	Comment
YUY2	1920x1080	6,15,30	Color Stream from RGB camera
	1280x720	6,15,30,60	
	960x540	6,15,30,60	

NOTE:

Color camera frame rates are expressed as nominal. Effective frame rates can vary depending on the exposure settings of the camera. Camera settings that increase the exposure time can decrease the effective frame rate.

Table 3-8. Color Camera Controls

Control	Description	Min	Max
Auto-Exposure Mode	Automatically sets the exposure time and gain for the frame.	0x1	0x8
Manual Exposure Time	Sets the absolute exposure time when auto-	1	10000

Control	Description	Min	Max
	exposure is disabled.		
Brightness	Sets the amount of brightness applied when auto-exposure is enabled.	-64	64
Contrast	Sets the amount of contrast based on the brightness of the scene.	0	100
Gain	Sets the amount of gain applied to the frame if auto-exposure is disabled.	0	4096
Hue	Sets the amount of hue adjustment applied to the frame.	-180	180
Saturation	Sets the amount of saturation adjustment applied to the frame.	0	100
Sharpness	Sets the amount of sharpening adjustment applied to the frame.	0	100
White Balance Temperature Control	Sets the white balance when AWB is disabled.	2800	6500
White Balance Temperature Auto (AWB)	Enables or disables the AWB algorithm.	0	1
Power Line Frequency	Specified based on the local power line frequency for flicker avoidance.	Disabled 50Hz 60Hz Auto	
Backlight Compensation	Sets a weighting amount based on brightness to the frame.	0	255

3.6 IMU Specification and Operating Modes

Table 3-9. Inertial Measurement Specifications

Parameter	Properties
Model	Bosch BMI085
Degrees of Freedom	6
Acceleration Range	±4g
Accelerometer Output Data Rate	100Hz/200Hz/400Hz

Gyroscope Range	±1000 Deg/s
Gyroscope Output Data Rate	100Hz/200Hz/400Hz
Data Format	32b Float

Accelerometer and gyroscope data streams from the onboard IMU are available via Intel® RealSense™ SDK 2.0.

3.7 L515 Device Firmware Update (DFU)

The firmware contains the operation instructions. Upon runtime, Vision ASIC loads the firmware and programs the component registers. If the Vision ASIC is configured for update or recovery, the unlocked R/W region of the firmware can be changed.

3.7.1 Update

During a firmware update, the firmware utility will issue a device firmware update command to the Vision ASIC. The Vision ASIC will then reset into firmware update mode. The firmware utility uses a single binary file to maintain the firmware image.

3.7.1.1 Update Limits

The firmware update engine does not allow infinite update cycles between older and current versions of firmware. The engine will establish a baseline version of firmware based on the latest firmware version installed. The engine will allow a return to a previous version or baseline version of firmware up to 20 times. After the 20th update, the engine will only allow an update to a firmware revision higher than the baseline version.

§§

4 Intel® RealSense™ LiDAR Camera L515 Hardware Specification

4.1 L515 Device Components

Table 4-1. Main components

Component	Description
BMI085	Accelerometer and Gyroscope in a single package
OV2740	RGB image sensor
IR emitter	860nm IR laser

4.2 Color Camera Properties

Table 4-2. Color Camera Properties

Parameter	Camera Sensor Properties
Color Image Signal Processor	Embedded*
Active Pixels	1920 X 1080
Sensor Aspect Ratio	16:9
Format	1/6”
F Number	2.0
Focal Length	1.88mm
Focus	Fixed
Shutter Type	Rolling Shutter
Signal Interface	MIPI CSI-2, 2X Lanes
Horizontal Field of View	69° +/-1°
Vertical Field of View	42° +/-1°

* arm This product uses Arm® Assertive Camera™ technology by Arm Limited.

4.3 Camera L515 Power Consumption

The Intel® RealSense™ LiDAR Camera L515 is powered through USB VBUS power connected to host platform via USB type-C connection. The same cable is used for data transfer.

Table 4-3. Power Requirements

Parameter		Min	Nom	Max	Unit
VCC	Supply Voltage	4.5	5	5.5	V

Table 4-4. Power Consumption

Model	Idle Power (W)	Normal Power (W) *Typical Usage Configuration (@ 25^{0}C)*	Notes
L515	0.8	3.0	Depth (VGA)
		3.2	Depth (VGA) + RGB (1080p, 30FPS)
		3.1	Depth (XGA)
		3.3	Depth (XGA) + RGB (1080p, 30FPS)

4.4 Camera Interface

The interface to L515 is USB 3.0 Type-C. Standard USB3 cables with max over-mold size of 6.5mmx12mm are supported.

4.5 Camera L515 Storage and Operating Conditions

Table 4-5. Storage and Operating Conditions

Parameter	Condition	Min	Max	Unit
Storage (Still Air), Not Operating	Sustained, Controlled (1)	0	50	ºC
	Short Exposure (2)	-20	70	ºC
	Humidity, Non-Condensing	Temperature/ RH: 40ºC / 90%		
Operating(3)(4)	Ambient temperature range when the device is streaming	0	30	ºC
Skin Temperature @ 25C Ambient(3)(4)	Camera housing temperature	N/A	50	ºC

NOTE:

(1) Controlled conditions should be used for long term storage of product.

(2) Short exposure represents temporary max limits acceptable for transportation conditions.

(3) Under typical indoor air flow.

(4) Depth and RGB enabled simultaneously.

4.6 Material, Vendor and Device ID

4.6.1 Camera L515 Product Identifier and Material Code

Table 4-6. Product Identifier and Material Code

Production	Product Material Code
Camera L515	999NGF

4.6.2 Vendor Identification (VID) and Device Identification (DID)

Table 4-7. Vendor ID and Device ID Table

Depth Module/Depth Camera	Vendor ID	Device ID
Intel® RealSense™ LiDAR Camera L515	8086	0x0B64

§§

5 Software (SDK)

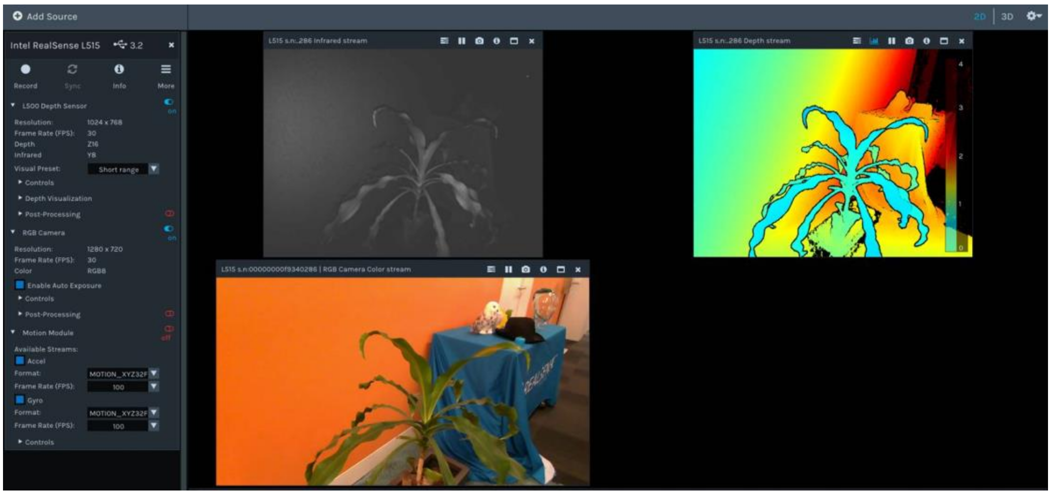
5.1 Intel® RealSense™ Software Development Kit 2.0

Intel® RealSense™ SDK 2.0 is a cross-platform library for working with Intel® RealSense™ LiDAR Camera L515. It is open source and available on https://www.intelrealsense.com/sdk-2/

The SDK at a minimum includes:

- **Intel® RealSense™ Viewer** - This application can be used view, record and playback depth streams, set camera configurations and other controls.
- **Depth Quality Tool** - This application can be used to test depth quality, including: distance to plane accuracy, Z accuracy, standard deviation of the Z accuracy and fill rate.
- **Debug Tools** - These command line tools gather data and generate logs to assist in debug of camera.
- **Code Examples** - Examples to demonstrate the use of SDK to include D400 Series camera code snippets into applications.
- **Wrappers** -Software wrappers supporting common programming languages and environments such as ROS, Python, Matlab, node.js, LabVIEW, OpenCV, PCL, .NET and more

Figure 5-1. RealSense Viewer – L515


Add Source
Intel RealSense L515
3.2
Record
Sync
Info
More
2D
3D
L515 s.n:..286 Infrared stream
L515 s.n:..286 Depth stream
L515 s.n:00000000f9340286 | RGB Camera Color stream
4
3
2
1
0
on
on
off

L500 Depth Sensor

Resolution: 1024 x 768
Frame Rate (FPS): 30
Depth Z16
Infrared Y8
Visual Preset: Short range

- Controls
- Depth Visualization
- Post-Processing

RGB Camera

Resolution: 1280 x 720
Frame Rate (FPS): 30
Color RGB8

☑ Enable Auto Exposure

- Controls
- Post-Processing

Motion Module

Available Streams:
☑ Accel
Format: MOTION_XYZ32F
Frame Rate (FPS): 100
☑ Gyro
Format: MOTION_XYZ32F
Frame Rate (FPS): 100

- Controls

§§

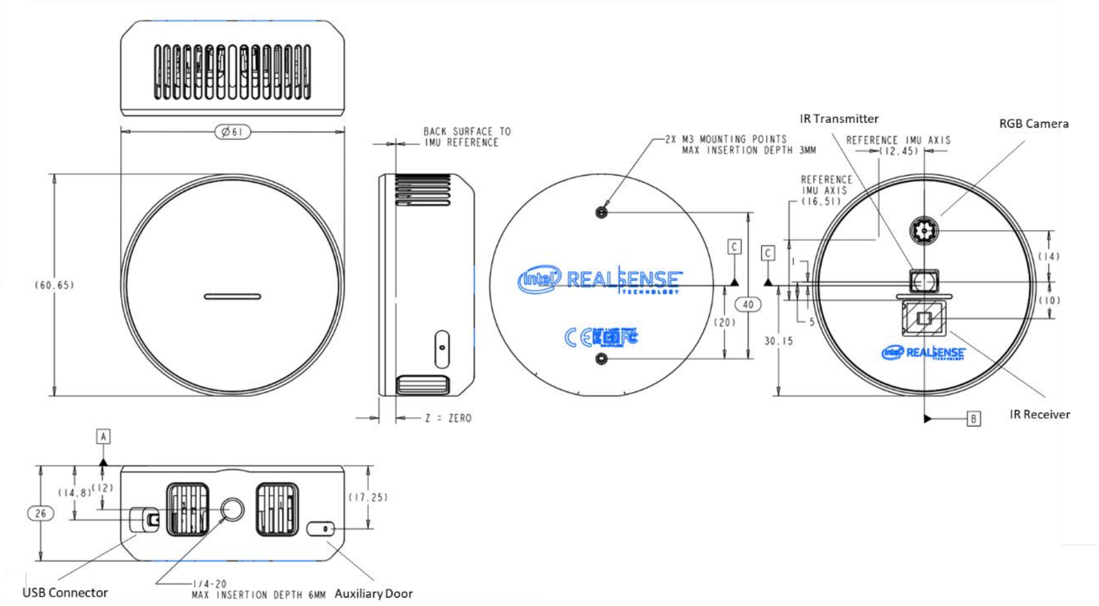
6 Mechanical Specifications

6.1.1 Mechanical Dimensions

Table 6-1. Intel® RealSense™ LiDAR Camera L515 Mechanical Dimensions

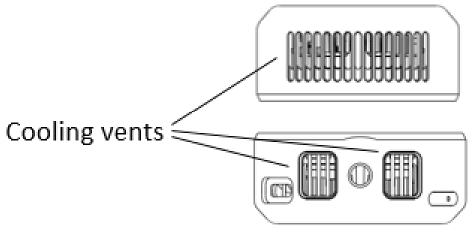
Dimension	Nominal	Unit
Diameter	61	mm
Height	26	mm
Weight	95	g

Figure 6-1. Intel® RealSense™ LiDAR Camera L515



When integrated into system, it is recommended that the L515 be secured via the two M3 mounting screw holes on the back of the product. The cooling vents need to remain unobstructed at all times.

Figure 6-2. Intel® RealSense™ LiDAR Camera L515 Cooling Vents



The cooling vents need to remain unobstructed at all times. Clearance of 12mm needed around the vents for airflow.

6.2 L515 Cover Material Cleaning Procedure

1. Do not use any chemical or water on the camera cover material
2. Remove dust and dirt as much as possible from the cover material with a lens blower brush.
3. Wipe with a dry, clean micro-fiber cloth.

§§

7 Regulatory Ecology Compliance

7.1 System Laser Compliance

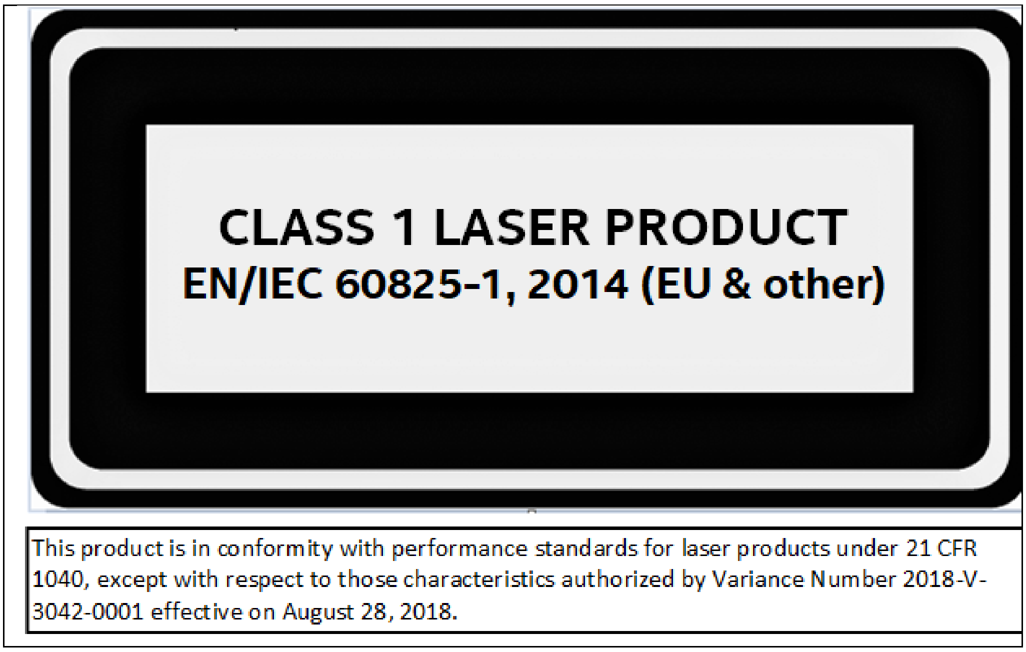
The Intel® RealSense™ LiDAR Camera L515 certification is transferable to the system and no system recertification is required. However, the following statements and labels must be included in the user manual of the end product.

7.1.1 Certification Statement

This product is classified as a Class 1 Laser Product under the EN/IEC 60825-1, Edition 3 (2014) internationally.

In the US, this product is in conformity with performance standards for laser products under 21 CFR 1040, except with respect to those characteristics authorized by Variance Number 2018-V-3042-0001 effective on August 28, 2018.

7.1.2 Explanatory Label



CLASS 1 LASER PRODUCT
EN/IEC 60825-1, 2014 (EU & other)

This product is in conformity with performance standards for laser products under 21 CFR 1040, except with respect to those characteristics authorized by Variance Number 2018-V-3042-0001 effective on August 28, 2018.

7.1.3 Cautionary Statements

⚠ System integrators should refer to their respective regulatory and compliance owner to finalize regulatory requirements for a specific geography.

⚠ **Caution** - Use of controls or adjustments or performance of procedures other than those specified herein may result in hazardous radiation exposure.

⚠

- Do not power on the product if any external damage was observed.
- Do not attempt to open any portion of this laser product. There are no user serviceable parts.
- Invisible laser radiation when opened. Avoid direct exposure to beam.
- There are no service/maintenance, modification, or disassembly procedures for the stereo module and infrared projector. The system integrator must either notify Intel or return modules before any failure analysis is performed.
- Modification or service of the stereo module, specifically the infrared projector, may cause the emissions to exceed Class 1.
- Do not try to update camera firmware that is not officially released for specific camera module SKU and revision.

7.1.4 Embedded Laser Information

- Wavelength (0-50°C): 844-875nm
- Beam divergence (without collimation): (6x10) deg to (15-21) deg; parallel x perpendicular
- Pulse duration and repetition rate:
  - 1ns pulse duration
  - 500 MHz repetition
  - Rise/Fall time: 300ps
- Maximum power or energy output: 240mW

7.1.5 US FDA Accession Number

Table 7-1. U.S. FDA Accession Number

Component	U.S. FDA accession numbers
Intel® RealSense™ LiDAR Camera L515	1820840

This accession number should be entered into Box B.1 of the Food and Drug Administration (FDA) 2877 Declaration for Imported Electronic Products Subject to Radiation Control Standards.

7.2 Regulatory Compliance

7.2.1 Manufacturer’s Information

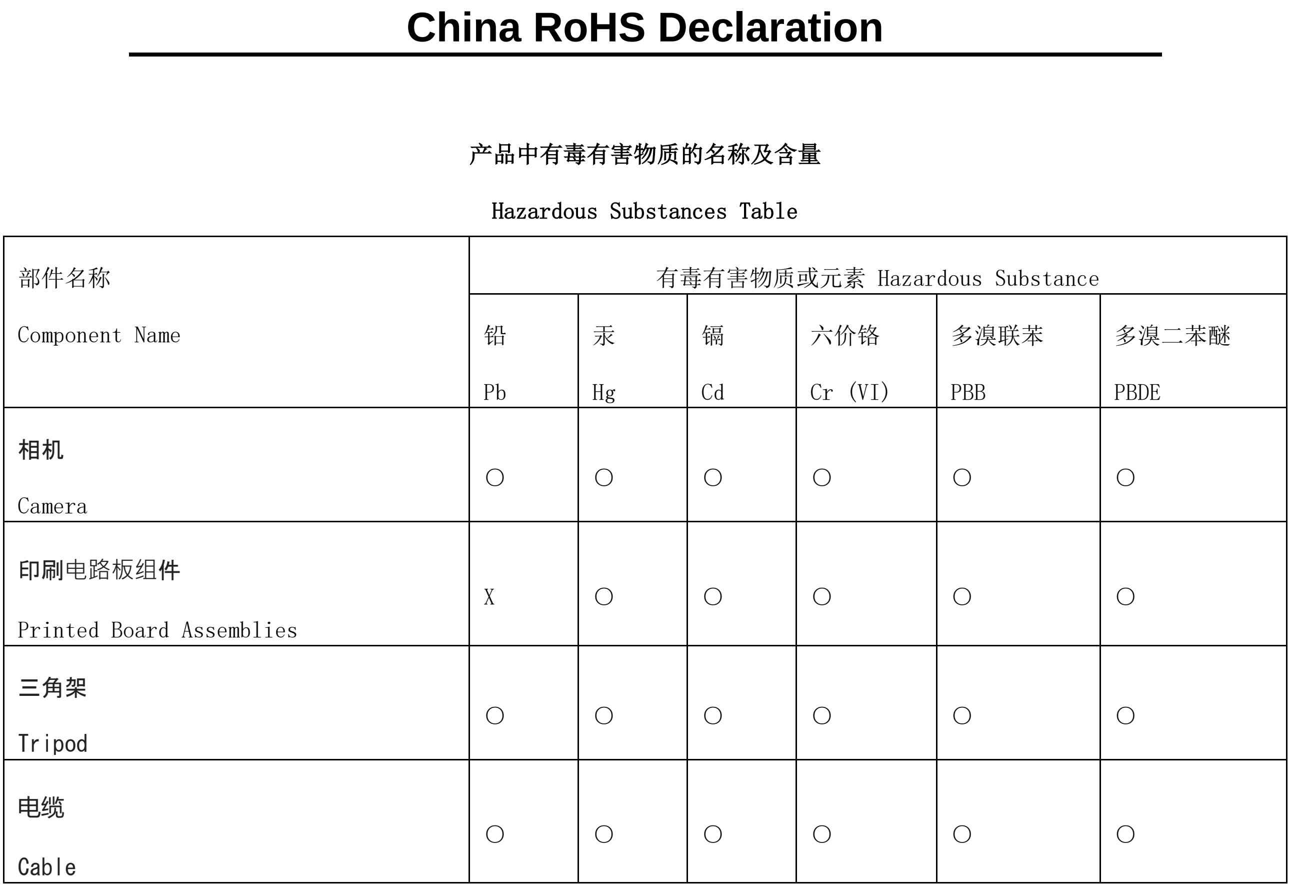
Intel Corporation:
Attn: Corp. Quality
2200 Mission College Blvd,
Santa Clara, CA 95054-1549, USA

7.2.2 EU Single Place of Contact

Att. Corp Quality
Intel Deutschland GmbH
Am Campeon 10-12
Neubiberg, 85579 – Germany

7.3 Ecology Compliance

7.3.1 China RoHS Declaration



China RoHS Declaration

产品中有毒有害物质的名称及含量

Hazardous Substances Table

部件名称 Component Name	有毒有害物质或元素 Hazardous Substance					
	铅 Pb	汞 Hg	镉 Cd	六价铬 Cr (VI)	多溴联苯 PBB	多溴二苯醚 PBDE
相机 Camera	○	○	○	○	○	○
印刷电路板组件 Printed Board Assemblies	X	○	○	○	○	○
三角架 Tripod	○	○	○	○	○	○
电缆 Cable	○	○	○	○	○	○

〇：表示该有毒有害物质在该部件所有均质材料中的含量均在GB/T 26572标准规定的限量要求以下。

〇：Indicates that this hazardous substance contained in all homogeneous materials of such component is within the limits specified in GB/T 26572.

×：表示该有毒有害物质至少在该部件的某一均质材料中的含量超出GB/T 26572标准规定的限量要求。

×： Indicates that the content of such hazardous substance in at least a homogeneous material of such component exceeds the limits specified in GB/T 26572.

对销售之日的所售产品,本表显示我公司供应链的电子信息产品可能包含这些物质。注意：在所售产品中可能会也可能不会含有所有所列的部件。

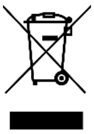
This table shows where these substances may be found in the supply chain of our electronic information products, as of the date of sale of the enclosed product. Note that some of the component types listed above may or may not be a part of the enclosed product.

除非另外特别的标注,此标志为针对所涉及产品的环保使用期限标志. 某些可更换的零部件可能会有一个不同的环保使用期限(例如,电池单元模块).

此环保使用期限只适用于产品在产品手册中所规定的条件下工作.

The Environment-Friendly Use Period (EFUP) for all enclosed products and their parts are per the symbol shown here, unless otherwise marked. Certain field-replaceable parts may have a different EFUP (for example, battery modules) number. The Environment-Friendly Use Period is valid only when the product is operated under the conditions defined in the product manual.

7.3.2 Waste Electrical and Electronic Equipment (WEEE)



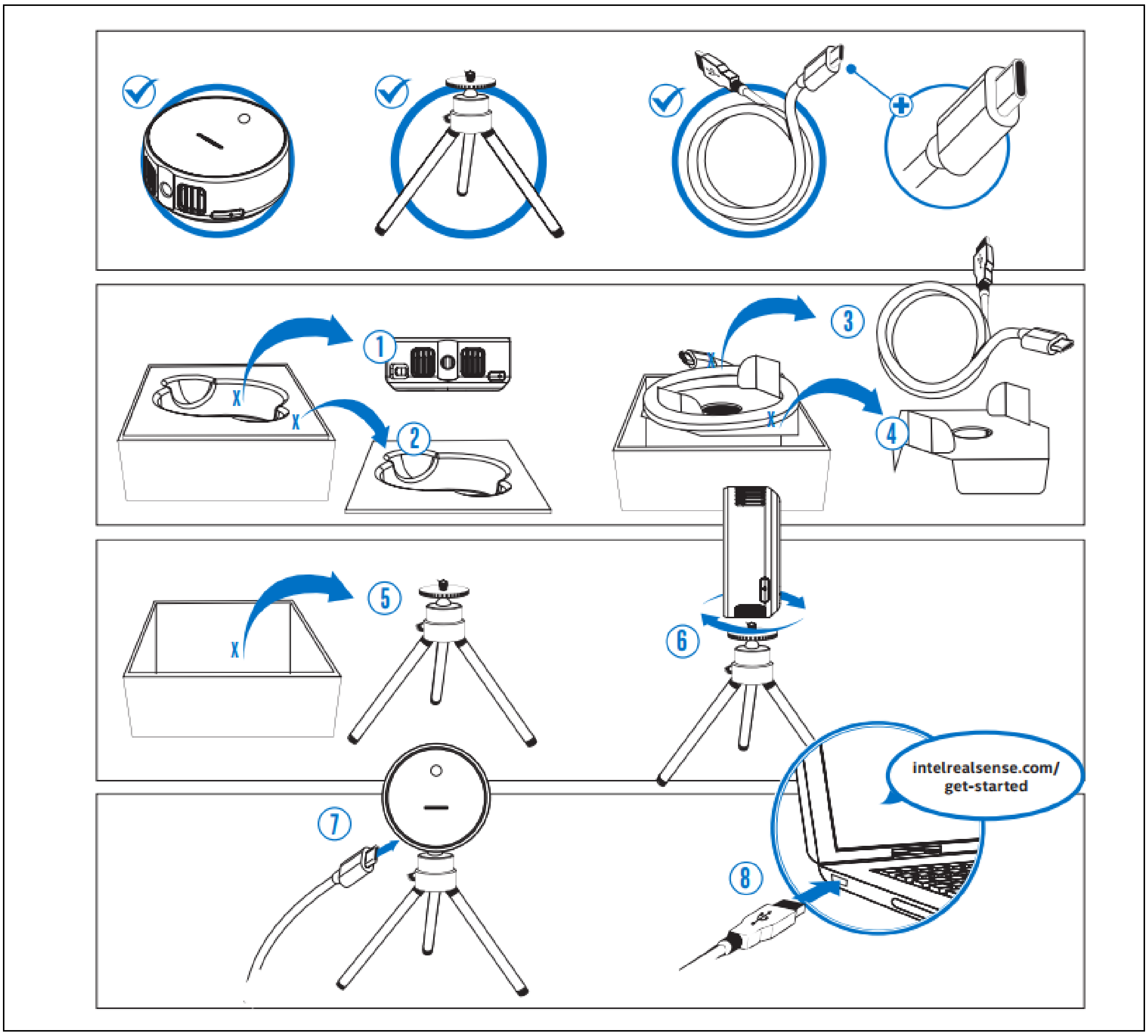
"In the EU, this symbol means that this product must not be disposed of with household waste. It is your responsibility to bring it to a designated collection point for the recycling of electrical and electronic equipment. For more information, contact the local waste collection center or your point of purchase of this product."

§ §

8 Appendix A – L515 Product Box

Inside Intel® RealSense™ LiDAR Camera L515 product box you will find the L515 camera, a tripod and a USB3 cable.

Figure 8-1. L515 Product Box



Appendix C

Zivid One$^+$ Large Datasheet

Zivid One+

Technical specification

Zivid One+ S	(ZVD1P-S)
Zivid One+ M	(ZVD1P-M)
Zivid One+ L	(ZVD1P-L)

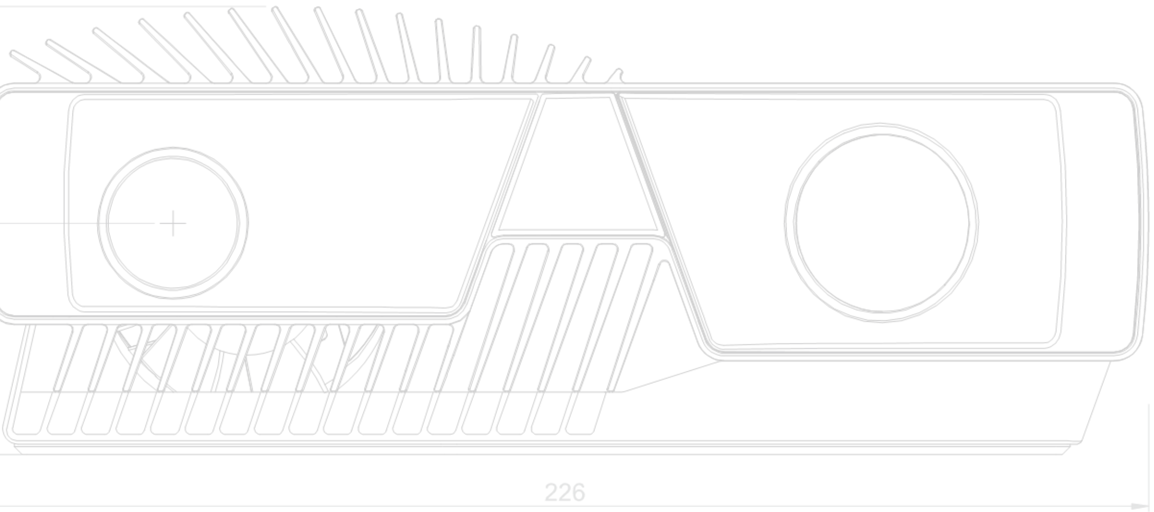


Table of Contents

General specifications

Model (Part number)	Zivid One+ S (ZVD1P-S) Zivid One+ M (ZVD1P-M) Zivid One+ L (ZVD1P-L)
3D technology	Structured light
Imaging	1920 x 1200 (2.3Mpixel) Native 3D Color
Point cloud output	3D (XYZ) + Color (RGB) + SNR
Exposure time (minimum per pattern projection)	6.500 ms
Aperture (A)	f/1.4 to f/32
Gain (G)	1x to 16x
Projector Brightness (B)	0.25x to 1.8x 1x = 400 lumens
Calibration	Factory calibrated
Safety and EMC	CE CB EN60950 FCC Class A
Typical capture time [1]	100 ms to 1 s

[1] From capture initialized until point cloud is ready to copy. Includes processing. Acquisition time can be shorter.

Operating distance and field of view

	S	M	L
Focus distance (mm)	500	1000	1800
Optimal working distance (mm)	350 to 700	700 to 1500	1200 to 2600
Recommended working distance (mm)	300 to 1000	500 to 2000	1200 to 3000
Field of view (mm)	164 x 132 at 300 350 x 220 at 500 621 x 439 at 1000	433 x 271 at 600 702 x 432 at 1000 1330 x 871 at 2000	843 x 530 at 1200 1252 x 783 at 1800 2069 x 1310 at 3000
Spatial resolution (mm)	0.18 at 500 4.00 × 10^{-4} per distance (z) in mm	0.37 at 1000 3.71 × 10^{-4} per distance (z) in mm	0.67 at 1800 3.67 × 10^{-4} per distance (z) in mm

Figure 1 - Zivid One+ S FOV

All values in degrees or mm.

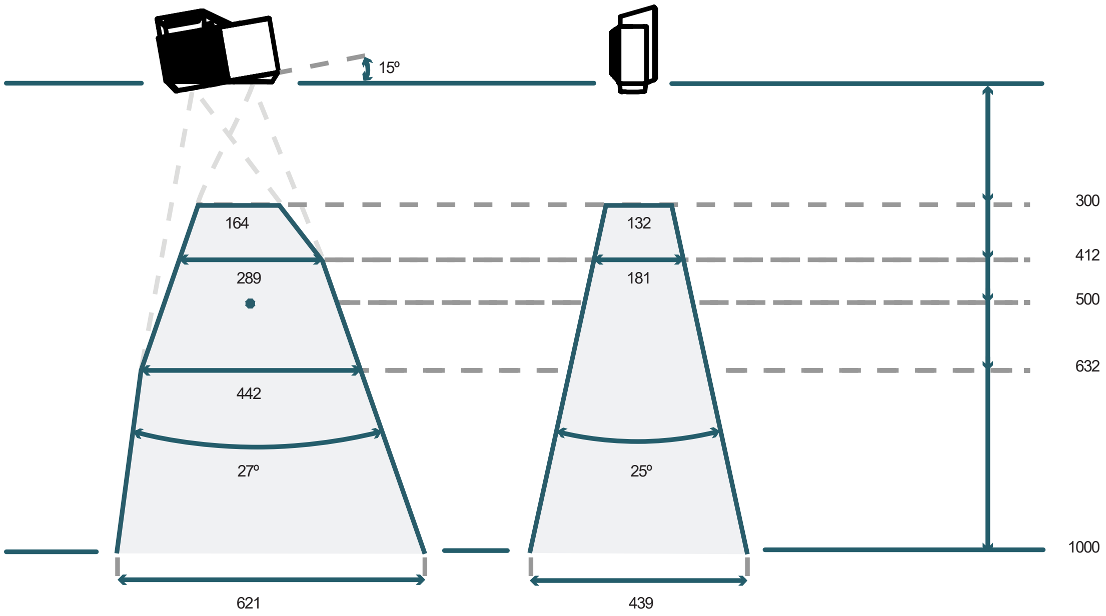


Figure 2 - Zivid One+ S Spatial Resolution vs. Distance

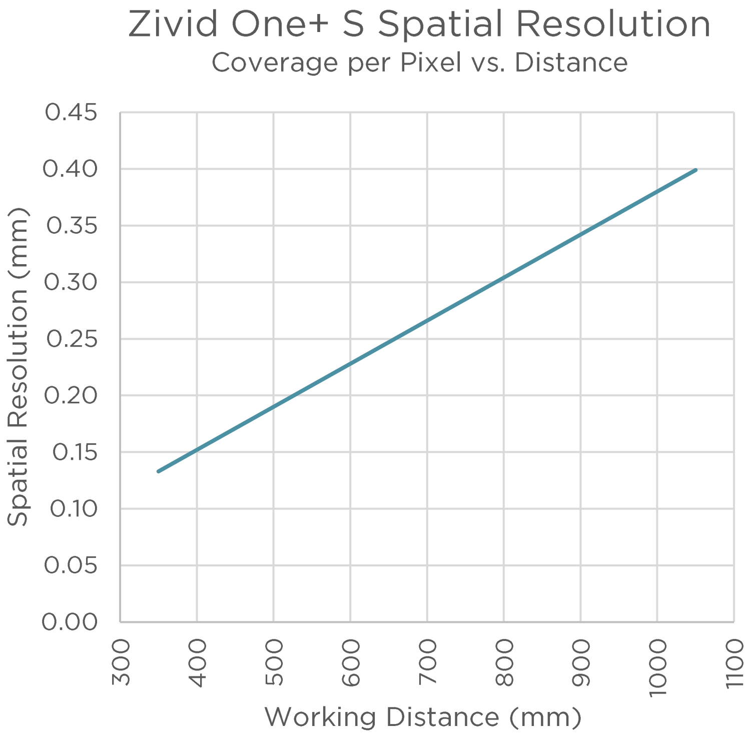


Figure 3 - Zivid One+ M FOV

All values in degrees or mm.

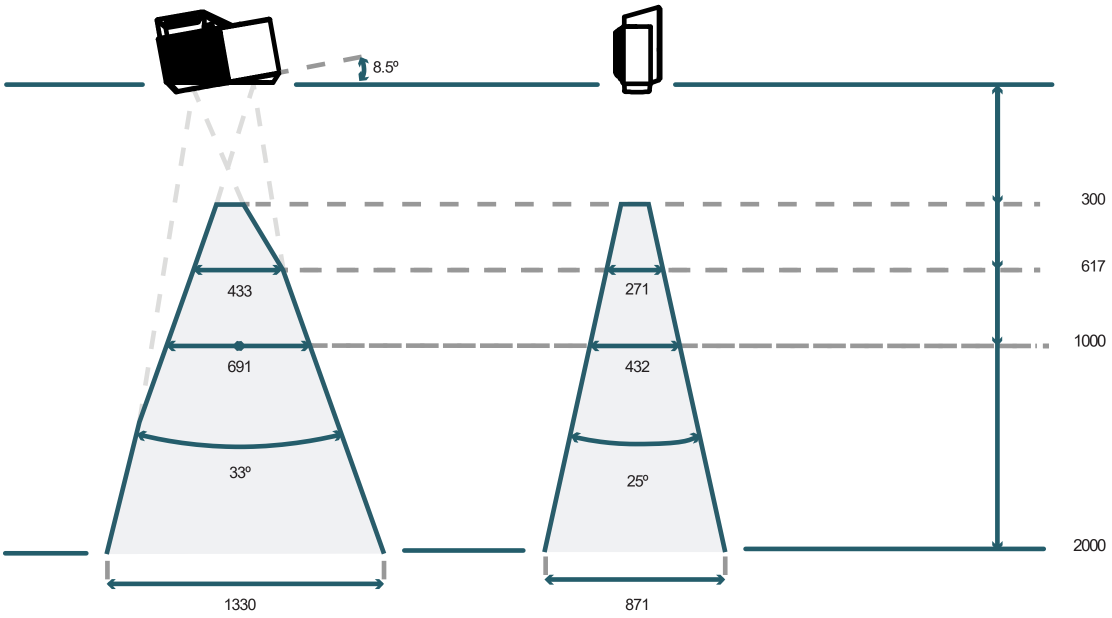


Figure 4 - Zivid One+ M Spatial Resolution vs. Distance

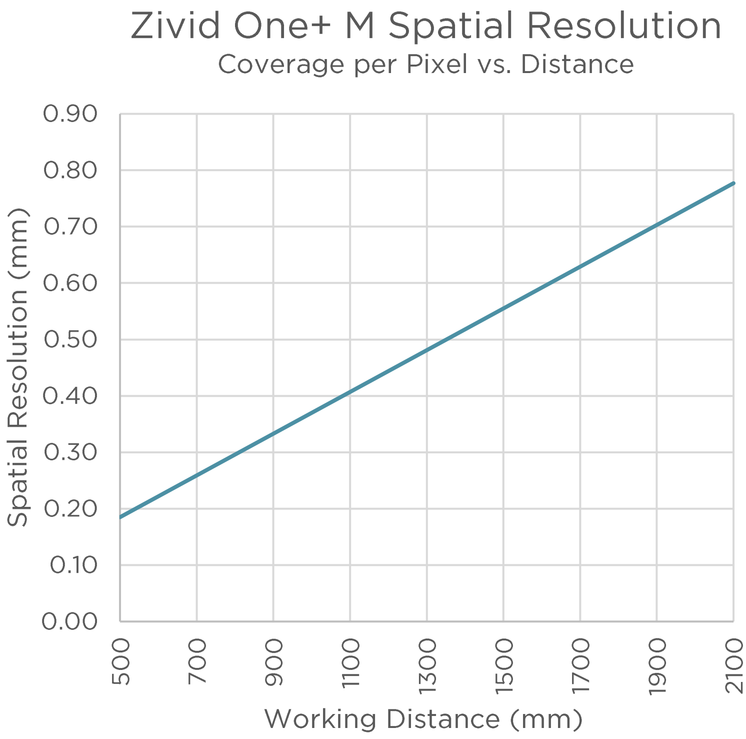


Figure 5 - Zivid One+ L FOV

All values in degrees or mm.

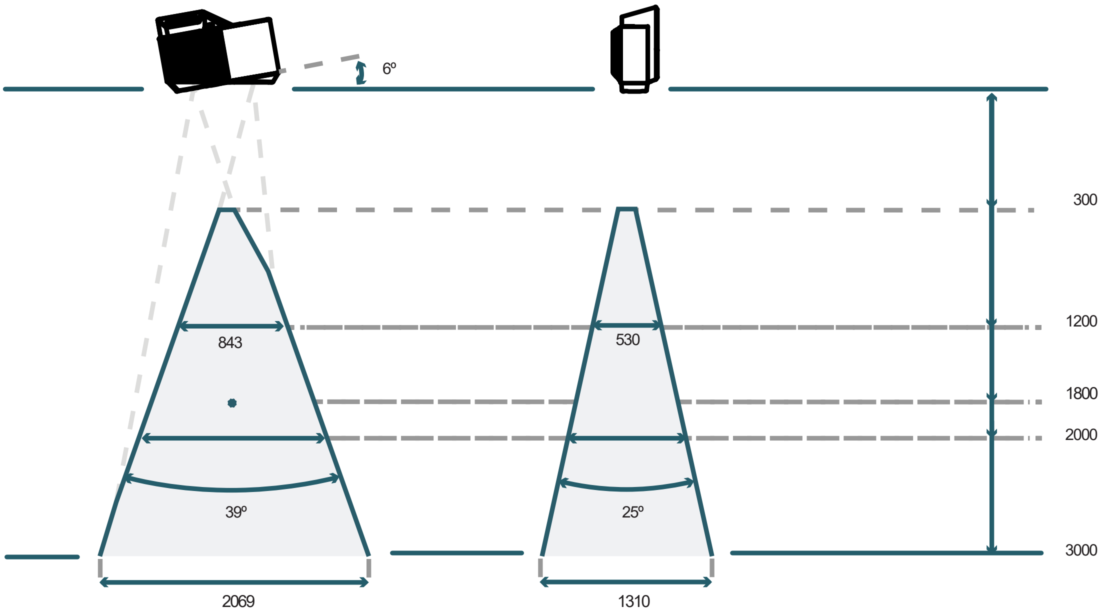
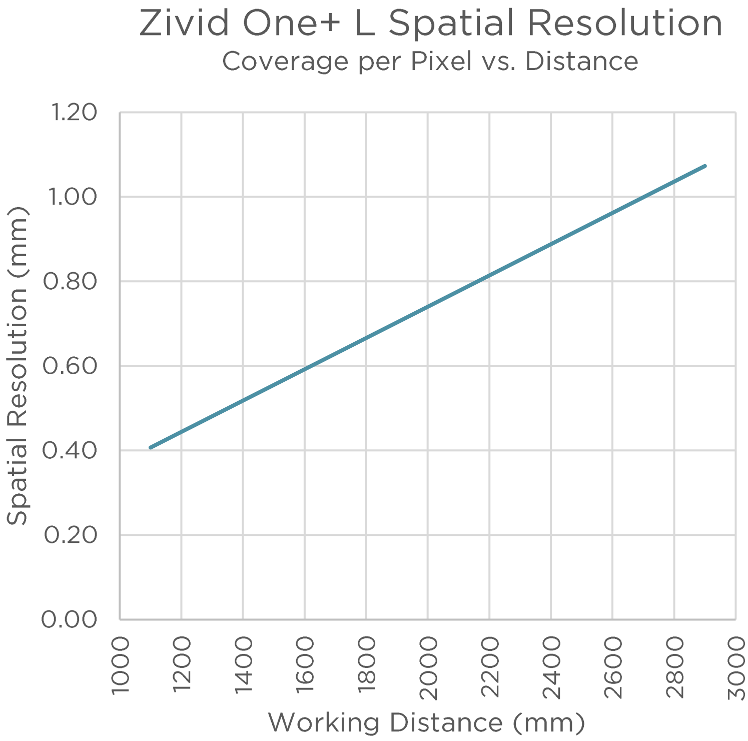


Figure 6 - Zivid One+ L Spatial Resolution vs. Distance



Accuracy specifications

Common conditions

The following table outlies the conditions applied under test and to all specifications unless otherwise stated.

Parameter	Description	Typical
Working distance (D)	Focus distance	Zivid One+ S: 500 mm Zivid One+ M: 1000 mm Zivid One+ L: 1800 mm
	Optimal working distance	Zivid One+ S: 350 – 700 mm Zivid One+ M: 700 – 1500 mm Zivid One+ L: 1300 – 2600 mm
Ambient temperature (Ta)	Typical temperature	15 - 30 °C
	Full temperature range	10 – 40 °C
Ambient light (La)		0 lux
Aperture (A)		f/8.0 – f/2.0
Gain (G)		1.0x
Projector Brightness (B)		1 – 1.8 x
Capture time	Acquisition time used during measurement	> 85 ms
	Capture time used during measurement	> 200 ms
Duty Cycle	Capture-to-Idle time ratio	5 - 30 %
Other		81% center crop (90% × 90%) HDR = off 10 min warm-up Applied in-field correction

Zivid One+ S Typical Specifications

Typical numbers are given at common conditions unless otherwise specified.

Property	Description	Typical
Warm-up time	Minimum recommended time needed for camera to stabilize from an idle state assuming capturing at a constant rate. Some trueness changes may be experienced during warm-up phase.	10 minutes
Point precision	1σ Euclidian distance variation for a point between consecutive measurements at focus distance, D. [2]	25 µm
Local Planarity Precision	1σ Euclidian distance variation from a plane for a set of points within a smaller local region at focus distance, D. [2]	40 µm
Global Planarity Trueness	Average deviation from a plane in field of view at focus distance, D.	< 100 µm
Dimension Trueness	70-percentile dimension error in field of view at focus distance, D, and typical temperature range.	< 0.15 %
	70-percentile dimension error in field of view within optimal working distance and typical temperature range.	< 0.20 %
	70-percentile dimension error in field of view within optimal working distance and full temperature range.	< 0.30 %

[2] Measured with Gaussian filter disabled.

Figure 7 – Typical Zivid One+ S Point Precision vs. Distance

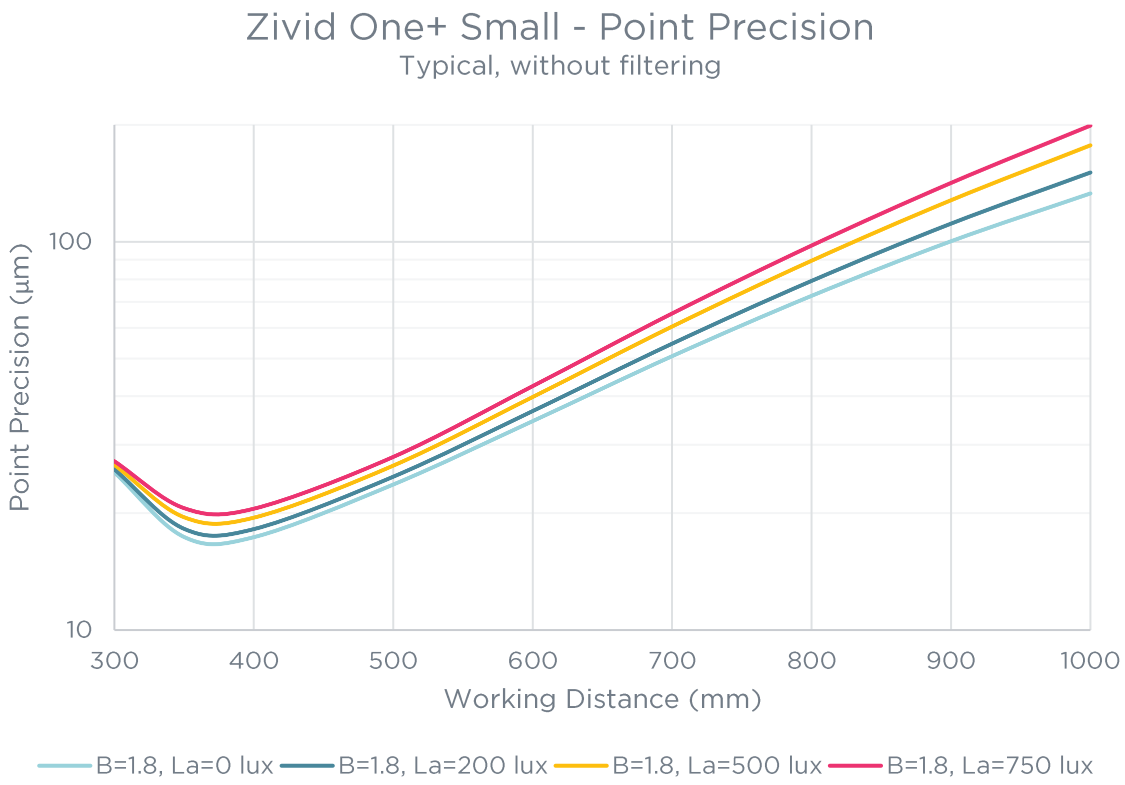


Figure 8 - Typical Zivid One+ S Local Planarity Precision vs. Distance

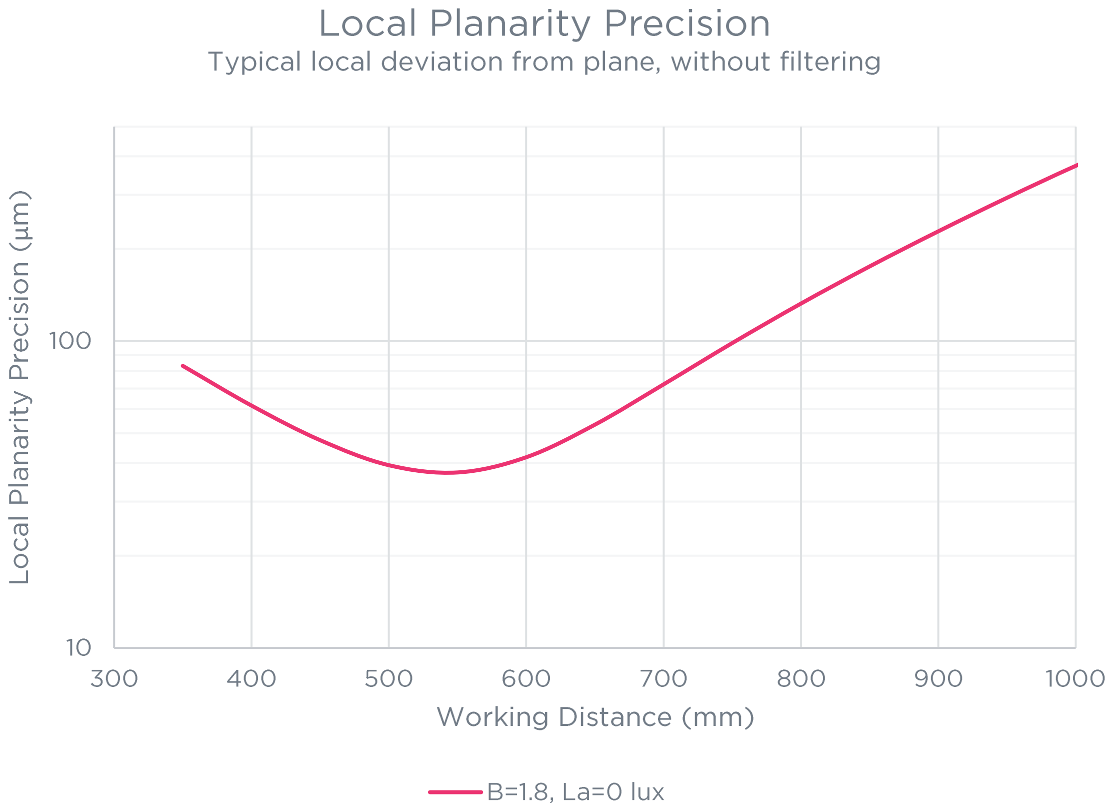


Figure 9 – Typical Zivid One+ S Global Planarity Trueness vs. Distance

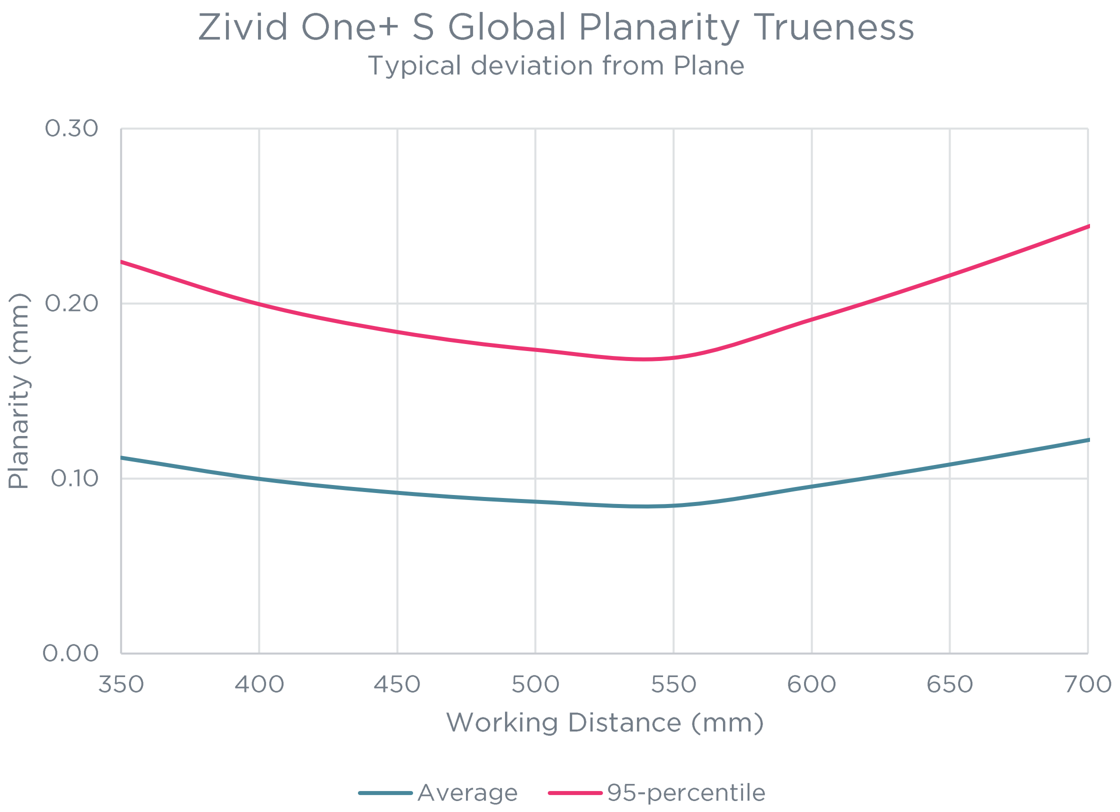
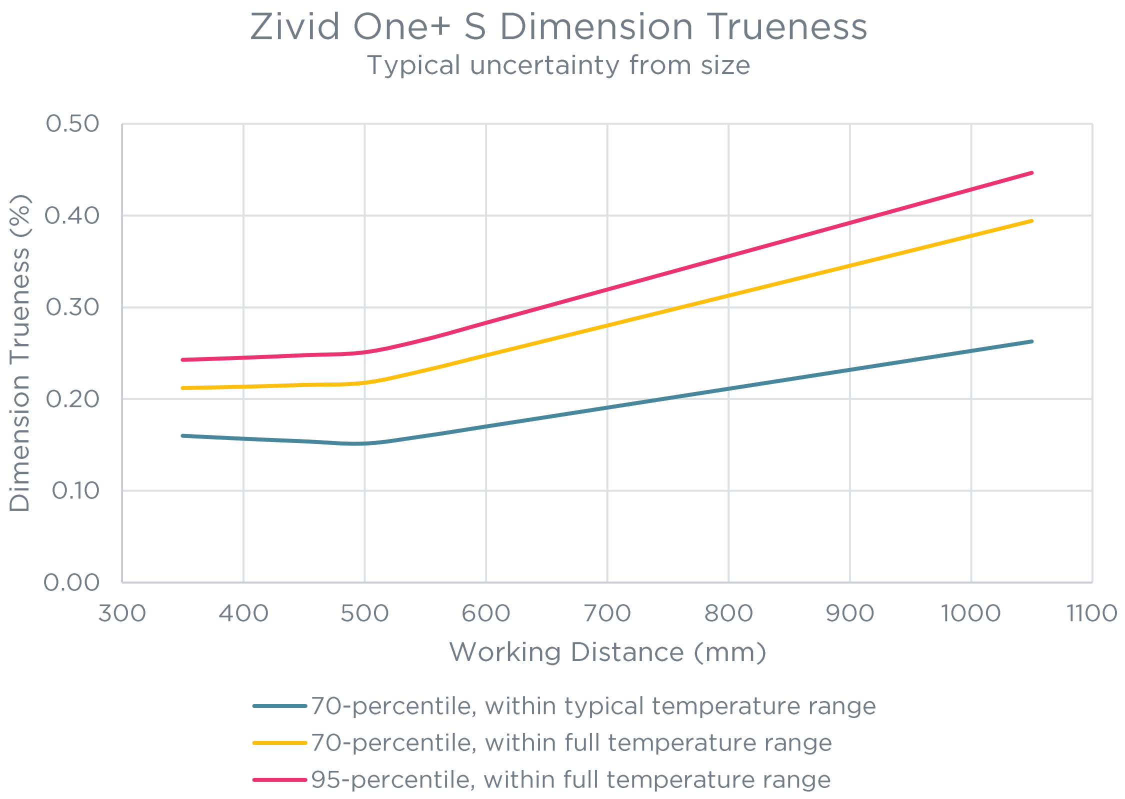


Figure 10 – Typical Zivid One+ S Dimension Trueness vs. Distance



Zivid One+ M Typical Specifications

Typical numbers are given at common conditions unless otherwise specified.

Property	Description	Typical
Warm-up time	Minimum recommended time needed for camera to stabilize from an idle state assuming capturing at a constant rate. Some trueness changes may be experienced during warm-up phase.	10 minutes
Point precision	1σ Euclidian distance variation for a point between consecutive measurements at focus distance, D. [3]	110 µm
Local Planarity Precision	1σ Euclidian distance variation from a plane for a set of points within a smaller local region at focus distance, D. [3]	190 µm
Global Planarity Trueness	Average deviation from a plane in field of view at focus distance, D.	< 100 µm
Dimension Trueness	70-percentile dimension error in field of view at focus distance, D, and typical temperature range.	< 0.30 %
	70-percentile dimension error in field of view within optimal working distance and typical temperature range.	< 0.40 %
	70-percentile dimension error in field of view within optimal working distance and full temperature range.	< 0.50 %

[3] Measured with Gaussian filter disabled.

Figure 11 – Typical Zivid One+ M Point Precision vs. Distance

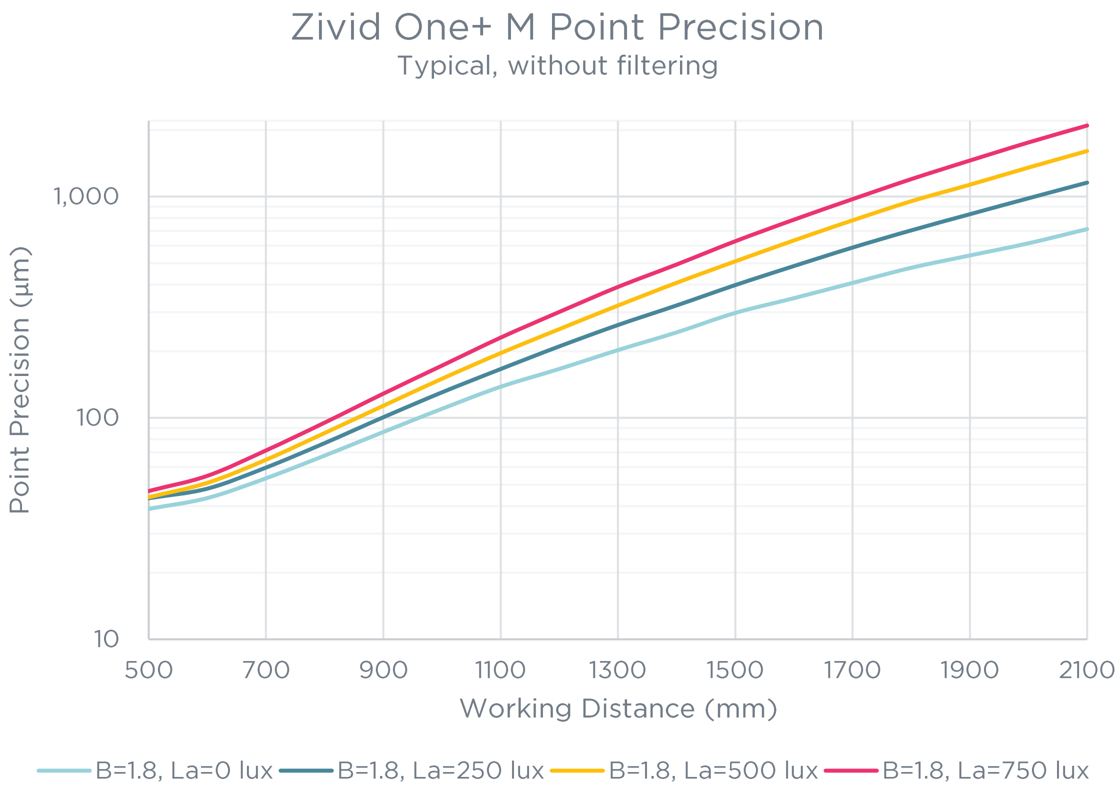


Figure 12 – Typical Zivid One+ M Local Planarity Precision vs. Distance

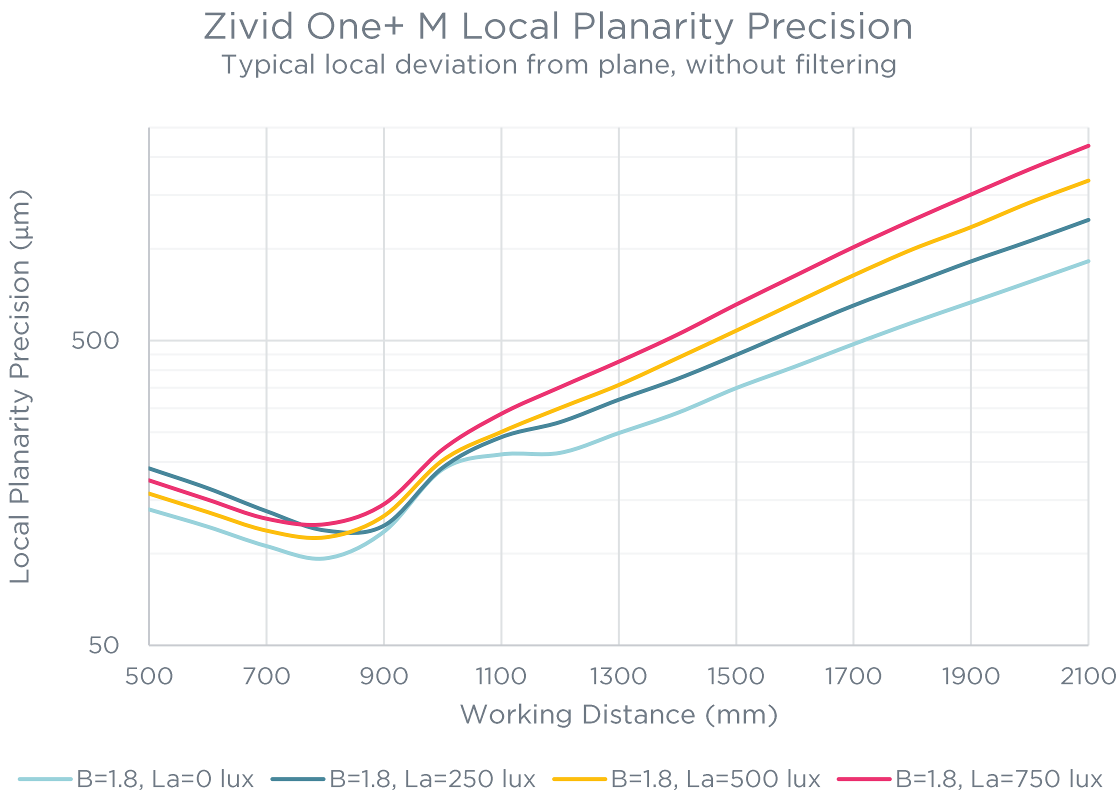


Figure 13 – Typical Zivid One+ M Global Planarity Trueness vs. Distance

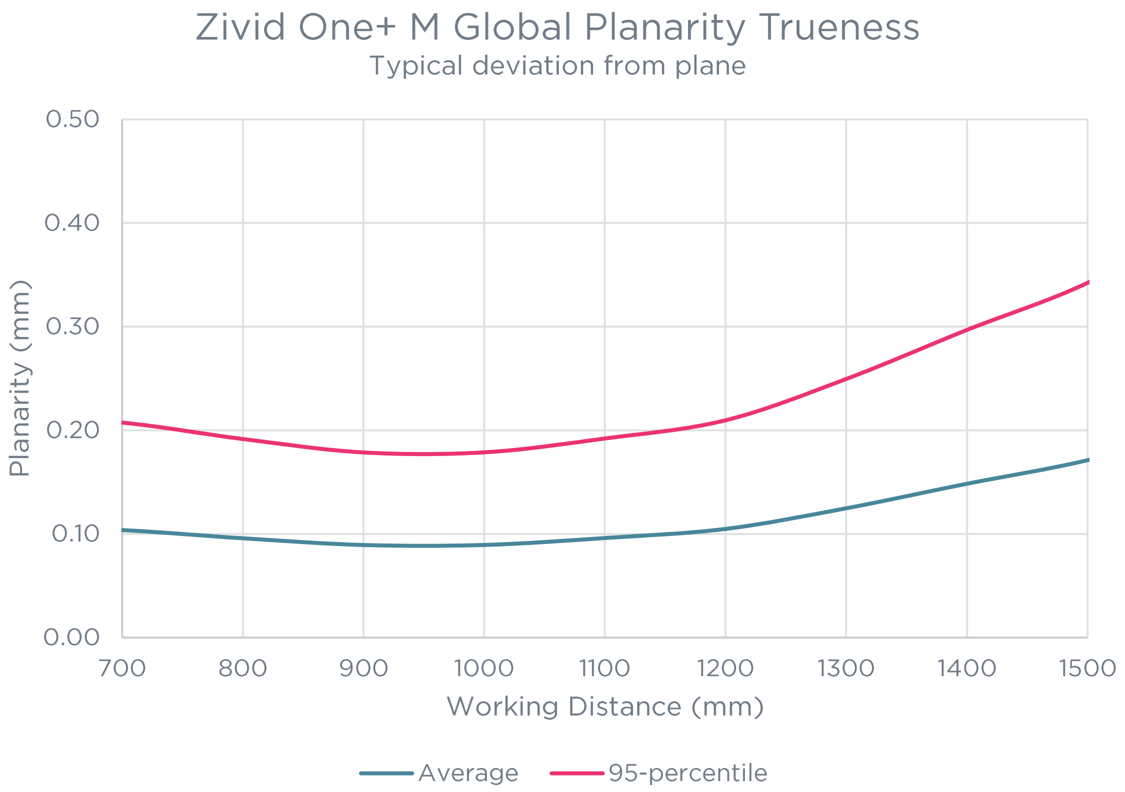
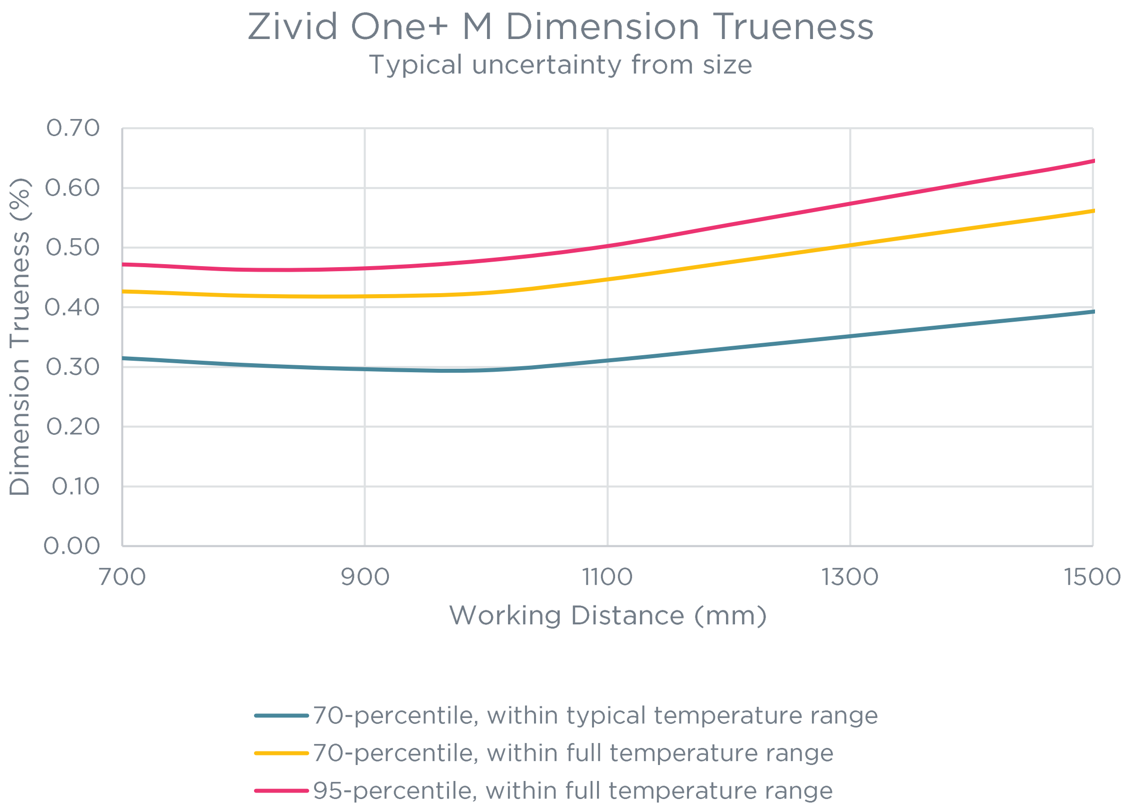


Figure 14 – Typical Zivid One+ M Dimension Trueness vs. Distance



Zivid One+ L Typical Specifications

Typical numbers are given at common conditions unless otherwise specified.

Property	Description	Typical
Warm-up time	Minimum recommended time needed for camera to stabilize from an idle state assuming capturing at a constant rate. Some trueness changes may be experienced during warm-up phase.	10 minutes
Point precision	1σ Euclidian distance variation for a point between consecutive measurements at focus distance, D. [4]	350 µm
Local Planarity Precision	1σ Euclidian distance variation from a plane for a set of points within a smaller local region at focus distance, D. [4]	700 µm
Global Planarity Trueness	Average deviation from a plane in field of view at focus distance, D.	< 350 µm
Dimension Trueness	70-percentile dimension error in field of view at focus distance, D, and typical temperature range.	< 0.50 %
	70-percentile dimension error in field of view within optimal working distance and typical temperature range.	< 0.60 %
	70-percentile dimension error in field of view within optimal working distance and full temperature range.	< 0.70 %

[4] Measured with Gaussian filter disabled.

FIGURE 15 – TYPICAL ZIVID ONE+ L POINT PRECISION VS. DISTANCE

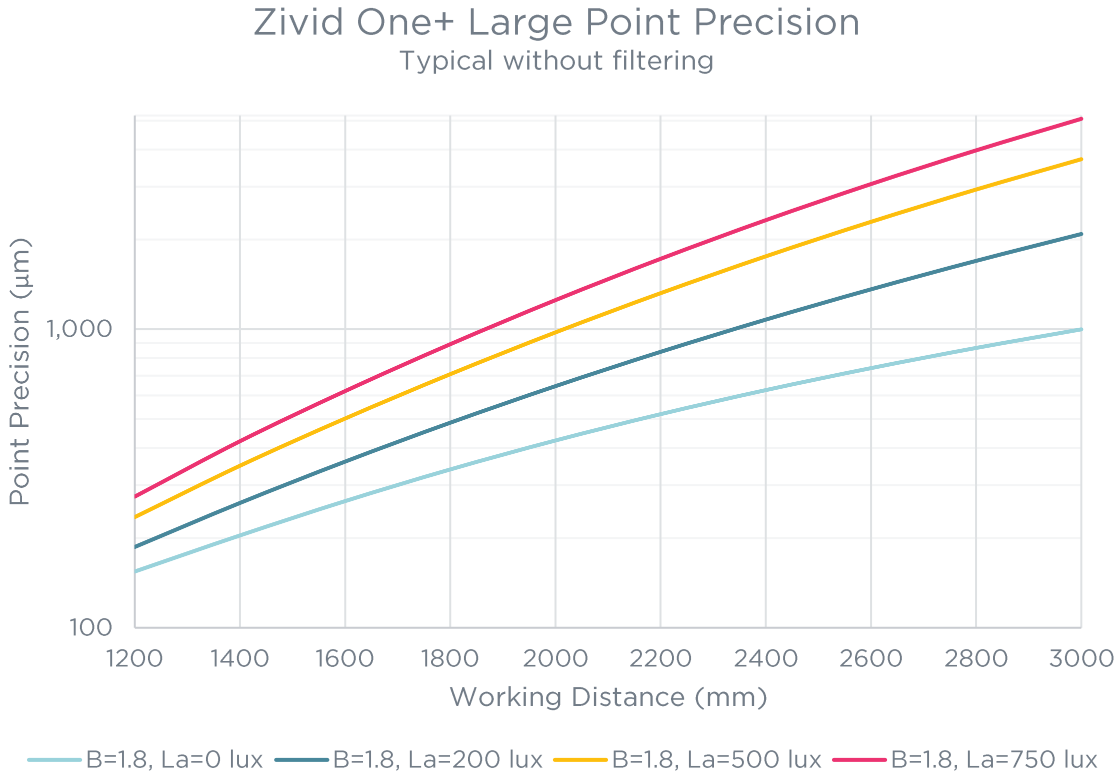


FIGURE 16 - TYPICAL ZIVID ONE+ L LOCAL PLANARITY PRECISION VS. DISTANCE

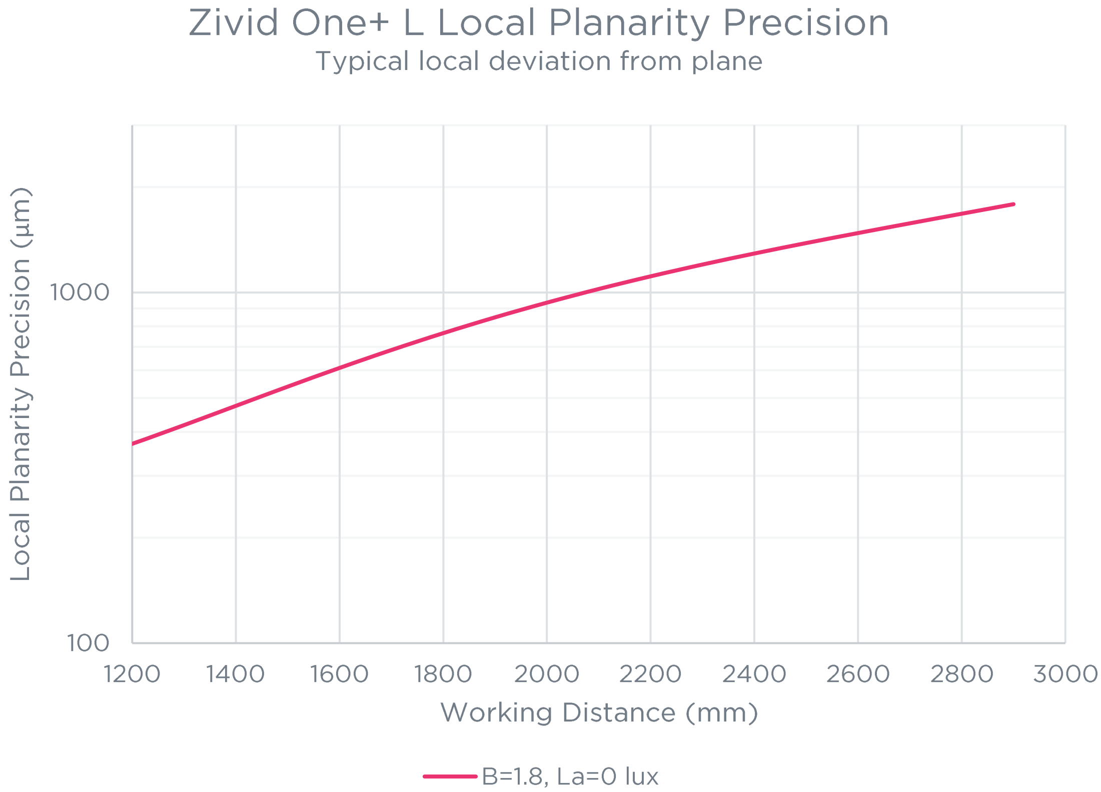


FIGURE 17 – TYPICAL ZIVID ONE+ L GLOBAL PLANARITY TRUENESS VS. DISTANCE

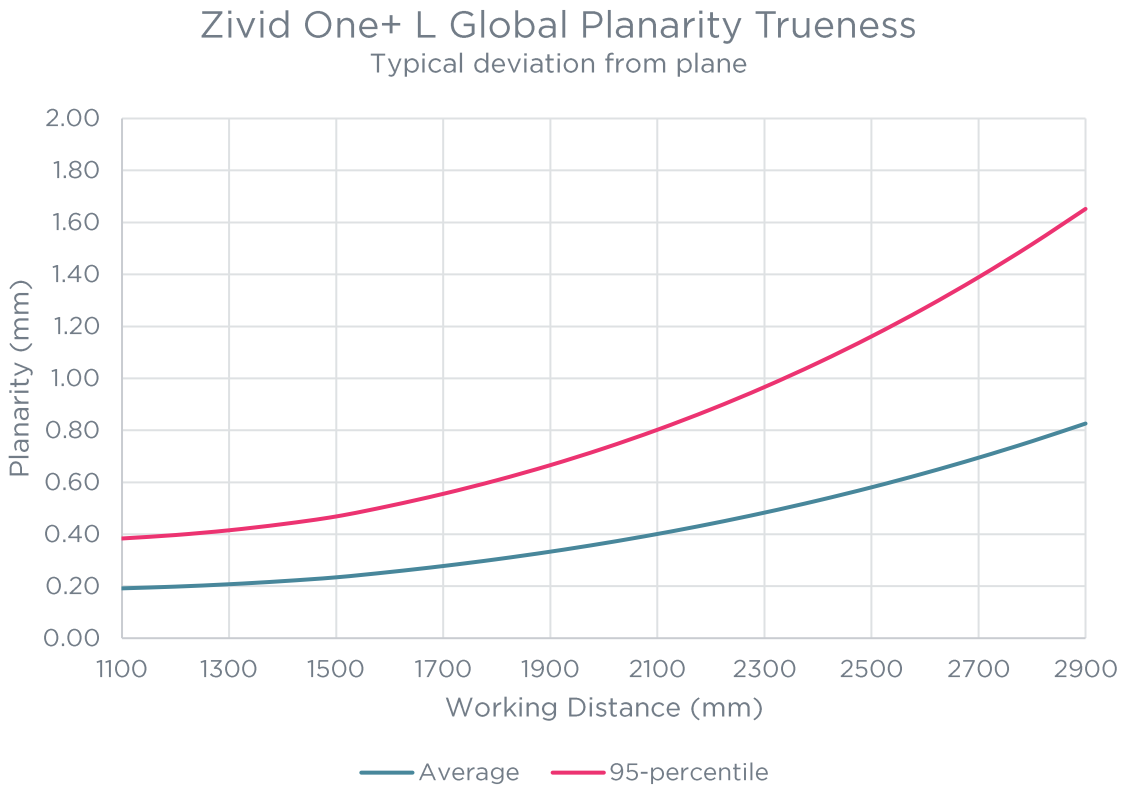
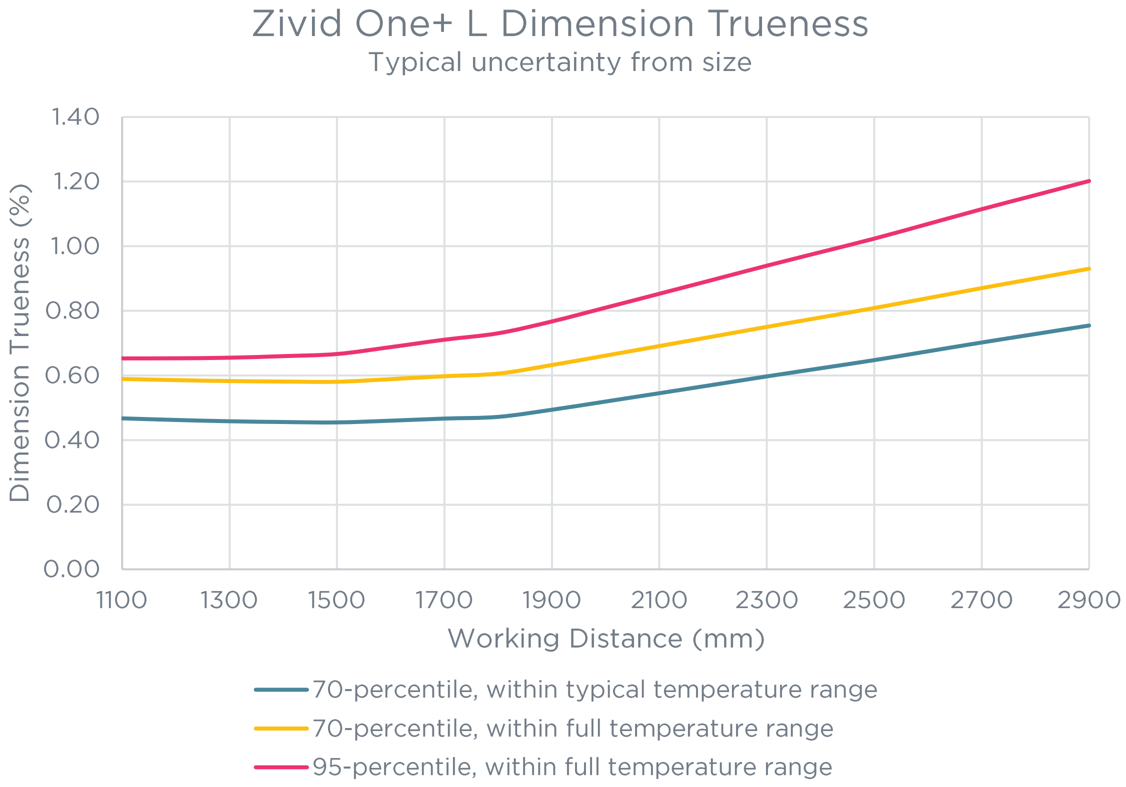


FIGURE 18 – TYPICAL ZIVID ONE+ L DIMENSION TRUENESS VS. DISTANCE



Physical specifications

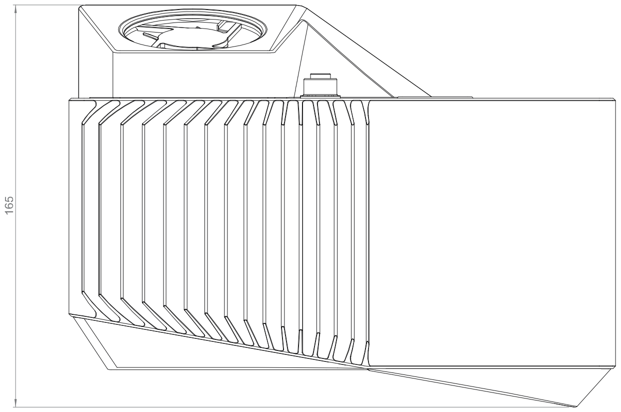
Size	226 mm x 165 mm x 86 mm
Weight	2 kg
Cable strain limit, power	90 N
Cable strain limit, data	30 N
Environmental	IP65 5 g sinusoidal [5] 15 g shock [6]
Operating temperature	10° to 40° C
Storage temperature	-20° to 60° C
Data connector	USB 3.0 SuperSpeed USB Type B Jack screw M2
Power connector	M12-5
Power adapter	24V ⎓ 5A EU, US, and UK power plug options
Power consumption, typical	15 W, Idle 45 W, TDP [7] 120 W, Peak

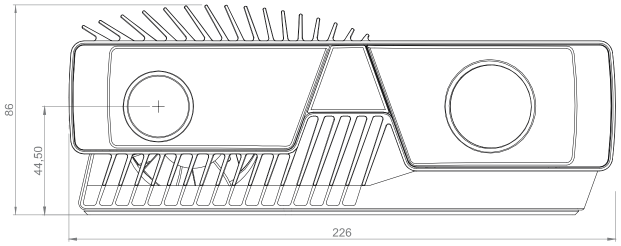
[5] IEC 60068-2-6, 10-150 Hz, 5 g, in X, Y and Z direction, 2 hour per axis. Sweep rate 1 octave per minute sweep rate.

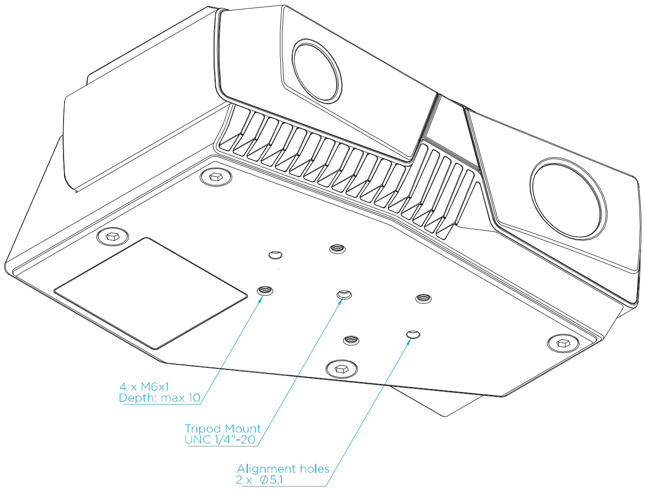
[6] IEC 60068-2-27, 15 g / 11 ms half sine shock pulses. 3 shocks per direction, 18 shocks in total.

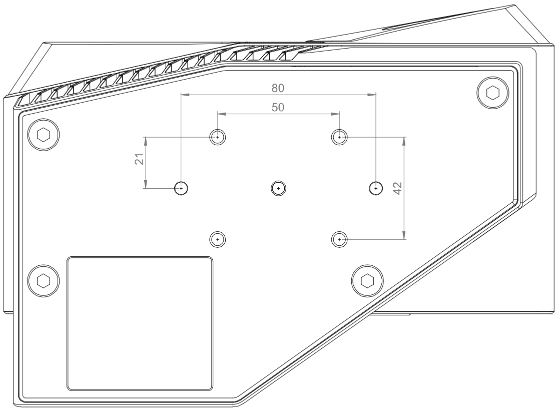
[7] Thermal Design Power is the maximum power consumed by the camera when capturing 3D images in a continuous stream.

Mechanical drawings

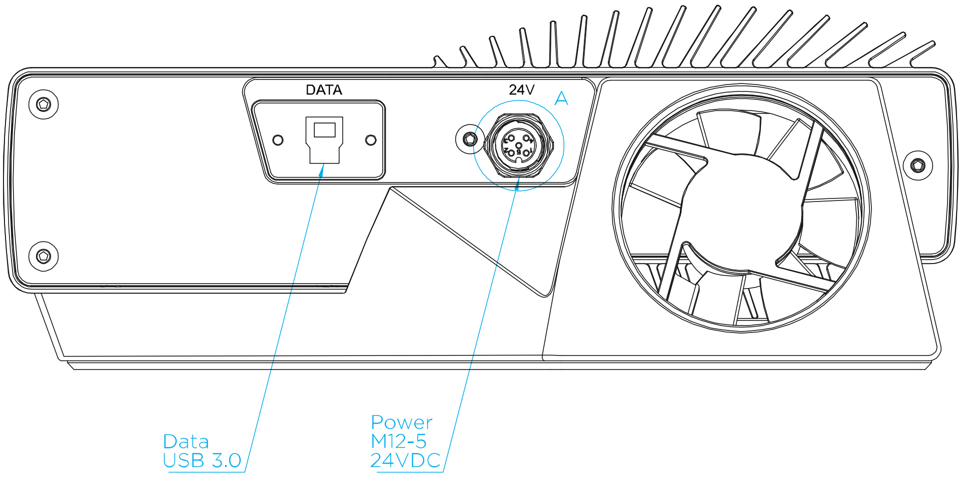


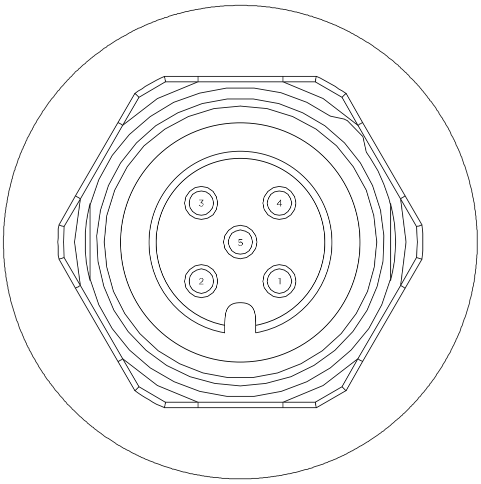






Connectors





DETAIL A
Power Inlet

Pin	
1	24V DC +/- 20% Max 4A
2	24V DC +/- 20% Max 4A
3	GND
4	GND
5	NC

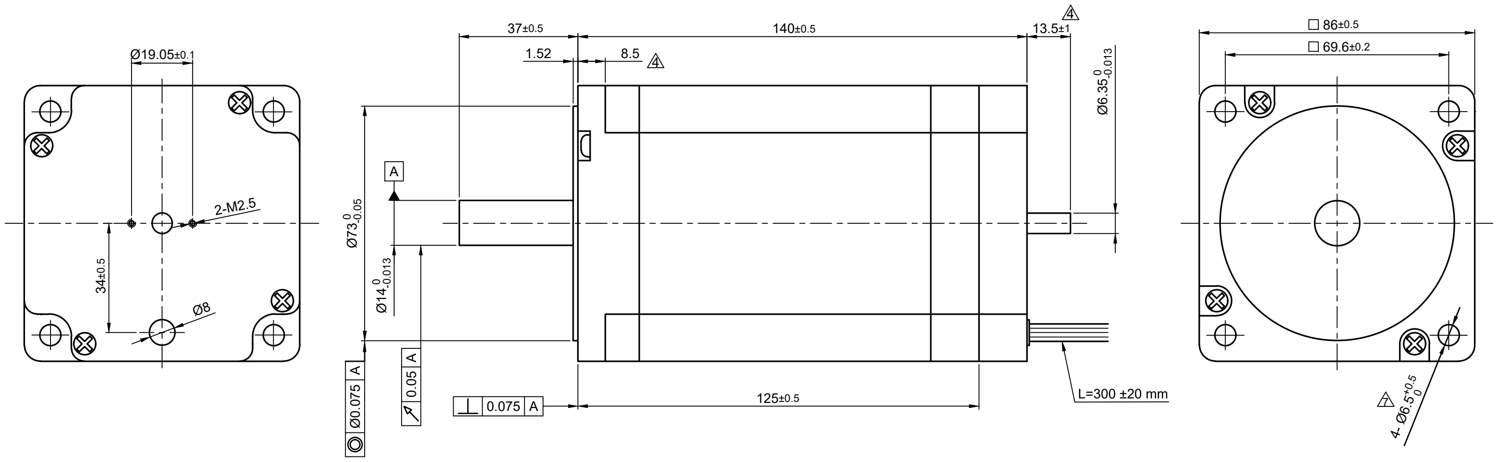
Revision history

Ver.	Date	Notes
1.0	05/21	Updated front page Updated table of contents Updated “General specifications” and added valid revision number Updated “Operating distance and field of view” Added figure 2-16 Updated “Accuracy specifications” and table “Common conditions” Updated table “Zivid One+ S Typical Specifications” Updated table “Zivid One+ M Typical Specifications” Updated table “Zivid One+ L Typical Specifications” Updated table “Physical specifications”
0.91	05/19	Added metrics for Zivid One+ Small. Added metrics for Zivid One+ Large. Updated Operating distance and field of view table. Updated Zivid One+ Medium spec. plots. Updated connector table.
0.9	04/19	Initial version.

Zivid AS
Gjerdrums vei 10A
N0484 Oslo
Norway

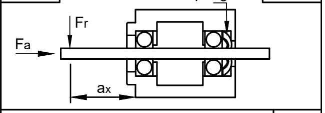
Appendix D

Nanotec DC motor datasheet



MOTOR SPECIFICATION		
No. of Poles		8
Rated Voltage	V DC	48
Current - Rated / Peak	A	17.95 / 53.85
Resistance Line to Line △6 △5 ±15%	Ω	0.097 △5 △6
Inductance Line to Line (1kHz) ±20%	mH	0.3 △5 △6
Torque - Rated / Peak	Nm	2.1 / 6.3
Torque Constant	Nm/A	0.117
Rated Power	W	660 △3
Speed - No Load / Rated ±10%	rpm	4500 / 3000
Rotor Inertia	kg m²	240 x10^{-6}

WIRING DIAGRAM			
	Colour	Function	Lead Gauge
Motor 8 Pol.	Ye	U	AWM3135 AWG16 △1 △2
	Rd	V	
	Bk	W	
Hall 24 Impl. per Rev.	Rd	+5V	UL1332 AWG22 △1
	Bu	H1	
	Wh	H2	
	Gn	H3	
	Bk	GND	



Max. Axial Force Fa	N	60
Max. Radial Force Fr (a2 = 20 mm)	N	220
Axial Play Fa = 4.5 N	mm	0.08
Radial Play Fr = 4.5 N	mm	0.02

GENERAL MOTOR SPECIFICATION		
Ambient Temperature	°C	-10 ...50
Max. Temperature Rise (at standstill - 2 phases energized)	°C	80
Max. Ambient Humidity (non condensing)	%	85
Insulation Class		B
Insulation Resistance	MΩ	100
Dielectric Strength (for 1 min - coil to case)	V AC	500

ISO 8015	ISO 1302	ISO 2768 cK	ISO 13715			
					Date	Name
				Drawn	04.12.2017	Import
				Reviewed	10.04.2018	Schneid_A
07	change diameter	Schneid_A	14.04.2022	Released	10.04.2018	Schneid_A
06	change induc./resist.					
REV	Rev. Text	Name	Date			

Nanotec® PLUG & DRIVE

Weight: ~4.0 kg

DB87L01-S

03000132

A4 Page 1

State: Released

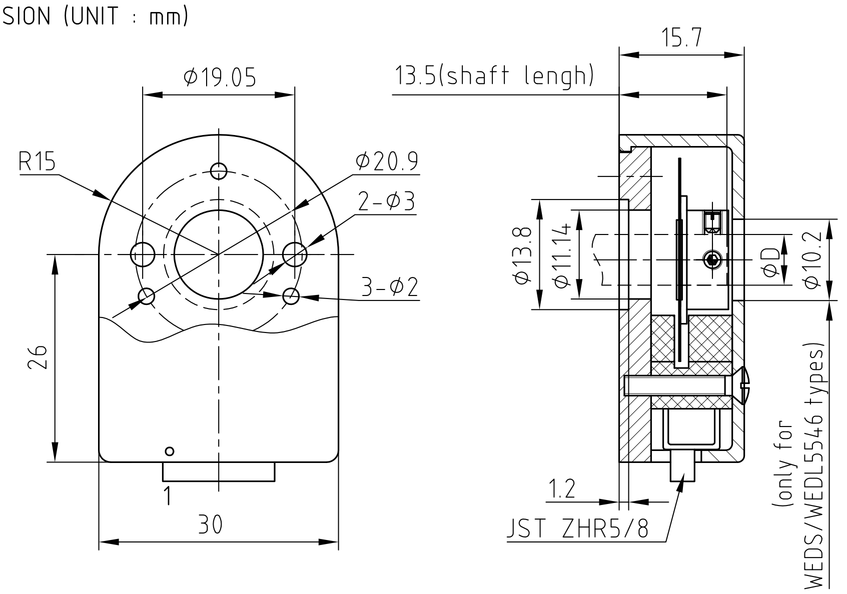
Rev: 07

CONFIDENTIAL

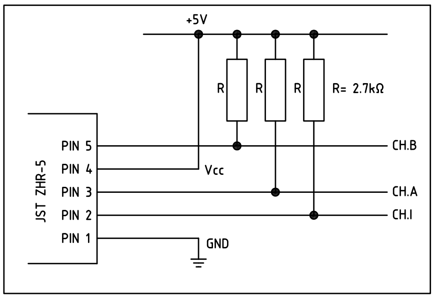
Appendix E

Nanotec encoder Datasheet

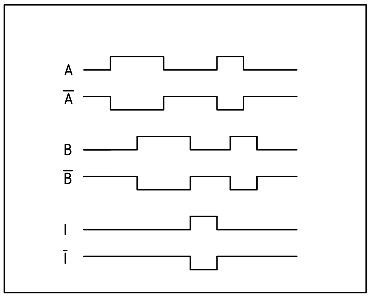
DIMENSION (UNIT : mm)



WEDS CONNECTOR CONFIGURATION



WEDL WITH LINE DRIVER OUTPUT SIGNALS



CONNECTOR CONFIGURATION

Driver output	0V	I	A	Vcc	B			
Coding system of the flat ribbon cable	1 (RD)	2	3	4	5			
Core color ZK-WEDS... cable	Black	Yellow	Green	Red	White			
Line driver output	0V	Vcc	A	A\	B\	B	I\	I
Coding system of the flat ribbon cable	1 (RD)	2	3	4	5	6	7	8
Core color ZK-WEDL... cable	Black	Red	Green	Brown	Grey	White	Yellow	Orange

				Nanotec® PLUG & DRIVE	SCALE FREE	APVD	S.R.	10.09.09	WEDL/WEDS 5541(5546)
					X ±0.5	CHKD			
A	-	22.03.16	A.S.		1PL ±0.2	DRN	J.W.	10.09.09	DWG.NO
REV	DESCRIPTION	DATE	DRN	WEDL/WEDS5541(500CPR)	2PL ±0.1 ANGLE ±30'	SIGNATURE		DATE	WEDL/WEDS5541 (500CPR)

Appendix F

Nanotec N5 2-1 motor controller quick guide

N5-1-1, N5-2-1

Short instructions
Original: de

Version 1.0.0

Nanotec Electronic GmbH & Co. KG
Kapellenstraße 6
85622 Feldkirchen, Germany

Phone: +49 (0)89-900 686-0
Fax: +49 (0)89-900 686-50
info@nanotec.de

Introduction

The *N5* is a controller for the *open loop* or *closed loop* operation of stepper motors and the *closed loop* operation of BLDC motors.

This document describes the installation and commissioning of the controller.You can find the detailed documentation for the product on the Nanotec website **us.nanotec.com**. The short instructions do not replace the technical manual oft he product.

Copyright, marking and contact

CE

Intended use

The *N5 controller* is used to control stepper and BLDC motors and is designed for use under the approved **Environmental conditions**.

Any other use is considered unintended use.

Note

Changes or modification to the controller are not permitted.

Warranty and disclaimer

Nanotec produces component parts that are used in a wide range of industrial applications. The selection and use of Nanotec products is the responsibility of the system engineer and end user. Nanotec accepts no responsibility for the integration of the products in the end system.

Under no circumstances may a Nanotec product be integrated as a safety controller in a product or construction. All products containing a component part manufactured by Nanotec must, upon delivery to the end user, be provided with corresponding warning notices and instructions for safe use and safe operation. All warning notices provided by Nanotec must be passed on directly to the end user.

Our general terms and conditions apply: **en.nanotec.com/service/general-terms-and-conditions/**.

Specialist staff

Only specialists may install, program and commission the device:

- Persons who have appropriate training and experience in work with motors and their control.
- Persons who are familiar with and understand the content of this technical manual.
- Persons who know the applicable regulations.

EU directives for product safety

The following EU directives were observed:

- RoHS directive (2011/65/EU, 2015/863/EU)
- EMC directive (2014/30/EU)

Other applicable regulations

In addition to this technical manual, the following regulations are to be observed:

- Accident-prevention regulations
- Local regulations on occupational safety

Safety and warning notices

Note

- Damage to the controller.
- Changing the wiring during operation may damage the controller.
- Only change the wiring in a de-energized state. After switching off, wait until the capacitors have discharged.

Note

- Fault of the controller due to excitation voltage of the motor.
- Voltage peaks during operation may damage the controller.
- Install suitable circuits (e.g., charging capacitor) that reduce voltage peaks.

Note

- There is no polarity reversal protection.
- Polarity reversal results in a short-circuit between supply voltage and GND (earth) via the power diode.
- Install a line protection device (fuse) in the supply line.

Note

- The device contains components that are sensitive to electrostatic discharge.
- Improper handling can damage the device.
- Observe the basic principles of ESD protection when handling the device.

Technical details and pin assignment

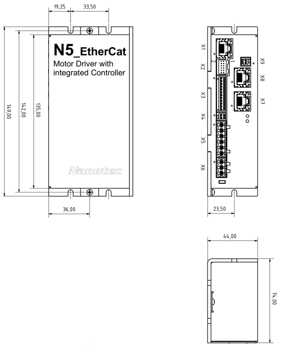
Environmental conditions

Environmental condition	Value
Protection class	IP20
Ambient temperature (operation)	-10 … +40°C
Air humidity (non-condensing)	0 … 95 %
Altitude of site above *sea level* (without drop in performance)	1500 m
Ambient temperature (storage)	-25 … +85°C

Electrical properties and technical data

Property	Description / value
Operating voltage	• 12 V-5% …72 V +4% DC for *low-current version* with designation N5-1-1 • 12 V - 48 V +/-5% DC for the *high-current version* with designation N5-2-1 and up to **hardware version** w007 • 12 V -5% …57.4 V DC for the *high-current version* with designation N5-2-1 and from **hardware version** w007b
Rated current	N5-1-1 (*low current*): 10 A_{rms} N5-2-1 (*high current*): 18 A_{rms}
Peak current	N5-1-1 (*low current*): 10 A_{rms} N5-2-1 (*high current*): 40 A_{rms} for 5 seconds
Commutation	Stepper motor – open loop, stepper motor – closed loop with encoder, BLDC motor – closed loop with Hall sensor, and BLDC motor – closed loop with encoder
Operating modes	*Profile Position Mode*, *Profile Velocity Mode*, *Profile Torque Mode*, *Velocity Mode*, *Homing Mode*, *Interpolated Position Mode*, *Cyclic Sync Position Mode*, *Cyclic Sync Velocity Mode*, *Cyclic Synchronous Torque Mode*, *Clock-Direction Mode*
Set value setting / programming	*EtherCAT*, *Ethernet (REST with the NanoIP user interface)*, *clock-direction*, *analog*, *NanoJ program*
Interfaces	EtherCAT, Ethernet
Inputs	• 4 inputs, 5 V/24 V (inputs 1 to 4) individually switchable by means of software, factory setting: 5 V • 2 inputs, wide range 5-24 V (inputs 5 and 6); • 2 analog inputs -10 to +10 V or 0–20 mA (switchable by means of software)
Outputs	2 outputs, (open drain, 0 switching, max. 24 V and 500 mA)
Encoder input	5 V or 24 V signal, differential or single-ended (switchable by means of software), max. resolution 65536 increments per revolution (16-bit)
Protection circuit	Overvoltage and undervoltage protection Overtemperature protection (> 75° Celsius on the power board) Polarity reversal protection: In the event of a polarity reversal, a short-circuit will occur between supply voltage and GND over a power diode; a line protection device (fuse) is therefore necessary in the supply line. The values of the fuse are dependent on the application and must be dimensioned • greater than the maximum current consumption of the controller • less than the maximum current of the voltage supply. If the fuse value is very close to the maximum current consumption of the controller, a medium / slow tripping characteristics should be used.

Dimensioned drawings



Overtemperature protection

Above a temperature of approx. 75°C on the power board (corresponds to 65–72°C outside on the cover), the power part of the controller switches off and the error bit is set . After cooling down and confirming the error , the controller again functions normally.

LED signaling

Power LED

Normal operation

In normal operation, the green power LED L1 flashes briefly once per second.

Case of an error

If an error has occurred, the LED turns red and signals an error number.

The following table shows the meaning of the error numbers.

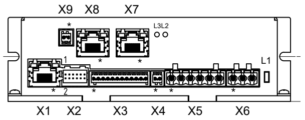
Flash rate	Error
1	General
2	Voltage
3	Temperature
4	Overcurrent
5	Controller
6	Watchdog-Reset

Note

For each error that occurs, a more precise error code is stored in object **1003**$_h$.

Pin assignment

Pin 1 is marked with an asterisk "*".



Connector	Function	Pin assignment / description
X1	Ethernet	Configuration interface
X2	Encoder and Hall sensor 5 V / 24 V DC signal Max. 1 MHz Switching thresholds: **5 V (factory setting)**: On: >3.8 V; Off: <0.26 V **24 V**: On: >14.42 V; Off: <4.16 V	**1.** GND **2.** Vcc: +5 V (factory setting) /24 V DC output, switchable with object 2059_h **3.** A **4.** B **5.** A\ **6.** B\ **7.** I **8.** I\ **9.** Hall 1 **10.** Hall 2 **11.** Hall 3 **12.** Shielding
X3	Inputs and outputs Switching thresholds for digital **inputs 1 - 4**: **5 V (factory setting)**: On: >3.8 V; Off: <0.26 V **24 V**: On: >14.42 V; Off: <4.16 V Switching thresholds for digital **inputs 5 - 6**: On: >3.25 V; Off: <2 V	**1.** GND **2.** Digital input 1; 5 V / 24 V Signal, switchable with object **3240**$_h$ **3.** Digital input 2; 5 V / 24 V Signal, switchable with object **3240**$_h$ **4.** Digital input 3: 5 V / 24 V, switchable with object **3240**$_h$, max. 1 MHz; *direction input* in clock/direction mode **5.** Digital input 4: 5 V / 24 V, switchable with object **3240**$_h$, max. 1 MHz; *clock input* in clock/direction mode **6.** Digital input 5; 5…24 V signal, not switchable **7.** Digital input 6; 5…24 V signal, not switchable **8.** Analog input 1: 10 Bit, 0-10 V oder 0-20 mA, switchable with object **3221**$_h$ **9.** Analog input 1: 10 Bit, 0-10 V oder 0-20 mA, switchable with object **3221**$_h$ **10.** Digital output 1: Open drain, max 24 V/500 mA **11.** Digital output 2: Open drain, max 24 V/500 mA **12.** Shielding
X4	Brake 24V Brakes have to be connected using an appropriate circuit if **+UB**>24 V!	**1.** Brake+: internally connected to **+UB** **2.** Brake -: PWM-controlled open-drain output, max 1.5 A

Connector	Function	Pin assignment / description
X5	Motor	**1.** Shielding **2.** A (Stepper) U (BLDC) **3.** A\ (Stepper) V (BLDC) **4.** B (Stepper) W (BLDC) **5.** B\ (Stepper) **6.** Shielding
X6	Voltage supply Permissible operating voltage: See *Electrical properties and technical data*	**1. Shielding** **2. +UB** **3. GND**
X7	EtherCAT IN	
X8	EtherCAT OUT	
X9	Supply for Encoder/ Hall sensor, external logic supply To be connected if 24V encoder is used or logic supply of the controller desired.	**1. +UB Logic / Encoder**: +24 V **2. GND**

Note

- EMC: For a DC power supply line longer than 30 m or when using the motor on a DC bus, additional interference-suppression and protection measures are necessary.
- An EMI filter is to be inserted in the DC supply line as close as possible to the controller/motor.
- Long data or supply lines are to be routed through ferrites.

Commissioning

The *Plug & Drive Studio* software offers you an option for performing the configuration and adapting the controller to the connected motor. You can find further information in document *Plug & Drive Studio: Quick Start Guide* at **us.nanotec.com**.

Observe the following note:

Note

- EMC: Current-carrying cables – particularly around supply and motor cables – produce electromagnetic alternating fields.
- These can interfere with the motor and other devices. Nanotec recommends the following measures:
- Use shielded cables and earth the cable shielding on both ends over a short distance.
- Use cables with cores in twisted pairs.
- Keep power supply and motor cables as short as possible.
- Earth motor housing with large contact area over a short distance.
- Lay supply, motor and control cables physically separate from one another.

Configuration via Ethernet

Establishing connection with the controller

Setting the IP address

Each of the connected devices (controller and communication partners) in an Ethernet network or with a point-to-point Ethernet connection requires a unique IP address. This can either be obtained automatically (DHCP) or generated (Auto-IP) or assigned statically. In the following, "communication partner" refers to a PC or laptop.

You can integrate the controller in an existing Ethernet network. To do this, you only need to establish the physical connection with a standard Ethernet cable. Provided DHCP and UPnP are activated on the controller (factory setting), the controller is also automatically detected on the network and can immediately be operated via a PC located on the network.

Setting DHCP/Auto-IP

IP addresses can be obtained dynamically in a network from a DHCP server or, for example, in the case of a PC direct connection, can be automatically self-generated without DHCP by the two communication devices (e.g., PC and controller). DHCP and UPnP are preset in the controller at the factory for automatically obtaining an IP address from a DHPC server or for automatic IP address generation. To establish the connection to the controller, it may only be necessary to make a few settings on the communication partner (e.g., PC or laptop). Settings using the Windows 7 operating system as an example:

1. Press the Windows Start button and select *Control Panel*.
2. Select *Network and Sharing Center*.
3. Select *Change adapter settings*.
4. A list of the available network adapters is displayed. Open the properties on the adapter to which the controller is connected (e.g., click with the right mouse button).
5. *Select Internet Protocol version 4 (TCP/IPv4)* and press the *Properties* button.
6. Select the *Obtain an IP address automatically* option.
7. Confirm acceptance of the entries with the *OK* button.

For the communication partner to automatically detect the controller in the entire network or for a point-to-point connection (PC direct connection), network discovery must be switched on and the UPnP service must be started on the communication partner (e.g., PC or laptop). No further settings are necessary on the controller. Settings using the Windows 7 operating system as an example:

1. Switching on network discovery:
   a. Press the Windows Start button and select *Control Panel*.
   b. Select *Network and Sharing Center*.
   c. Select *Change advanced sharing settings*.
   d. Open the *Public* section.
   e. Under *Network discovery*, select the *Turn on network discovery* option.
2. Activating the UPnP service:
   a. Press the Windows Start button and then right-click on *Computer* and select *Manage*.
   b. Open the *Services and Applications* node and select *Services*.
   c. Double-click the *UPnP device host* service to open.
   d. As *Startup type*, select *Automatic* and press the *Start* button.
   e. Confirm acceptance of the entries with the OK button.

Configuration via EtherCAT

Software connection

Tip

The following description assumes that an EtherCAT master from Beckhoff with the *TwinCAT* software is used.

1. Connect the EtherCAT master to the controller, see **Technical details and pin assignment**.
2. Supply the controller with voltage.
3. Obtain the *ESI file* that corresponds exactly to the used **firmware version** from the following sources:
   a. Via the Nanotec homepage **us.nanotec.com**. The current version of the firmware and the *ESI file* can be found in the *Plug & Drive Studio* download folder.
   b. From Nanotec support.
4. Close the *TwinCAT* system manager if it is open.
5. Then copy the *ESI file* to the *TwinCAT* subfolder:
   - If you use *TwinCAT* version 2, use folder `<TWINCAT INSTALL DIR>/Io/EtherCAT`
   - If you use *TwinCAT* version 3, use folder `<TWINCAT INSTALL DIR>/3.1/Config/Io/EtherCAT`

   Example

   Example: If *TwinCAT* 2 is installed on your computer under path `C:\TwinCAT\`, copy the *ESI file* to path `C:\TwinCAT\Io\EtherCAT\`.
6. Open the *ESI file* with an editor. Find the *AddInfo* parameter. Enter:
   - the value "2" if you would like to integrate the controller as *Box* (factory settings)
   - the value "0" if you would like to integrate the controller as *NC-Axis*

   Save and close the file.
7. Now restart the *TwinCAT system manager*. The *ESI files* are read in again following a restart.

Note

The cycle time of the sync signal must always be set to 1 ms. You can set the bus cycle time (and, consequently, the interpolation time in **60C2**$_h$) to integer multiples of 1 ms.

Setting the motor data

Prior to commissioning, the motor controller requires a number of values from the motor data sheet.

- Number of pole pairs: Object **2030**$_h$:00$_h$ (pole pair count) The number of motor pole pairs is to be entered here. With a stepper motor, the number of pole pairs is calculated using the step angle, e.g., 1.8° = 50 pole pairs, 0.9° = 100 pole pairs (see step angle in motor data sheet). With BLDC motors, the number of pole pairs is specified directly in the motor data sheet.
- Setting the motor current / motor type:
  - Stepper motor only: Object **2031**$_h$:00$_h$: Rated current (bipolar) in mA (see motor data sheet)
    - ▸ Object **2031**$_h$:00$_h$: Rated current (bipolar) in mA (see motor data sheet)
    - ▸ Object **3202**$_h$:00$_h$ (Motor Drive Submode Select): Defines motor type stepper motor, activates current reduction on motor standstill: 0000008h.
    - ▸ Object **2037**$_h$ (Open Loop Current Reduction Value/factor): the root mean square is specified to which the rated current is to be reduced if current reduction is activated in *Open Loop*.
  - BLDC motor only:
    - ▸ Object **2031**$_h$:00$_h$ Peak current in mA (see motor data sheet)
    - ▸ Object **203B**$_h$:01$_h$ Rated current in mA (see motor data sheet)
    - ▸ Object **203B**$_h$:02$_h$ Maximum duration of the peak current in ms (for initial commissioning, a value of 100 ms is recommended; this value is to be adapted later to the specific application).
    - ▸ Object **3202**$_h$:00$_h$ (Motor Drive Submode Select): Defines motor type BLDC: 00000041h
- Motor with encoder: Object **20592059**$_h$:00$_h$ (Encoder Configuration): Depending on the encoder version, one of the following values is to be entered (see motor data sheet):
  - Supply voltage 5V, differential: 00000000h
  - Supply voltage 24V, differential: 00000001h
  - Supply voltage 5V, single-ended: 00000002h
  - Supply voltage 24V, single-ended: 00000003h
- Motor with brake: Object **3202**$_h$:00$_h$ (Motor Drive Submode Select): The brake control is activated for the initial commissioning. Depending on the specific application, this configuration can be deactivated later if necessary. One of the following values is to be entered depending on the motor type:
  - Stepper motor, brake control (and **current reduction** while at standstill) activated: 0000000Ch
  - BLDC motor, brake control activated: 00000044h

Auto setup

To determine a number of parameters related to the motor and the connected sensors (encoders/Hall sensors), an auto setup is performed. **Closed Loop** operation requires a successfully completed auto setup.

Note

- Note the following prerequisites for performing the auto setup:
- The motor must be load-free.
- The motor must not be touched.
- The motor must be able to turn freely in any direction.
- No NanoJ programs may be running (object 2300$_h$:00$_h$ bit 0 = "0", see **2300h NanoJ Control**).

Tip

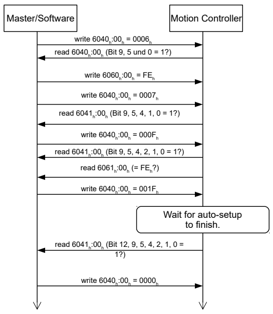
As long as the motor connected to the controller or the sensors for feedback (encoders/Hall sensors) are not changed, auto setup is only to be performed once during initial commissioning.

Execution

1. To preselect the *auto setup* operating mode, enter the value "-2" (="FE$_h$") in object 6060$_h$:00$_h$.
   The *power state machine* must now switch to the *Operation enabled* state.
2. Start *auto setup* by setting bit 4 *OMS* in object 6040$_h$:00$_h$ (controlword). While the auto setup is running, the following tests and measurements are performed in succession:

   To determine the values, the direction of the measurement method is reversed and edge detection re-evaluated.

   Value 1 in bit 12 *OMS* in object 6041$_h$:00$_h$ (statusword) indicates that the auto setup was completely executed and ended. In addition, bit 10 *TARG* in object 6041$_h$:00$_h$ can be used to query whether (= "1") or not (= "0") an encoder index was found.


Master/Software
Motion Controller
write 6040h:00h = 0006h
read 6040h:00h (Bit 9, 5 und 0 = 1?)
write 6060h:00h = FEh
write 6040h:00h = 0007h
read 6041h:00h (Bit 9, 5, 4, 1, 0 = 1?)
write 6040h:00h = 000Fh
read 6041h:00h (Bit 9, 5, 4, 2, 1, 0 = 1?)
read 6061h:00h (= FEh?)
write 6040h:00h = 001Fh
Wait for auto-setup to finish.
read 6041h:00h (Bit 12, 9, 5, 4, 2, 1, 0 = 1?)
write 6040h:00h = 0000h

CAUTION

- After executing auto setup mode, the internal coordinate system is no longer valid.
- *Homing* alone does not suffice! If the controller is not restarted, unexpected reactions may result.
- Restart the device after an auto setup!

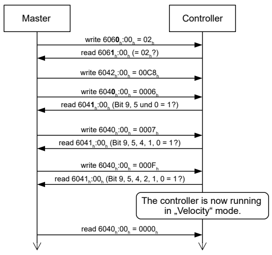
Test run

As an example, the **Velocity** operating mode is used.

The values are transferred from your *EtherCAT master* or to the controller. After every transfer, the *master* should use the status objects of the controller to ensure successful parameterization.

1. Select the *Velocity* mode by setting object **6060**$_h$ (Modes Of Operation) to the value "2".
2. Write the desired speed in **6042**$_h$.
3. Switch the *power state machine* to the *Operation enabled* state.

   The following sequence starts *Velocity* mode; the motor turns at 200 rpm.



4. To stop the motor, set controlword (**6040**$_h$) to "0".

Appendix G

Source code, field of view software

```python
"""
README

This project is developed to be used for proof-of-concept in my bachelor's thesis.
A pallet is represented by an array of 3D points.
A camera is represented by a single point in space facing a specified direction along with a specified field of view.
The resulting 3D-plot shows green and red points. Green means seen, red means unseen.
In fov check-mode the resulting plot will show points within the camera's FOV as blue, regardless if it can actually
be seen or not. This mode is used to verify camera placements where the objective is to see as much of the points as
possible.
"""

import time
import numpy as np
import matplotlib.pyplot as plt
import multiprocessing


class Queue:
    def __init__(self):
        self.queue = []

    # Check to see whether or not queue is empty
    def isEmpty(self) -> bool:
        return True if len(self.queue) == 0 else False

    # Return the first element of the queue
    def front(self):
        return self.queue[0]

    # Return the last element of the queue
    def rear(self):
        return self.queue[-1]

    # Return the value of and remove the element passed to the method
    def pop(self, value):
        return self.queue.remove(value)

    # Return length of queue
    def length(self) -> int:
        return len(self.queue)

    # Return a copy of the queue
    def copy(self):
        return self.queue

# Class includes all specification for camera and methods used for FOV calculations
class Camera:
    def __init__(self, x, y, z, fov, focus, range):
        self.x = x
        self.y = y
        self.z = z
        self.fov = fov
        self.focus = focus
        self.min_range = range[0]
        self.max_range = range[1]
        self.checked_points = []

    # Return list of points not seen by the camera
    def check_fov(self, point, possible_obstructions, planar_equations, step, tC, tP):
        x, y, z = self.calculate_coordinates(point, step)
        confirmed_obstructions = self.check_for_obstructions(planar_equations, possible_obstructions, x, y, z, point, step, tC, tP)

        return confirmed_obstructions

    # Return array of coordinates for the ray between camera and point
    def calculate_coordinates(self, point, step):
        x = np.linspace(self.x, point[0], step, endpoint=False)
        y = np.linspace(self.y, point[1], step, endpoint=False)
        z = np.linspace(self.z, point[2], step, endpoint=False)

        return x, y, z

    # Checks if ray passes through any surface
    @staticmethod
    def check_for_obstructions(equations, surfaces, x, y, z, point,
                               step, threshold_crossing, threshold_point):
        for j in range(step):
            for k, l in enumerate(equations):
                planar_result = l[0] * (x[j] - l[1]) + l[2] * (y[j] - l[3]) + l[4] * (z[j] - l[5])
                crossing_plane = abs(planar_result) < threshold_crossing
                if crossing_plane:
                    # Calculates coordinates of margin-box around surface
                    corners_x = (np.min([surfaces[k][0][0], surfaces[k][1][0],
                                         surfaces[k][2][0], surfaces[k][3][0]]),
                                 np.max([surfaces[k][0][0], surfaces[k][1][0],
                                         surfaces[k][2][0], surfaces[k][3][0]]))
                    corners_y = (np.min([surfaces[k][0][1], surfaces[k][1][1],
                                         surfaces[k][2][1], surfaces[k][3][1]]),
                                 np.max([surfaces[k][0][1], surfaces[k][1][1],
                                         surfaces[k][2][1], surfaces[k][3][1]]))
                    corners_z = (np.min([surfaces[k][0][2], surfaces[k][1][2],
                                         surfaces[k][2][2], surfaces[k][3][2]]),
                                 np.max([surfaces[k][0][2], surfaces[k][1][2],
                                         surfaces[k][2][2], surfaces[k][3][2]]))
                    # Checks whether or not ray passes through surface and that the surface is not part of the same one
                    # as the point itself
                    confirmed_obstruction = (corners_x[0] - threshold_point) <= x[j] <= (corners_x[1] + threshold_point) and \
                             (corners_y[0] - threshold_point) <= y[j] <= (corners_y[1] + threshold_point) and \
                             (corners_z[0] - threshold_point) <= z[j] <= (corners_z[1] + threshold_point) and \
                             not (point[0] - threshold_point <= x[j] <= point[0] + threshold_point and
                                  point[1] - threshold_point <= y[j] <= point[1] + threshold_point and
                                  point[2] - threshold_point <= z[j] <= point[2] + threshold_point)

                    if confirmed_obstruction:
                        return True

        return False

    # Calculates field of view vectors and determines which points are within the field of view
    def check_points_in_fov(self, points):
        focus_vector = self.get_focus_vector()
        focus_axis_1 = self.get_focus_axis_1(focus_vector)
        focus_axis_2 = get_focus_axis_2(focus_vector, focus_axis_1)
        fov_vectors = get_fov_vectors(focus_axis_1, focus_axis_2, focus_vector, self.fov)
        unit_vectors = []
        for i in range(len(fov_vectors)-1):
            normal_vector = np.cross(fov_vectors[i], fov_vectors[i+1])
            unit_vector = normal_vector / (normal_vector**2).sum()**0.5
            unit_vectors.append(unit_vector)
        normal_vector = np.cross(fov_vectors[-1], fov_vectors[0])
        unit_vector = normal_vector / (normal_vector ** 2).sum() ** 0.5
        unit_vectors.append(unit_vector)

        points_outside_fov = []
        for i in points:
            dist = np.sqrt((i[0] - self.x) ** 2 + (i[1] - self.y)**2 + (i[2] - self.z) ** 2)
            if self.min_range <= dist <= self.max_range:
                for j in unit_vectors:
                    planar_result = np.dot(np.array([i[0], i[1], i[2]]) - np.array([self.x, self.y, self.z]), j)
                    if planar_result > 0:
                        points_outside_fov.append(i)
                        break
                    else:
                        pass
            else:
                points_outside_fov.append(i)

        possible_points = []
        for i in points:
            if i not in points_outside_fov:
                # Points that are not outside of field of view are within
                possible_points.append(i)

        return possible_points, points_outside_fov, unit_vectors

    # Return vector between camera and focus point
    def get_focus_vector(self):
        vector = np.array([self.focus[0]-self.x, self.focus[1]-self.y, self.focus[2]-self.z])
        unit_vector = vector / np.linalg.norm(vector)

        return unit_vector

    # Return vector perpendicular to the focus vector, along the XY-plane
    @staticmethod
    def get_focus_axis_1(focus_vector):
        spread_radians = np.pi
        rotational_matrix = [
            [np.cos(spread_radians / 2), -np.sin(spread_radians / 2), 0],
            [np.sin(spread_radians / 2), np.cos(spread_radians / 2), 0],
            [0, 0, 0]
        ]

        vector = np.matmul(rotational_matrix, focus_vector)

        if (vector**2).sum()**0.5 == 0 and focus_vector[2] == 1:
            return np.array([-1, 0, 0])

        elif (vector**2).sum()**0.5 == 0 and focus_vector[2] == -1:
            return np.array([1, 0, 0])

        else:
            return vector / (vector**2).sum()**0.5


# Class used to represent pallets as objects
class Pallet:
    def __init__(self, faces):
        self.faces = faces
        self.equations = get_planar_equations(faces)


# Shows points as seen or not seen in 3D-plot
def show_plots(seen_points, obstructed_points, cameras, unit_vectors=None, color=None):
    X_seen, Y_seen, Z_seen = [], [], []
    for j in seen_points:
        X_seen.append(j[0])
        Y_seen.append(j[1])
        Z_seen.append(j[2])

    X_obs, Y_obs, Z_obs = [], [], []
    for j in obstructed_points:
        X_obs.append(j[0])
        Y_obs.append(j[1])
        Z_obs.append(j[2])

    plt.figure()
    ax = plt.axes(projection='3d')
    plt.xlim(-40, 100)
    plt.ylim(-40, 100)
    ax.set_zlim(-50, 50)
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.set_zlabel("z")
    # Add points to plot
    if color is None:
        ax.scatter(X_seen, Y_seen, Z_seen, color="green", marker=".")
    else:
        ax.scatter(X_seen, Y_seen, Z_seen, color=color, marker=".")
    ax.scatter(X_obs, Y_obs, Z_obs, color="red", marker=".")
    # Add field of view vectors to the plot
    for i in cameras:
        ax.scatter(i.x, i.y, i.z, color="black")
        fov_vectors = get_camera_fov(i)
        ax.plot([i.x, i.x + fov_vectors[0][0] * 75], [i.y, i.y + fov_vectors[0][1] * 75],
                [i.z, i.z + fov_vectors[0][2] * 75], color="blue")
        ax.plot([i.x, i.x + fov_vectors[1][0] * 75], [i.y, i.y + fov_vectors[1][1] * 75],
                [i.z, i.z + fov_vectors[1][2] * 75], color="blue")
        ax.plot([i.x, i.x + fov_vectors[2][0] * 75], [i.y, i.y + fov_vectors[2][1] * 75],
                [i.z, i.z + fov_vectors[2][2] * 75], color="blue")
        ax.plot([i.x, i.x + fov_vectors[3][0] * 75], [i.y, i.y + fov_vectors[3][1] * 75],
                [i.z, i.z + fov_vectors[3][2] * 75], color="blue")
        if unit_vectors is not None:
            for j in unit_vectors:
                ax.plot([i.x, i.x + j[0] * 10], [i.y, i.y + j[1] * 10],
                        [i.z, i.z + j[2] * 10], color="orange")

    ax.view_init(30, 220)
    plt.show()


# Calculate planar equations for each of the surfaces of the pallet
def get_planar_equations(surfaces):
    equations = []
    for j in surfaces:
        vector_1 = [j[1][0] - j[0][0], j[1][1] - j[0][1], j[1][2] - j[0][2]]
        vector_2 = [j[2][0] - j[0][0], j[2][1] - j[0][1], j[2][2] - j[0][2]]
        normal = np.cross(vector_1, vector_2)
        normal_hat = normal / (normal ** 2).sum() ** 0.5
        equations.append(get_plane(j[0], normal_hat))

    return equations


# Return the vectors representing the field of view of the camera
def get_camera_fov(camera):
    focus_vector = camera.get_focus_vector()
    focus_axis_1 = camera.get_focus_axis_1(focus_vector)
    focus_axis_2 = get_focus_axis_2(focus_vector, focus_axis_1)
    fov_vectors = get_fov_vectors(focus_axis_1, focus_axis_2, focus_vector, camera.fov)

    return fov_vectors

# Return components of planar equation as individual elements
def get_plane(point, normal):
    return normal[0], point[0], normal[1], point[1], normal[2], point[2]


# Return second focus axis perpendicular to the first and the focus vector
def get_focus_axis_2(focus_vector, focus_axis):
    return np.cross(focus_vector, focus_axis)


# Return field of view vectors for each camera as a list
def get_fov_vectors(vector_1, vector_2, focus_vector, fov):
    left_turn_radians = fov[0] * np.pi / 180 / 2
    top_turn_radians = fov[1] * np.pi / 180 / 2

    leftwards = np.tan(left_turn_radians)
    rightwards = -leftwards
    upwards = np.tan(top_turn_radians)
    downwards = -upwards

    vector_top_left = leftwards * vector_1 + upwards * vector_2 + focus_vector
    vector_bottom_left = leftwards * vector_1 + downwards * vector_2 + focus_vector
    vector_bottom_right = rightwards * vector_1 + downwards * vector_2 + focus_vector
    vector_top_right = rightwards * vector_1 + upwards * vector_2 + focus_vector

    return [vector_top_left, vector_bottom_left, vector_bottom_right, vector_top_right]



# Return list of seen points based on camera specifications and pallet surfaces
def check_fov(camera, current_camera, pallet, remaining_points, steps, threshold_point, threshold_surface, seen_points=[], queue=[], multi_processing=False):
    # If the setup specifies not using multi processing
    if not multi_processing:
        camera_progress = 0
        for j in remaining_points:
            point_obstructed = camera.check_fov(j, pallet.faces, pallet.equations, steps, threshold_surface, threshold_point)
            if point_obstructed:
                pass
            else:
                seen_points.append(j)
            camera_progress += 1
            print("Camera " + str(current_camera) + " " + str(
                round(camera_progress / len(remaining_points) * 100, 2)) + "% done")

        return seen_points
    # If setup specifies using multi processing
    else:
        while len(queue) > 0:
            if queue[0] not in camera.checked_points:
                current_point = queue.pop(0)
                camera.checked_points.append(current_point)
            elif len(queue) > 1:
                queue_copy = queue
                pos = 1
                rounds = 0
                checked = True
                while checked:
                    if pos < len(queue_copy):
                        if queue_copy[pos] not in camera.checked_points:
                            current_point = queue_copy[pos]
                            try:
                                queue.remove(current_point)
                            except ValueError:
                                pass
                            else:
                                camera.checked_points.append(current_point)
                                checked = False
                        else:
                            pos += 1
                    # When three passes of the remaining points do not return any unchecked points
                    elif rounds == 3:
                        print(f"Camera {current_camera} finished")
                        return

                    else:
                        pos = 1
                        rounds += 1
                        print(f"Camera {current_camera} finished {rounds} rounds without new points")
                        time.sleep(2)
            # If no points remain unseen
            else:
                print(f"Camera {current_camera} finished")
                return
            # Print remaining number of unseen points
            print(len(queue))
            point_obstructed = camera.check_fov(current_point, pallet.faces, pallet.equations, steps, threshold_surface, threshold_point)

            if point_obstructed:
                queue.append(current_point)
            else:
```

```python
                seen_points.append(current_point)


# Import .txt-file and return surfaces and points as arrays
def read_file(file, additional_points):
    with open(file) as palletFile:
        pallet_surface_text = palletFile.read()
        pallet_surface_text = pallet_surface_text.replace("[", "")
        pallet_surface_text = pallet_surface_text.replace("]", "")
        pallet_surface_text = pallet_surface_text.replace(" ", "")
        pallet_surface_text = pallet_surface_text.split(",")
        pallet_surface_num = []
        initial_points = []
        for i in range(int(len(pallet_surface_text) / 12.0)):
            pallet_surface_temp = []
            # Every four points represent a surface
            for j in range(4):
                initial_points.append([float(pallet_surface_text[i * 12 + j * 3]),
                                        float(pallet_surface_text[i * 12 + 1 + j * 3]),
                                        float(pallet_surface_text[i * 12 + 2 + j * 3])])
                pallet_surface_temp.append([float(pallet_surface_text[i * 12 + j * 3]),
                                          float(pallet_surface_text[i * 12 + 1 + j * 3]),
                                          float(pallet_surface_text[i * 12 + 2 + j * 3])])
            pallet_surface_num.append(pallet_surface_temp)
        # Removes duplications of points
        for j in initial_points:
            if initial_points.count(j) > 1:
                initial_points.remove(j)
        # Add specified amount of points along sides of surfaces
        for j in pallet_surface_num:
            if j[0][0] != j[1][0]:
                for k in np.linspace(j[0][0], j[1][0], additional_points, endpoint=False):
                    initial_points.append([k, j[0][1], j[0][2]])
                for k in np.linspace(j[2][0], j[3][0], additional_points, endpoint=False):
                    initial_points.append([k, j[2][1], j[2][2]])

                if j[1][1] != j[2][1]:
                    for k in np.linspace(j[1][1], j[2][1], additional_points, endpoint=False):
                        initial_points.append([j[1][0], k, j[1][2]])
                    for k in np.linspace(j[3][1], j[0][1], additional_points, endpoint=False):
                        initial_points.append([j[3][0], k, j[3][2]])

                if j[1][2] != j[2][2]:
                    for k in np.linspace(j[1][2], j[2][2], additional_points, endpoint=False):
                        initial_points.append([j[1][0], j[1][1], k])
                    for k in np.linspace(j[3][2], j[0][2], additional_points, endpoint=False):
                        initial_points.append([j[3][0], j[3][1], k])

            if j[0][1] != j[1][1]:
                for k in np.linspace(j[0][1], j[1][1], additional_points, endpoint=False):
                    initial_points.append([j[0][0], k, j[0][2]])
                for k in np.linspace(j[2][1], j[3][1], additional_points, endpoint=False):
                    initial_points.append([j[3][0], k, j[2][2]])

                if j[1][0] != j[2][0]:
                    for k in np.linspace(j[1][0], j[2][0], additional_points, endpoint=False):
                        initial_points.append([k, j[1][1], j[1][2]])
                    for k in np.linspace(j[3][0], j[0][0], additional_points, endpoint=False):
                        initial_points.append([k, j[3][1], j[3][2]])

                if j[1][2] != j[2][2]:
                    for k in np.linspace(j[1][2], j[2][2], additional_points, endpoint=False):
                        initial_points.append([j[1][0], j[1][1], k])
                    for k in np.linspace(j[3][2], j[0][2], additional_points, endpoint=False):
                        initial_points.append([j[3][0], j[3][1], k])

            if j[0][2] != j[1][2]:
                for k in np.linspace(j[0][2], j[1][2], additional_points, endpoint=False):
                    initial_points.append([j[0][0], j[0][1], k])
                for k in np.linspace(j[2][0], j[3][0], additional_points, endpoint=False):
                    initial_points.append([[j[2][0]], j[2][1], k])

                if j[1][0] != j[2][0]:
                    for k in np.linspace(j[1][0], j[2][0], additional_points, endpoint=False):
                        initial_points.append([k, j[1][1], j[1][2]])
                    for k in np.linspace(j[3][0], j[0][0], additional_points, endpoint=False):
                        initial_points.append([k, j[3][1], j[3][2]])

                if j[1][1] != j[2][1]:
                    for k in np.linspace(j[1][1], j[2][1], additional_points, endpoint=False):
                        initial_points.append([j[1][0], k, j[1][2]])
                    for k in np.linspace(j[3][1], j[0][1], additional_points, endpoint=False):
                        initial_points.append([j[3][0], k, j[3][2]])

        for j in initial_points:
            if initial_points.count(j) > 1:
                initial_points.remove(j)

    return pallet_surface_num, initial_points


# Setup and main program
def main():
    debug = False  # Debug mode
    fov_check = True  # FOV check mode
    use_multi_processing = False  # Multi processing mode
    steps = 5000  # Number of discretization points between camera and point
    additional_points = 20  # Additional points along each surface
    threshold_surface = 1/steps*20  # Margins of the planar equations
    threshold_point = 0.02  # Margin for deciding whether point is part of a crossed surface
    start_time = time.time()
    pallet_surface_num, initial_points = read_file("euro.txt", additional_points)
    cameras = []
    # Mode used to check if camera FOV is correct
    if debug:
        cameras.append(Camera(-10, -10, 0, (1, 1), (0, 0, 100), (0, 150)))
        p1 = Pallet(pallet_surface_num)

        points_in_fov, not_possible_points, unit_vectors = cameras[0].check_points_in_fov(initial_points)
        seen_points_camera = check_fov(cameras[0], 1, p1, points_in_fov, steps, threshold_point, threshold_surface)

        obstructed_points = []
        for i in initial_points:
            if i not in seen_points_camera:
                obstructed_points.append(i)
        show_plots(points_in_fov, not_possible_points, cameras, unit_vectors)
    # Mode used to check which points are within field of view. Does not check for obstructions
    elif fov_check:
        cameras.append(Camera(-80, -60, 0, (33, 25), (40, 60, 10), (50, 200)))
        points_in_fov, not_possible_points, unit_vectors = cameras[0].check_points_in_fov(initial_points)
        show_plots(points_in_fov, not_possible_points, cameras, unit_vectors, color="blue")
    # Checks all point to see whether or not they are seen by any of the cameras
    else:
        cameras.append(Camera(-142, -122, -104, (39, 25), (40, 40, 5), (120, 300)))
        cameras.append(Camera(-142, -122, 238, (39, 25), (40, 40, 5), (120, 300)))
        cameras.append(Camera(222, 242, -104, (39, 25), (40, 40, 5), (120, 300)))

        p1 = Pallet(pallet_surface_num)
        # Checks seen points using multi processing
        if use_multi_processing:
            manager = multiprocessing.Manager()
            seen_points = manager.list()
            queue = manager.list()

            for i in initial_points:
                queue.append(i)
            process_pool = []
            for i, c in enumerate(cameras):
                remaining_points, obstructed_points, unit_vectors = c.check_points_in_fov(initial_points)
                process_pool.append(multiprocessing.Process(target=check_fov, args=(c, i+1, p1, remaining_points, steps, threshold_point, threshold_surface, seen_points, queue, True)))
                process_pool[i].start()
            for process in process_pool:
                process.join()
        # Checks seen points one camera at a time
        else:
            seen_points = []
            remaining_points = initial_points
            for i, c in enumerate(cameras):
                points_in_fov, _, _ = c.check_points_in_fov(remaining_points)
                seen_points_camera = check_fov(c, i+1, p1, points_in_fov, steps, threshold_point, threshold_surface)
                remaining_points = []
                for j in seen_points_camera:
                    seen_points.append(j)
                for j in initial_points:
                    if j not in seen_points:
                        remaining_points.append(j)

        obstructed_points = []
        # All unseen points is added to seperate list
        for i in initial_points:
            if i not in seen_points:
                obstructed_points.append(i)

        print("Total time used:", time.time() - start_time)
        # Displays results in 3D-plot
        try:
            show_plots(seen_points, obstructed_points, cameras)
        except KeyboardInterrupt:
            pass


if __name__ == "__main__":
    main()
```

Appendix H

Source code, Webots simulator

Source code, Point cloud acquisition

```python
"""
README

This simulation script was written to be used for moving a camera between two positions
using a linear axis, during the prototyping in the bachelor thesis. This program
i written using a python in the simulation software Weebots.
"""
"""camera controller."""

from controller import Robot, Camera, RangeFinder
import math
import numpy as np

#lagger en klasse som brukes for alle robotene, navnet på child komponentene må vere likt
class platform (Robot):
    timestep = 32
    def __init__(self):
        super(platform, self).__init__()
        #setter opp kamera
        self.camera = self.getDevice('camera')
        self.camera.enable(4*self.timestep)
        #setter opp rangefinderen
        self.rangeFinder = self.getDevice('3d')
        self.rangeFinder.enable(4*self.timestep)
        #setter opp keyboard detection
        self.keyboard.enable(self.timestep)
        self.keyboard = self.getKeyboard()
        #lar deg gjøre forskjellige ting med hver robot
        self.ID = self.getName()
        if self.ID == 'framework1':
            print('robot1 identified')
            self.filename = ['cam1_dist.txt','cam1_rgb.txt']
            image_width = self.rangeFinder.getWidth()
            image_height = self.rangeFinder.getHeight()
            focal_length = 0.5 * image_width * (1 / math.tan(0.5 * self.rangeFinder.getFov()))
            k_matrix = np.array([
                [focal_length, 0, image_width / 2],
                [0, focal_length, image_height / 2],
                [0, 0, 0]
            ])
            file = open('k_matrix.txt', "w+")
            for row in k_matrix:
                np.savetxt(file, row)
            file.close()
        elif self.ID == 'framework2':
            print('robot2 identified')
            self.filename = ['cam2_dist.txt','cam2_rgb.txt']
        elif self.ID == 'framework3':
            print('robot3 identified')
            self.filename = ['cam3_dist.txt','cam3_rgb.txt']

    def run(self):
        while True:
            #2d array med alle dybder
            self.depth = self.rangeFinder.getRangeImageArray()
            #array med RGB info for siste bilde
            #formen er array[x][y][n] hvor n = 0 er rød, 1 er grøn og 2 er blå
            self.image = self.camera.getImageArray()

            k = self.keyboard.getKey()

            if k == ord('Q'):
                file = open(self.filename[0], "w+")
                for row in self.depth:
                    np.savetxt(file, row)
                file.close()
                file = open(self.filename[1], "w+")
                for row in self.image:
                    np.savetxt(file, row)
                file.close()


            if self.step(self.timestep) == -1:
                break

controller = platform()
controller.run()

"""linear axis controller."""
from controller import Robot, Motor, Camera, RangeFinder
from math import pi, sin

class platform (Robot):
    timestep = 32

    def __init__(self):
        super(platform, self).__init__()

        self.motorAngle = self.getDevice("motor")
        self.motorAngle.setPosition(float('inf'))
        self.motorAngle.setVelocity(0.0)

        self.pSensor = self.getDevice("ps")
        self.pSensor.enable(self.timestep)

        self.vSensor = self.getDevice("vs")
        self.vSensor.enable(self.timestep)

        self.motor = self.getDevice("linear")
        #setter opp kamera
        self.camera = self.getDevice('camera1')
        self.camera.enable(4*self.timestep)
        #setter opp rangefinderen
        self.rangeFinder = self.getDevice('3d1')
        self.rangeFinder.enable(4*self.timestep)
        #setter opp keyboard detection
        self.keyboard.enable(self.timestep)
        self.keyboard = self.getKeyboard()
        #lar deg gjøre forskjellige ting med hver robot
        self.ID = self.getName()
        if self.ID == 'metal_pole':
            print('The linear axis identified')
            self.filename = ['cam1_dist.txt','cam1_rgb.txt']

    def run(self):
        while True:
            #2d array med alle dybder
            self.depth = self.rangeFinder.getRangeImageArray()
            #array med RGB info for siste bilde
            #formen er array[x][y][n] hvor n = 0 er rød, 1 er grøn og 2 er blå
            self.image = self.camera.getImageArray()

            k = self.keyboard.getKey()


            if k == ord('Q'):
                file = open(self.filename[0], "w+")
                content = str(self.depth)
                file.write(content)
                file.close()
                file = open(self.filename[1], "w+")
                content = str(self.image)
                file.write(content)
                file.close()
            if self.step(self.timestep) == -1:
                break
    def camera_movement(self):


        speed=1
        vertical_value = 0
        horizontal_value = 0
        position = 0.0
        constant = 1
        while self.step(self.timestep) != -1:

            n = self.keyboard.getKey()

            # Verical controle
            self.motor.setPosition(position)
            vertical_value = self.vSensor.getValue()
            print("Vertical position", vertical_value)

            # Angle controle
            self.motorAngle.setVelocity(speed)
            horizontal_value = self.pSensor.getValue()
            print("Horizontal angle", horizontal_value)

            if n == ord('S'):
                print("Going down")
                position -= constant
                speed -= 1



            if n == ord('W'):
                print("Going up")
                position += constant
                speed += 1


            pass


controller = platform()
controller.camera_movement()
```

Source code, LiDAR testing simulator

```
"""
README

This simulation script was written to be used for testing several methods
of obtaining 3D data. This program is written using a python in the simulation software Weebots.

lidar_test_controller controller.
"""

from controller import Robot

def run_robot(robot):
    timestep = 32
    max_speed = 6.28

    left_motor = robot.getDevice('motor_1')
    right_motor = robot.getDevice('motor_2')

    left_motor.setPosition(float('inf'))
    right_motor.setPosition(float('inf'))

    left_motor.setVelocity(0.0)
    right_motor.setVelocity(0.0)

    lidar = robot.getLidar('lidar')
    lidar.enable(timestep)
    lidar.enablePointCloud()


    # Main loop:

    while robot.step(timestep) != -1:

        range_image = lidar.getRangeImage()
        print('{}'.format(range_image))

        left_motor.setVelocity(max_speed*0.25)
        right_motor.setVelocity(max_speed*0.25)
```

```python


if __name__== "__main__":

    my_robot = Robot()
    run_robot(my_robot)
```

Appendix I

Source code, point cloud generator

```python
"""
README

This python script takes the depth and color data from the webots simulator and converts it into ply files by utilizing the intrinsic camera parameters from the camera. This file is set up for the zivid one+ large camera.
"""

import numpy as np
import open3d as o3d


def get_calibration_matrix():
    file_data = np.loadtxt(’zivid_test/k_matrix.txt’).reshape(3, 3)
    return file_data


# Uses the depth array and k matrix to create a point cloud,
# the axis are rotated to correspond to the orientation in webots
def get_point_cloud(depth_image, k_matrix):
    inv_fx = 1.0 / k_matrix[0, 0]
    inv_fy = 1.0 / k_matrix[1, 1]
    ox = k_matrix[0, 2]
    oy = k_matrix[1, 2]
    image_height, image_width = depth_image.shape
    points = np.zeros((image_width * image_height, 3), dtype=np.float32)
    counter = 0
    for y in range(image_height):
        for x in range(image_width):
            dist = depth_image[y, x]
            points[counter, 2] = -np.float32((x - ox) * dist * inv_fx)
            points[counter, 1] = -np.float32((y - oy) * dist * inv_fy)
            points[counter, 0] = np.float32(dist)
            counter += 1
    return points[:counter].astype(np.float32)


# Reads array written from webots, needs the file name and shape of array
def read_array(file_name, x_px, y_px, z):
    if z == 0:
        file_data = np.loadtxt(file_name).reshape(x_px, y_px)
    else:
        file_data = np.loadtxt(file_name).reshape(x_px, y_px, z)
    return file_data


def save_point_cloud(name, pcd):
    o3d.io.write_point_cloud(name, pcd)


# Creates coordinate frames for the origin and cameras and scales them down, so they don’t dominate the view
def create_cameras():
    camera1 = o3d.geometry.TriangleMesh.create_coordinate_frame()
    camera2 = o3d.geometry.TriangleMesh.create_coordinate_frame()
    camera3 = o3d.geometry.TriangleMesh.create_coordinate_frame()
    origin_frame = o3d.geometry.TriangleMesh.create_coordinate_frame()
    origin_frame.scale(0.3, center=origin_frame.get_center())
    camera1.scale(0.3, center=camera1.get_center())
    camera2.scale(0.3, center=camera2.get_center())
    camera3.scale(0.3, center=camera3.get_center())
    return origin_frame, camera1, camera2, camera3


def create_transformation(geometry, quat, translation):
    T = np.eye(4)
    T[:3, :3] = geometry.get_rotation_matrix_from_quaternion(quat)
    T[0, 3] = translation[0]
    T[1, 3] = translation[1]
    T[2, 3] = translation[2]
    return T


```

```python
def filter_point_cloud(pointcloud, radius):
    points = np.asarray(pointcloud.points)
    colours = np.asarray(pointcloud.colors)
    center = np.array([0, 0, 0])
    distances = np.linalg.norm(points - center, axis=1)
    pointcloud.points = o3d.utility.Vector3dVector(points[distances <= radius])
    pointcloud.colors = o3d.utility.Vector3dVector(colours[distances <= radius])
    return pointcloud


# camera resolution
width = 1920
height = 1200

# collects the images from text files

k_matrix = get_calibration_matrix()
dist_image_2 = read_array('zivid_test/cam2_dist.txt', width, height, 0)
point_cloud_2 = get_point_cloud(dist_image_2, k_matrix)
colour_img_2 = read_array('zivid_test/cam2_rgb.txt', width, height, 3)
colour_2 = colour_img_2.reshape(height*width, 3)
dist_image_1 = read_array('zivid_test/cam1_dist.txt', width, height, 0)
point_cloud_1 = get_point_cloud(dist_image_1, k_matrix)
colour_img_1 = read_array('zivid_test/cam1_rgb.txt', width, height, 3)
colour_1 = colour_img_1.reshape(height*width, 3)
dist_image_3 = read_array('zivid_test/cam3_dist.txt', width, height, 0)
point_cloud_3 = get_point_cloud(dist_image_3, k_matrix)
colour_img_3 = read_array('zivid_test/cam3_rgb.txt', width, height, 3)
colour_3 = colour_img_3.reshape(height*width, 3)

# creates the geometries and displays them
pcd1 = o3d.geometry.PointCloud()
pcd2 = o3d.geometry.PointCloud()
pcd3 = o3d.geometry.PointCloud()
pcd1.points = o3d.utility.Vector3dVector(point_cloud_1)
pcd1.colors = o3d.utility.Vector3dVector(colour_1/255)
pcd2.points = o3d.utility.Vector3dVector(point_cloud_2)
pcd2.colors = o3d.utility.Vector3dVector(colour_2/255)
pcd3.points = o3d.utility.Vector3dVector(point_cloud_3)
pcd3.colors = o3d.utility.Vector3dVector(colour_3/255)

origin, cam1, cam2, cam3 = create_cameras()

pcd1 = filter_point_cloud(pcd1, 3.49)
pcd2 = filter_point_cloud(pcd2, 3.49)
pcd3 = filter_point_cloud(pcd3, 3.49)
save_point_cloud('zivid_top_view.ply', pcd1)
save_point_cloud('zivid_bot_view_1.ply', pcd2)
save_point_cloud('zivid_bot_view_2.ply', pcd3)

T1 = create_transformation(pcd1, [0.87, 0.13, 0.36, -0.31], [-0.636, 2.6, 3.43])
cam1.transform(T1)
pcd1.transform(T1)
T2 = create_transformation(pcd2, [0.87, -0.17, -0.34, -0.30], [-0.588, 2.51, -0.202])
cam2.transform(T2)
pcd2.transform(T2)
T3 = create_transformation(pcd3, [0.25, 0.25, -0.03, 0.93], [2.74, 0.476, 0.417])
pcd3.transform(T3)
cam3.transform(T3)

o3d.visualization.draw_geometries([pcd1, pcd2, pcd3, cam1, cam2, cam3, origin])

combined_pcd = o3d.geometry.PointCloud()
combined_pcd = pcd1 + pcd2 + pcd3

o3d.io.write_point_cloud("Zivid_combined.ply", combined_pcd)
```

```python
"""
README

This python script takes the generated point clouds and tries to fit them together. Based on the example code from open3D found here: http://www.open3d.org/docs/latest/tutorial/Advanced/multiway_registration.html
```

```
"""

import open3d as o3d
import numpy as np


# Creates a transformation matrix from axis angles
def create_transformation(geometry, angles, translation):
    T = np.eye(4)
    T[:3, :3] = geometry.get_rotation_matrix_from_xyz(angles)
    T[0, 3] = translation[0]
    T[1, 3] = translation[1]
    T[2, 3] = translation[2]
    return T


# Creates coordinate frames for the origin and cameras and scales them down, so they don't dominate the view
def create_cameras():
    camera1 = o3d.geometry.TriangleMesh.create_coordinate_frame()
    camera2 = o3d.geometry.TriangleMesh.create_coordinate_frame()
    origin_frame = o3d.geometry.TriangleMesh.create_coordinate_frame()
    origin_frame.scale(0.3, center=origin_frame.get_center())
    camera1.scale(0.3, center=camera1.get_center())
    camera2.scale(0.3, center=camera2.get_center())
    cameras = [origin_frame, camera1, camera2]
    return cameras


# Reads a sequence of point clouds
def load_point_clouds(voxel_size=0.0):
    pcds = []
    for i in range(2):
        j = i + 1
        pcd = o3d.io.read_point_cloud("zivid_bot_view_%d.ply" %
                                      j)
        pcd_down = pcd.voxel_down_sample(voxel_size=voxel_size)
        pcd_down.estimate_normals()
        pcds.append(pcd_down)
    pcd = o3d.io.read_point_cloud("zivid_top_view.ply")
    pcd_down = pcd.voxel_down_sample(voxel_size=voxel_size)
    pcd_down.estimate_normals()
    pcds.append(pcd_down)
    return pcds


def load_full_point_clouds():
    pcds = []
    for i in range(2):
        j = i + 1
        pcd = o3d.io.read_point_cloud("zivid_bot_view_%d.ply" %
                                      j)
        pcds.append(pcd)
    pcd = o3d.io.read_point_cloud("zivid_top_view.ply")
    pcds.append(pcd)
    return pcds


def pairwise_registration(source, target):
    print("Apply point-to-plane ICP")
    icp_coarse = o3d.pipelines.registration.registration_icp(
        source, target, max_correspondence_distance_coarse, np.identity(4),
        o3d.pipelines.registration.TransformationEstimationPointToPlane())
    icp_fine = o3d.pipelines.registration.registration_icp(
        source, target, max_correspondence_distance_fine,
        icp_coarse.transformation,
        o3d.pipelines.registration.TransformationEstimationPointToPlane())
    transformation_icp = icp_fine.transformation
    information_icp = o3d.pipelines.registration.get_information_matrix_from_point_clouds(
        source, target, max_correspondence_distance_fine,
        icp_fine.transformation)
    return transformation_icp, information_icp

```

```python
def full_registration(pcds, max_correspondence_distance_coarse,
                      max_correspondence_distance_fine):
    pose_graph = o3d.pipelines.registration.PoseGraph()
    odometry = np.identity(4)
    pose_graph.nodes.append(o3d.pipelines.registration.PoseGraphNode(odometry))
    n_pcds = len(pcds)
    for source_id in range(n_pcds):
        for target_id in range(source_id + 1, n_pcds):
            transformation_icp, information_icp = pairwise_registration(
                pcds[source_id], pcds[target_id])
            print("Build o3d.pipelines.registration.PoseGraph")
            if target_id == source_id + 1:  # odometry case
                odometry = np.dot(transformation_icp, odometry)
                pose_graph.nodes.append(
                    o3d.pipelines.registration.PoseGraphNode(
                        np.linalg.inv(odometry)))
                pose_graph.edges.append(
                    o3d.pipelines.registration.PoseGraphEdge(source_id,
                                                             target_id,
                                                             transformation_icp,
                                                             information_icp,
                                                             uncertain=False))
            else:  # loop closure case
                pose_graph.edges.append(
                    o3d.pipelines.registration.PoseGraphEdge(source_id,
                                                             target_id,
                                                             transformation_icp,
                                                             information_icp,
                                                             uncertain=True))
    return pose_graph


voxel_size = 0.04
pcds_down = load_point_clouds(voxel_size)

o3d.visualization.draw_geometries(pcds_down)
print("Full registration ...")
max_correspondence_distance_coarse = voxel_size * 15
max_correspondence_distance_fine = voxel_size * 1.5
with o3d.utility.VerbosityContextManager(
        o3d.utility.VerbosityLevel.Debug) as cm:
    pose_graph = full_registration(pcds_down,
                                   max_correspondence_distance_coarse,
                                   max_correspondence_distance_fine)

print("Optimizing PoseGraph ...")
option = o3d.pipelines.registration.GlobalOptimizationOption(
    max_correspondence_distance=max_correspondence_distance_fine,
    edge_prune_threshold=0.25,
    reference_node=0)
with o3d.utility.VerbosityContextManager(
        o3d.utility.VerbosityLevel.Debug) as cm:
    o3d.pipelines.registration.global_optimization(
        pose_graph,
        o3d.pipelines.registration.GlobalOptimizationLevenbergMarquardt(),
        o3d.pipelines.registration.GlobalOptimizationConvergenceCriteria(),
        option)

print("Transform points and display")
# Prints out the transformations found by the optimiser
for point_id in range(len(pcds_down)):
    print(pose_graph.nodes[point_id].pose)

pcds = load_full_point_clouds()
pcd_combined = o3d.geometry.PointCloud()
for point_id in range(len(pcds)):
    pcds[point_id].transform(pose_graph.nodes[point_id].pose)
    pcd_combined += pcds[point_id]
pcd_combined_down = pcd_combined.voxel_down_sample(voxel_size=voxel_size)
o3d.io.write_point_cloud("combined_views.ply", pcd_combined)
o3d.visualization.draw_geometries([pcd_combined])
```

Appendix J

Source code, PLC program

1 PLC program

1.1 POV

1.1.1 MAIN (PRG)

```
"""""
README

This project was developed to be used for moving a camera between two positions
using a linear axis, during the protyping in my bachelor's thesis. This program
i written using a ESI-file from the specific motor used, and therefore used the
Tc2_MC2 library from backhoff to link the motor to an axis.

"""

PROGRAM MAIN
VAR
        Machine : FB\_Machine;
END_VAR
```

```
""    Machine();
```

1.1.2 FB_Machine (FB)

```
""FUNCTION_BLOCK FB_Machine
VAR
        Tower : FB_Tower;
END_VAR
```

```
""    Tower();
```

1.1.3 FB_Tower (FB)

```
""FUNCTION_BLOCK FB_Tower
VAR_INPUT
        resetButton                     : BOOL;
        startButton                     : BOOL;
        stopButton                      : BOOL;
        forwardButton           : BOOL;
        backwardButton          : BOOL;
        fManualVelocity         : LREAL;

        // velocity constant for HIGH and LOW value.
        fManualLowSpeed         : LREAL := 0.05;
        fManualHighSpeed        : LREAL := 0.1;
END_VAR
VAR_OUTPUT
END_VAR
VAR
        // Libraries used
        stAxis                          : AXIS_REF;
        MCJog                           : MC_Jog;
        McPower                         : MC_Power;
        McReset                         : MC_Reset;
        stErrorData                     : ST_ErrorData;

        // Real axis data feedback
        Enabled                 : BOOL;
        ActualVel               : LREAL;
        ActualPos                       : LREAL;

        //ActualTorque          : LREAL;
        bPositiveDirection      : BOOL;
        bNegativeDirection      : BOOL;
        AxisMovement            : BOOL;
        bMinSoftLimit           : BOOL;
        bMaxSoftLimit           : BOOL;

        // Error id
        mc_errorID                      : UDINT := 1;
        mc_error_string         : STRING;
        error_header_string     : STRING;

        // For blinking lamp
        light_pulse             : BOOL;
        light_constant          : BOOL;
        tBlink                          : TON;
        blink1000                       : BOOL;

        //Adjusting speed on axis
        SpeedSetting            : E_Speed;

        // TO_STRING for velocity

        ESpeed                  : E_Speed;
    nCurrentValue       : INT;
    sCurrentValue       : STRING;
    wsCurrentValue      : WSTRING;
    sComponent          : STRING;
    wsComponent         : WSTRING;

        // TO_STRING for states
        EMachineState           : E_MachineState;
        nCurrentState           : INT;
        sCurrentState           : STRING;
        wsCurrentState          : WSTRING;
        sComponentState         : STRING;
        wsComponentState        : WSTRING;

        {IF defined (Emulation)}
        StateTower1                                     AT %I* : UINT := 8;
        {ELSE}
        StateTower1                                     AT %I* : UINT;
        {END_IF}
END_VAR
```

```
""MCJog( Axis :=         stAxis, Velocity := fManualVelocity, Mode := MC_JOGMODE_CONTINOUS );
McPower(Axis := stAxis);
McReset(Axis := stAxis);
ActUpdateAxisData();
ActUpdateManVelocity();
ActBlinkingTimer();
ActToStringSpeedAndState();
CASE GVL.eMachineState OF

        // The machine starts up in Error state
        E_MachineState.Error:

        light_constant := FALSE;
        EnableAxis(FALSE);
        mc_error_string := F_NcErrorMessage(mc_errorID);
        error_header_string := F_HeaderErrorMessage(mc_errorID);

        IF resetButton THEN
                SetState( E_MachineState.Ready );
        END_IF

        // Ready state is for enabling the axis
        E_MachineState.Ready:
        light_constant := FALSE;
        McResetAxis();

        IF startButton THEN
                light_constant := TRUE;
                EnableAxis(TRUE);
                SetState( E_MachineState.Running );
        END_IF

        // Running state is for driving the axis in both postive and negative Y direction.
        E_MachineState.Running:

        MCJog.JogForward        := forwardButton;
        MCJog.JogBackwards      := backwardButton;

        IF stopButton THEN
                EnableAxis(FALSE);
                SetState( E_MachineState.Ready );
        END_IF
END_CASE
```

FB_Tower: ActBlinkingTimer (Action)

```
""tBlink(IN := NOT tblink.Q, PT := T#1S);
blink1000 := tBlink.ET > T#0.5S;

light_pulse := light_constant OR blink1000;
```

FB_Tower: ActToStringSpeedAndState (Action)

```
""nCurrentValue := ESpeed;

sCurrentValue  := TO_STRING(ESpeed);
wsCurrentValue := TO_WSTRING(ESpeed);

sComponent     := TO_STRING(E_Speed.High);
wsComponent    := TO_WSTRING(E_Speed.Low);

nCurrentState := EMachineState;

sCurrentState  := TO_STRING(EMachineState);
wsCurrentState  := TO_WSTRING(EMachineState);

sComponentState      := TO_STRING(E_MachineState.Ready);
wsComponentState     := TO_WSTRING(E_MachineState.Running);
```

FB_Tower: ActUpdateAxisData (Action)

```
""// Data from NC interface
stAxis.ReadStatus();
Enabled                 := McPower.Status;
ActualPos                       := stAxis.NcToPlc.ActPos;
ActualVel                       := stAxis.NcToPlc.ActVelo;

//ActualTorque          := ActualTorqueNm;
bPositiveDirection      := stAxis.Status.PositiveDirection;
bNegativeDirection      := stAxis.Status.NegativeDirection;
AxisMovement            := stAxis.Status.Moving;
bMinSoftLimit           := stAxis.Status.SoftLimitMinExceeded;
bMaxSoftLimit           := stAxis.Status.SoftLimitMaxExceeded;
```

FB_Tower: ActUpdateManVelocity (Action)

```
""CASE SpeedSetting OF

        E_Speed.Low:

                IF GVL.UI_EnableHighSpeed THEN
                        fManualVelocity                 := fManualHighSpeed;
                        SpeedSetting                    := E_Speed.High;

                ELSE
                        fManualVelocity                 := fManualLowSpeed;
                END_IF

        E_Speed.High:

                IF NOT GVL.UI_EnableHighSpeed THEN
```

```
                        fManualVelocity                 := fManualLowSpeed;
                        SpeedSetting                    := E_Speed.Low;
                ELSE
                        fManualVelocity                 := fManualHighSpeed;
                END_IF

END_CASE
```

FB_Tower: EnableAxis (Method)

```
""METHOD EnableAxis : BOOL
VAR_INPUT
        axisBoolStatus : BOOL;
END_VAR
```

```
""McPower.Enable                        := axisBoolStatus;
McPower.Enable_Positive := axisBoolStatus;
McPower.Enable_Negative := axisBoolStatus;
```

FB_Tower: ErrorUpdate (Method)

```
""METHOD ErrorUpdate
VAR_INPUT
        eState                          : E_Axis;
        nErrorId                        : UDINT;
        sErrorMessageBody       : STRING[300];
END_VAR
VAR
        sErrorMessageHeader     : STRING[300];
END_VAR
```

```
""stErrorData.eLastState                        := eState;
stErrorData.nErrorID                    := nErrorId;
stErrorData.sErrorMessageBody   := sErrorMessageBody;
stErrorData.sErrorMessageHeader := F_HeaderErrorMessage(nErrorId);
```

FB_Tower: McResetAxis (Method)

```
""METHOD McResetAxis : BOOL
VAR_INPUT
END_VAR
```

```
""McReset.Execute := TRUE;

IF McReset.Done THEN
        McReset.Execute := FALSE;
END_IF
```

FB_Tower: SetState (Method)

```
""METHOD SetState
VAR_INPUT
        eState  : E_MachineState;
END_VAR
```

```
""GVL.eMachineState     := eState;
```

1.1.4 F_NcErrorMessage (FUN)

```
""FUNCTION F_NcErrorMessage : STRING[2048]
VAR_INPUT
        Error           : UDINT;
END_VAR
```

```
""CASE Error OF
                0               :               F_NcErrorMessage        := 'None';

(* --> User defined errors for entire machine <--*)
                1               :               F_NcErrorMessage        := 'The machine is in error at startup - please reset. ';

                ELSE
                        IF 36844 < Error AND Error < 36857 THEN
                                F_NcErrorMessage:='NC3_Internal_Unexpected INTERNAL Error';
                        ELSE
                                F_NcErrorMessage:='Msg Unknown';
                        END_IF
END_CASE
```

1.1.5 F_HeaderErrorMessage (FUN)

```
""FUNCTION F_HeaderErrorMessage : STRING
VAR_INPUT
        ErrorID         : UDINT;
END_VAR
```

```
""IF ErrorID = 0 THEN
        F_HeaderErrorMessage := 'None';
ELSIF ErrorID = 1 THEN
        F_HeaderErrorMessage := 'Startup safety';
END_IF
```

1.2 GLV

```
""VAR_GLOBAL
        eMachineState                                    : E_MachineState;
        UI_EnableHighSpeed                               : BOOL;
END_VAR
```

1.3 DUTs

1.3.1 E_axis(ENUM)

```
""TYPE E_Axis :
(
        Disabled,
        Enabled,
        Idle,
        MoveAbs,
        Stop,
        Error,
        Reset,
        ManualMode
);
END_TYPE
```

1.3.2 E_MachineState

```
""TYPE E_MachineState :
(
        Error,
        Ready,
        Running
);
END_TYPE
```

1.3.3 E_Speed

```
""TYPE E_Speed :
(
        Low,
        High
);
END_TYPE
```

1.3.4 ST_ErrorData

```
""TYPE ST_ErrorData :
STRUCT
        nErrorID                         : UDINT;
        sTimeStamp                       : STRING;
        sErrorMessageBody        : STRING[300];
        sErrorMessageHeader      : STRING[300];
        eLastState                       : E_Axis;
END_STRUCT
END_TYPE
```

Appendix K

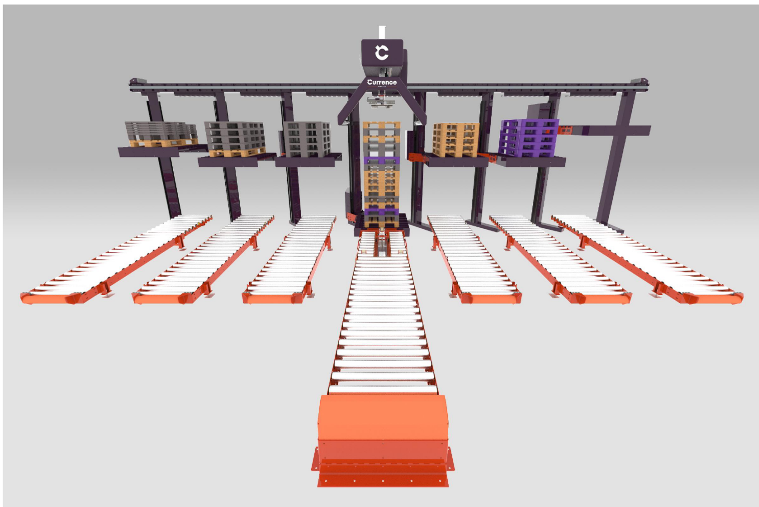
Bachelor poster

NTNU

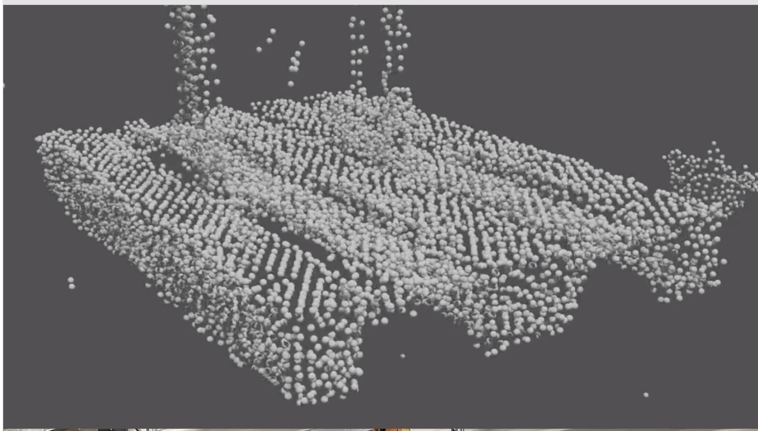
Institutt for IKT og realfag

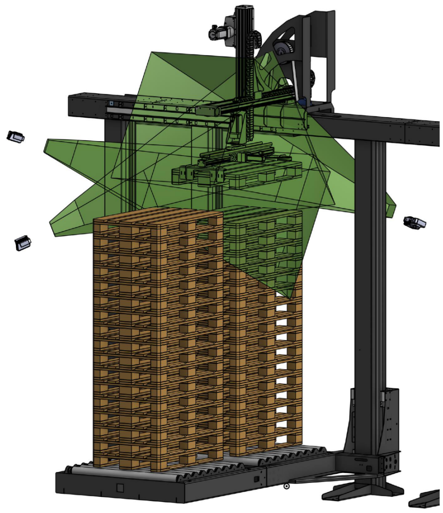
Redesigning the Point Cloud Acquisition for Sort

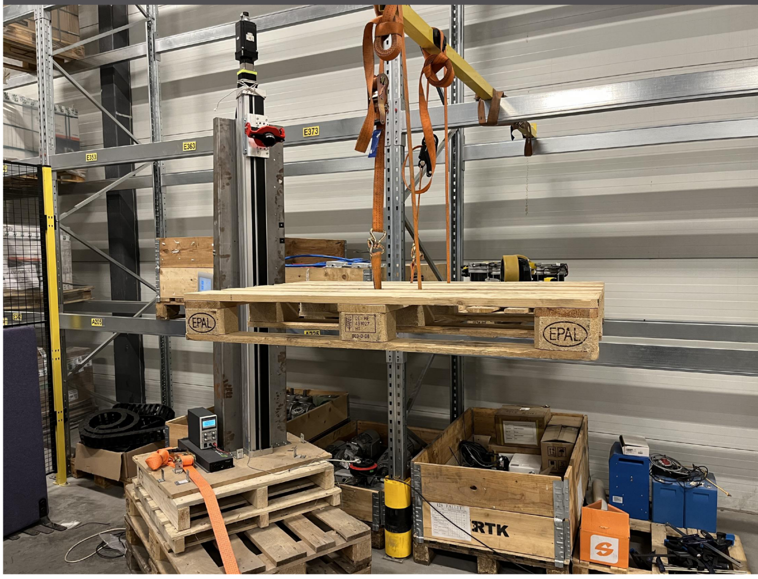
Sort™ is an automized pallet sorter designed by Solwr. They currently use six Intel Realsense L515 cameras in order to classify and evaluate used pallets. However, the current cameras are discontinued, which forms the basis for this thesis.



Method: The new acquisition is utilizing two Zivid structured light cameras. Meshing several point clouds to form a 3D model of the pallet. The pallet will be classified by its type and status.







Result:
The group designed and built a working prototype using a three image solution and a linear axis. The result was a detailed point cloud within Solwr's requirements.

Pål-André Furnes   Marius Høyer Melaas   Odd Arne Skjeret Tynes

Appendix L

Cooperation agreement

NTNU
Knowledge for a better world

Currence Robotics

IELEA2920 - bachelor thesis

COOPERATION AGREEMENT

Odd Arne Tynes, Pål-André Furnes,
Marius Høyer Melaas

A document including all rules and agreements between group members during bachelor period.

January 2022

↩

1 Members of the agreement

- Marius Høyer Melaas
- Pål-André Furnes
- Odd Arne Tynes

2 Cooperation agreement

2.1 Duration of agreement

The agreement duration is from the 10th of January 2022 until 20th of May 2022, when the thesis is to be delivered.

2.2 Purpose of the agreement

The group is writing it's bachelor thesis with Currence Robotics. This agreement is formalized to achieve accountability for each group member to follow set rules in chapter 2.3.

2.3 Rules

- Each group member must write a daily log describing the work they have done, in relation to the bachelor thesis.
- Each group member must dedicate two hours each week to write the project thesis.
- If a group member is unable to meet up with the group at the agreed upon time and location, the member shall notify as soon as possible.
- Each group member shall work an average of four workdays each week. Totaling 32 hours each week, including lunch breaks.
- Each group member shall attend sprint planning every other week. These meetings specify what each member should be working on the coming weeks.

2.4 Responsibilities of each group member

- Daily log - Marius
- Weekly writing of thesis - Odd Arne
- Updating Gantt diagram - Pål-André

3 Projects roles

Project roles within the bachelor group	
Marius Høyer Melaas	Project leader
Pål-André Furnes	Secretary
Odd Arne Tynes	Archivist

Table 1: Students and their corresponding roles

3.1 Responsibilities and tasks for the project leader

- Internal communication between the bachelor group and Currence Robotics.
- Ordering or buying parts needed.

3.2 Responsibilities and tasks for the secretary

- Send out meeting invitations, update Gantt diagram, and agenda.
- Book rooms for meetings.
- Write meeting reports.

3.3 Responsibilities and tasks for the archivist

- To upload Gantt diagram before every meeting
- Write sprint review and sprint planning.

Marius Høyer Melaas

Marius Høyer Melaas

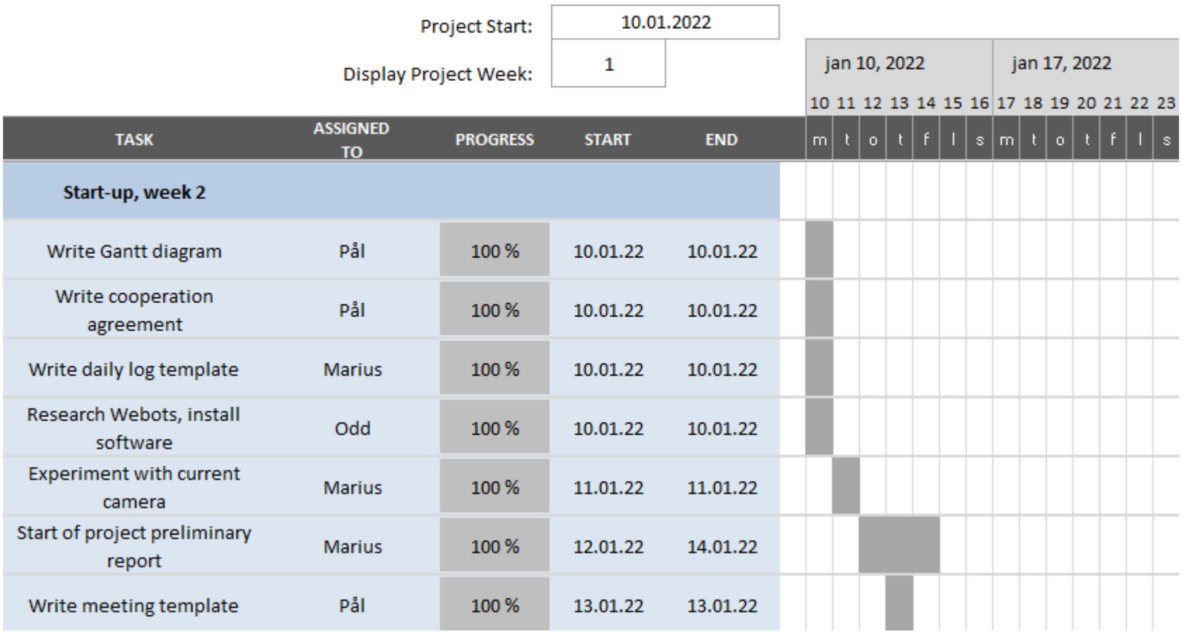
Odd Arne S. Tynes

Odd Arne Skjeret Tynes

Pål-André Furnes

Appendix M

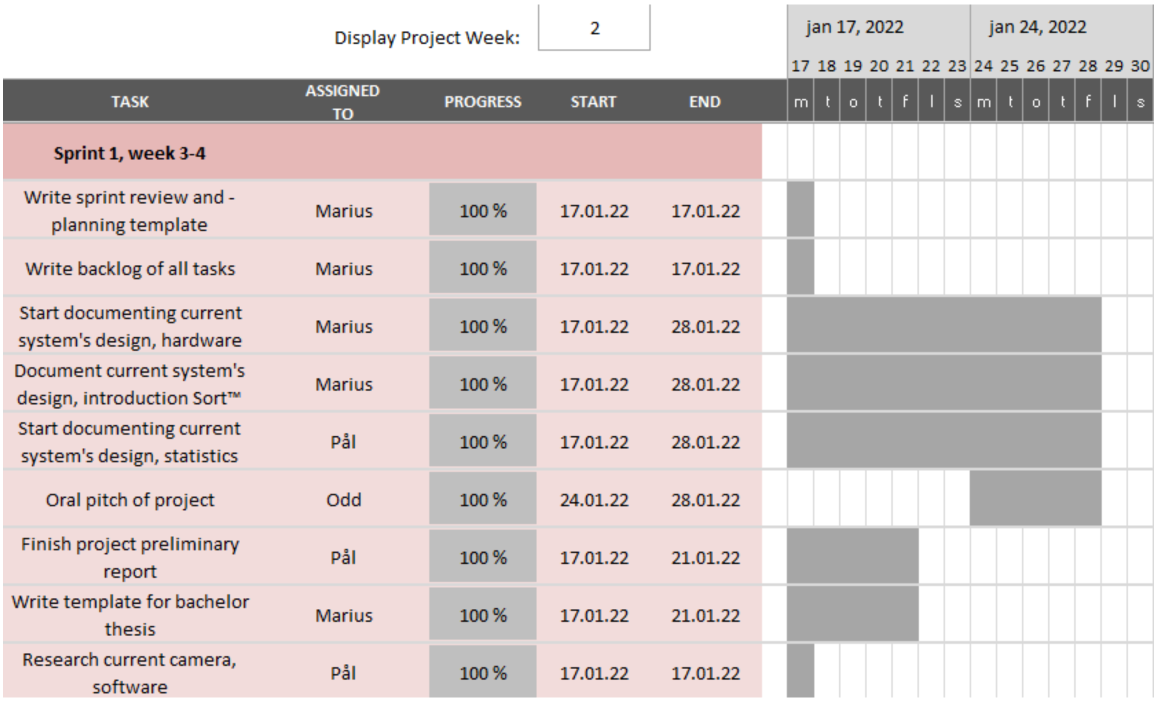
Final Gantt diagram & daily logs

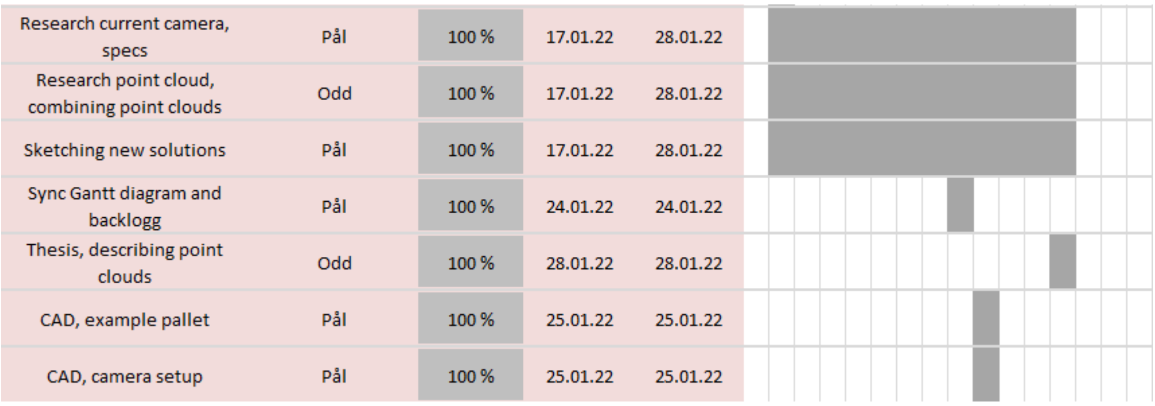


Project Start: 10.01.2022

Display Project Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END	jan 10, 2022							jan 17, 2022						
					10	11	12	13	14	15	16	17	18	19	20	21	22	23
					m	t	o	t	f	l	s	m	t	o	t	f	l	s
Start-up, week 2																		
Write Gantt diagram	Pål	100 %	10.01.22	10.01.22	■													
Write cooperation agreement	Pål	100 %	10.01.22	10.01.22	■													
Write daily log template	Marius	100 %	10.01.22	10.01.22	■													
Research Webots, install software	Odd	100 %	10.01.22	10.01.22	■													
Experiment with current camera	Marius	100 %	11.01.22	11.01.22		■												
Start of project preliminary report	Marius	100 %	12.01.22	14.01.22			■	■	■									
Write meeting template	Pål	100 %	13.01.22	13.01.22				■										





Display Project Week: 2

TASK	ASSIGNED TO	PROGRESS	START	END	jan 17, 2022							jan 24, 2022						
					17	18	19	20	21	22	23	24	25	26	27	28	29	30
					m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 1, week 3-4																		
Write sprint review and -planning template	Marius	100 %	17.01.22	17.01.22	■													
Write backlog of all tasks	Marius	100 %	17.01.22	17.01.22	■													
Start documenting current system's design, hardware	Marius	100 %	17.01.22	28.01.22	■	■	■	■	■	■	■	■	■	■	■	■		
Document current system's design, introduction Sort™	Marius	100 %	17.01.22	28.01.22	■	■	■	■	■	■	■	■	■	■	■	■		
Start documenting current system's design, statistics	Pål	100 %	17.01.22	28.01.22	■	■	■	■	■	■	■	■	■	■	■	■		
Oral pitch of project	Odd	100 %	24.01.22	28.01.22								■	■	■	■	■		
Finish project preliminary report	Pål	100 %	17.01.22	21.01.22	■	■	■	■	■									
Write template for bachelor thesis	Marius	100 %	17.01.22	21.01.22	■	■	■	■	■									
Research current camera, software	Pål	100 %	17.01.22	17.01.22	■													
Research current camera, specs	Pål	100 %	17.01.22	28.01.22	■	■	■	■	■	■	■	■	■	■	■	■		
Research point cloud, combining point clouds	Odd	100 %	17.01.22	28.01.22	■	■	■	■	■	■	■	■	■	■	■	■		
Sketching new solutions	Pål	100 %	17.01.22	28.01.22	■	■	■	■	■	■	■	■	■	■	■	■		
Sync Gantt diagram and backlogg	Pål	100 %	24.01.22	24.01.22								■						
Thesis, describing point clouds	Odd	100 %	28.01.22	28.01.22												■		
CAD, example pallet	Pål	100 %	25.01.22	25.01.22									■					
CAD, camera setup	Pål	100 %	25.01.22	25.01.22									■					

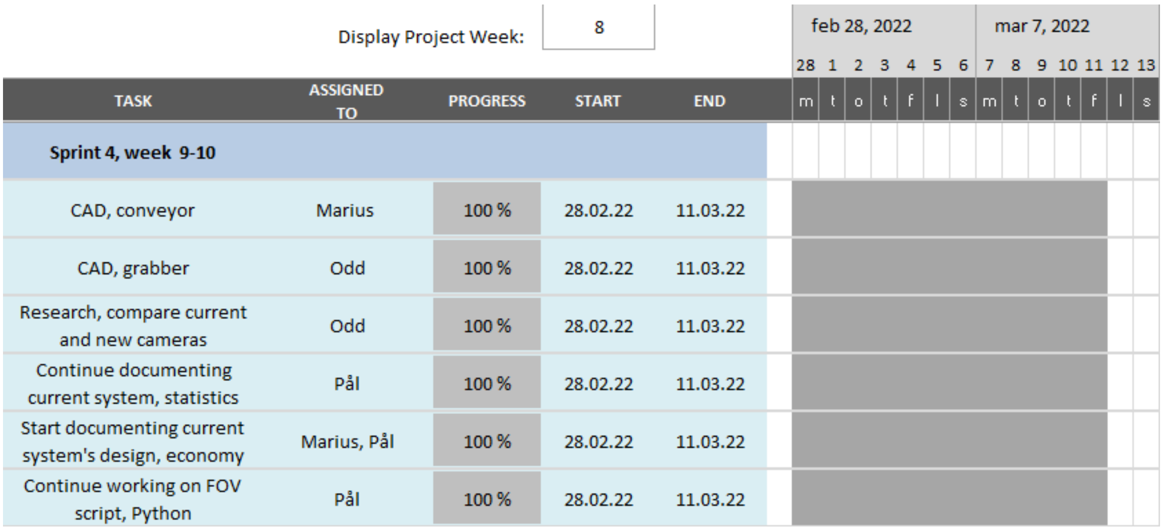
Project Start: 10.01.2022

Display Project Week: 4

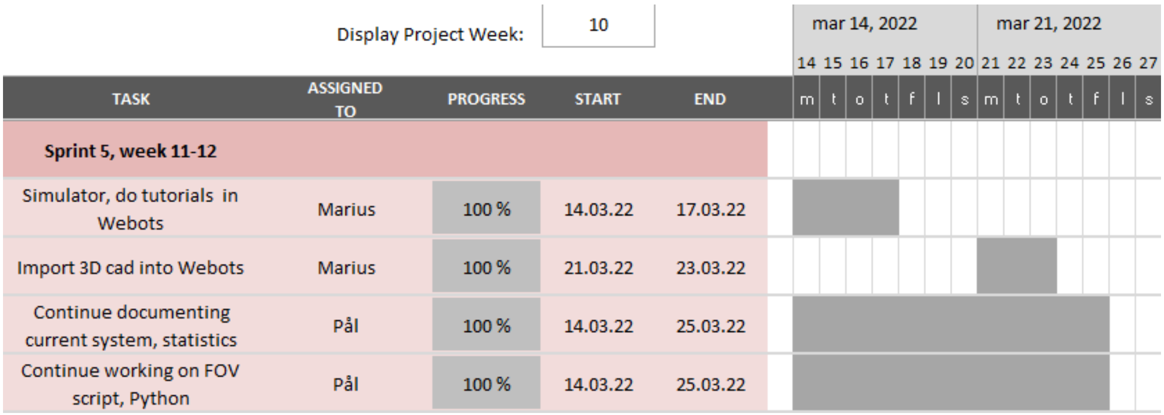
					jan 31, 2022							feb 7, 2022						
					31	1	2	3	4	5	6	7	8	9	10	11	12	13
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 2, week 5-6																		
Document current system's design, image acquisition	Odd	100 %	31.01.22	04.02.22	▨	▨	▨	▨	▨									
Start documenting current system's design, economy	Marius, Pål	100 %	31.01.22	11.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continue documenting current system, statistics	Pål	100 %	31.01.22	11.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Sketching new solutions	Pål	100 %	31.01.22	11.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
3D-modelling ideas	Pål	100 %	31.01.22	11.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Research new camera, specs	Odd	100 %	07.02.22	11.02.22								▨	▨	▨	▨	▨		
Research new camera, software	Odd	100 %	07.02.22	11.02.22								▨	▨	▨	▨	▨		
Thesis, describing new concepts	Pål	100 %	31.01.22	04.02.22	▨	▨	▨	▨	▨									
Thesis, describing current economy of system	Marius, Pål	100 %	07.02.22	11.02.22								▨	▨	▨	▨	▨		
Thesis, describing current economy of system	Marius, Pål	100 %	07.02.22	11.02.22								▨	▨	▨	▨	▨		
Thesis, finish describing current system, hardware	Marius	100 %	31.01.22	11.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Start programming FOV check in Python	Pål	100 %	03.02.22	11.02.22				▨	▨	▨	▨	▨	▨	▨	▨	▨		
Look into programming of FOV check in Python	Pål	100 %	02.02.22	03.02.22			▨	▨										

Display Project Week: 6

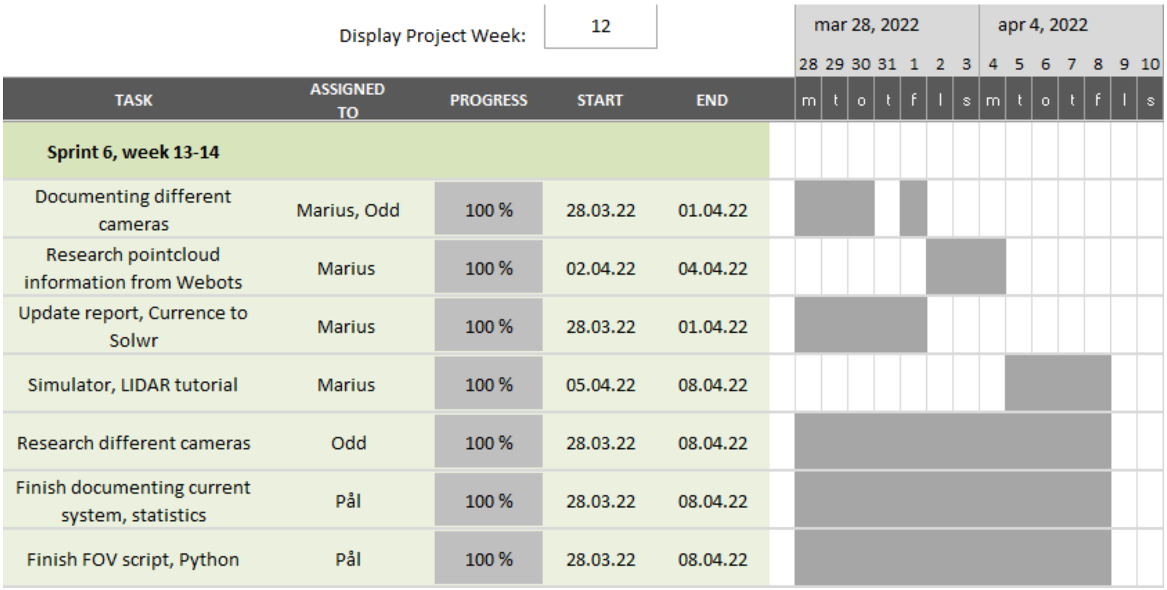
					feb 14, 2022							feb 21, 2022						
					14	15	16	17	18	19	20	21	22	23	24	25	26	27
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 3, week 7-8																		
CAD, frame	100	100 %	14.02.22	25.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
CAD, stack platform	Pål	100 %	14.02.22	25.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
CAD, camera	Pål	100 %	14.02.22	25.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continue working on FOV script, Python	Pål	100 %	14.02.22	25.02.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		



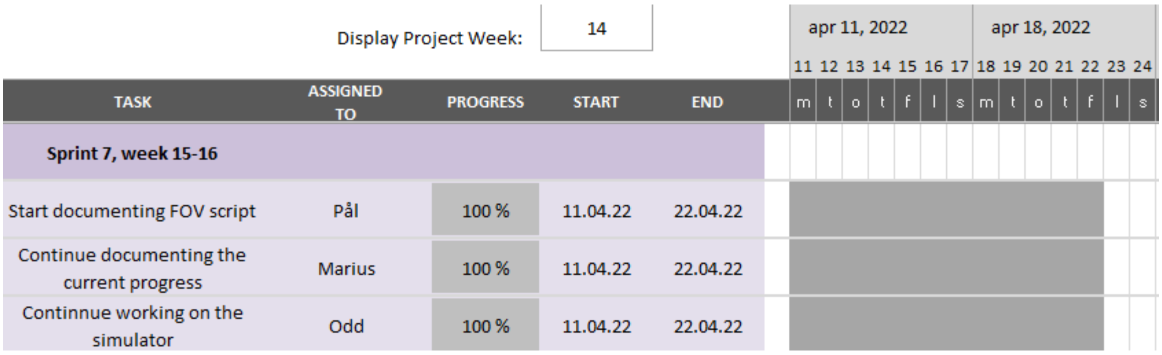
		Display Project Week:	8		feb 28, 2022							mar 7, 2022						
					28	1	2	3	4	5	6	7	8	9	10	11	12	13
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 4, week 9-10																		
CAD, conveyor	Marius	100 %	28.02.22	11.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
CAD, grabber	Odd	100 %	28.02.22	11.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Research, compare current and new cameras	Odd	100 %	28.02.22	11.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continue documenting current system, statistics	Pål	100 %	28.02.22	11.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Start documenting current system's design, economy	Marius, Pål	100 %	28.02.22	11.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continue working on FOV script, Python	Pål	100 %	28.02.22	11.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		



		Display Project Week:	10		mar 14, 2022							mar 21, 2022						
					14	15	16	17	18	19	20	21	22	23	24	25	26	27
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 5, week 11-12																		
Simulator, do tutorials in Webots	Marius	100 %	14.03.22	17.03.22	▨	▨	▨	▨										
Import 3D cad into Webots	Marius	100 %	21.03.22	23.03.22								▨	▨	▨				
Continue documenting current system, statistics	Pål	100 %	14.03.22	25.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continue working on FOV script, Python	Pål	100 %	14.03.22	25.03.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		

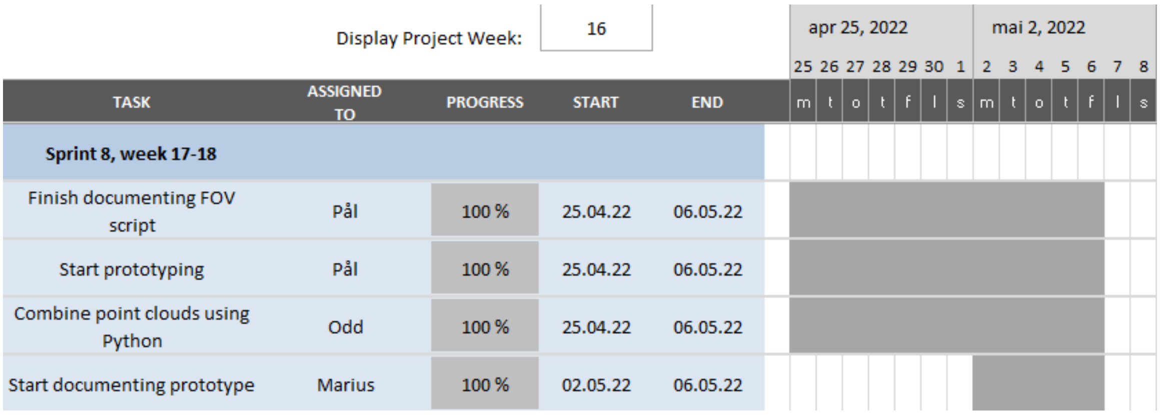


		Display Project Week:	12		mar 28, 2022							apr 4, 2022						
					28	29	30	31	1	2	3	4	5	6	7	8	9	10
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 6, week 13-14																		
Documenting different cameras	Marius, Odd	100 %	28.03.22	01.04.22	▨	▨	▨		▨									
Research pointcloud information from Webots	Marius	100 %	02.04.22	04.04.22						▨	▨	▨						
Update report, Currence to Solwr	Marius	100 %	28.03.22	01.04.22	▨	▨	▨	▨	▨									
Simulator, LIDAR tutorial	Marius	100 %	05.04.22	08.04.22									▨	▨	▨	▨		
Research different cameras	Odd	100 %	28.03.22	08.04.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Finish documenting current system, statistics	Pål	100 %	28.03.22	08.04.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Finish FOV script, Python	Pål	100 %	28.03.22	08.04.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		



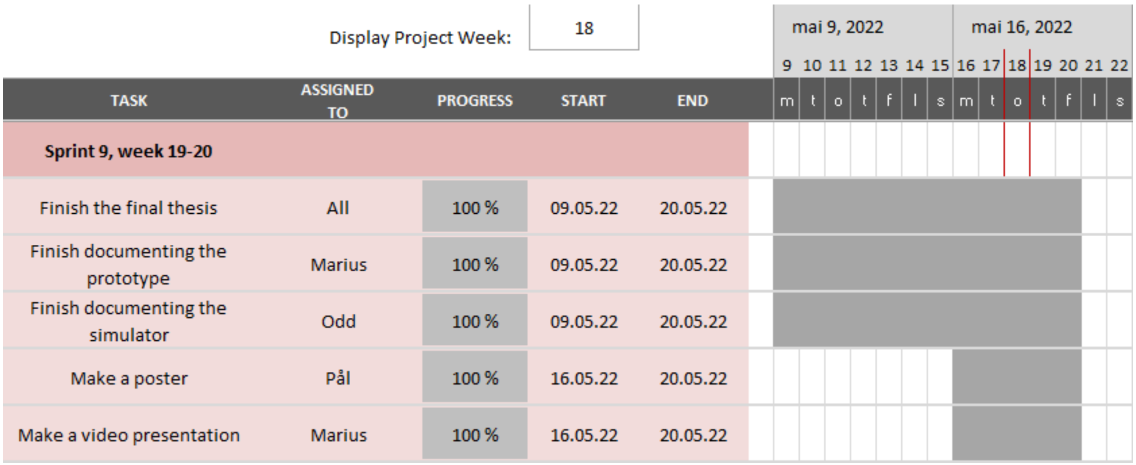
Display Project Week: 14

					apr 11, 2022							apr 18, 2022						
					11	12	13	14	15	16	17	18	19	20	21	22	23	24
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 7, week 15-16																		
Start documenting FOV script	Pål	100 %	11.04.22	22.04.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continue documenting the current progress	Marius	100 %	11.04.22	22.04.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Continnue working on the simulator	Odd	100 %	11.04.22	22.04.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		



Display Project Week: 16

					apr 25, 2022							mai 2, 2022						
					25	26	27	28	29	30	1	2	3	4	5	6	7	8
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 8, week 17-18																		
Finish documenting FOV script	Pål	100 %	25.04.22	06.05.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Start prototyping	Pål	100 %	25.04.22	06.05.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Combine point clouds using Python	Odd	100 %	25.04.22	06.05.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Start documenting prototype	Marius	100 %	02.05.22	06.05.22								▨	▨	▨	▨	▨		



Display Project Week: 18

					mai 9, 2022							mai 16, 2022						
					9	10	11	12	13	14	15	16	17	18	19	20	21	22
TASK	ASSIGNED TO	PROGRESS	START	END	m	t	o	t	f	l	s	m	t	o	t	f	l	s
Sprint 9, week 19-20																		
Finish the final thesis	All	100 %	09.05.22	20.05.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Finish documenting the prototype	Marius	100 %	09.05.22	20.05.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Finish documenting the simulator	Odd	100 %	09.05.22	20.05.22	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Make a poster	Pål	100 %	16.05.22	20.05.22								▨	▨	▨	▨	▨		
Make a video presentation	Marius	100 %	16.05.22	20.05.22								▨	▨	▨	▨	▨		

Appendix N

Original Gantt diagram

Full schedule		
Write bachelor application	21.11.21	21.11.21
Make Gantt diagram	10.01.22	10.01.22
Write cooperation agreement	10.01.22	10.01.22
Write daily log template	10.01.22	10.01.22
Write meeting template	13.01.22	13.01.22
Write sprint review and - planning template	17.01.22	17.01.22
Write backlog of all tasks	17.01.22	17.01.22
Experimenting with current camera, assembly	10.01.22	14.01.22
Research Webots, install software	10.01.22	14.01.22
Write template for bachelor thesis	17.01.22	21.01.22
Write project preliminary report	11.01.22	21.01.22
Oral pitch of project	28.01.22	28.01.22
Document current system's design, statistics	17.01.22	28.01.22
Document current system's design, resulting pointcloud	17.01.22	28.01.22
Document current system's design, pros and cons	17.01.22	28.01.22
Document current system's design, image acquisition	17.01.22	28.01.22
Document current system's design, economy	17.01.22	28.01.22
Research current camera, software	17.01.22	28.01.22
Research current camera, specs	17.01.22	28.01.22
Research new camera, software	31.01.22	11.02.22
Research new camera, specs	24.01.22	28.01.22
Research, compare current and new cameras	14.02.22	25.02.22
Research point cloud, combining point clouds	17.01.22	28.01.22
Sketch new solutions	17.01.22	11.02.22
CAD, frame	14.02.22	11.03.22

↩

CAD, example pallet	14.02.22	11.03.22
CAD, camera setup	14.02.22	11.03.22
CAD, conveyor	14.02.22	11.03.22
CAD, stack platform	14.02.22	11.03.22
CAD, grabber	14.02.22	11.03.22
CAD, camera	14.02.22	11.03.22
Simulator, do tutorials in Webots	14.03.22	17.03.22
Simulator, LIDAR tutorial	18.03.22	18.03.22
Simulator, program camera	21.03.22	22.04.22
Simulator, program movement feeding	21.03.22	22.04.22
Simulator, program grabber movement	21.03.22	22.04.22
Simulator, program pallet in-feed	21.03.22	22.04.22
Simulator, program color detection	21.03.22	22.04.22
Simulator, program defect detection	21.03.22	22.04.22
Simulator, moving pallets to specified location	21.03.22	22.04.22
Simulator, document results	20.04.22	22.04.22
Thesis, describing current system	21.01.22	21.01.22
Thesis, describing point clouds	28.01.22	28.01.22
Thesis, describing current image acquisition	28.01.22	28.01.22
Thesis, describing current economy of system	04.02.22	04.02.22
Thesis, describing potential cost of system	11.02.22	11.02.22
Thesis, describing new concepts	11.02.22	11.02.22
Thesis, describing how to test designs	18.02.22	18.02.22
Thesis, results of each design	22.04.22	22.04.22
Thesis, describing software developed by group	22.04.22	22.04.22
Thesis, conclusion of all simulators	22.04.22	22.04.22

Thesis, recommendation for which design to pursue	22.04.22	22.04.22
Prototype, assembly	18.04.22	13.05.22
Prototype, mount camera	18.04.22	13.05.22
Fysisk modell, testing	18.04.22	13.05.22
Thesis, polishing	16.05.22	20.05.22
Posterpresentation	20.05.22	20.05.22
Oral presentation of result	20.05.22	20.05.22

Appendix O

Hours worked

		Pål-André	Marius	Odd Arne
Week 2	10.01.2022	5,00	7	3,5
	11.01.2022	6,25	7,25	5,75
	12.01.2022	1,00	6,5	1
	13.01.2022	7,50	6,75	8
	14.01.2022	6,25	7	7
Week 3	17.01.2022	7,58	7,83	6,25
	18.01.2022	7,75	7,75	8
	19.01.2022	3,25	7,5	2,75
	20.01.2022	7,75	0	8
	21.01.2022	6,67	7,25	5,25
	22.01.2022	0,00	0,75	0
Week 4	24.01.2022	7,00	4	6
	25.01.2022	3,25	3,75	0,75
	26.01.2022	0,75	4	1
	27.01.2022	5,75	0	7,75
	28.01.2022	6,00	5,75	6,75
Week 5	31.01.2022	4,00	0	2
	01.02.2022	4,50	4,5	2
	02.02.2022	0,00	7	0
	03.02.2022	7,75	0	6
	04.02.2022	6,00	5,75	6
	05.02.2022	1,50	0	0
	06.02.2022	2,00	0	0
Week 6	07.02.2022	1,50	3	1
	08.02.2022	3,00	2	2
	09.02.2022	4,00	2	2
	10.02.2022	5,75	0	8
	11.02.2022	6,50	7	6,5
Week 7	14.02.2022	2,50	3,25	3,5
	15.02.2022	5,00	0	1,5
	17.02.2022	7,75	0	8
	18.02.2022	4,75	0	6
Week 8	21.02.2022	0,00	0	0
	22.02.2022	0,00	4,25	0
	23.02.2022	0,00	7,75	0
	24.02.2022	7,75	6	8
	25.02.2022	6,50	7,25	8

Week 9	28.02.2022	0,75	1,5	0
	01.03.2022	0,00	5,5	0
	02.03.2022	0,00	5	0
	03.03.2022	7,75	2	0
	04.03.2022	7,75	7,25	0
Week 10	10.03.2022	7,75	7,5	7,75
	11.03.2022	7,75	4,5	0
Week 11	14.03.2022	0,00	0	0
Week 12	23.03.2022	0,00	7,5	0
	24.03.2022	7,75	0	0
	25.03.2022	7,75	6	0
Week 13	28.03.2022	7,25	7	11,25
	29.03.2022	5,75	6	5,75
	30.03.2022	0,00	6,5	0
	31.03.2022	7,75	2	7,75
	01.04.2022	6,00	6	6
Week 14	04.04.2022	6,75	4,75	6
	05.04.2022	7,75	2	7,75
	06.04.2022	0,00	7,75	0
	07.04.2022	7,75	7,75	7,75
	08.04.2022	7,25	7,25	7,25
Week 15	11.04.2022	7,75	7,75	7,75
	12.04.2022	7,25	7,25	3,75
	13.04.2022	7,75	7,75	3,75
	14.04.2022	3,75	3,75	7,75
	15.04.2022	7,75	7,75	7,75
Week 16	18.04.2022	7,75	7	7,75
	19.04.2022	7,75	3,75	7,75
	21.04.2022	4,00	1,5	4
	22.04.2022	5,50	6,75	2
Week 17	25.04.2022	6,00	6	6
	26.04.2022	7,50	0,75	7,5
	27.04.2022	7,50	1	7,5
	28.04.2022	0,00	7,75	0

	29.04.2022	7,50	6,75	7,5
Week 18	02.05.2022	7,00	0	7
	03.05.2022	5,25	3,5	5,25
	04.05.2022	0,00	2,5	0
	05.05.2022	7,75	7,5	7,75
	06.05.2022	5,25	7,5	6,75
	07.05.2022	0,00	7,5	0
Week 19	09.05.2022	7,00	9	7
	10.05.2022	7,75	7	7,75
	11.05.2022	9,75	7,75	7,75
	12.05.2022	7,75	7,75	9,75
	13.05.2022	6,00	9,75	5,5
	14.04.2022	5,00	7	0
	15.04.2022	3,00	6	7
Week 20	16.05.2022	7,00	7	7
	17.05.2022	6,00	8,25	9
	18.05.2022	7,75	7,75	11
	19.05.2022	14	9,5	12,5
	Total hours:	**460**	**437,33**	**407,25**

Appendix P

Sprint review & planning

NTNU
Knowledge for a better world

Solwr

IELEA2920 - bachelor thesis

SPRINT REVIEW & PLANNING

Odd Arne Tynes, Pål-André Furnes,
Marius Høyer Melaas

A document including all sprint reviews and planning during the bachelor thesis.

May 2022

↩

Contents

Startup sprint, start 10.01.22, week 2

Main goal / purpose for this sprint

Having a project preliminary report that is almost done and ready to be evaluated by teachers.

Planned activities this sprint

- Downloading needed software
- Experimenting with software
- Experimenting with current LIDAR cameras
- Writing templates for documents.
  - Bachelor thesis
  - Meeting reports
  - Sprint reviews
- Writing a project preliminary report

Actually conducted activities this sprint

- Downloading needed software
- Experimenting with software
- Experimenting with current LIDAR cameras
- Writing templates for documents.
  - Meeting reports.
- Writing a project preliminary report

Description of / justification for deviation between planned and real activities

The sprint review template was made the first day of the next sprint. The main thesis template will be done by the end of week 3 when we are going to start writing the final product.

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

No desired changes.

Main experience from this period

The group has gained some fundamental knowledge about point clouds and how they are made. The group used Intel RealSense SDK to view the camera output. It has also started building 3D-structures in Webots, the simulator software that is going to be used going forward.

Main focus next period

- Sketching possible solutions for new camera setups
- Understanding the current camera setup
- Project preliminary report

Planned activities next period

- Finish project preliminary report.
- Deliver the project preliminary report to mentors for evaluation.
- Review mentors comments on the project preliminary report.
- Contact Solwr for possible resources in form of simulators and CAD files, that the group can utilize.
- Make a template for the bachelor thesis.
- Start writing in the bachelor thesis.
- Document the functionality of the current camera systems, on Sort^{tm}.
- Document a rough estimate of the current cost and maintenance for the current camera system, on Sort^{tm}.
- Brainstorm on concepts for the new camera setup.
- Sketch ideas for new camera setups.

Other

Nothing to note.

Wish / need for counseling

Nothing in particular.

1 Sprint 1, start 17.01.22, week 3

Main goal / purpose for this sprint

Produce sketches of ideas to be presented to mentors and employer, and document current solution.

Planned activities this sprint

- Finish project preliminary report.
- Deliver the project preliminary report to mentors for evaluation.
- Review mentors comments on the project preliminary report.
- Contact Solwr for possible resources in form of simulators and CAD files, that the group can utilize.
- Make a template for the bachelor thesis.
- Start writing in the bachelor thesis.
- Document the functionality of the current camera systems, on Sort™.
- Document a rough estimate of the current cost and maintenance for the current camera system, on Sort™.
- Brainstorm on concepts for the new camera setup.
- Sketch ideas for new camera setups.

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- Finish project preliminary report. ✓
- Deliver the project preliminary report to mentors for evaluation. ✓
- Review mentors comments on the project preliminary report. ✓
- Contact Solwr for possible resources in form of simulators and CAD files, that the group can utilize. ✓
- Make a template for the bachelor thesis. ✓
- Start writing in the bachelor thesis. ✓
- Document the functionality of the current camera systems, on Sort™. ✗
- ~~Document a rough estimate of the current cost and maintenance for the current camera system, on Sort™.~~
- Brainstorm on concepts for the new camera setup. ✓
- Sketch ideas for new camera setups. ✓
- **3D-model sketches in Onshape.** ✓

Description of / justification for potential deviation between planned and real activities

Not enough data about the current system has been gathered to write about the total costs of the system.

Not enough time to 100% document the current system functionality. To fulfill the required quality in the thesis, this was more work then anticipated. However, the task is approximate 50% completed.

In order to better present the ideas, it was decided that 3D-modelling sooner would be better.

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

3D-modelling moved up one week because it was better to present 3D-models than just sketches in the meeting.

To add documentation of current solution in next sprint.

Main experience from this period

Good progress in regards to learning Onshape. The selected 3D-modeling software for this project.

Good communication with the employer, and valuable feedback on the project preliminary report from mentors.

Main purpose / focus next period

- touching up on the previous ideas and improving the 3D-models
- new brainstorm session for new ideas
- documenting the current solution
- documenting new cameras

Planned activities next period

- research new camera
- sketching new solutions
- 3D-modelling ideas
- finish writing about the current system: hardware
- writing about the current camera: software
- documenting the current system: economy
- documenting the current system: image acquisition
- documenting the current system: statistics
- documenting new cameras

Other

Nothing to note.

Wish / need for counseling

The group wishes to get a better insight into the economics of the current system in order to document the changes in cost of a new system.

The group wishes to get a better insight into the software, coding and algorithm behind the current system. This is to better understand liming factors when designing new solutions.

2 Sprint 2, start 31.01.22, week 5

Main goal / purpose for this sprint

- touching up on the previous ideas and improving the 3D-models
- new brainstorm session for new ideas
- documenting the current solution
- documenting new cameras

Planned activities this sprint

- Finish documenting image acquisition of the current system
- Document the current system's economy
- Document the current system's statistics
- Sketch and CAD possible solutions
- Research and document new camera specs
- Document new solutions in the thesis

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Skipped: ~~crossed out~~, Added: **Bold text**

- Finish documenting image acquisition of the current system✓
- Document the current system's economy✗
- Document the current system's statistics✓
- Sketch and CAD possible solutions✓
- Research and document new camera specs✓
- Document new solutions in the thesis✓
- **Document how many images are needed to see all required surfaces✓**

Description of / justification for potential deviation between planned and real activities

The current system's economy need to be supplied by Solwr, and the group time for this task was used on a added task for the sprint. The added task was that Solwr specifically asked for us to complete our analysis of how many cameras would be needed, at minimum.

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

Because of our digression regarding the documentation of the current system's economy, we will need to postpone this to next sprint. We will be more focused on a specific solution. This frees up more time to obtain the necessary documentation.

Main experience from this period

We've had more talks with Solwr, they've clarified their expectations for this project and what immediate tasks we should focus on.

Main purpose / focus next period

- 3D model our most promising solution in Onshape.
- Establish new meeting routines
- Investigate if Zivid two is a viable option.

Planned activities next period

- CAD, camera portal
- CAD, cameras
- CAD, conveyor
- CAD, stacking platform
- CAD, grabber
- finish FOV checker in Python

Other

Going forward, the meetings associated with each sprint will be held on Tuesdays the following week. This is due to schedules not lining up.

Wish / need for counseling

3 Sprint 3, start 14.02.22, week 7

Main goal / purpose for this sprint

- 3D model our most promising solution in Onshape.
- Establish new meeting routines
- Investigate if Zivid two is a viable option.

Planned activities this sprint

- CAD, camera portal
- CAD, cameras
- CAD, conveyor
- CAD, stacking platform
- CAD, grabber
- finish FOV checker in Python

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- CAD, camera portal ✓
- CAD, cameras ✓
- CAD, conveyor ✓
- CAD, stacking platform ✓
- CAD, grabber ✗
- finish FOV checker in Python ✗

Description of / justification for potential deviation between planned and real activities

We found out that we did not need all models in order to progress in regards to the simulation. We will be able to test a basic version with the models we completed and from that we will know how detailed we need to make the other ones.

The FOV script also proved more time intensive than first imagined. Due to another course running parallel to the bachelor thesis, we had to deprioritize the script. It currently functions as a proof-of-concept. The only thing missing is to make the script handle the field-of-view of cameras and not 360°/360°coverage.

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

We will be moving up the work with the simulator. This is because we have gotten feedback that this is a very time consuming activity and we do not see 3D-modelling in great detail to serve our goal.

Main experience from this period

The group has gained experience in simulating and 3D-modelling.

Main purpose / focus next period

Get an understanding of how to simulate our solution in Webots. The group will also finish importing the needed components to Webots.

Planned activities next period

- Do tutorials in Webots
- Program camera in Webots
- Import 3D-models to Webots
- Program movement of pallets in Webots

Other

None

Wish / need for counseling

None

4 Sprint 4, start 28.02.22, week 9

Main goal / purpose for this sprint

Get an understanding of how to simulate our solution in Webots. The group will also finish importing the needed components to Webots.

Planned activities this sprint

- Do tutorials in Webots
- Program camera in Webots
- Import 3D-models to Webots
- Program movement of pallets in Webots

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- Do tutorials in Webots ✓
- ~~Program camera in Webots~~
- Import 3D-models to Webots ✓
- ~~Program movement of pallets in Webots~~

Description of / justification for potential deviation between planned and real activities

The group has an intensive course running alongside the bachelor thesis. It was decided that in order to get good grades in that subject, the group would have to change their focus. Not a lot of progress has been made to the thesis this sprint.

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

Shift the Gantt diagram back a few weeks while the other subject concludes

Main experience from this period

Making a simulator will be time intensive. The group finds it likely that this will occupy their time all the way up to the delivery deadline of the thesis.

Main purpose / focus next period

Give status to Solwr regarding the group's progress

Planned activities next period

- Meeting with Solwr
- Continue working on FOV-script

Other

None

Wish / need for counseling

None

5 Sprint 5, start 14.03.22, week 11

Main goal / purpose for this sprint

Give status to Solwr regarding the group's progress

Planned activities this sprint

- Meeting with Solwr
- Continue working on FOV-script

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- ✓Meeting with Solwr
- ✓Continue working on FOV-script

Description of / justification for potential deviation between planned and real activities

None

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

None other than previously mentioned. The Gantt diagram will be shifted a few weeks back in order for the group to focus on another subject.

Main experience from this period

Solwr has provided some additional manufacturers they wanted the group to look into.

Main purpose / focus next period

Begin programming the simulator

Planned activities next period

- Finish FOV-script
- Finish simulator tutorials
- Simulate movement of 3D-objects in Webots
- Research different cameras
- Finish documenting current system's statistics
- Obtain a point cloud from a Webots object

Other

None

Wish / need for counseling

None

6 Sprint 6, start 28.03.22, week 13

Main goal / purpose for this sprint

Program a simulator

Planned activities this sprint

- Finish FOV-script
- Finish simulator tutorials
- Simulate movement of 3D-objects in Webots
- Research different cameras
- Finish documenting current system's statistics
- Obtain a point cloud from a Webots object

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- Finish FOV-script ✓
- Finish simulator tutorials ✓
- Simulate movement of 3D-objects in Webots ✓
- Research different cameras ✓
- Finish documenting current system's statistics ✓
- Obtain a point cloud from a Webots object ✓

Description of / justification for potential deviation between planned and real activities

None

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

None

Main experience from this period

The group has made a lot of progress regarding the simulator. The main experience from this sprint is about generating and exporting data from a simulator.

Main purpose / focus next period

Finish simulation

Planned activities next period

- Complete simulation
- Document current progress

Other

None

Wish / need for counseling

None

7 Sprint 7, start 11.04.22, week 15

Main goal / purpose for this sprint

Finish simulator

Planned activities this sprint

- Complete simulation
- Document current progress

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- Complete simulation ✗
- Document current progress ✗

Description of / justification for potential deviation between planned and real activities

None

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

None

Main experience from this period

A lot of progress has been made in regards to the simulation. The group has been focusing on different methods of gathering distance and color data in Webots.

Main purpose / focus next period

Complete the simulator

Planned activities next period

- Finish simulator
- Continue documenting progress in the report

Other

None

Wish / need for counseling

None

8 Sprint 8 start 25.04.22, week 17

Main goal / purpose for this sprint

Complete the simulator

Planned activities this sprint

- Finish simulator
- Continue documenting progress in the report

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- Finish simulator ✓
- Continue documenting progress in the report ✓

Description of / justification for potential deviation between planned and real activities

None

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

None

Main experience from this period

The prototype was a big learning experience. Connecting the theoretical to the practical.

Main purpose / focus next period

Completing the thesis

Planned activities next period

- Finish the thesis
- Make a poster
- Make a video presentation

Other

None

Wish / need for counseling

None

9 Sprint 9 start 09.05.22, week 19

Main goal / purpose for this sprint

Completing the thesis

Planned activities this sprint

- Finish the thesis
- Make a poster
- Make a video presentation

Actually conducted activities this sprint

Completed: ✓, In progress: ✗, Postponed: ~~crossed out~~, Added: **Bold text**

- Finish the thesis ✓
- Make a poster ✓
- Make a video presentation ✓

Description of / justification for potential deviation between planned and real activities

None

Description of / justification for changes that is desired in the projects content or in the further plan of action – or progress report

None

Main experience from this period

Writing a bachelor is a time-consuming process. The final weeks included a lot of hours outside of the prospected working hours. The group should have written more as they progressed.

Other

None

Wish / need for counseling

None

Appendix Q

Bachelor meetings report

NTNU

Knowledge for a better world

Solwr

IELEA2920 - bachelor thesis

MEETING REPORTS

Odd Arne Tynes, Pål-André Furnes,
Marius Høyer Melaas

A report including all official meetings regarding the bachelor thesis.

May 2022

↩

Contents

1 Sprint meeting 14.01.2022, Week 2

1.1 Main topic: Startup meeting with all involving partners

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D., Ottar L.O., Erlend Magnus L.C., Lars Ivar H.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D., Ottar L.O., Erlend Magnus L.C., Lars Ivar H.

Preliminary questions/answers:

- Do Currence Robotics require a written agreement regarding the bachelor thesis?
  - No.
- What cameras should the group base their simulation and model on?
  - The group will use Intel's cameras. They produce results similar enough to Zivid's.
- Will the group have access to any desk or office space at Driw?
  - No answer at this time. Driw is currently quite full and closed due to corona restrictions, but we will come back to this after corona restrictions are lifted.
- Should the group have a local contact person at Currence?
  - We should try using Slack as the main communication medium. Use it often. If we know someone has a lot of knowledge on a certain subject, we can direct questions to them specifically.
- Do we have a budget for ordering parts we may need?
  - Currence Robotics has a lot of equipment already, try using that first. Ordering new equipment is also possible after dialog with company representative.
- Do Currence already have a finished simulator in Webots?
  - Yes, a model is ready. Direct questions about Webots to Harald. Alternatively Trondheim.
- How should the group distribute Gantt diagram, agenda and progress report?
  - When inviting to a meeting, the group will upload the relevant documents to that meeting's files.

Meeting agenda:

- General introduction
- Deciding boundaries for the thesis
- Establishing routine for meetings
- Starts first sprint (sprint planning this Monday)

Meeting summary:

- We should be using Intel's cameras, the old ones. The field of view is similar enough to be used as a replacement and will give a good enough proof-of-concept.
- The group is advised to produce many different concepts, no matter how stupid, and present them to Currence Robotics so that they can decide which ones to pursue.
- The group should take care to clearly define the scope of the project. This can be modifies as time passes, but the initial scope needs to be clear to everyone involved.
- Meetings will be held every other Friday. The project group will handle invitations and meeting agenda.

2 Sprint meeting 28.01.22, Week 4

2.1 Main topic: Showing off possible new solutions

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D., Ottar L.O., Erlend Magnus L.C., Lars Ivar H.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Ottar L.O., Lars Ivar H.

Preliminary questions/answers:

- What should we call ”Styringsgruppa” in English?
  - Reference group.
- How to include the preliminary report in appendix?
  - Include it in the final thesis. Not in a seperate file.
- What is your opinion on the group's way of writing references, and should they offer additional info?
  - Some examples have been put into the thesis. We should include date accessed.
- Should we include a separate chapter in the final thesis for Sort, or should we write it in ”Theory”?
  - Put it in Material and method. Use a design process where looking at previous designs is part of it. This gives a natural transition into results.
- When should the group distribute the Gantt diagram and progress report?
  - It should be available by lunch time the day before.
- Can we push the meeting start by half an hour going forward 09:30 - 10:30?
  - Schedule with everyone by e-mail. The weekday is likely to change. Meetings should be shorter, about 30 minutes, going forward.
- How should the group handle copying text from the project preliminary report?
  - Refer to appendix, and text in cursive.

Meeting agenda:

- progress report
- general questions
- showing a sample of sketches and 3D-models
- feedback on said models

Meeting summary:

- Got a tip about using draw.io for flowcharts.
- The group showed some sketches, and discussed some additional ideas to be sketched up in the coming sprint.
- The group notified the reference group that the sprint is ambitious, and this is intentional to reach for high results.
- The meeting schedule should be changed. Looking at possibility of having it on Wednesdays.
- We shoould present ideas to Currence Robotics as they come, by Slack, and not wait for a scheduled meeting.
- We should use a design process to document the progress we make, and how we make it. This gives the reader of the final thesis a sense of timeline of when and how we reached our final result.
- We should look into Simplygon. A simple 3D-modelling software to easily present possible solutions
- New ideas shared by reference group:
  - Using a belt with camera attached to move it into multiple locations.
  - Using ball joints on the frame sketched in design 1 and 5. This gives an alternative to having the camera move up and down.
  - Use the gripper to move pallet into position. This is in order to reduce having to move the camera(s).
  - Using a rotating surface with camera attached that moves the camera in a circular pattern.
  - Using rope control to move camera. (I.e: BR Automation's CableEndy)

3 Progress meeting 04.02.22, Week 5

3.1 Main topic: Discussing possible solutions with employer

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D.

Preliminary questions/answers:

- Will it be possible for the group to use Driw as a workplace?
  - The group was asked to send mail to Andrea to check if there were any available space at Driw.
- Ask employer to elaborate on the point cloud from Sigmund.
  - They explained why there was missing pices in the point cloud.
- Will sprint planning meeting at Wednesday 13:00 work for the employer?
  - Should send mail to all participants in order to see what time is best for everyone.
- What solution is most promising, and the group should pursue?
  - The group should verify what can be seen using two cameras first. Based on this the design will either have two, four or more cameras.

Meeting agenda:

- Discuss adding a new arm/grabber
- Discuss rotating while lifting
- Discussing if a much higher pallet count should be a motivation for our designs
- Show Gantt diagram and plans forward.

Meeting summary:

- The group discussed some of the solutions
- The employer wanted the group to create a 3D CAD of FOV on Onshape
- Andrea will look in to workspace on Driw.
- The group created a Doodle sheet to find a new time for the biweekly sprint meeting.

4 Sprint meeting 15.02.22, Week 7

4.1 Main topic: General progress

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Ottar L.O., Erlend Magnus L.C., Lars Ivar H., Olivier R.D.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Lars Ivar H.

Preliminary questions/answers:

- Should we go deep into the theory of point clouds or keep it more surface level?
  - Explain it in enough details so that any reader understands what we are presenting.
- Since the Zivid camera has a higher resolution, will this result in a longer computation time?
  - Not answered

Meeting agenda:

- Discuss progress the group has made
- Discuss possible solutions and CADs

Meeting summary:

The group showed the progress they had made. In regards to the thesis, programming, and the most promising camera setup.

5 Sprint meeting 01.03.22, Week 9

5.1 Main topic: General progress

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Ottar L.O., Erlend Magnus L.C., Lars Ivar H.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Lars Ivar H.

Preliminary questions/answers:

- None

Meeting agenda:

- Discuss progress since last meeting
- Discuss the plan moving forward
- Discuss feedback from Currence

Meeting summary:

- The group has not made the progress they envisioned. This is due to the wrapping up of a different subject. This should be better once the subject is concludes.
- The group was reminded on how big of a task simulating a physical system is. They should keep this in mind when allocating time to this.
- If the group for some reason does not have enough time to finish a simulator, they should look into implementing a ray tracing script in Unity.

6 Progress meeting 10.03.22, Week 5

6.1 Main topic: Discussing progress with employer

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D., Solwr team

---

Preliminary questions/answers:

Meeting agenda:

- Discuss most promising solution
- Receive feedback on the solution
- Plan the next step of the project

Meeting summary:

- The most promising design seems to be viable. No major flaws can be found.
- One of the cameras would need to be moved, but this is unproblematic. It has to be moved due to collisions with the wall behind Sort™.
- The group should start looking for different manufacturers in order to make the new design cheaper or better.

7 Sprint meeting 29.03.22, Week 13

7.1 Main topic: Discussing progress and focus going forward

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Erlend Magnus L.C., Lars Ivar H.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Lars Ivar H.

Preliminary questions/answers:

- None

Meeting agenda:

- Discussing progress
- Discussing meeting with Solwr
- Discussing focus going forward

Meeting summary:

- The group gave an update on their progress. Good progress regarding FOV script.
- The group has just finished it's second subject. They are just now getting started with the bachelor again.
- The group will focus on comparing different cameras and deciding which one is the most suited for their purpose, for the time being.

8 Progress meeting 05.04.22, Week 14

8.1 Main topic: Discussing possible cameras

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D., Thomas S.M., Rodrigo U.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Thomas S.M., Rodrigo U.

Preliminary questions/answers:

- Software used for point-clouds in today's system.
    - Solwr has developed their own software, and website to display all point-clouds from cameras at Sort™.

Meeting agenda:

- Debating alternative cameras other then Zivid
- Check status for Zivid camera order
- Discuss placement number of cameras max pricing
- Discuss today's solution of displaying point-clouds

Meeting summary:

- Solwr was happy with the groups research. Together we concluded to reach out to the companies with viable solutions.
- The group presented the plan going forward:
    - Display multiple LIDAR point-clouds
    - Mimic a real camera spec
    - Make simulation with 3 cameras
    - Test various camera setups for best point-cloud
    - Documentation of all work in thesis
    - Play around with the Intel camera, so we are ready when the Zivid arrives
    - Testing Zivid cameras when they arrive
    - Display a 3D LIDAR point-cloud

9 Meeting 12.04.22, Week 15

9.1 Main topic: Discussing the plan for the final two sprints

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Ottar L.O., Erlend Magnus L.C., Lars Ivar H.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Erlend Magnus L.C., Lars Ivar H.

Preliminary questions/answers:

- How do we turn in the code for the software we have made?
    - The reference group will come back to this.

Meeting agenda:

- Discussing the group's progress
- Deciding whether or not to build a prototype

Meeting summary:

- The group has made a lot of progress
- The simulator is showing promising results

10 Meeting 26.04.22, Week 17

10.1 Main topic: Prototyping

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Olivier R.D., Ottar L.O., Erlend Magnus L.C., Lars Ivar H.

Attended parties: Marius H.M, Odd Arne T., Pål-André F., Erlend Magnus L.C., Lars Ivar H.

Preliminary questions/answers:

- Should the group spend time building a prototype?
  - Yes, a physical prototype would benefit the thesis greatly.

Meeting agenda:

- Discussing the group's progress
- Deciding whether or not the group should build a prototype

Meeting summary:

- The group has made a lot of progress in regards to the simulator.
- The group should build a prototype.

11 Meeting 10.05.22, Week 19

11.1 Main topic: The final period

Invited parties: Marius H.M, Odd Arne T., Pål-André F., Erlend Magnus L.C., Lars Ivar H.

Attended parties:

Preliminary questions/answers:

- How should the group turn in the software they have produced?
    - We should add the source code to the appendix. Given that it is less than 15 pages.
- When should we have the poster ready?
    - The deadline will be announced.
- Is the current way of writing okay?
    - We should try to make it less sections, combining the current sections that are within the same area.

Meeting agenda:

- Discussing the group's progress
- Discussing the prototype

Meeting summary:

- The current setup is too divided. Making it fewer, but bigger, sections would be more appropriate.
- Overall, the group has made good progress. They should now focus on getting all the data into the thesis.

Appendix R

Project preliminary report

Currence Robotics

IELEA2920 - bachelor thesis

PROJECT PRELIMINARY REPORT

Odd Arne Tynes, Pål-André Furnes,
Marius Høyer Melaas

January 2022

↩

Contents

List of Figures

List of Tables

1 Introduction

The project is to develop, test and evaluate enhancements to a pallet sorting machine produced for logistics centers across Norway by Currence Robotics.

The group will primarily work with designing a new image acquisition solution. This is due to the current cameras being discontinued by the manufacturer. The current setup uses 6 cameras to map each pallet from multiple angles. The camera that is considered to be the replacement is too expensive for this setup, due to competing systems' pricing. The group will therefore attempt to minimize the amount of cameras needed, without compromising the system's accuracy.

This will involve testing different setups. E.g. 4 stationary cameras, 2 cameras with actuators or 2D-cameras with AI image recognition. In addition to this, the group needs to be wary of lighting conditions in different warehouses. The group must decide whether or not to mount lighting on the new setup to get consistent data from the cameras.

The new concept will be tested using the current cameras. If successful, it will be implemented in the next design of the pallet sorter.

If there is enough time, the group will also look into adding functionality to the existing robot. E.g. adding sensors for weight and RFID. This is to sort out water-logged pallets and aid error detection.

Currence Robotics has a working algorithm for recognition of pallets, but the group might write an algorithm of their own. This is for educational purposes, but also to investigate alternate ways to detect pallet defects.

Another assignment the group is considering adding is assembling and setup of a new conveyor system. This will include mechanical work, programming of PLCs, and electrical wiring. Each modification of the original goal will be discussed with the group's mentors and representatives from Currence Robotics in advance.

2 Project organization

2.1 Project group & mentors

Project group's student ID	
Marius Høyer Melaas	ID: 498780
Pål-André Furnes	ID: 510340
Odd Arne Tynes	ID: 522481

Table 1: Student names and their student ID

Mentors & business representative		
Mentor	Lars Ivar Hatledal	NTNU
Mentor	Erlend Magnus Lervik Coates	NTNU
Senior advisor	Ottar Laurits Osen	NTNU
Employer (CTO)	Olivier Roulet-Dubonnet	Currence Robotics

Table 2: Key personnel and their role in the project

2.2 The group

The group consists of three members on their last semester of an automation bachelor program, at NTNU Ålesund. The members have quite different backgrounds, which the group sees as a strength. Earlier, the group has done several projects together, with good communication and results.

2.2.1 Marius Høyer Melaas

Marius is 26-years-old from Molde, which is 1,5 hour away from Ålesund. He finished high-school with a certificate of apprenticeship in automation, and afterwards moved to Oslo to finish his second certificate of apprenticeship in electro. Thereafter he got drafted for the navy, and was responsible for electrical and mechanical maintenance aboard a minesweeper named Hinnøy.

After his service in the navy he started ”Forkurs for ingeniørfag”-course at NTNU during summer, and therefore could start directly on the automation bachelor program in august 2019.

2.2.2 Pål-André Furnes

Pål-André is a 24-year-old raised on Vigra, close to Ålesund. He decided to be an engineer when he was 10 years old, and has now almost reached that goal. He spends his days either working out or working a part-time job.

He finished high-school with a certificate of apprenticeship in Automation. After high-school he got drafted for military service where he spent his days maintaining the weapons systems of a coast guard vessel in the northern parts of Norway, utilizing his previous experiences in automation. Before his bachelor's degree he spent one year taking the "Forkurs for ingeniørfag"-course at NTNU.

2.2.3 Odd Arne Tynes

Odd Arne is 27 years old and grew up in Sykkylven, one of Ålesund's neighbors. He finished high-school with a stud-spes diploma. After this he went on to study physics at the university of Bergen.

After finishing a bachelor degree in physics He decided he wanted to focus on a different field and landed on automation engineer. While studying he has been working part time at various places.

2.3 Employer

Currence Robotics is a Norwegian tech company that develops and manufactures robots for the warehouse industry. They have offices both in Ålesund and Trondheim. In Ålesund, they share office space with Driw AS, at a building named "Driw-huset".

Currence Robotics is a well-established company with a range of strong industrial partners, suppliers and professional investors. Including a wide spread expertise with a team of mathematicians, engineers, scientists, full stack developers, machine learners and logistic experts. In addition, they well supported by Innovation Norway and the Norwegian Research Council [1].

2.4 Projects roles

Project roles within the bachelor group	
Marius Høyer Melaas	Project leader
Pål-André Furnes	Secretary
Odd Arne Tynes	Archivist

Table 3: Students and their corresponding roles

2.4.1 Responsibilities and tasks for the project leader

- Internal communication between the bachelor group and Currence Robotics.
- Ordering or buying parts needed.

2.4.2 Responsibilities and tasks for the secretary

- Send out meeting invitations, update Gantt diagram, and agenda.
- Book rooms for meetings.
- Write meeting reports.

2.4.3 Responsibilities and tasks for the archivist

- To upload Gantt diagram before every meeting
- Write sprint review and sprint planning.

3 Arrangements and agreements

3.1 Bachelor's contract with Currence Robotics

Currence Robotics did not consider it necessary to write a contract. If proprietary information is shared, a Non-Disclosure-Agreement can be considered.

3.2 Workplace and resources

Due to Covid restrictions, the employees at Currence Robotics works remotely. Therefore, it was not possible for the group to have their own assigned workspace at Driw at project start. When the restrictions are lifted, the employer will make a new evaluation.

The project group can, with permission, use the equipment already at Currence Robotics' storage area. Currence Robotics can order new components as the group needs them, if the components are necessary for the project.

The group is welcome to contact Currence Robotics employees directly or ask questions on a common server on Slack, a communications platform.

3.3 Cooperation agreement

3.3.1 Duration of agreement

The agreement duration is from the 10th of January 2022 until 20th of May 2022, when the thesis is to be delivered.

3.3.2 Purpose of the agreement

The group is writing it's bachelor thesis with Currence Robotics. This agreement is formalized to achieve accountability for each group member to follow set rules in chapter 3.3.3.

3.3.3 Rules

- Each group member must write a daily log describing the work they have done, in relation to the bachelor thesis.
- Each group member must dedicate two hours each week to write the project thesis.
- If a group member is unable to meet up with the others at the agreed upon time and location, they must say so as soon as possible.
- Each group member must work an average of four workdays each week. Totaling 32 hours each week, including lunch breaks.
- Each group member must attend sprint planning every other week. These meetings specify what each member should be working on the coming weeks.

3.3.4 Responsibilities of each group member

- Daily log - Marius
- Weekly writing of thesis - Odd Arne
- Updating Gantt diagram - Pål-André

4 Project description

4.1 Goals and purpose

The group's main goal for this project is to develop a new image acquisition solution. If successful, the solution is to be implemented in Currence Robotics' pallet sorter, named Sort™.

The group will be working with the version of Sort™ located at H.I. Giørtz, illustrated in figure 1. It was built in 2019 and continuously updated with improvements. This instance of Sort was originally built as a prototype, but has since been developed into a commercial product.

The purpose of Sort™ is to sort pallets based on their characteristics into separate stacks as seen in figure 1. In addition to this, Sort™ also determines whether or not a pallet is defective, either due to damage or dirt.

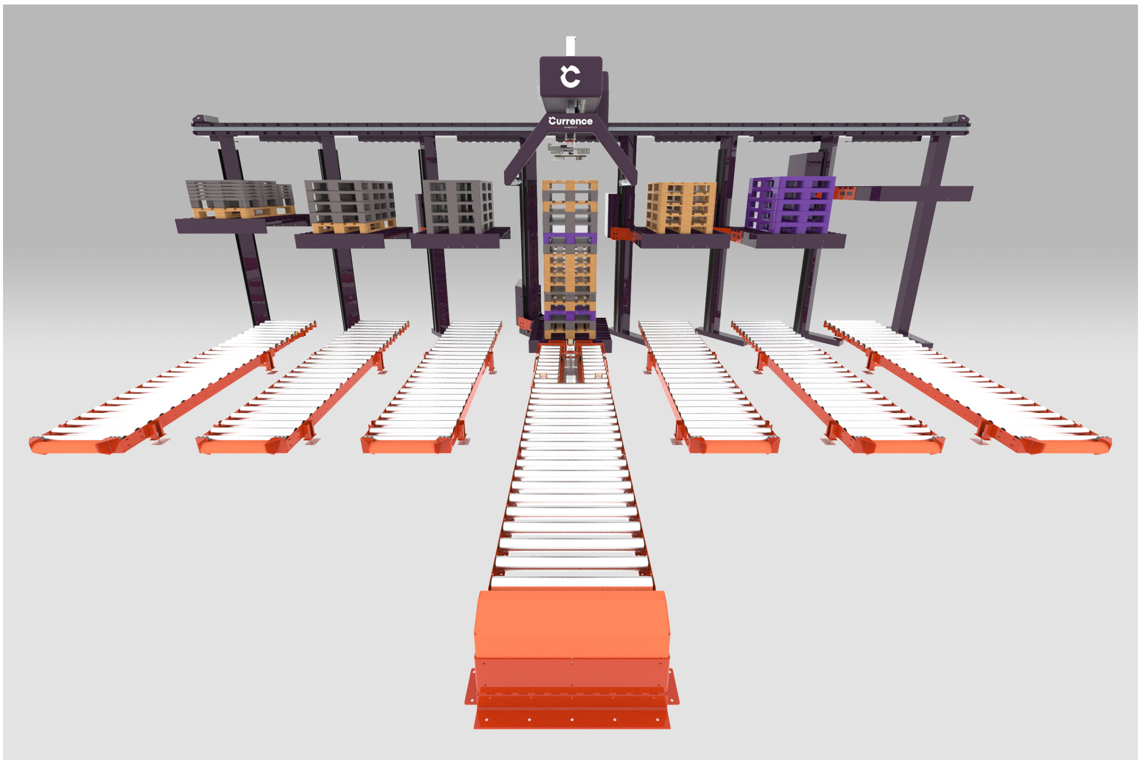


Figure 1: Mockup of Sort

The reason for implementing a new image acquisition solution is that the current camera is discontinued by the manufacturer. Another reason is restrictions using the USB protocol. There are issues with both cable length and data handling from multiple cameras. A third reason is the current camera is not an industrial camera, and therefore is lacking industrial standards in lifespan and robustness.

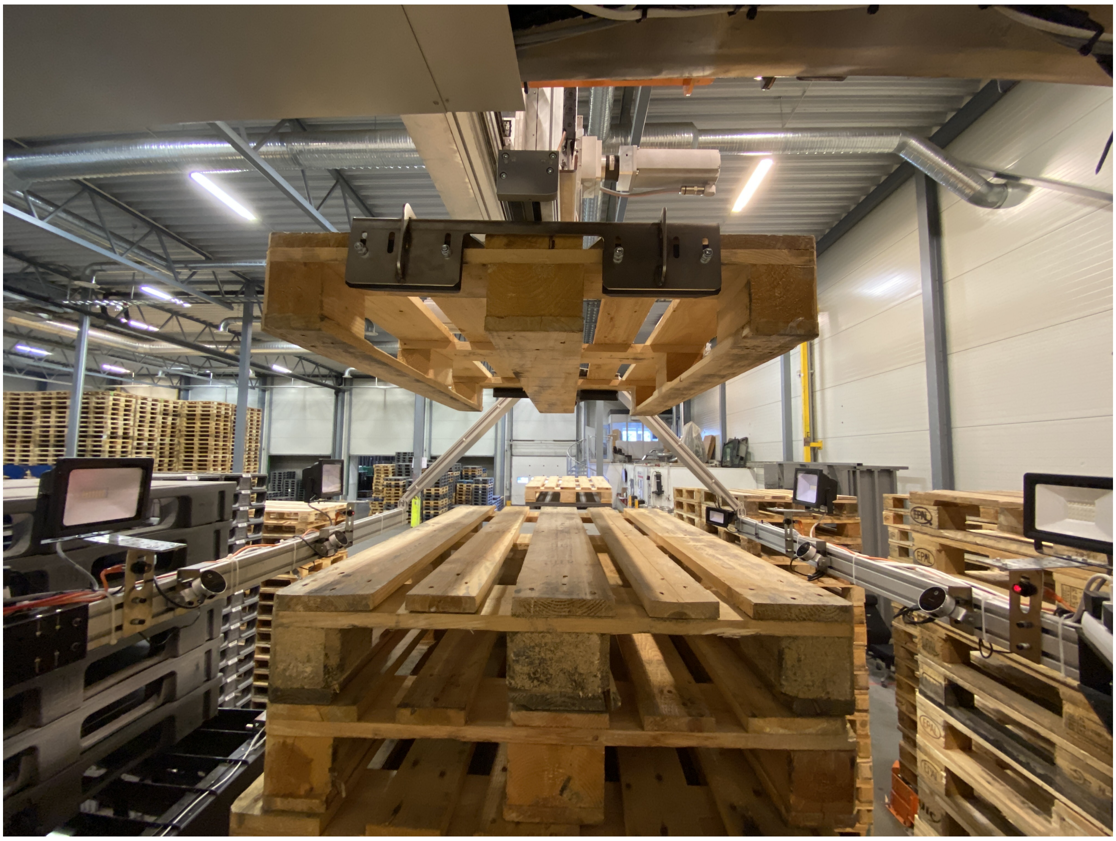


Figure 2: Sort™ in action

The camera that is most likely to replace the current ones are too expensive to use with the current solution, because of competitive pricing. Because of this, the goal of this project is to achieve the same system accuracy with fewer cameras. The price for the current camera is 520 euros [2], while the potential new camera, Zivid One+, is between 6.500 and 11.850 euros [3], respectively. The Zivid camera pricing may vary, due to negotiations between Zivid and Currence Robotics.

During the project period, the group has set out to achieve several milestones.

Milestones:

- reconstruct a 3D-structure from multiple images
- simulate possible solutions
- building prototypes based on simulations

4.2 Project requirements

- low latency
- high computational speed
- high accuracy
- must not be overly expensive
- reduce the number of cameras, without compromising the system's accuracy

4.3 Project development

The project group will brainstorm and make several sketches of different solutions. Thereafter, the sketches will be modeled in a browser-based 3D-modelling software, named Onshape. Based on the mentors' and employer's feedback, one or several solutions will be simulated for further research. The simulation will be built in Webots. Webots supports user defined robots and several different programming languages. Thereafter, a prototype will be built based on the most promising solution. Depending on the remaining time of the project, the prototype will be built either by the project group or Currence Robotics.

In order to share software between group members and the employer, the group will be using GitHub. This is a development- and version control tool. Using GitHub will also make data loss highly unlikely due to it all being stored online and in consecutive versions.

4.4 Crucial factors

One crucial factor for the project success is a good information- and feedback process, with the team at Currence Robotics and the group mentors. This is due to their experience and different fields of expertise.

In addition to this, all group members must fully commit to the project and it's agreement. Following the agreement signed on the first day of the project. If any member does not contribute as expected, the workload may increase beyond the group's capacity.

4.5 Main activities

Main activities during the project period:

- researching the current camera setup
- researching point clouds generated by LIDAR-cameras
- researching viable new cameras
- produce new image acquisition solutions
- simulate in Webots
- simulate error detection
- prototyping

The planned timeline for these activities can be seen in the attached Gantt diagram in chapter 9. The tasks within the Gantt diagram will be assigned at the start of each sprint.

↩

4.6 Project schedule

4.6.1 Description of milestones

Reconstructing a 3D-structure from multiple images:
In order to sort each pallet correctly, the software needs images from multiple angles to make a 3D point cloud. A 3D point cloud is a collection of points that creates a 3D structure. With today's point cloud accuracy it's possible to determine type of pallet, material, cracks, cleanliness and other defects. All of these parameters will decide the pallet placement, and if the pallet should be discarded.

Simulate possible solutions:
The simulators accuracy and ability to recreate a real-life scenario, is critical for prototyping. The simulator will be built in Webots, and will be the testing platform for possible solutions. Therefore, the simulated environment should match real-life conditions as closely as possible. The group has chosen Webots as their simulator software and Onshape as their 3D-modelling software because that is the same software as their employer. This allows the group to benefit from the employer's experience and file repository.

Building a prototype:
If a simulator meets the system requirements, the project group will start creating a physical model as a proof-of-concept. Thereafter, Currence Robotics can improve and implement the concept into their design.

4.6.2 Progress tracking

To monitor and document the group's progress, the group will continuously update the Gantt diagram and write daily logs. This enables the group to be structured towards a common goal, easily document methods that did not meet the requirements, and make it easier for each group member to see how the team is progressing. This also enables the group to hold each other accountable, and see where help is needed to reach set goals.

4.6.3 Evaluation

The project group will have a running dialogue with both their mentors at NTNU and Currence Robotics. All ideas the group presents are discussed and considered. The ones that are deemed unnecessary complex or too expensive will not be pursued further.

4.6.4 Decision making

All significant decisions will be decided by consulting the group mentors and Currence Robotics. Thereafter, the group will apply a democratic practice.

↩

5 Documentation

5.1 Reports, routines and technical documents

5.1.1 Documentation of finished project:

- bachelor application
- project preliminary report
- bachelor meeting reports
- progress reports
- Gantt diagram
- daily logs
- bachelor thesis

5.1.2 Routines:

- Progress meetings with mentors at the end of every 2-week sprint.
- Each member will write a daily log.
- Each group member must dedicate two hours each week to writing the bachelor thesis.
- The secretary will write reports and send out invites for every meeting.
- The Archivist will make sure the Gantt diagram is up to date, and write agenda for every meeting. In addition, take charge on writing sprint review and sprint planning in a progress report.

5.1.3 Storage

The group will use a shared locker at campus for small items, and keep expensive components at Driw or H.I. Giørtz.

6 Planned meetings and reports

The group will be using 2-week sprints. Sprints are a method of producing a set amount of work within a set amount of time. Each sprint has specified tasks and people delegated to each of them. Each sprint will be concluded with a meeting with mentors and a representative from Currence Robotics.

The project group is responsible for arranging meetings and sending out the invites to all relevant participants. Before every meeting, all participants will have the opportunity to view the group's current progress in the form of a Gantt Diagram. It will also detail the progress during the current sprint and the goal for the next.

Each sprint will end on even-numbered weeks. After the scheduled meeting, the group will write a report of what was discussed, and distribute it to all participants.

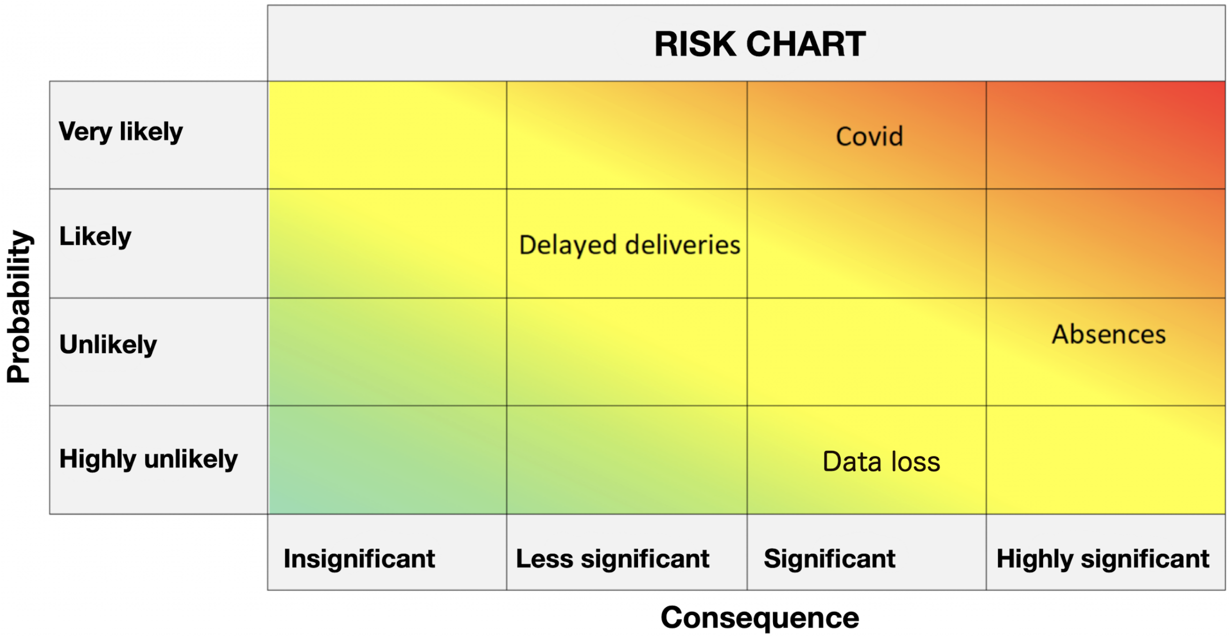
Meeting schedule:

- Meeting 1, week 2, 14th of January 09:00-10:00
- Meeting 2, week 4, 28th of January 09:00-10:00
- Meeting 3, week 6, 11th of February 09:00-10:00
- Meeting 4, week 8, 25th of February 09:00-10:00
- Meeting 5, week 10, 11th of March 09:00-10:00
- Meeting 6, week 12, 25th of March 09:00-10:00
- Meeting 7, week 14, 8th of April 09:00-10:00
- Meeting 8, week 16, 22nd of April 09:00-10:00
- Meeting 9, week 18, 6th of May 09:00-10:00

↩

7 Planned non-conformance handling

There remains a degree of unpredictability in the Covid pandemic, that most likely will last throughout the project period. Therefore, it's necessary to have procedures in place, in case of another national or local lockdown. One obstacle might be a lockdown that will limit the group's workspace, and will force the group to work remotely. This might limit the group's ability to prototype. Another obstacle will be if one or several group members becomes absent due to quarantine or illness. In addition there is a constant evaluation of parts needed, considering the high chance of a long delivery-time.



RISK CHART

Probability	Insignificant	Less significant	Significant	Highly significant
Very likely			Covid	
Likely		Delayed deliveries		
Unlikely				Absences
Highly unlikely			Data loss	

Consequence

Figure 3: Non-conformance events, probability and consequence.

7.1 Covid-19

The group has several procedures to prepare and prevent damage on the project progress. One procedure is a well-established file-sharing platform for files and daily video meetings. Another procedure is the groups scheduling in a Gantt diagram and daily logs to keep all members updated on the progress, and when tasks are meant to be completed. Should one of the members feel too ill to work, the tasks are easily re-distributed to a another member.

The project files are shared over Microsoft Teams, Overleaf, while all software are hosted in GitHub. With an Overleaf subscription the group gets access to version control on all shared documents. GitHub also offers version control, for the software.

Each member has received the booster dose against Covid-19, to lessen the chance of infection and serious illness.

7.2 Absence

If one of the group members is absent for an extended period of time or indefinitely, the workload for the remaining members will be shifted. This might reduce the amount of total work produced by the group. Meaning that some of the initial goals are not met. In addition, having one less group member will mean that ideas are not as heavily discussed, possibly resulting in a lesser result.

To minimize the impact of this event, the group is using a Gantt diagram. All remaining tasks are listed in the order of which they need to be completed, and a preliminary timeline for each of them. The remaining group members can easily view and claim the uncompleted tasks. In addition, proper log keeping will allow for easier transfer of tasks, as there is a paper trail of what exactly has been completed.

7.3 Slow deliveries and missing components

In case parts are hard to obtain or get delayed the project might be held back, and can create challenges with prototyping. Therefore it's important to plan in detail, and constantly evaluate and identify which parts to order. To prevent set backs, the group will utilize parts that either are already accessible, or can be obtained locally. Another solution would be to buy alternative parts, and perform mechanical adjustments manually. Without a prototype, the group will lose out on data confirming function, or highlighting flaws in the design.

7.4 Data loss

If files or other data were to be lost, the group risks having to redo several weeks of progress. The CAD-files and simulators are all time consuming to produce.
In order to avoid loss of progress due to losing key files, the group will be using cloud storage with version control wherever possible. This includes using Overleaf for the group's PDFs, GitHub for software, Onshape for 3D-modeling and Microsoft Teams for all remaining files.

8 Equipment & implementation needs

8.1 Tools and hardware

- Currence Robotics will provide the group with needed tools and equipment if it's already in their storage. If the group requires further equipment, the employer will evaluate the price and importance. Based on this decision the employer will make the order.
- The group will utilize Intel RealSense™ 515 for this bachelor thesis, and it's the camera which is currently used on the Sort™ robot located on H.I.G. New cameras will not be ordered due to it's high price, and similarity. If the concept is proven used the Intel cameras, it will be easily implemented with alternative new cameras such as Zivid One+.

(a) Intel RealSense™ L515

(b) Zivid One+

Figure 4: LIDAR cameras

8.2 Software and license

- Currence Robotics will provide the group with necessary CAD files of Sort™.
- The employer wishes the group to update their progress on Slack, and this might require a license. Currently the group uses a 90 day free trial on the Slack application.
- For the project simulator the group will be using Webots which is an open source platform. Webots supports several programming languages, and the group will be using Python.

References

[1] C. Robotics, “Who we are.”

[2] Intel, “Intel realsense l515.”

[3] Zivid, “Zivid one+.”

9 Appendix

Project zip folder:

- Documents
  - Gantt chart with daily logs

NTNU
Kunnskap for en bedre verden