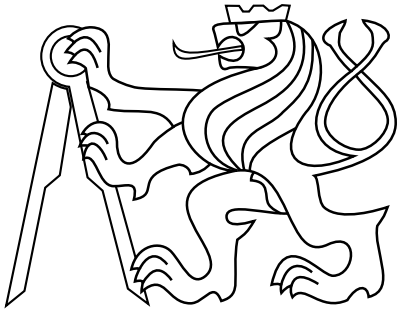
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics



Estimating the Pose of a Pallet with Bricks Using Sensors of a Pallet Truck

Bachelor's Thesis

Richard Randák

Prague, May 2024

Study programme: Cybernetics and Robotics

Supervisor: Ing. Vladimír Smutný, Ph.D.

Acknowledgements

Firstly, I would like to express my deepest gratitude to Ing. Vladimír Smutný, Ph.D., for his patience and invaluable guidance throughout this project. His support has been instrumental in the completion of this thesis.

I would also like to thank everyone from the CIIRC Robot perception group that played a part in the creation of this thesis. Namely to Ondřej Holešovský MSc., Ing. Miroslav Uller. and Ing. Libor Wagner.

To the author of a concurrently written sister thesis Adam Basl I wish a successful graduation and all the best in his future endeavours.

Lastly, I wish to express thanks to my friends and family for supporting me over the years.

And finally to the readers of this thesis, thank you for your curiosity and time.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Randák Richard** Personal ID number: **507379**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Estimating the Pose of a Pallet with Bricks Using Sensors of a Pallet Truck

Bachelor’s thesis title in Czech:

Odhadování polohy palety s cihlami ze senzorů paletového vozíku

Guidelines:

1. Get familiar with the problem of automatic loading and unloading of the pallet truck specified by the customer.
2. Get familiar with the methods of pose estimation using available sensors, used algorithms, and Robot Operating System 2 (ROS).
3. Participate in the sensors selection for a given task.
4. Propose and implement algorithms for pose estimation in ROS.
5. Test implemented algorithms in simulation and using real data.
6. Evaluate test results and make conclusions.

Bibliography / sources:

[1] Lynch, Liam, et al. "Automated ground vehicle (agv) and sensor technologies-a review." 2018 12th International Conference on Sensing Technology (ICST). IEEE, 2018.
[2] Li, Y.; Wang, D.; Li, Q.; Cheng, G.; Li, Z.; Li, P. Advanced 3D Navigation System for AGV in Complex Smart Factory Environments. Electronics 2024, 13, 130. https://doi.org/10.3390/electronics13010130
[3] Roberts, Jonathan & Corke, Peter. (2000). Obstacle Detection for a Mining Vehicle using a 2D Laser.
[4] G. Antonelli, F. Caccavale, F. Grossi and A. Marino, "Simultaneous calibration of odometry and camera for a differential drive mobile robot," 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 2010, pp. 5417-5422, doi: 10.1109/ROBOT.2010.5509954.

Name and workplace of bachelor’s thesis supervisor:

Ing. Vladimír Smutný, Ph.D. Robotic Perception CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **12.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Vladimír Smutný, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Declaration

I declare that presented work was developed independently, and that I have listed all sources of information used within, in accordance with the Methodical instructions for observing ethical principles in preparation of university theses.

I also declare that artificial intelligence was used for grammar correction and text reformulation (ChatGPT, Grammarly). Artificial intelligence was also utilised when writing the code of this thesis (ChatGPT). All artifical inteligences were used in accordance with the Guidelines for the use of Artificial Intelligence at CTU.
Prague 24. 5. 2024

Prague 24.05.2024 .....

Abstract

This thesis explores the design of the vision system for a visual servoing setup used to load pallets of bricks with a pallet truck in outdoor construction site conditions. We employed the Luxonis OAK-D Pro stereo camera to gather measurement data. A customised YOLOv5 neural network was trained to detect pallets of bricks in colour images. Using these detections along with depth data, we estimated the poses of the pallets. We've utilised RANSAC algorithm to process the generated point cloud, together with other techniques to improve the estimation accuracy. The system was developed using ROS to simplify the integration with other components of the larger project. Additionally, we examined the behaviour of the Luxonis OAK-D Pro stereo camera as part of this research.

Keywords Pose estimation, Visual servoing, Stereo camera, Pallet truck, YOLO, RANSAC, ROS, outdoors

Abstrakt

Tato práce zkoumá návrh vizuálního systému pro vizuální servořízení, který se používá k nakládání palet cihel s paletovým vozíkem v podmínkách venkovního stavebního prostředí. K získávání měřicích dat jsme použili stereo kameru Luxonis OAK-D Pro. Přizpůsobili jsme neuronovou síť YOLOv5 pro detekci palet cihel v barevných obrazech. Pomocí těchto detekcí spolu s hloubkovými daty jsme odhadli polohy palet. Na generovaný mrak bodů jsme použili algoritmus RANSAC spolu s dalšími metodami pro zlepšení přesnosti odhadu. Systém byl vyvinut s využitím ROS pro zjednodušení integrace s dalšími komponenty většího projektu. Krom toho jsme v rámci tohoto výzkumu zkoumali chování stereo kamery Luxonis OAK-D Pro.

Klíčová slova Odhad polohy, Vizuální servořízení, Stereo kamera, Paletový vozík, YOLO, RANSAC, ROS, venkovní prostředí

Abbreviations

FOV Field of View

IMU Inertial Measurement Unit

lidar Light Detection and Ranging

ROS Robot Operating System

AGV autonomous ground vehicle

YOLO You Only Look Once: Unified, Real-Time Object Detection [0]

ROI Region of interest

RANSAC Random sample consensus

QR code quick-response code

PC point cloud

NN Neural network

ABLR Automatic brick laying robot

SE3 Special Euclidean group in three dimensions

IMU Inertial measurement unit

fps frames per second

id identifier

GUI graphical user interface

λ inliers per meter

FOV Field of view

$\mathbf{HD_o}$ high detail version of YOLO recorded on Luxonis OAK-D Pro

$\mathbf{HD_{pc}}$ high detail version of YOLO made on our PC after the recordings were made

$\mathbf{LD_o}$ low detail version of YOLO recorded on Luxonis OAK-D Pro

$\mathbf{LD_{pc}}$ low detail version of YOLO made on our PC after the recordings were made

OLD original version of You Only Look Once: Unified, Real-Time Object Detection [0] (YOLO) that existed before we expanded the training dataset

stereo camera Luxonis OAK-D Pro

Contents

1 Introduction

The concept of an Automatic brick laying robot has already existed for some time. However, recent improvements to the robot and its deployment in the field demonstrated a need for a system that would automatically supply the Automatic brick laying robot with building materials.

Existing automated pallet trucks do not meet the design specifications required for outdoor operations under varying light conditions and the unpredictability of construction sites. Given these limitations, we decided to design an in-house system capable of full customisation to meet our specific needs.

Given the complexity of creating a full-fledged autonomous ground vehicle, we initially focused on solving the loading and unloading processes of pallets. Human operators will handle the movement of the pallet truckbetween brick storage and the Automatic brick laying robot, as depicted in Figure 1.1 and Figure 1.2, respectively.

Figure 1.1: Stored bricks.

Figure 1.2: Automatic brick-laying robot [43]

In this thesis, we explore and implement various techniques for accurately determining the relative pose (position and orientation) of the pallet of bricks and pallet truck, enabling precise loading operations.

2 State of the art

2.1 Already existing designs

There are many companies already offering products of autonomous forklifts, palette jacks and forkless palette carriers. These companies offer autonomous ground vehicles (AGVs) of their own design that promise to proficiently handle standard wooden pallets. The installation of these systems is usually also offered [53]. Their products are designed for use in warehouses as tool for reducing the required manpower, increasing speed and reducing the accident rate [17]. Another potential benefit of automation is that it may decrease the damage to the pallet caused by improper handling [20]. As these products are made by for-profit companies the designs are not very easily accessible. But the existence of these products on the commercial market may make us hopeful that a solution to our problem exists. On the other hand, their requirements are quite different than those of ours. A well-optimized warehouse is very different from a construction site outdoors.

As the task of developing AGVs is a common one, there is much literature on the subject and its different components, that shall be referenced further in the text.

The next part of the text shall be an overview of the basic task of creating a functioning sensor suite for our robot. We should learn some theory behind the basic principles and methods we will later utilise.

2.2 Detection of points of interest

As a point of interest, we shall classify the position and volume encompassed by any object physical or not we would like to know the location of. It may be a pallet loaded with bricks, a wall, or a hole in the ground i.e. our targets and potential obstacles. In the future would also like to know the location of our unloading stations.

Sensors

For the purposes of estimating location in space AGVs we can utilise many different sensors and methods. In the upcoming section we shall compare some of the sensors we took into consideration when choosing the sensors for our project.

Lidars

A thorough review of 3D TOF lidars can be seen in [18].

Lidars have the advantages of high accuracy and general widespread usage in industry. Its main issue lies in overcoming the powerful weather dependent sunlight outdoors. Airborne particles might also prove to be an issue. Rain, snow, dust and fog may decrease the accuracy of measurement [19]. Range of the Light Detection and Rangings (lidars) can differ from five metres on a poorly reflecting surface with [26] to 245 metres [24]. Lidars also have issues working well with poorly reflecting surfaces, even as common as water puddles [29]. The different internal structure of lidars influences their characteristics. They make a trade-off between size, lifespan, capabilities and price. The price variation can easily range from some $300 to over $5 000 [38], [41].

Lidars can also be divided into planar 2D lidars and camera-like 3D sensors. While 2D lidars measure ranges in one plane only, 3D sensors measure in multiple planes at once. 3D lidars often reduce the horizontal field of view, so that it is no longer full 360°.

Cameras

Cameras are a widespread, accurate and affordable form of sensor. The inner workings of the sensors are well described in [11],[27].

They gather a large amount of data points and require significant resources to process correctly. They also have issues with focus and setting a correct light exposure. Creating settings for a camera to work well at all levels of ambient light is no simple task. Thankfully most devices automatically adjust these settings during operation. Cameras also create another issue, they do not capture the distance of the object from the camera. Additional methods must thus be applied.

Stereo cameras

Stereo cameras work on the principle of binocular vision. Usually, by combining two images captured at the same time in cameras at a known range from each other. The cameras try to pair their detections onto each other, computing which pixels correspond to each other. The distance between the pixels is known as disparity. Stereo cameras prefer for the seen points to be distinctly different to decrease any ambiguity in pairing the pixels [37]. To solve this issue some models also include an active component that helps with detection both under low light conditions and on untextured surfaces [23], [47]. The problem of correctly calculating disparities will be further discussed in Section 5.3.

An additional difficulty may represent transparent objects [30]. Stereo cameras also have a limited range and can have issues with working under direct sunlight [6][21]. In summary, cameras represent an affordable and potentially well-performing option.

Honorable mentions

- Global positioning system

GPS provides an accuracy of about 5 m and would require us to create a dynamic map of our environment. The accuracy is also reduced by proximity to buildings [42]. And we would still somehow need to measure the positions of our targets.

- Path planning

In unchanging environments AGVs are commonly programmed to follow predefined paths. These methods need to be set up in advance [51] which makes them unusable in our dynamic environment.

- radar

We could utilize larger wavelengths for our measurements. "Weather phenomena such as fog, rain, snow, and hail impede visual perception" [35]. This would most certainly also include dust, which could also be expected on a construction site. We have decided that this would not be a common enough problem and thus chose more accurate sensors. Radar has the potential for a larger operating range, but we wouldn't utilise it anyway.

Evaluating Measured Data

When it comes to evaluating the measured data there are many possible approaches. A key requirement is for the entire vision module of the project to work in real-time. And most importantly we should be able to detect and estimate the pose of our bricks pallet.

The first thing to consider is whether or not we know the dimensions and texture of our points of interest. If these properties are known beforehand we can measure whether or not and where is the object located in space. Otherwise, we can really only pronounce the existence of some object at some point in space, not much else.

Rigid Object 6DOF Pose Estimation

Many models have recently been created that use a color image or even a point cloud to estimate the poses of objects [2]. These are usually pre-trained models that compare a database of known objects to the measured data [7], [14]. Recent advances have allowed for some of these models to potentially even detect objects in real time, though they are still prohibitively computationally demanding. Another obstacle is the need to create a 3D model of sufficient quality. Keeping our target in a condition corresponding closely to the provided mesh may be difficult.

Combined Methods

We could also combine colour and depth detections. Use the colour image to find the rough position in space of the object and then fit the shape of our object onto the depth measurements.

Many methods can be utilised to detect objects in an image. From simple algorithms like KNN to trained Neural network (NN) [5]. NNs require training data, these could be annotated manually of by using a segment anything model [9], [15].

2.3 Calibration

All detection methods give us the ability to measure the transformation between the sensor and the measured object. This alone does not allow us to navigate the robot successfully. We also need to measure the transformation between the sensors and the robot base. Some detection methods also require us to measure the intrinsic properties of the sensors themselves, these are sometimes provided by the manufacturer. Calibration methods usually represent trade off between the ease of repeatability, the need for additional hardware, accuracy and simplicity of implementation.

One may try to rigidly mount the sensors, never move them and read the transformation by measuring the distances. This would not be a very accurate method, that does not allow for easy modification or reliability, but it would be very easy to implement. A common option makes the use of a calibration board covered with markers, which can also make recalibrating cumbersome.

An common issue of target based methods is the fact that they are performed offline and thus they cannot correct calibration errors that occur during operation. Furthermore, these methods become impractical in mass-produced systems, especially if manufacturing tolerances result in variations of the calibration parameters among systems to affect further processing

and data fusion. In such cases, these methods would require individual calibration of each system, increasing the manufacturing cost [28].

As seen in Section 2.2.1 the three sensors the most applicable to our task are the camera, 3D and 2D lidar. As discussed in Section 2.2.2 we would prefer to use a depth sensor in combination with a camera's picture, thus the calibration of depth sensors shall be done in conjunction with a camera.

Camera Calibration

The camera calibration task is a common one. A camera is considered calibrated once its intrinsic properties are known, another related task is measuring the transformation between the camera and the robot base.

Intrinsic Parameters

Intrinsic camera calibration aims to find the distortion in the camera image and the conversion between pixels and real-world units [3]. Intrinsic measurements have the property, that they should not change much between calibrations. Camera calibration methods can be grouped by their use of targets and whether or not they allow us to determine the transformation between the camera and the robot base.

Targetless methods tend to rely on feature recognition of an unknown object. Thus an object of favourable qualities is required. A relatively large amount of images fulfilling difficult constraints also need to be captured including rolling the camera [33]. Targetless methods can also be motion-based [36]. These require the use of precise time matching and measurement of motion.

Methods that specific calibration targets are much more common. The target models are manufactured with precisely known dimensions and easily recognisable features. The corners of a chessboard or circles are shapes suitable for measurement and detection by cameras. Markers can also simplify the pixel model pairing process [31], [4].

Extrinsic Parameters

The extrinsic camera properties can be calibrated using eye-in-hand calibration. It works on the principle of capturing many images with many different poses. This would allow us to calculate the transformation between the reference frame of our camera and our pallet truck's origin [13]. Another approach would be to fix the target onto the robot itself. This might allow for fast calibration of the camera, but creating a consistent and durable placement of a target might prove to be difficult.

A common approach is also to manually measure the transformation. Creating an automatic method requires a lot of upfront cost in time and this investment may not always return.

2.4 Project History

The work discussed in this thesis is part of a larger project that began well before this thesis' contributions started. In large projects like this, many people are involved, which complicates the attribution of individual contributions. The next section provides context to the work discussed here.

Figure 2.1: A row of pallets

Initial Developments

Ayane Hokari and Ryohei Tozaki, exchange students during the winter semester of 2023-2024, undertook the initial phase of our project. Their task focused on moving a pallet truckusing a planar lidar and camera combination. Their work did not overlap with the timeframe of this thesis. They primarily focused on obtaining real-life measurements and subsequently used this data to develop a simulation, albeit without the ability to physically move the pallet truck. It's worth noting that their objective centred around picking up a single lone standing pallet rather than one integrated into a row of pallets such as one seen in Figure 2.1.

Their implementation followed a structured approach for pallet pose estimation, which involved:

- Pairing the detected lidar and camera data by their timestamps.
- Locating the pallet in the image using YOLO NN.
- Running Random sample consensus (RANSAC) through the lidar data.
- Fitting a pallet's shape onto the detected lines.

Additionally, they implemented a tracking mechanism utilising the poses of the fitted pallets. While the tracker was not perfect, it was not their priority given the use case.

The used solution seemed promising and we decided to continue with this approach after their departure.

Transition to Current Work

In its final form, the lidar solution had one major issue: it did not reliably detect pallets standing in a row. There also were limitations with using lidar outdoors. Thus later our design specifications changed and we chose to explore what other sensors we could use.

We have also decided to split the task into two distinct parts, a vision module described in this thesis and a control module described in the sister thesis [1].

After examining existing sensors, stereo cameras emerged as a promising choice. Luxonis OAK-D Pro (stereo camera) seemed like a good option as it did not require an active projector.

Research into the inner workings of the OAK-D Pro, its communication with Robot Operating System (ROS), and performance under direct sunlight was conducted by Ondřej Holešovský, MSc. He also implemented a version of the lidar solution that utilised the rgbd sensor. Finally he managed to make our neural networks utilise a computing chip on the stereo camera.

Thus, we had a basic version of a pallet pose estimator. Which, however, did not work entirely reliably. Furthermore, we lacked a tracker, and our You Only Look Once: Unified, Real-Time Object Detection [0] (YOLO) model did not reliably detect our pallets. Moreover, we had no communication system, no real outdoor data, and no good means to evaluate the data. Such was the state of the project at the beginning of our study.

In our project, we also worked with a physical pallet truck. Ing. Libor Wagner and KM Robotics s.r.o. worked together on making the physical pallet truckoperational and controllable. In the timeframe of this thesis we were not able to incorporate our systems together.

Another significant contributor was Ing. Miroslav Uller. His primary role was to assist with the creation of both the simulation and the Robot Operating System (ROS) communication. Uniting various components in ROS would have been most challenging without his assistance. ROS integration also required code cleanup which he also played a large role in. The vision module of the project while requiring complete integration in ROS did not need to understand the entire structure of the ROS communication. As such it will not be too deeply discussed. A more detailed explanation of the ROS communication can be found in the sister thesis [1]. Finally, a simulation was created. The simulation was mostly used in the control module.

Furthermore, our work planning involved weekly meetings within our research group. These sessions included group discussions and interpretation of measured data, enabling the team to stay abreast of ongoing developments.

Summary of My Role

To summarise, my contributions mostly included improving pallet detection and pose estimation, implementing a tracker, interpreting results, and creating documentation.

3 Project Specifications

3.1 Project Requirements

The task specifications dictated the creation of a pallet truck capable of automatically loading a pallet from storage once placed in its vicinity by a human operator. The same vehicle should be able to unload the pallet of bricks near the Automatic brick laying robot (ABLR). In the unloading task, the sensors onboard the ABLR could also be utilised.

Both loading and unloading tasks essentially fall under the term "visual servoing" as outlined in the following steps:

- Make continuous measurements using cameras.
- Control the onboard motors to reach the desired point in space.

The goal of this thesis is to address the vision module of our problem, while the control aspect is covered in the sister thesis [1]. The primary objective of the vision module is to make measurements quickly, accurately, and cost-effectively[1].

Here we will also note that both loading and unloading tasks are in essence the same task. When loading a pallet we needed to use the sensors on the pallet truckto estimate the pallet's location. When unloading we want to estimate the pose of our pallet truckin relation to the ABLR, but the pallet truckalready has a pallet of bricks loaded. We can thus use the same algorithms both for loading and unloading of the pallets as all we care about is the transformation between sensors and the pallets of bricks. During this thesis, we shall continue with describing only the loading task, but in actuality, the unloading task will also benefit from our work.

3.2 Bricks

Our target was a pallet of bricks, as seen in Figure 3.1. These pallets would be located in storage, closely packed side by side, one pallet high. (The data we gathered later would include stacks of multiple pallets on top of each other. In the final solution, we would understandably not be loading pallets obstructed in such a way.) A drawing of the dimensions of the wooden pallets can be seen in Figure 3.2.

Specially designed bricks were stacked on the pallet and the entire package was covered in protective film. The pallet of bricks had the shape of a box with side lengths of 1000 and 1200 mm and a height of 1390 mm. The shorter side of 1000 mm was the loading side. A quick-response code (QR code) code was placed on the short side of the pallet. These QR codes contained information that would help an operator choose the correct pallet. Such a system would also help manage storage inventory, but these concerns are outside the scope of this thesis.

[1] Considering price and power consumption.

Figure 3.1: A pallet of bricks (here the long side of a pallet is pictured, we will not be loading the pallet from this side)

3.3 Used Pallet Truck Prototype

Pallet Truck

It was decided that we would use and modify a pallet truck made by the company DW Forklift. The original pallet truck is depicted in Figure 3.3. Additional details from the manufacturer can be found at [40].

This is a cropped drawing made originally by Ryohei Tozaki.

Modifications were made to the pallet truck to accommodate our sensors. A mounting board was added atop the pallet truck to facilitate this purpose. It was designed such that mounting and dismounting the board did not affect much the position of the sensors relative to the rest of the pallet truck. The mounting board is illustrated in Figure 3.4.

Additional motors were also integrated to control the back wheel of the pallet truck. However, the entire system could not be made operational in the timeframe of this thesis.

Detailed internal mechanical workings of the pallet truck are not within the scope of this thesis. For a more in-depth discussion on the mechanical workings of the pallet truck, refer to the sister thesis [1].

The complete pallet truck with all modifications is depicted in Figure 3.5. A drawing was also created and can be seen in Figure 3.6.

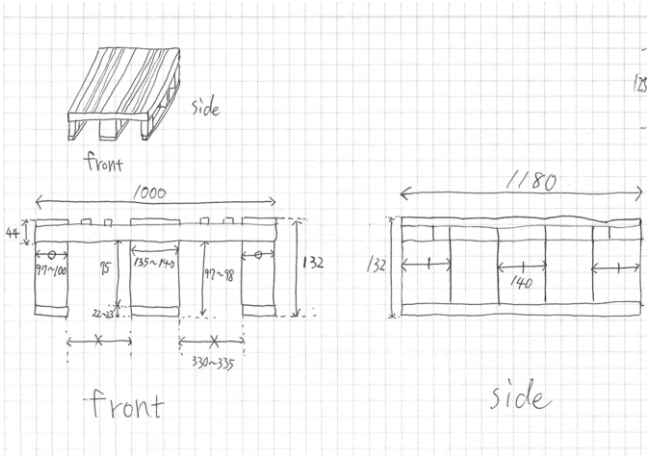


Figure 3.2: A drawing with the dimensions of our wooden pallets[2].

Figure 3.3: A picture of original pallet truck from [40].

Battery and Power Management

Our prototype utilised several different batteries, each serving specific functions. The electric pallet truck had its own battery, powering the motors and internal hydraulics. An additional battery was added to increase onboard power capacity and simplify integration with other components. The additional battery supplied power to a computer and our sensors, extending their operational duration.

The onboard battery, seen in Figure 3.7, had a capacity of 0.96 kWh.

As we were not able to integrate our modifications to the pallet truck operational by the deadline, we can only speculate on the future performance. The pallet truck is expected to move less and at slower speeds, potentially conserving charge. However, added modifications increased weight, potentially raising power consumption.

The additional battery Figure 3.8, we added had capacity of 0.108 kWh [22]. We will add that in our task, the vision module would not require continuous operation, allowing for power-saving strategies.

Sensors

An RGBD stereo camera, Luxonis OAK-D Pro, was chosen for this project, where the documentation can be seen in [47]. The camera was mounted on top of the board in its own specially designed mounting, as can be seen in Figure 3.9. It must also be noted that the camera in this image is mounted upside down, and this discrepancy had to be fixed in code.

The stereo camera could capture colour images at a resolution of up to 12MP[3]. However, the resolutions primarily utilized were 720p[4] and 2K[5]. The choice was driven by its compatibility with the native resolution of the stereo camera, as both are a multiple of 720p[4]. Field of view (FOV) of the cameras can be seen in Table 3.1.

The stereo camera automatically calculated the depths in the frames. The depths were

[3] 4000×3000 px
[4] 1280×720 px
[5] 2560×1440 px

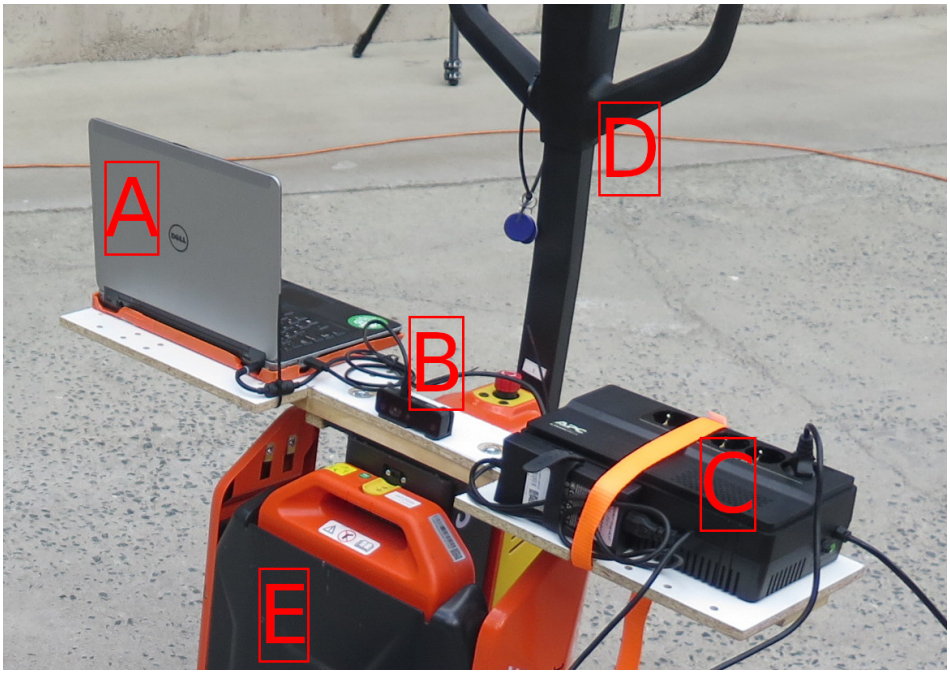


Figure 3.4: The pallet truck mounting board A - notebook, B - stereo camera stereo camera, C - Sensor battery, D - Pallet truck handle, E - Pallet truck battery

Figure 3.5: Pallet truck with all of our modifications.

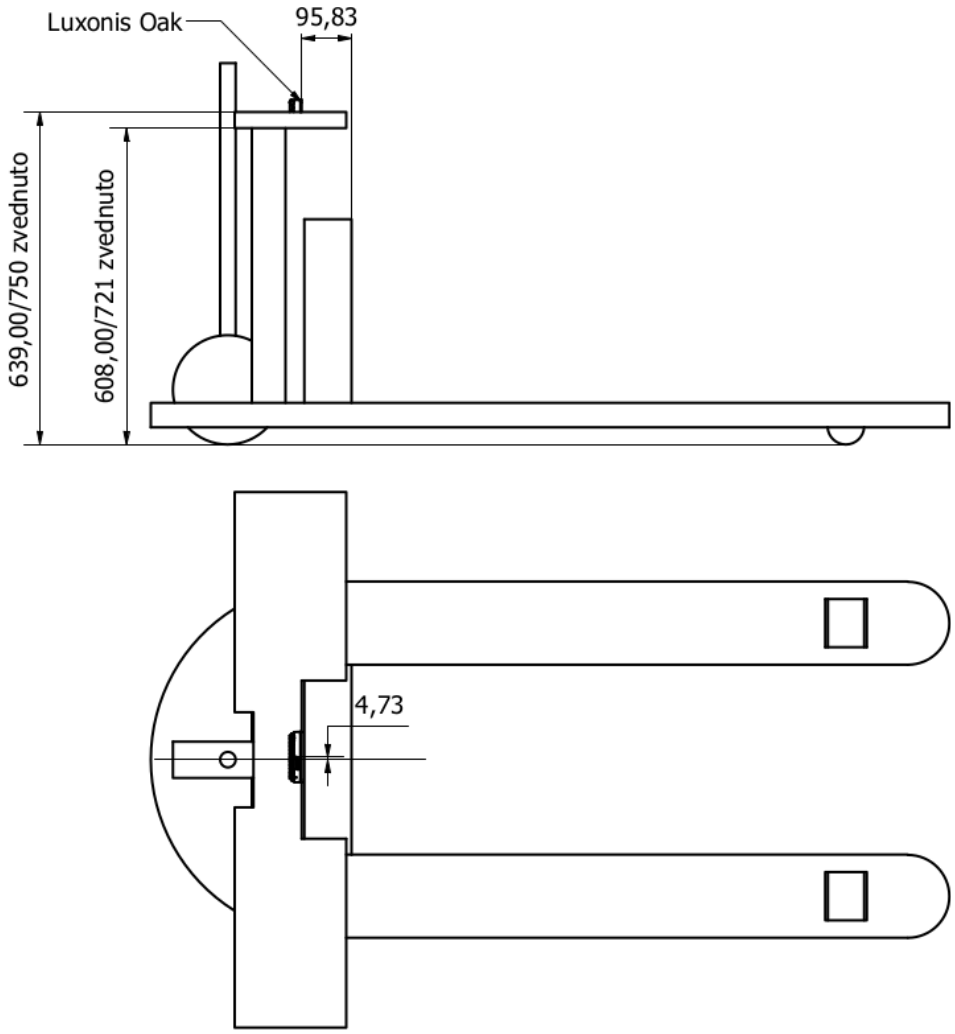


Figure 3.6: A drawing of the pallet truck and its dimensions.

Figure 3.7: the onboard battery

Figure 3.8: the sensor battery

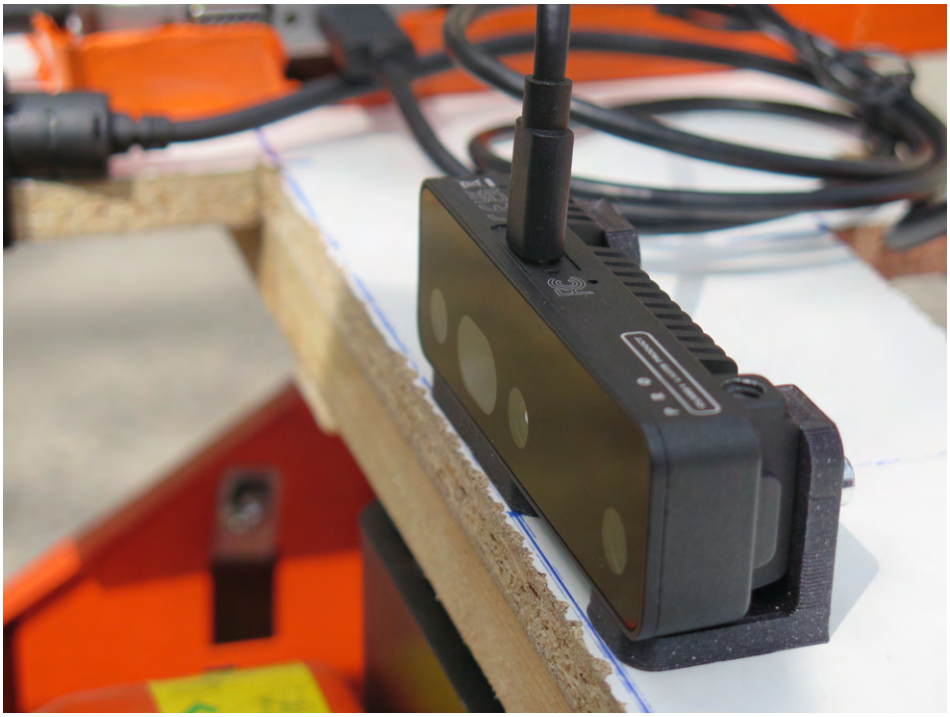


Figure 3.9: stereo camera in its mount

Luxonis OAK-D Pro FOVs	colour camera	stereo camera
Diagonal FOV	81°	89°
Horizontal FOV	69°	80°
Vertical FOV	55°	55°

Table 3.1: Table describing the FOVs of the stereo camera

transformed into the reference frame of the colour camera, such that each pixel had its corresponding depth in the z direction.

The camera offered various settings in terms of what messages to send, including sending images from the left and right cameras separately, sending image disparities instead of their depth, multiple subpixel settings, and adjustable frames per second (fps). All of the options can be found at[6][46].

The stereo camera also featured an onboard additional computational unit, allowing the stereo camera to run a NN in the camera itself. The specifications of the Robotics Vision Core 2 can be found at [12]. Additionally, the stereo camera included an Inertial measurement unit (IMU).

Significant problems were later encountered with how the stereo camera pairs the detected pixels, which will be further described in Section 5.3.

[6]In actuality, all of the settings were gathered by running the command *ros2 run rqt_reconfigure rqt_reconfigure*

Figure 3.10: Pallet truck computer in its mount on the pallet truck.

- **Computers**

To record our data we mounted a notebook onto the mounting board. This allowed us to easily control the onboard sensors.

Pallet Truck Computer

The computed mounted on top of the pallet truck can be seen in Figure 3.10. Full specifications of the notebook mounted on the pallet truck can be found at [32].

Short summary:

- OS: Ubuntu 22.04.04.LTS,
- CPU: Intel® Core™ i7-4610M CPU @ 3.00GHz × 4,
- GPU: Mesa Intel® HD Graphics 4600 (hsw gt2),
- MEMORY: DDR3L 16GiB,
- STORAGE: SAMSUNG SSD SM841N mSATA 256GB SED .

The capacity of the notebooks battery was quite limited and as such we did not list it in Section 3.3.2.

Testing computer

This was the computer used for programming and testing of the vision module of our task. Full specifications can be found at [16].

Short summary:

- OS: Ubuntu 22.04.04.LTS,
- CPU: 13th Gen Intel® Core™ i7-13700HX × 24,
- GPU: NVIDIA GeForce RTX 4070,
- MEMORY: 32 GiB DDR5,
- STORAGE: 1 TB SSD M.2 PCIe 4.0 NVMe.

3.4 System Architecture

The components of our prototype pallet truck mostly communicate using Robot Operating System (ROS). A detailed description of internal ROS communication between the different components can be seen in the sister thesis [1] and in Section 5.1.5.

4 Proposed Solution

4.1 System Goal

We have decided to continue in the direction of our predecessors, as seen in Section 2.4, with the difference of us using a combined RGBD stereo camera instead of a camera and lidar combination. Our goal was, naturally, to make estimations of the pallets' position as precisely and quickly as possible.

4.2 System Overview

The majority of the code could be implemented entirely as an usual Python script. With the constraint that it must have been created in such a way that ROS could easily communicate with it.

The inputs to the main function were the taken colour image, an already calculated depth image, and the detected Regions of interest (ROIs), all received from the stereo camera. The received ROIs solved the problem of detecting our pallets. Using the depth data we estimated the poses of the detected pallets. We also needed to create a system that would have allowed us to track one specific pallet for us to load. And finally, we needed a way for the user to input the desired pallet for loading.

Once implemented, we needed to encompass the entirety of the code into ROS.

4.3 Implementation Overview

This section shall give an overview of the required parts to implement in our solution without going into too much detail. The required parts shall be listed in the same order they traveled through our system. A more detailed implementation can be seen in Section 5.1.

Pallet Detection

We already had a system that detected pallets very quickly with limited resources using just a colour image – a trained YOLO neural network. Since our network still had some limitations, we had to expand its training dataset and retrain it. An additional requirement we also needed to address was making the network use the computation unit onboard the stereo camera.

Pallet Pose Estimation

As described in Section 2.4.2, the basic structure of this part had been created before we began the work on this thesis. But a lot of work still needed to be done. Mostly made up of bug fixing and various other improvements to increase reliability.

The basic structure of the pallet pose estimation was the following:

- convert the measured depth data into a PC,
- rotate the PC by the extrinsic rotation matrix,
- fit planes onto the detected PCs,
- check if the fitted planes are realistic,

- fit a pallet onto the detected planes,
- save the detected pose.

Most of these steps were relatively simple to achieve, their implementation shall be discussed in Section 5.1.2.

- **Pallet Tracking**

We had to keep a memory of the previous detections. Using this memory, we were able to establish a link between the detections made in the current and previous iterations. The tracker had to be able to cope with a changing amount of pallets in time. That is to handle new detections, missing frames and losing pallets for being out of frame. And lastly, in the future our tracker would granted access to the odometry measurements made in between our iterations.

Most certainly, we had to decide what features to track. In essence, two options presented themselves.

ROI Tracking

We could have tried to use the ROIs detected by the YOLO. This would have also allowed us to filter the detected pallets before we made any additional calculations onto them. However, the ROIs could move unpredictably in the image. Quick rotations of the pallet truck made large changes in the movement of the ROIs in the image. We anticipated that it would be rather difficult to incorporate the odometry data to predict the movement of the ROIs.

Pose Tracking

Another method for us to use was to track the detected pallet poses themselves. Thus we estimated the poses of all of our detected pallets and utilised these to track them.

This method was slower than the ROI tracking because we needed to estimate the poses for all of the pallets. However, the movement was easier to work with. And we predicted that incorporating the odometry data would be much easier on the pallet poses.

- **Pose Refinement**

Despite our best efforts, the pose estimations we created still had some errors. We can classify these into multiple classes.

- noise created by RANSAC
- noise in the depth created by the stereo camera
- noise of the ROIs effected by both the YOLO and the stereo camera
- systematic errors

Noise

We were able to keep a moving average of a few recent samples. A moving average was quick to calculate and proved sufficient for the needs of visualising our results.

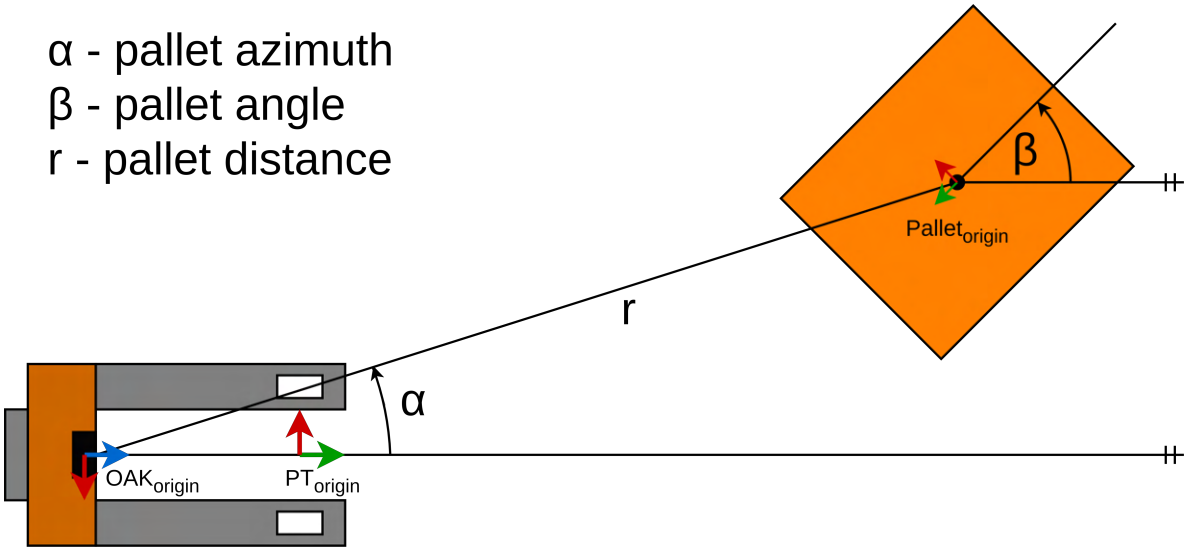


Figure 4.1: Diagram depicting the terminology used in the following section.

Systematic Errors

Sometimes the received data was just not salvageable in code. The stereo camera would simply not provide us usable depth data. Or the bounding box received from the YOLO missed the edges of the pallets.

Some problems we were able to solve using additional information, others less so.

The most common problem for us to solve was the misidentification of short and long sides of our pallets. The result of this was a rotation of the pallet by 90 degrees. And a small movement of the centre of the pallet. This specific issue we corrected by utilising the QR codes located on the short sides of our pallets. We used a NN similar to YOLO to detect QR codes.

- **ROS Communication**

As described in Section 2.4.2 for the purpose of converting our code to Robot Operating System (ROS) we have had significant help from Ing. Miroslav Uller. The ROS communication played a much larger part in the control module of our task. As part of the vision module, we created a simple method that took the input images from stereo camera and outputted the detections of our tracked pallets. These would then be taken as input by the control module. A high-level visual depiction can be seen in Figure 4.2

A challenge for us to deal with was the different frequencies of the segments. The frequency of our camera was not the same as that of our code and would not necessarily send its measurements all at once. The control module had a similar problem where our vision module gave detections at a much slower frequency than the controlled motors. A more detailed look into the control module can be seen in the sister thesis [1]. And our detailed implementation in Section 5.1.5.

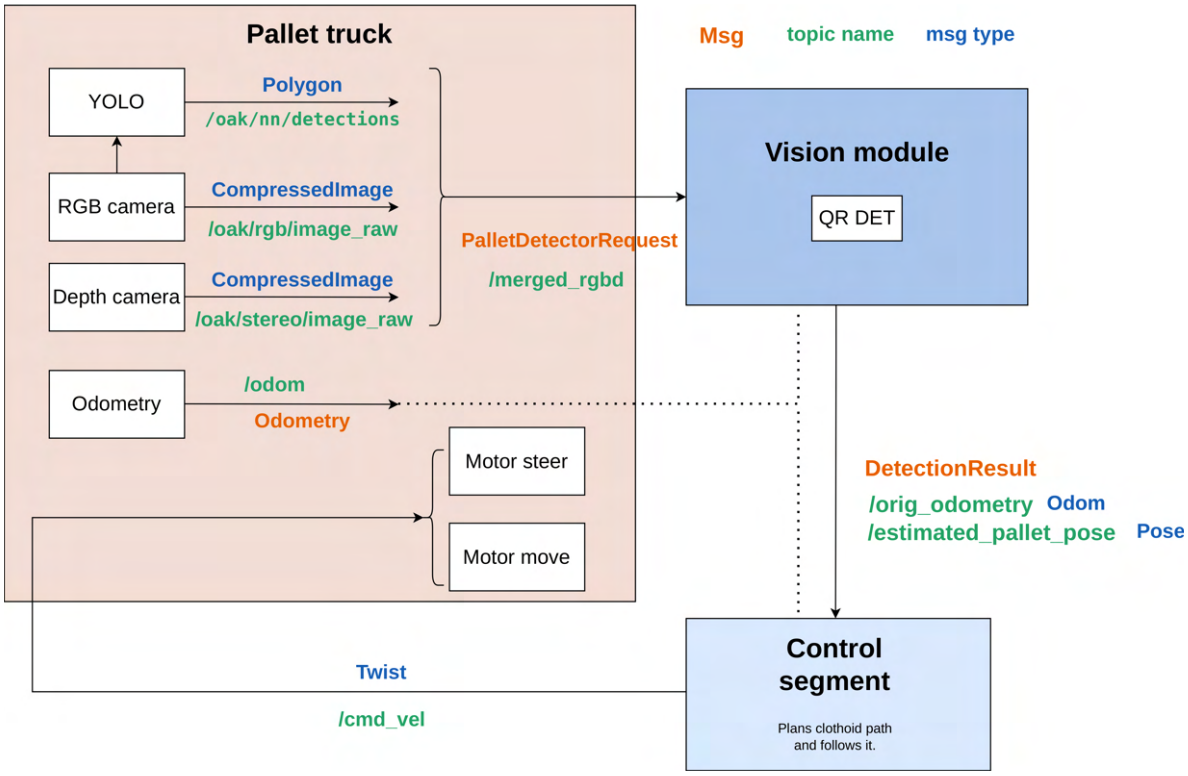


Figure 4.2: Diagram depicting a high level view on the ROS communication[1].

Camera Calibration

The intrinsic parameters of stereo camera were provided by the manufacturer. To unite the vision and the control modules of our project, we needed to establish a unified frame of reference. The origin point of our pallet truck is shown in Figure 4.1 and was defined as a point between the front wheels of the pallet truck. What we needed was to find the Special Euclidean group in three dimensions (SE3) transformation between our pinhole model camera and the pallet truck origin. Another useful measure was the orientation of the camera to the ground. We used this measurement when fitting the pallets onto the point cloud (PC).

Transformation

Some methods were described in Section 2.3.1. They can be split into costly to create, repeatable and presumably accurate methods. And less accurate ones that require less setup. As driving a physical model was not a target we accomplished by this thesis' deadline, we have decided not to create a reliable method of calibration yet.

[1]This diagram was based on one depicted in the sister thesis [1].

Orientation

The camera's orientation to the ground was inherently linked to the transformation between the camera and the pallet truck origin. If the ground were to be uneven it would also affect our measurements. Another measure of interest could also be the slope of the ground. The slope and terrain may also be useful to the control modules of our project.

Two rotations that interested us in the vision module were:

- the rotation of the camera in its mount and the rotation of its mount on the pallet truck,
- the rotation of the pallet truck to the ground.

The first one was static and already discussed in Section 4.3.6 the other was not and we needed to solve it.

The options that present themselves were to:

- find the ground in the measured point cloud,
- use the Inertial measurement unit (IMU) on board the stereo camera,
- assume the ground is flat and the pallet truck stands on it flat.

There were issues with those options. The ground was not always in frame. The IMU also measured the slope of the ground and any small changes in the movement of the truck, thus the measurements were not as useful.

We have also made the large presumption that the slope of the ground would be the same for both the pallet truck and the pallet of bricks.

4.4 Prototype Development

Once we were able to record data from the camera and created a mount for it described in Section 3.3.1 we gathered some real data. These consisted of recording of driving to and from pallets of bricks. In these recordings, we tried to replicate the assumed movement of the PC. After we have recovered data we optimised the behaviour of the vision moduleon the measured data.

Once the control module and simulation have progressed far enough we covered the entire vision module into ROS and tested it in simulation.

5 Implementation and Experiments

5.1 Implementation detailed

The code created in this project can be accessed at [10].

This section will give a more detailed explanation of how we accomplished the sub-goals of our task together with some results. A more general overview is described in Section 4.3. And the following section will keep to the same structure.

Pallet Detection

A more general overview can be seen in Section 4.3.1. For the retraining process, we used tools from Roboflow [52] where we expanded our previous training dataset. Later we trained a new You Only Look Once: Unified, Real-Time Object Detection [0] (YOLO) Neural network (NN) on our new expanded dataset which can be found at [8]. We trained multiple variants of the YOLO based on our hardware and speed requirements.

YOLO would give as its output a Region of interest (ROI) in the format of a bounding box together with its confidence. We filtered out all detections with confidence below 80% and all detections closer than 5 pixels to the edges of the frames[1]. These values gave us a good balance between making the required detections and suppressing most of the false positives. The YOLO did not always fit the bounding boxes precisely onto the pallets' edges. It also had issues when the pallets were obstructed. More information on the performance together with visuals can be seen in Section 6.2.1.

In the following section visualisations will use the most accurate of our detections calculated on a computer instead of the stereo camera.

Pallet Pose Estimation

A more general overview can be seen in Section 4.3.2.

The conversion from our depth map to a point cloud (PC) can be written into the following equation:

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & y_n \end{pmatrix} = \begin{pmatrix} d_1 & d_2 & \cdots & d_n \\ d_1 & d_2 & \cdots & d_n \\ d_1 & d_2 & \cdots & d_n \end{pmatrix} \circ \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} u_1 & u_2 & \cdots & u_n \\ v_1 & v_2 & \cdots & v_n \\ 1 & 1 & \cdots & 1 \end{pmatrix} . \quad (5.1)$$

Where $\mathbf{u}, \mathbf{v}$ are indexes of our pixels and $\mathbf{d}$ their corresponding depths. The $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are coordinates of our points in space.

We then converted the measured data from mm to m. We also needed to rotate the resultant PC using our extrinsic rotation matrix. The method by which we measured the extrinsic rotation matrix is described in Section 5.1.6.

Afterwards, we removed points from the PC both close to the ground and to the top edge of the pallets. This was done to suppress any errors caused by YOLO if it were to give

[1] Detecting a pallet near the edges of our frame had the effect of changing the side lengths of our detected pallet. A 1.2 m long side would seemingly look to be 1 m long.

us larger bounding boxes than desired. Otherwise, we might have tried to fit planes through the ground, wooden pallets or even other pallets of bricks if some were located on top of each other[2].

The pallets we detected could have either one or two of their sides visible. At random we chose at most 6000 points through which we then tried to fit planes. To achieve this we iteratively applied Random sample consensus (RANSAC) onto the PC removing inliers after the first iteration and applying a second if the amount of our remaining points was over the threshold of 1500. As inliers, we classified all points closer to the plane than our precalculated inlier threshold. The inlier threshold (t) was calculated by using the mean distance of the points in the z direction and inputting it into the following formula:

$$t = \max(0.02, \min(0.00417 \cdot z^2, 0.5)) \,. \tag{5.2}$$

Which we derived from the stereo depth accuracy of the stereo camera [45] and that can be seen in Figures 5.1 and 5.2. For our RANSAC we used the library pyRANSAC-3D [48]. The following parts of the code had access for each Region of interest (ROI) to both the equations describing our planes and to their inlier points.

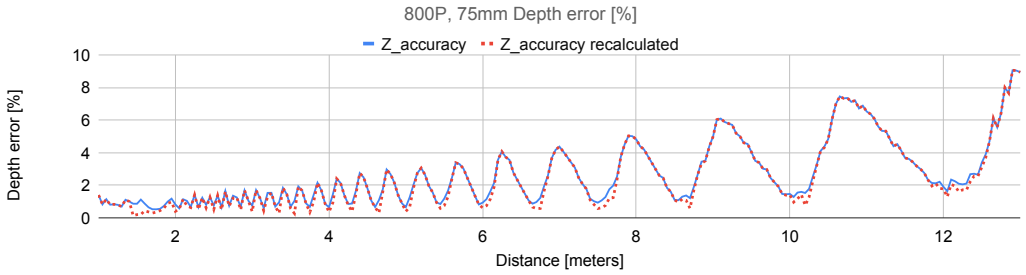


Figure 5.1: Depth accuracy for our used camera where we recalculated the measured values provided by the manufacturer[3][45].

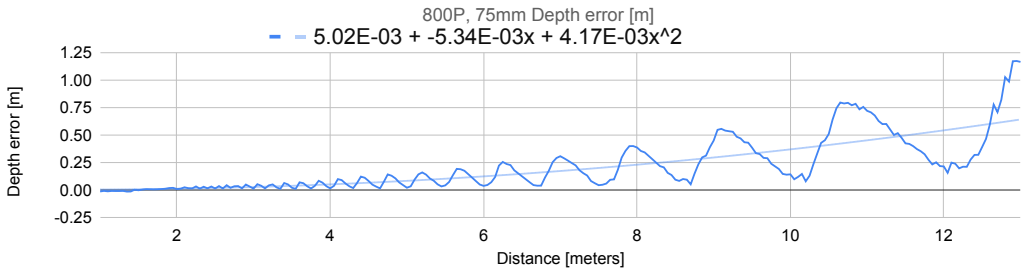


Figure 5.2: Depth accuracy for our used camera converted to distances with its second polynomial representation.

Our next step was to filter out wrongly detected planes. Firstly we checked the angles between the planes and the ground, whether or not they are close to perpendicular i.e. in range $[75°, 105°]$. If multiple planes were detected we also calculated the angle between them, hoping for them to be in the same range $[75°, 105°]$.

[2]We would not pick up pallets with a second layer above them but as most of our gathered data had two layers of pallets we still needed to be able to detect them.

[3]As we can see there is a slight difference in the high accuracy spots between the original values and our calculations of the accuracy. Surprisingly our measurements came out more accurate. As to the cause, we can not say.

Afterwards, we needed to find the lengths of the sides of our pallets. This proved to be a significant issue as the pallets' side lengths of 100 cm and 120 cm were quite similar. YOLO while great at detecting pallets gave us only a bounding box as its output. Thus the assigned area tended to not perfectly follow the edges of our pallets. See Section 5.1.1 for more detail. In essence, the inlier points of our planes could paint us the wrong picture as we can see in Figures 5.3 and 5.4.

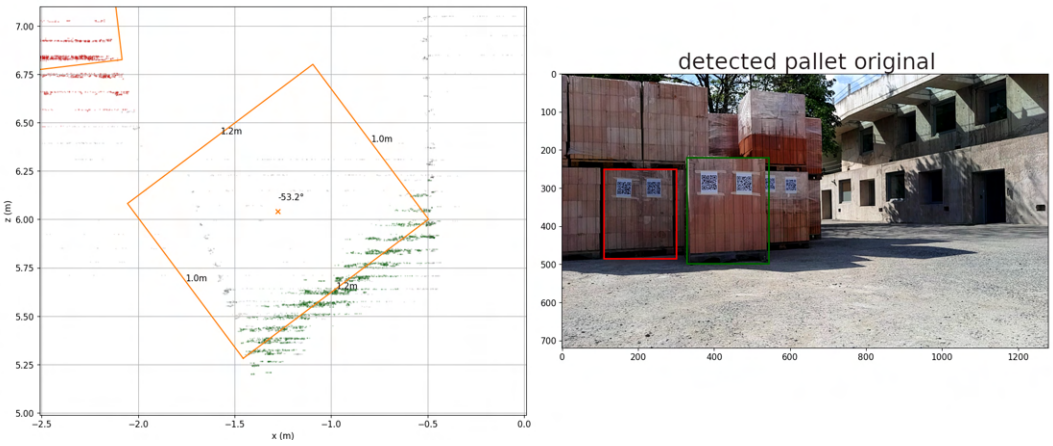


Figure 5.3: Detection of our pallet where we can see that an improperly fitted long side describes our PC reasonably well.

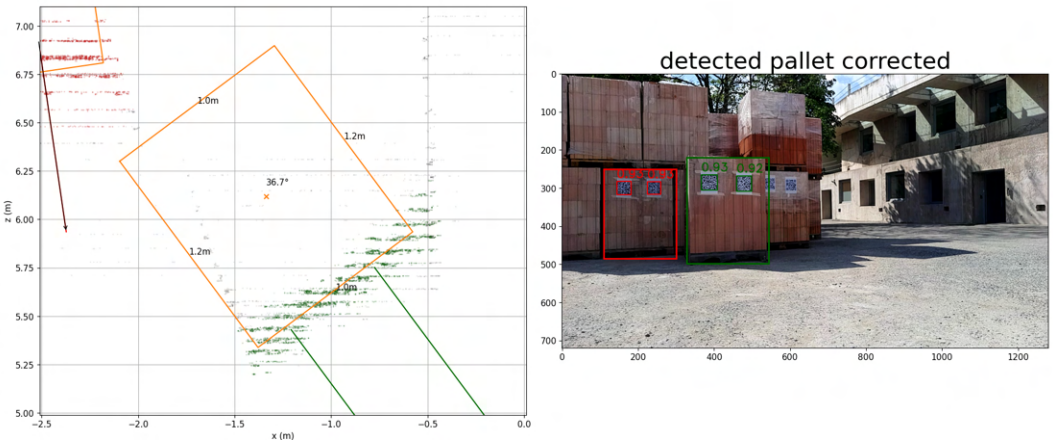


Figure 5.4: Here we corrected the detection and can see how a correct length side can look short in the PC.

We resolved to create a system that would correct the issue of wrongfully assigning the side lengths by using the QR codes located on the pallets. See Section 5.1.4 for more information.

The methods by which we determined the endpoints of the sides differed by whether or not we detected one or two sides. We have implemented other approaches seen in Section 5.2.1, but these did not improve our results much.

One Side

First, we transformed all of the inlier points onto their corresponding planes. A transformed PC can be seen in Figure 5.5. In the coordinate frame of the plane, we found the median of the points in the x direction. And then we moved in its vicinity trying to find the best centres for side lengths both 1 m and 1.2 m. The best centre was the one where the most points (inliers) were part of an interval formed by the side lengths. The best centre was on average 4 cm away from the median. We then calculated which of the two side lengths described our data better.

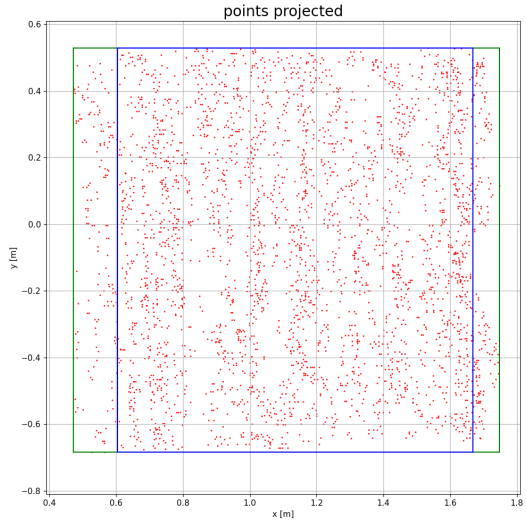


Figure 5.5: Inlier points projected onto their fitted plane with original and top 90th percentile edges visualised

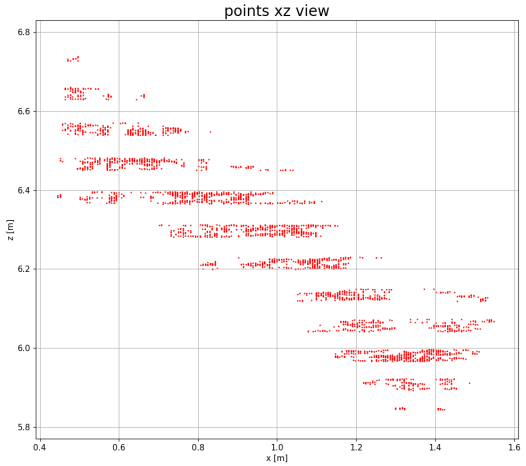


Figure 5.6: Top down view of the inlier points pictured in Figure 5.5.

The number of inliers we divided by the tested lengths of the sides, thus we received inliers per meter (λ) values. A visualisation of theoretical λ values can be seen in Figure 5.7. We decided that the measured side would be one meter long if the following inequality was fulfilled.

$$\lambda_{\text{short}} \cdot \lambda_{\text{margin}} < \lambda_{\text{long}} \tag{5.3}$$

The theoretical value of λ_{margin} would be 0.917 but a value of 0.93 gave us better results and was used. Our method made some basic presumptions about the data such as it following a uniform distribution and was not always correct.

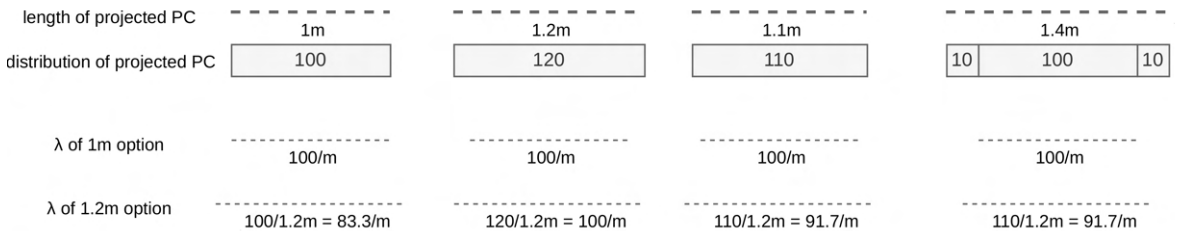


Figure 5.7: Different theoretical inliers per meter values measure on PCs.

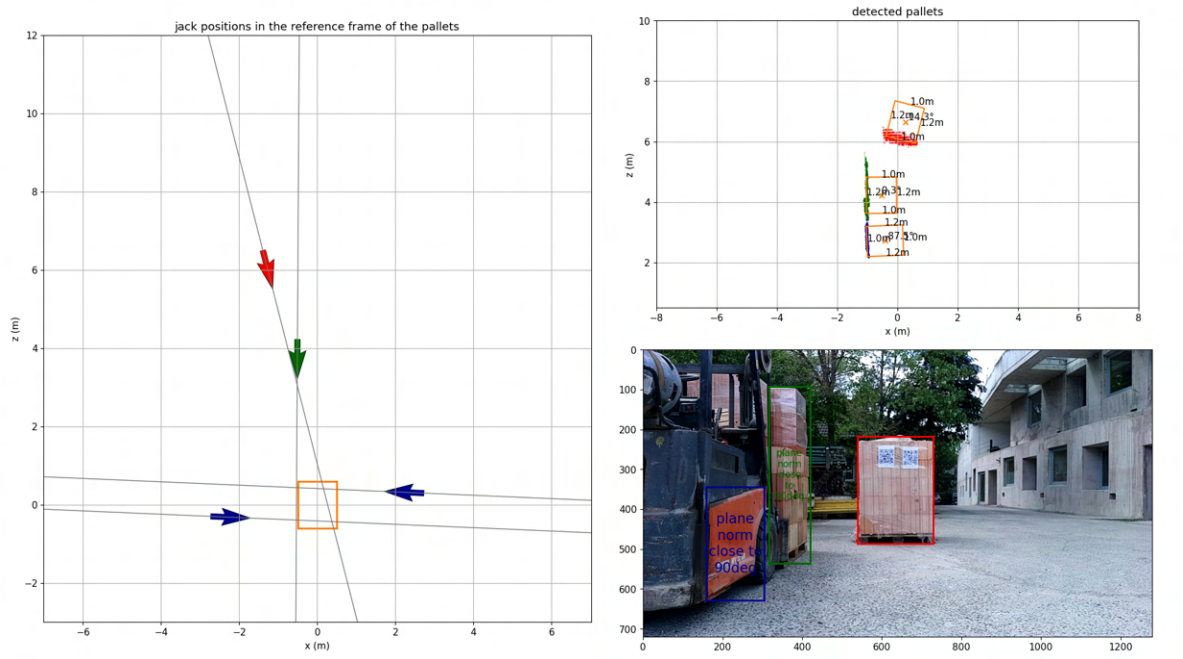


Figure 5.8: Pallets being fitted to the wrong side of PC after RANSAC calculated a wrong slope of a plane[5].

After we had computed the side length we checked if the inlier ratio (the amount of inliers in the entirety of our PC) was over 70%. We also measured a "slack" value. The difference in length between the length of our fitted side and the outer edges of the projected inlier PC. To reduce the influence of random noise on the outer edges we calculated the 90th percentile[4] of points closest to the mean of the points and used this reduced value. And filtered for slack values smaller than 0.3 m.

We then transformed the calculated centre and bounds back to the world coordinates.

Two Sides

Here we first found the corner of our pallet of bricks by calculating the intersection of the two detected planes. We again projected the PCs onto the planes together with the calculated corner. Then we tried which combination of the side lengths gave us a better inlier ratio. We filtered for inlier ratios above 66% and converted them to the world reference frame.

From the added key points we calculated the locations of the rest of the pallet's corners, its centre and angle. Thus finally we estimated the pallet's pose.

One notable shortcoming we noticed was an unreliability of our code when dealing with pallets close to parallel to the camera's orientation. Near parallel angles noise had a significant effect. And could even make the slope of our planes have a wrong sign i.e. be positive instead of negative. A result would be the pallets being fitted to the wrong side of out PC as can be seen in Figure 5.8. These issues thankfully arose only when we couldn't see the front side of our pallet and thus weren't very common.

[4]This was done by measuring the top 5th and 95th percentile in the x direction.
[5]We can see that in this image YOLO wrongly decided that the orange of the forklift is a pallet of bricks.

- **Pose Tracking**

A more general overview can be seen in Section 4.3.3

To simplify our task we have decided to use a preexisting library namely Motpy [49]. While choosing this route offered us fewer customisation options than creating our own tracker, it allowed us to focus on other tasks.

The Motpy tracker took rectangles as an input. Using its internal memory predicted both new sizes and locations in space of the rectangles. It handled both missing frames and the loss of tracks. What interested us were the internal identifiers (ids) of the tracks, we needed to remember them and pair them with our detections. When certain conditions such as being close enough to a pallet at a good azimuth angle with a legible QR code code we would get the option to establish a track. This would constitute saving the id of the tracker. We also needed to decode the QR code located on the pallets. In the future text on the QR code would help with reporting which pallets we used and with cataloguing the entire bricks storage. The final structure which we decided to create was such:

- detect and locate the pallets,
- input the locations to the tracker,
- pair the trackers internal locations to our detected ones,
- find a pallet with a matching id and output its pose.

For the pairing of our detections and the trackers ids we used the predicted and detected locations of our pallets. We found the shortest euclidean distances between the centres of the pallets to the centres of the tracks. We also created a logic that handled both duplicate and non-existent tracks.

A special ROS node that would handle the track selection with a special graphical user interface (GUI) was planned to be implemented later.

A notable drawback of using motpy with our position tracker is that we could not utilise the rotations of our pallets as motpy did not allow for it. While a more suitable tracker might have indeed existed our used solution proved to be sufficient. As we will see in Section 6.2.2 the rotations are the features we are the least sure of.

- **Pose Refinement**

A more general overview of how to improve our detections can be seen in Section 4.3.4.

Pose Refinement Using QR Codes

We decided to utilise the QR codes located on the pallets to improve our pallet pose estimation.

The QR codes had one important feature - they were fitted only on the short sides of our pallets. Sadly we could not ensure that the locations of the QR code codes within a pallet would be the same for all pallets. Otherwise, we might have been able to do more.

An important shortcoming we needed to address was switching short and long sides of our pallet. If such a misidentification were to happen to our target pallet we might have a significant problem.

As we already had to decode the QR code codes to properly establish track we hoped that refining our detections using QR codes wouldn't be too costly.

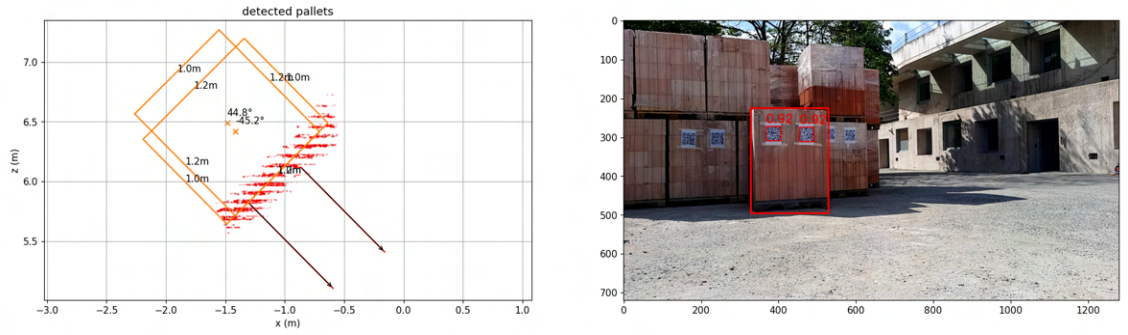


Figure 5.9: In this figure we see both the original and the refined detections of a pallet.

To improve the range of our refinement we decided to not only try to decode, but to only detect the QR code codes. The text on the pallets was not nearly as important to us as the knowledge of the QR codes existence. For both the tasks of QR code detections and decoding we used the library QReader [39].

The QR codes would be used like the following:

- Detect the pallets.
- Find their poses.
- Filter for the pallets we are interested in (such as the one we track).
- Crop the input colour image to the ROI corresponding to our filtered pallet.
- Detect and decode QR codes in the cropped image.
- Crop the depth image using the bounding box provided by our QR code code detector.
- Convert the depth data to a point cloud (PC) in world coordinates (as described in Section 5.1.2).
- Find the mean point in space of the QR code.
- Find the closest plane on the corresponding pallet to the mean point.
- Calculate the rejection vector of the mean point to the plane.
- Compare the orientation of the vector with the rotation of the pallet and rotate the pallet if required.

The result of our QR code refinement can be seen in Figure 5.9.

Other Methods of Pose Refinement

We have tried to remove as much noise as possible in all stages of the code while maintaining its speed. And the large noise of the stereo camera at long distances did not make our task easy. As we can see in Figure 5.13 the noise is rather substantial. Thus even RANSAC could not remove it entirely. Increasing the number of iterations and the size of the converted PC improved the noise suppression of RANSAC at a great cost in required resources.

A solution to this issue could be to remember the previous detections to remove some of the noise. We could adaptively change the amount the resources allocated to the RANSAC. Or estimate the poses in two iterations, one to establish a track and the other to improve it. These features were not implemented in the timeframe of this thesis.

- **ROS communication**

A more general overview can be seen in Section 4.3.5.

To make our code communicate with the Luxonis OAK-D Pro and the rest of the systems we have decided to utilise ROS. We have set up the stereo camera to send colour, and depth images at the same frequency, the camera then applied YOLO onto the colour images. The transported data couldn't have all been transported from the stereo camera at once. Thus the frames were equipped with precise timestamps. We've paired the received data together by the timestamps. We have created a method of reading the measured data using ROS. This node then communicated with another that calculated pallet positions from the received images. We also later planned to utilise measured odometry in the vision module, this would also be useful in simulation. At the end of this thesis, our code did not incorporate the odometry measurements. The control module described in the sister thesis would however require it [1]. A diagram with an overview of communication inside the vision modulecan be seen in Figure 5.10.

A more detailed diagram can be seen in Figure 5.11. The "Concentrator client" would keep in memory the last detected frames. Each update it would look if the client received any new images in the time before the previous update. If it did receive them it would rewrite the previous frames in memory. This made sure we always made our calculations only on the most recent frames. In the future, we want to update the Concentrator with each new odometry measurement; however, this feature was not implemented by the end of this thesis. For each update, the Concentrator would also try to send an action request to the "Pallet detector & tracker client". Actions are one of the communication types in ROS 2 and are intended for long-running tasks" [50]. The server would answer to the client after a request was made whether or not it accepted the signal[6]. If it did the client would clear the sent frames from memory. Thus the client wouldn't send the same frames to the server multiple times. After the detections were made the server sent its result back to the client. The client would then pass the detections further along.

[6]We assumed communication between the ROS nodes would be very reliable as all of the ROS nodes ran on the same machine.

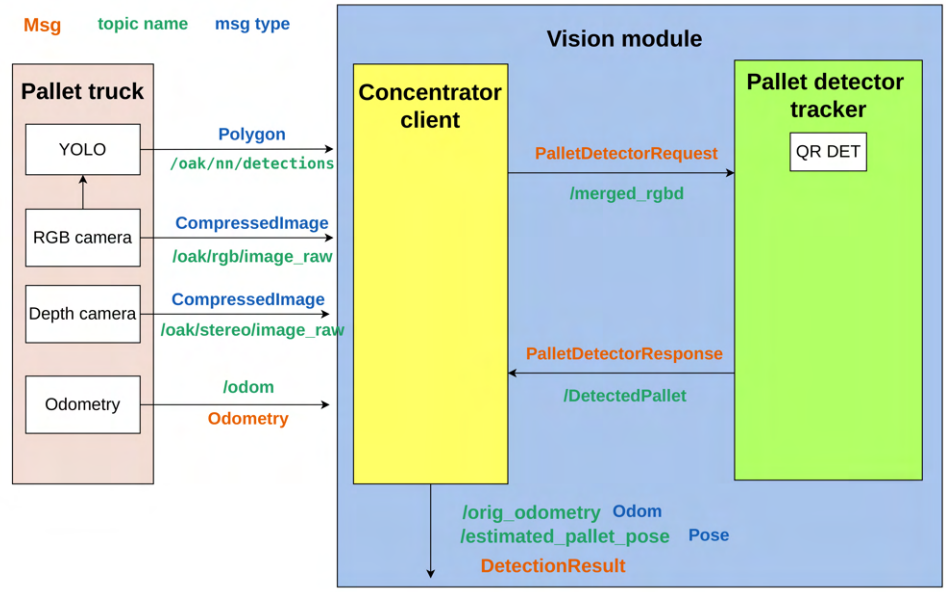


Figure 5.10: ROS communication with the vision module.

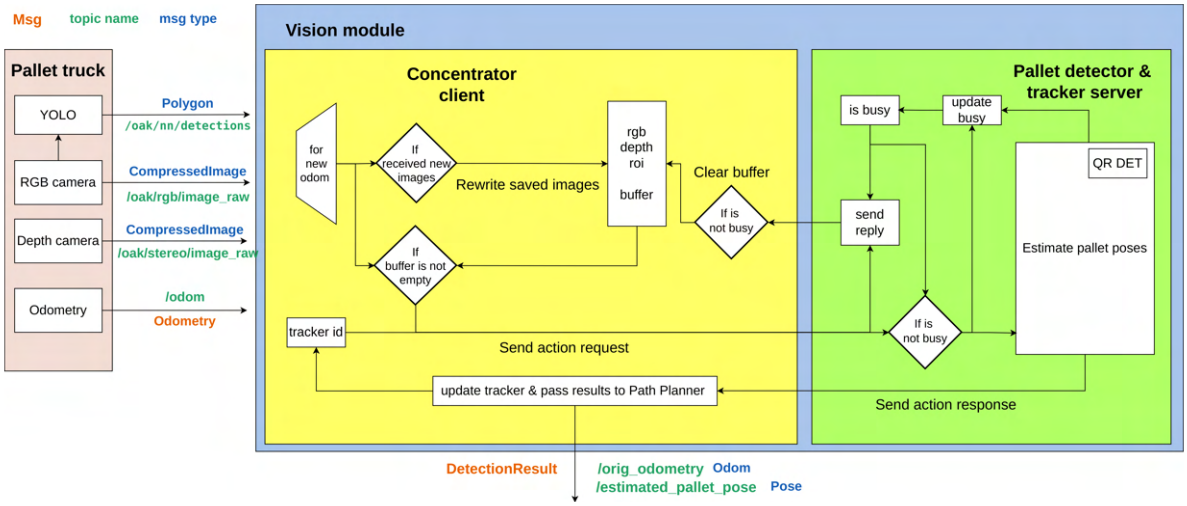


Figure 5.11: Detailed scheme portraying the ROS communication inside the vision module.

Camera Calibration

The overview can be seen in Section 4.3.6.

As by the time of this thesis deadline, we were not able to control the physical pallet truck we've decided not to focus on calibrating the camera.

We've decided to proclaim that the pallet truck would stand flat on the ground. To use our measured data, however, we still needed to measure the rotation of the stereo camera inside the pallet truck. For this purpose, we found the rotation between the plane of the ground and the plane defined by the world z-axis. This measurement we then used to rotate our measured PC.

5.2 Unsuccessful Solutions

Deciding the Length of a PC

Unlike the solution described in Section 5.1.2 where we used the inliers per meter measurement to compute the length we have also tried another approach. Namely to measure the length of the densest part of the PC and declare it to be the length of the side.

As before we projected the PC onto the sides plane. We calculated edges by taking the top 5th and 95th percentile in the x direction and measured the width of the cropped PC. Visualisation can be seen in Figure 5.5 This gave us similar results to the used solution.

ROI Tracking

Initially, we implemented a method of tracking that utilised the bounding boxes created by YOLO instead of the pallet poses we later utilised in Section 5.1.3. The ROI tracking worked reasonably well and fast but was more susceptible to YOLO noise, large rotations of pallet truck and did not allow for later improvement. It worked the same way as the later used solution.

QR Code Refinement

We have tried other approaches than the one chosen in Section 5.1.4.

The main thought behind them was still to find the direction in which the QR code points.

First, we tried to find the QR codes vector by calculating the cross product of two vectors starting at the same corner making the edges (or diagonal) of our QR code. This did not work reliably because of the high noise of the measured depth data.

Another method was to fit a plane directly onto the QR codes PC. Here we encountered the issue that at long distances the noise of our depth data in the direction of the camera was greater than the area of our QR code. The noise of detection at a large distance can be seen in Figure 5.12. Thus the planes tended to be perpendicular to the pallets' planes.

As we've seen solution was to utilise the planes already calculated on a much larger area.

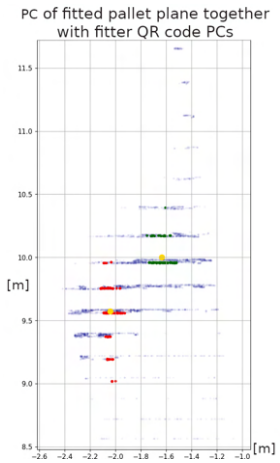


Figure 5.12: Diagram displaying detected PC of a QR code in a pallet.

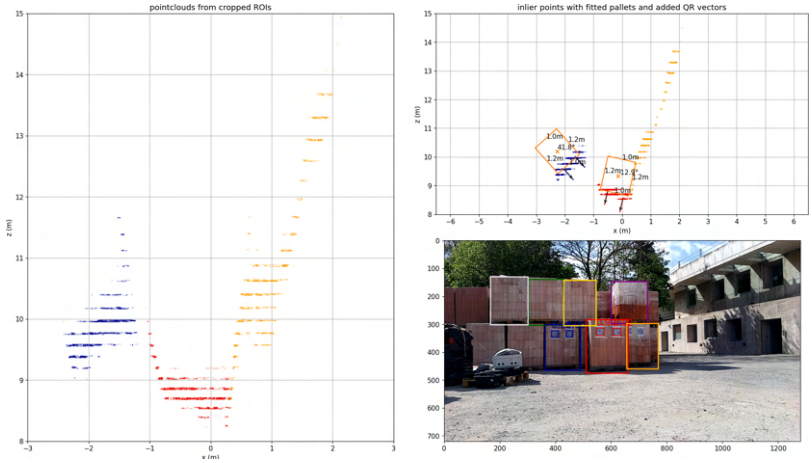


Figure 5.13: Diagram displaying detected PC of a pallet viewed from the top.

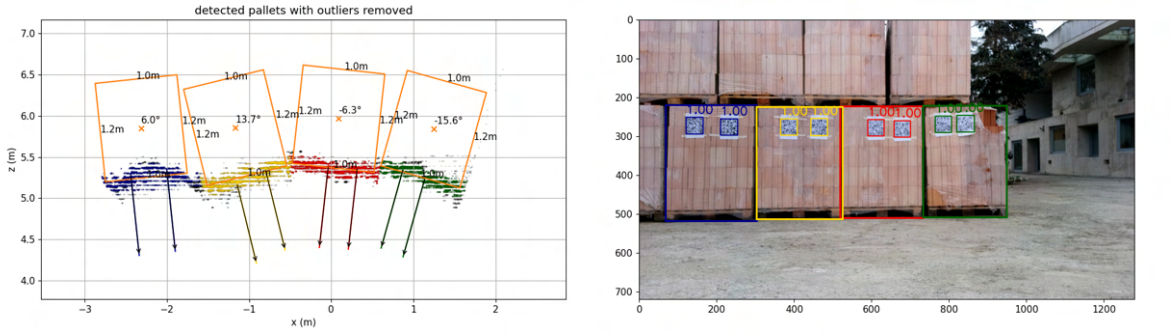


Figure 5.14: Pallets tilted when standing in a straight line. Black is used to depict the entire PC.

5.3 Stereo Camera Depth Issues

In the later parts of the work on this thesis, we have noticed discrepancies between how we assumed and how the stereo camera did work.

When detecting a wall of straight standing pallets we have noticed an issue, where the pallets are correctly fitted onto their point clouds (PCs) but not standing straight side by side as can be seen in Figure 5.14.

Straight Wall Experiment

Thus we have decided to conduct an experiment, we found a well-textured straight piece of wall seen in Figure 5.15. On it measured the depth on the wall, created a PC[7] from the measured data and removed all the points with bad texture. We then projected the points onto the XZ plane which can be seen in Figure 5.16 and 5.17. As we can see Luxonis OAK-D Pro did not measure the straight wall straight. The measurements look wavy at both close and far ranges.

[7]Unlike in Section 5.1.2 here we did not rotate the PC by the extrinsic rotation matrix.

Figure 5.15: A straight wall with a good texture.

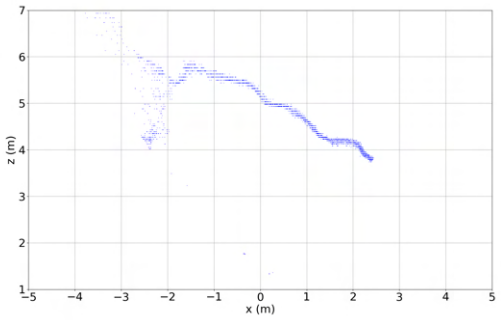


Figure 5.16: A well-textured straight wall.

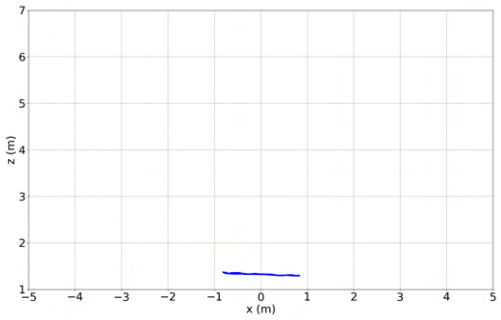


Figure 5.17: A well-textured straight wall close.

We've decided to plot all of the distances we measured in our recording into one large histogram seen in Figure 5.18. As we can see not all of the distances appear with the same frequency. Peaks have formed and are further apart at larger distances. Our camera is set up to use 3 sub-pixel bits i.e. a disparity of one pixel is split into 8 sub-segments and by no coincidence every eighth measured distance made a peak. We have decided to examine the measurements at a distance of 5 m more closely.

- **The effects of camera disparity**

We want to calculate the theoretical effects of camera disparity on our depth measurement. Namely how much does the distance change when the disparity changes by one pixel. The diagram seen in Figure 5.20 displays the used terminology.

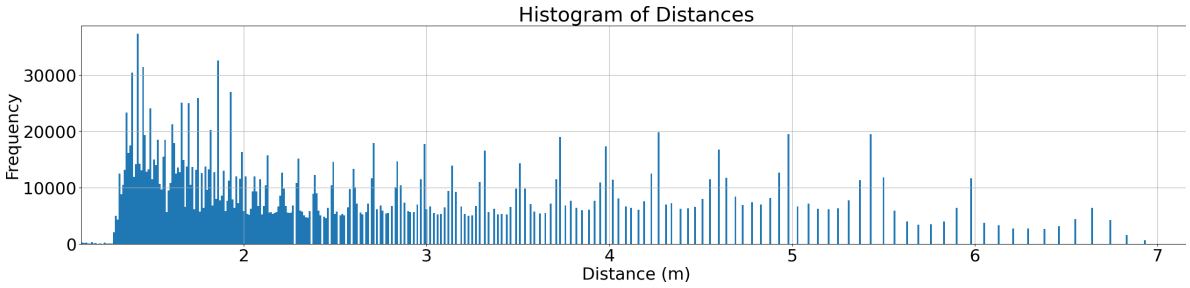


Figure 5.18: A histogram showing the frequency of measured distances on a straight wall.

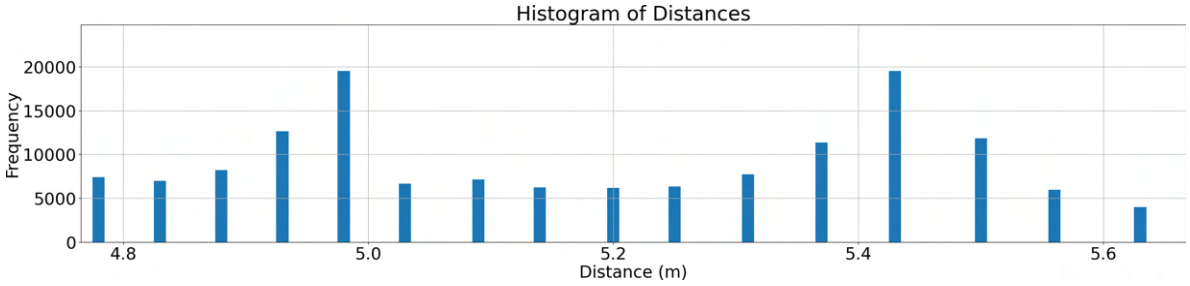


Figure 5.19: A detail of the histogram seen in Figure 5.18 around the distance 5 m.

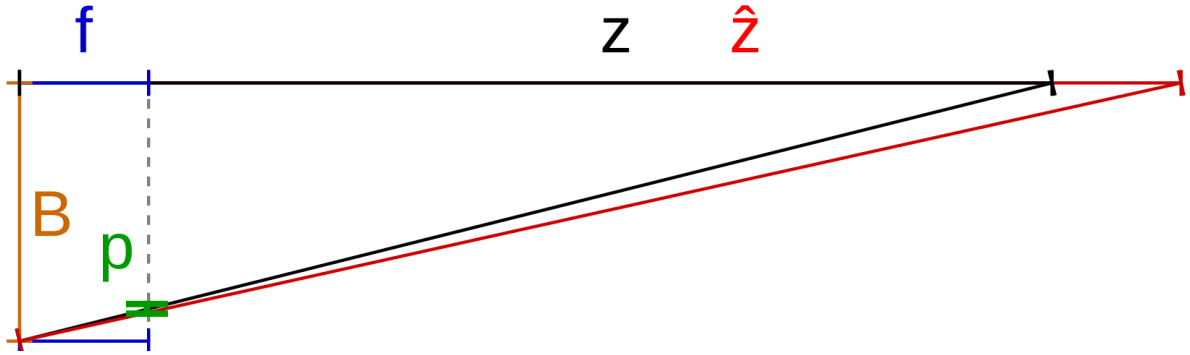


Figure 5.20: A diagram showing the notation used in the following section.

The values for stereo camera are:

$$\text{Pixel size} : p = 3\,\mu\text{m}, \tag{5.4}$$
$$\text{Effective Focal Length} : f = 2.35\,\text{mm}, \tag{5.5}$$
$$\text{Baseline distance} : B = 75\,\text{mm}, \tag{5.6}$$
$$\text{Measured distance} : z, \tag{5.7}$$
$$\text{Distance after one pixel change} : \hat{z}\,. \tag{5.8}$$

Stereo cameras measure the distance using the following equation: [25], [44]

$$z = \frac{fB}{d}\,. \tag{5.9}$$

We can convert this equation into the following:

$$\begin{aligned} \hat{d} &= \frac{fB}{z} - p, \\ \hat{z} &= \frac{fB}{\hat{d}}\,. \end{aligned} \tag{5.10}$$

When we input the values at original $z = 5000\,\text{mm}$ we get:

$$\hat{z} = \frac{2.35 \cdot 75}{0.03225} \approx 5465\,. \tag{5.11}$$

so $\Delta z \approx 465\,\text{mm}$.

According to [34] the value would be some 425 mm. These values correspond well to the peaks we see in Figure 5.19.

Disparity Experiments

The stereo camera also gave us the option to get not the distances, but the disparities as our results. We have made both a long recording with varied scenery and again captured images of a straight textured surface. We can see the distribution of disparities in a recording in Figure 5.21. Again we see peaks in the recording made up of every 8th measurement, thus we grouped every eighth measurement together to find the general distribution. In Figure 5.22 we can see that the distribution of the disparities is not uniform. And in Figure 5.23 we can see the effect on a textured surface.

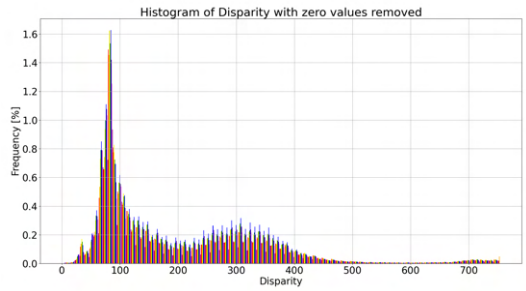


Figure 5.21: distribution of disparities in a recording

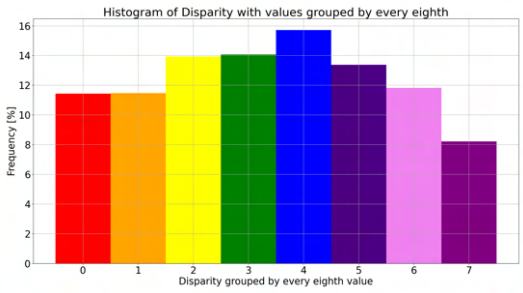


Figure 5.22: histogram of disparities every 8th grouped together.

We have seen the effect in Figure 5.16 i.e. a straight wall is not straight. It could also change the corner angle of our pallets as pictured in Figures 5.24 and 5.25.

The manufacturer does provide some data that support our findings [45] but nothing quite as clear as our findings.

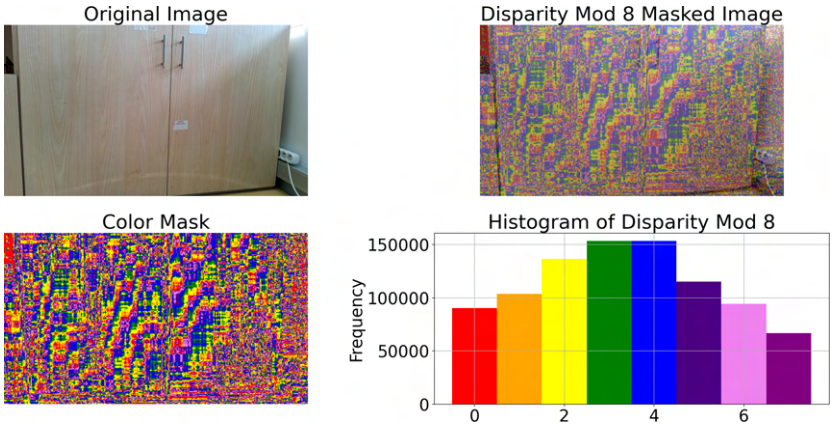


Figure 5.23: a textured cabinet together with a histogram of grouped disparities

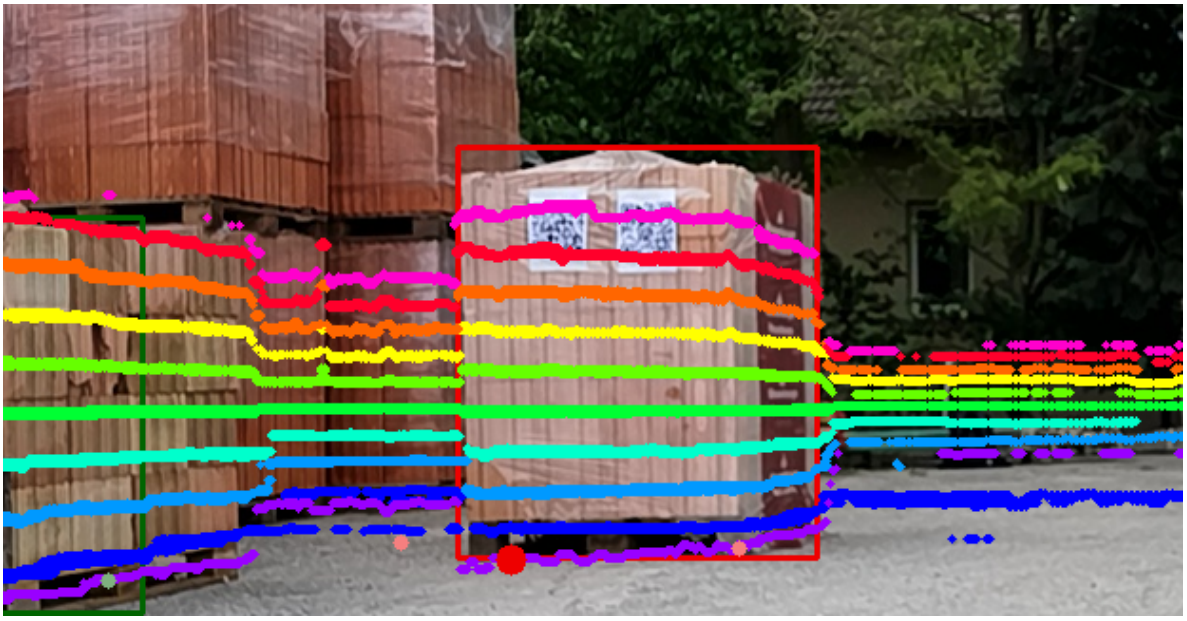


Figure 5.24: Measured depth data projected onto their corresponding colour image. Each colour band corresponds to a new height and is 20 mm tall.

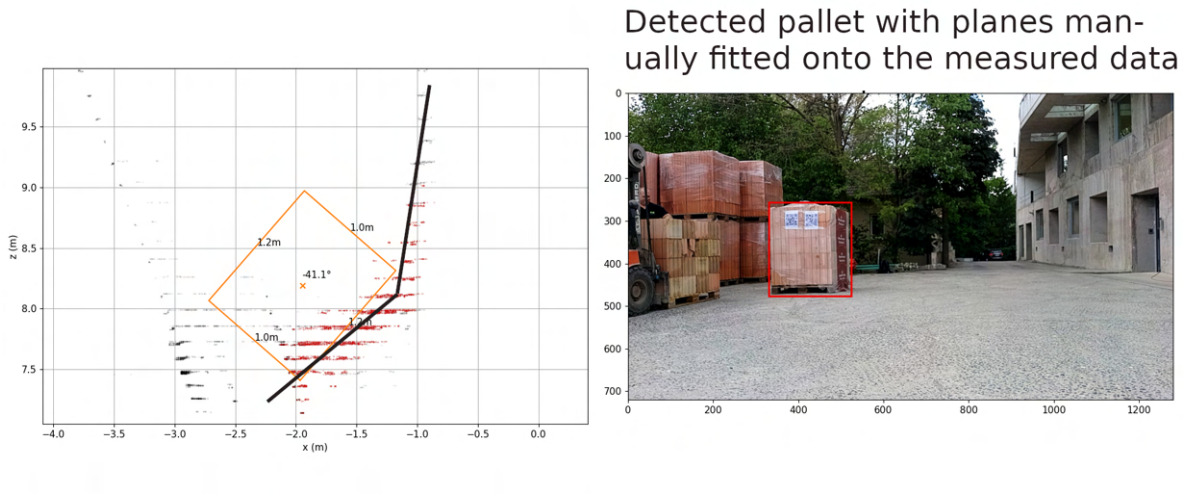


Figure 5.25: A top-down view of the detected PC onto which we manually fitted planes.

6 Results

6.1 Data Collection

For the vision module of our project we gathered real on-location data[1]. We mounted the camera into its mount on the pallet truck and made recordings of us moving to and from pallets of bricks. We tried to simulate as closely as possible future usage, i.e. we moved at a similar speed, and at similar trajectories to the ones the control module would produce. The camera was also set up the same way we predicted it would be in the future.

The recordings we unpacked and tested on the testing computer described in Section 3.3.4.

In the following sections, we shall describe our results and our used methodology. A discussion of the results is in Section 6.3.

6.2 Presentation of Results

In the following section, we will be using the terminology described in Figure 6.1.

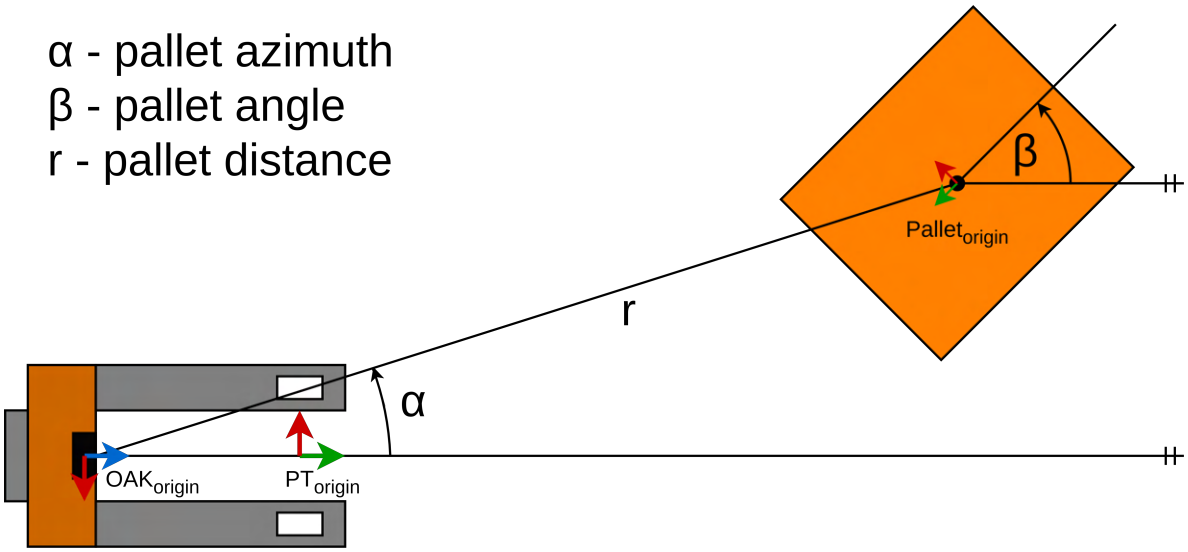


Figure 6.1: Diagram depicting the terminology used in the following chapter.

[1] 49.95438012104425, 14.663088931364516

Pallet detection

First, we shall present how our trained You Only Look Once: Unified, Real-Time Object Detection [0] (YOLO) performs. There are five distinct YOLOs we shall compare:

- original version of YOLO that existed before we expanded the training dataset (OLD),
- high detail version of YOLO recorded on Luxonis OAK-D Pro (HD_o),
- low detail version of YOLO recorded on Luxonis OAK-D Pro (LD_o),
- high detail version of YOLO made on our PC after the recordings were made (HD_{pc}),
- low detail version of YOLO made on our PC after the recordings were made (LD_{pc})[2].

The difference between low and high detailed versions was the size of the input image YOLO used. We were not able to calculate both the low and high detailed versions on stereo camera at the same time. Thus the recording using them were slightly different. The behaviour of YOLO can be seen in Figures 6.2 to 6.4 and a behaviour over a larger amount of detections in Figure 6.5 and Table 6.1.

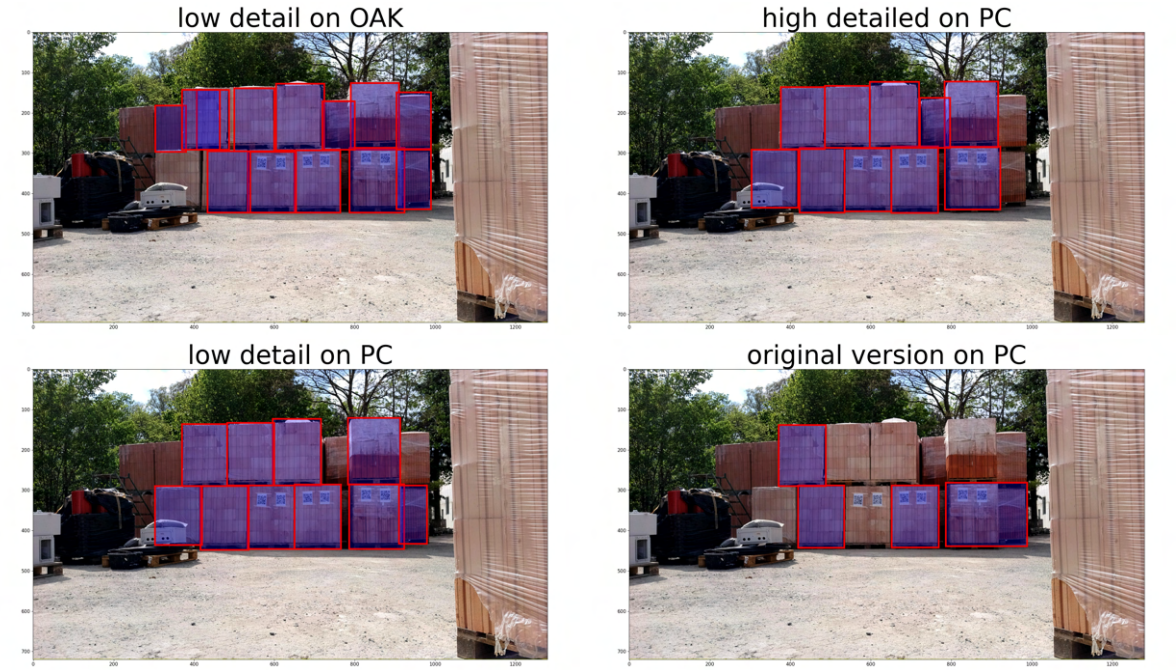


Figure 6.2: A row of pallets that used HD_o, same scene as in Figure 6.4, all graphs in Figure 6.5 using LD_o use this recording

[2]This version was created only so that LD_o had a good comparison.

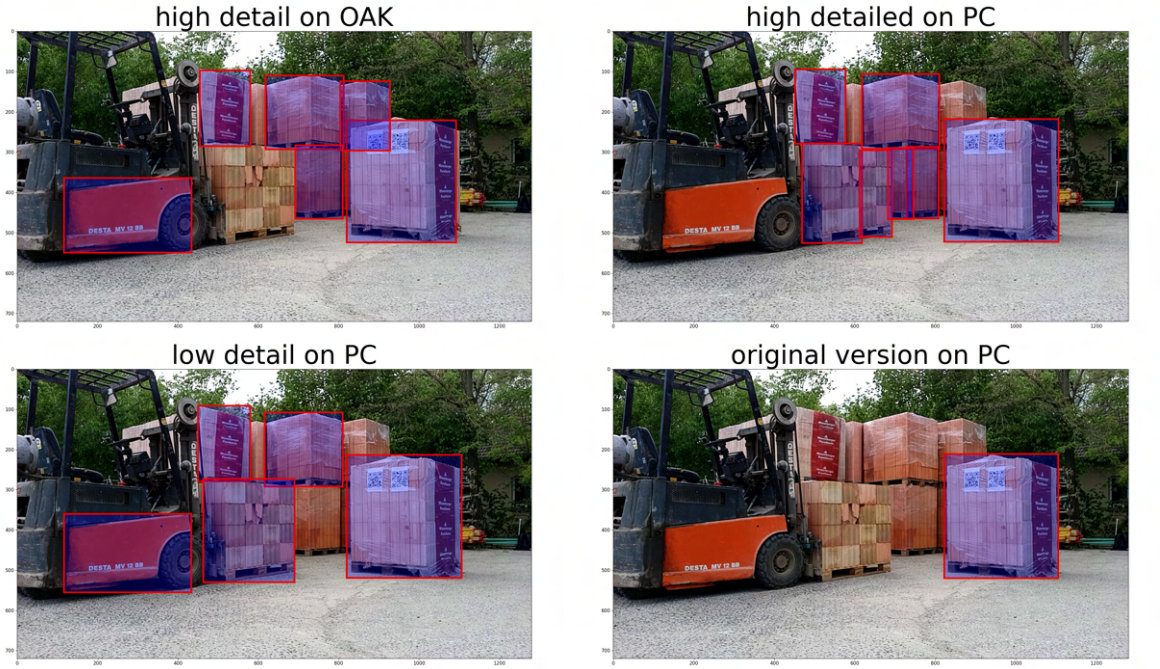


Figure 6.3: A solo loading of a pallet (not pictured in Figure 6.5)

Figure 6.4: A row of pallets that used HD_o, same scene as in Figure 6.2, all graphs in Figure 6.5 using HD_o use this recording

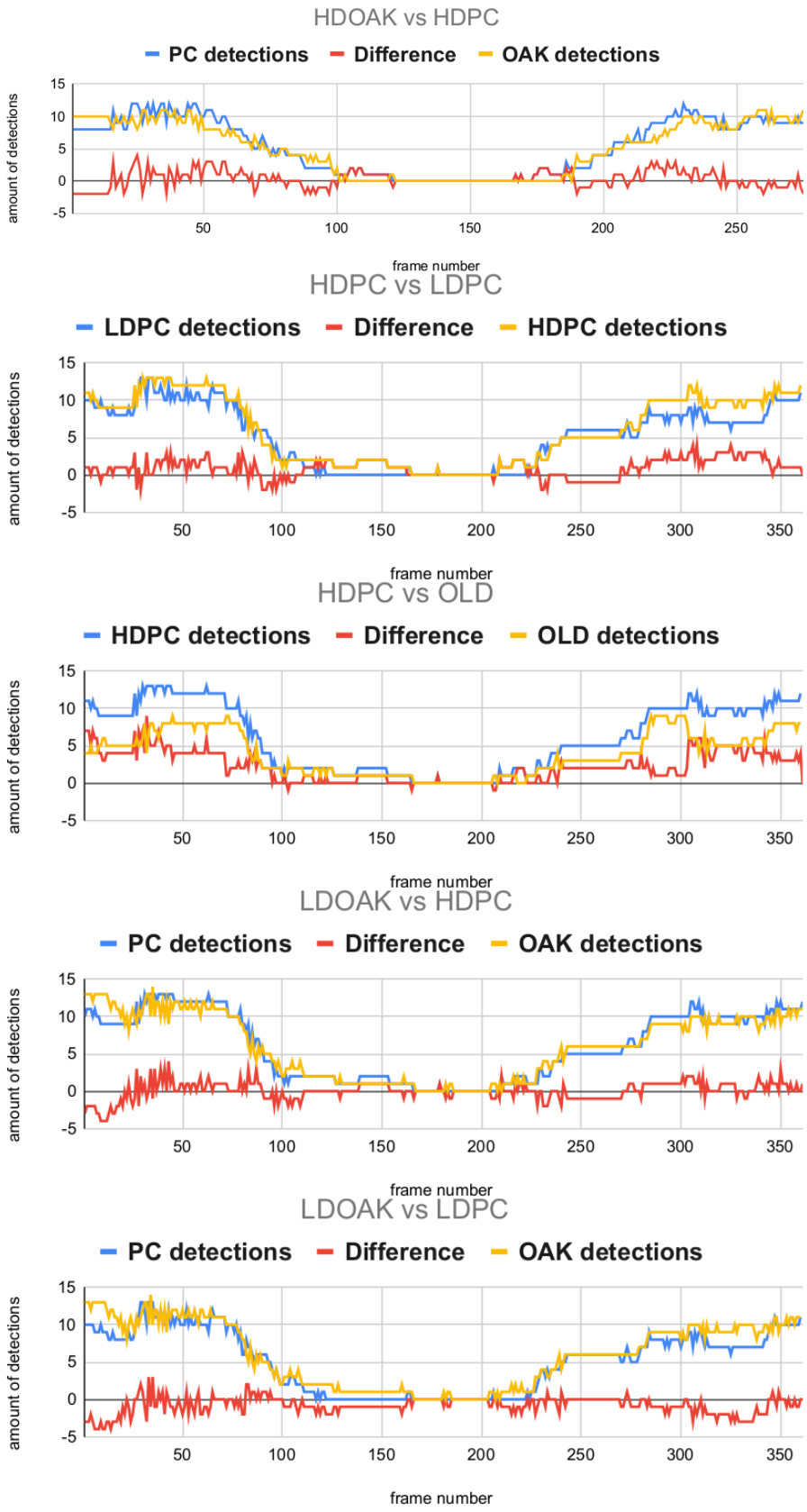


Figure 6.5: Comparison of evolutions of different YOLO variants. The amount of detection in scenes from Figures 6.2 and 6.4. HD_o vs HD_{pc} used Figure 6.4, all others Figure 6.2.

Recording comparisons	HD_o-HD_{pc}	HD_{pc}-LD_{pc}	HD_{pc}-OLD	LD_o-HD_{pc}	LD_o-LD_{pc}
detections amount avg	5.4529	6.0720	6.0720	6.0720	6.0332
det diff avg	0.2271	0.8615	2.1884	0.0388	-0.8144
det diff stdev	1.1272	1.2529	2.0105	1.1896	1.1311
size avg [px] $\times 10^3$	111.598	139.051	139.051	139.051	157.647
size diff avg [px] $\times 10^3$	43.894	-81.447	-3.956	-18.596	-100.050
size diff stdev [px] $\times 10^3$	151.512	176.592	93.929	120.864	213.673

Table 6.1: We calculated the depicted values by subtracting the values measured in each frame. Thus we got average distances per frame and not for the entire recording. The "detections amount avg" and "size avg" rows were calculated over the entire recordings. The values depicted preferred to use values in the order of HD_{pc}, LD_{pc}. A graphical representation can be seen in Figure 6.5

- **Per frame detection noise**

This section will present how strong is the noise of our detection noise in one frame.

We have estimated the poses of pallets multiple times on the same frames. The noise differed depending on the distance of the pallets from our camera. We measured the noise of pallets at camera-centre distances around 3 m, 6 m and 8 m. We also tried to see what effect detecting pallets at an angle had. And what was the effect of finding two planes to fit our pallet onto. In Figures 6.10 and 6.11 we can see the difference between us fitting one or two planes. The results can be seen in Table 6.2.

Measured images	close 6.6	medium 6.8	far 6.9	med 45° 1 plane 6.7	med 45° 2 planes 6.7
centre distance [m]	3.46	6.02	8.54	6.03	6.05
x distance average [m]	-1.00	0.39	-0.65	2.11	2.07
x standard deviation [mm]	5.15	22.99	45.96	17.82	12.27
z distance average [m]	3.31	6.00	8.51	5.65	5.69
z standard deviation [mm]	2.58	14.39	21.64	18.00	8.39
angle average [deg.]	-4.47	-3.62	7.55	-46.18	-46.61
angle standard deviation [deg.]	0.53	2.17	4.65	2.73	2.20

Table 6.2: Noise of our pallet pose estimation in different scenes

Figure 6.6: pallet at far distance 8.54 m at an angle of 7.55°

Figure 6.7: pallet at medium distance 6.03 m at an angle of -46.61°

Figure 6.8: pallet at medium distance 6.02 m at an angle of -3.62°

Figure 6.9: pallet at close distance 3.46 m at an angle of -4.47°

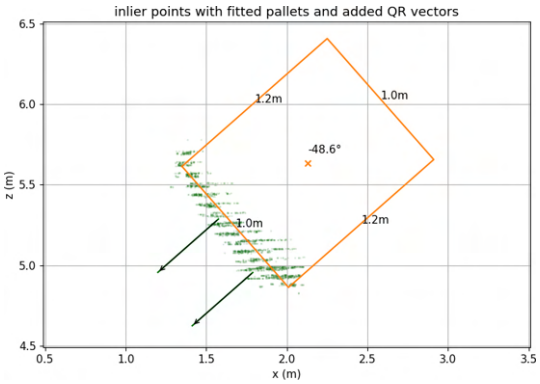


Figure 6.10: Pallet seen in Figure 6.7 with only one plane fitted

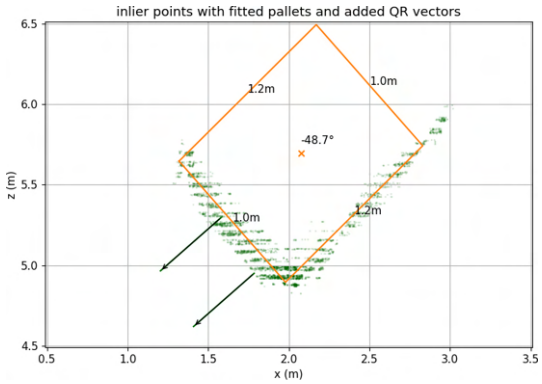


Figure 6.11: Pallet seen in Figure 6.7 with both planes fitted

Simulation performance

Unlike the control module [1], our vision module did not work with the simulation by the end of this thesis. We also were not able to connect the vision and control module in time. But as the simulation was already created we gathered some data from it and tested our algorithm onto it. The results of the scene seen in Figures 6.12 and 6.13 give us an indication that the pallets are fitted onto the depth data fairly precisely. While a noise still persists, it is rather minimal, as we can see in Table 6.3.

As mentioned earlier the simulation was not a focus of ours and other than in this section we did not utilise it.

Measured images	simulation
centre distance [m]	6.28
x distance average [m]	-1.89
x standard deviation [mm]	1.46
z distance average [m]	5.99
z standard deviation [mm]	0.97
angle average [deg.]	19.85
angle standard deviation [deg.]	0.15

Table 6.3: Noise of our pallet pose estimation on simulated data

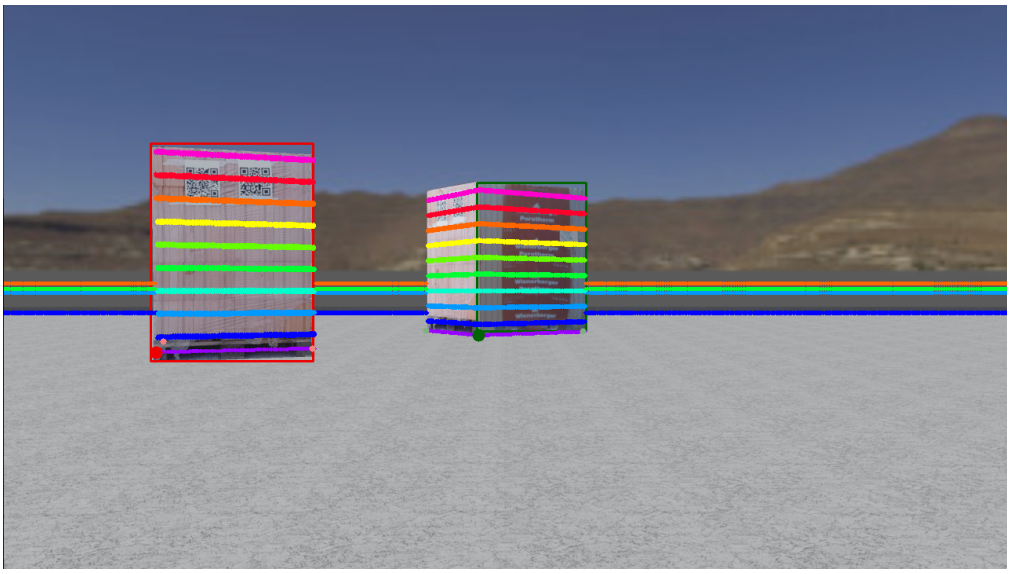


Figure 6.12: The scene of our simulated data (we tracked the red pallet on the left)

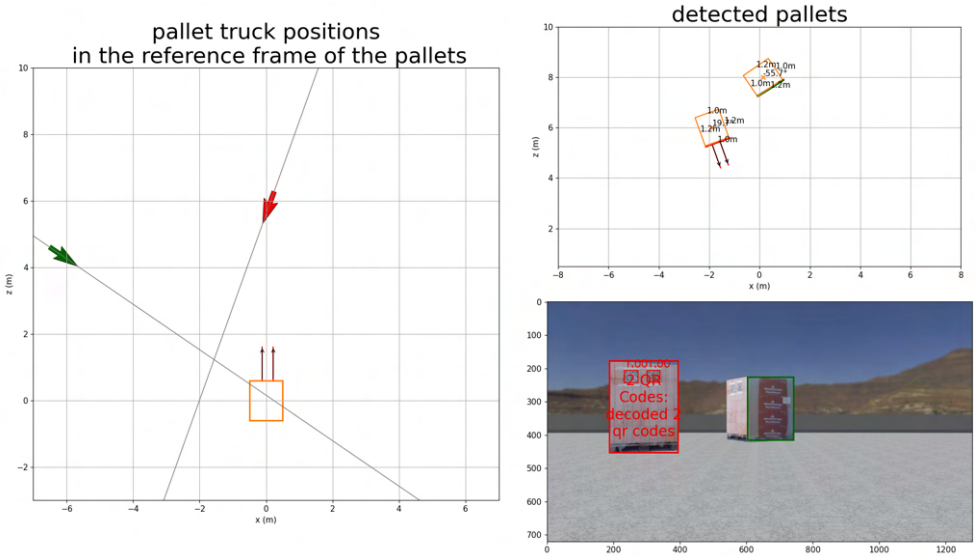


Figure 6.13: The detection on our simulated data

- **Per recording detection noise**

We did not have any ground truth results to compare to. Thus we have chosen to create an approximation. We have applied a rolling average over the last five data points using the following formula:

$$\mathbf{x_{avg}} = \frac{(1,1,1,1,1)}{5} * (x_1, x_2, \mathbf{x}, x_{n-1}, x_n) \tag{6.1}$$

We expanded the original vector $\mathbf{x}$ so that $\mathbf{x_{avg}}$ would keep the same dimensions.

Three evolutions of loading and unloading operations can be seen in Figures 6.16 to 6.18. A view of the scenes can be seen in Figures 6.7 to 6.9.

- **Time**

The results of our time measurements can be seen in following table:

Task	Time
Fitting planes through a ROI	0.0264 sec
Finding pallet pose per ROI	0.044 sec
Tracker system per frame	0.009 226 sec
QR code detection and decoding per ROI	0.18 sec
Calculating refinements per frame	0.0016 sec
Overall time per frame	0.297 sec
HD_oYOLO on the stereo camera	0.28 sec
LD_oYOLO on the stereo camera	0.19 sec
$HD_{pc}YOLO$ on the testing computer[3]	0.007 039 sec

Calculating the QR code detection was not as simple, as the required time is dependent on the size of the used image. A visualisation of the time required can be seen in Figure 6.14, 6.15.

On average one frame took us 0.297 sec to calculate of which QR code detection took 60.4%.

The HD_oYOLO calculated on the stereo camera took another 0.28 sec. And LD_oYOLO 0.19 sec. Thus on average, our vision module had a time delay of either 0.477 sec or 0.567 sec.

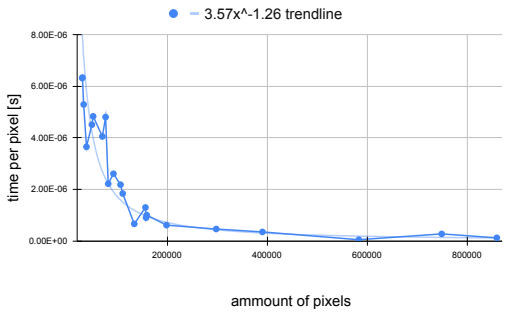


Figure 6.14: Time required for QR code detection per pixel with a fitted trendline

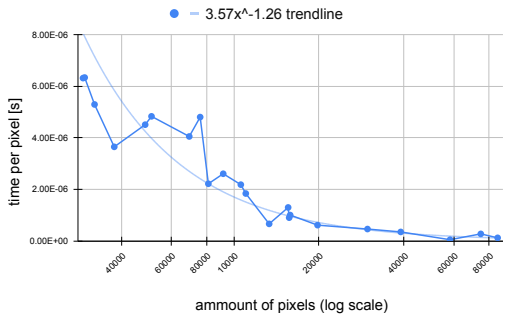


Figure 6.15: Time required for QR code detection per pixel with a fitted trendline where x-axis uses log scale

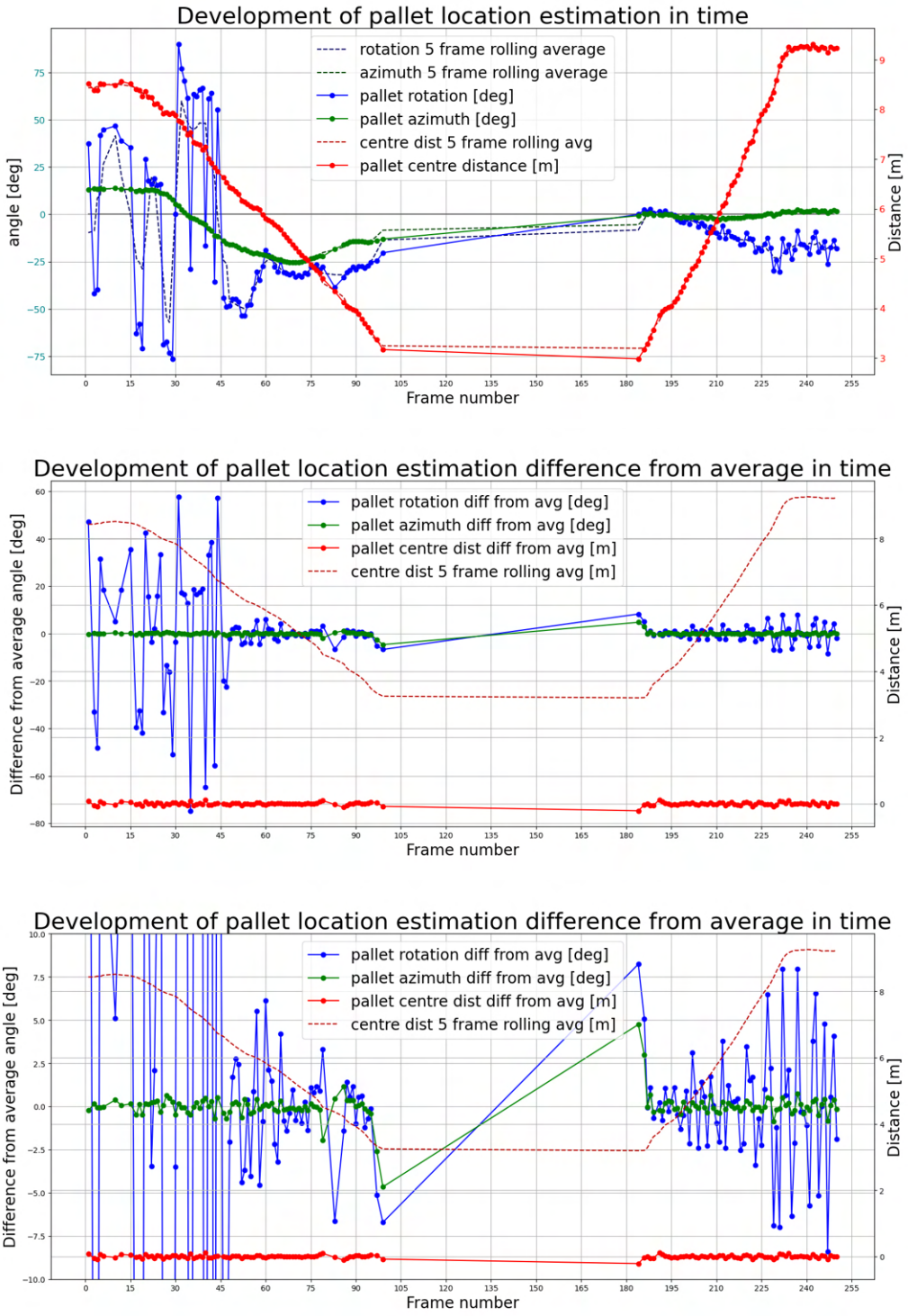


Figure 6.16: Evolution of pose estimations in scene seen in Figure 6.7

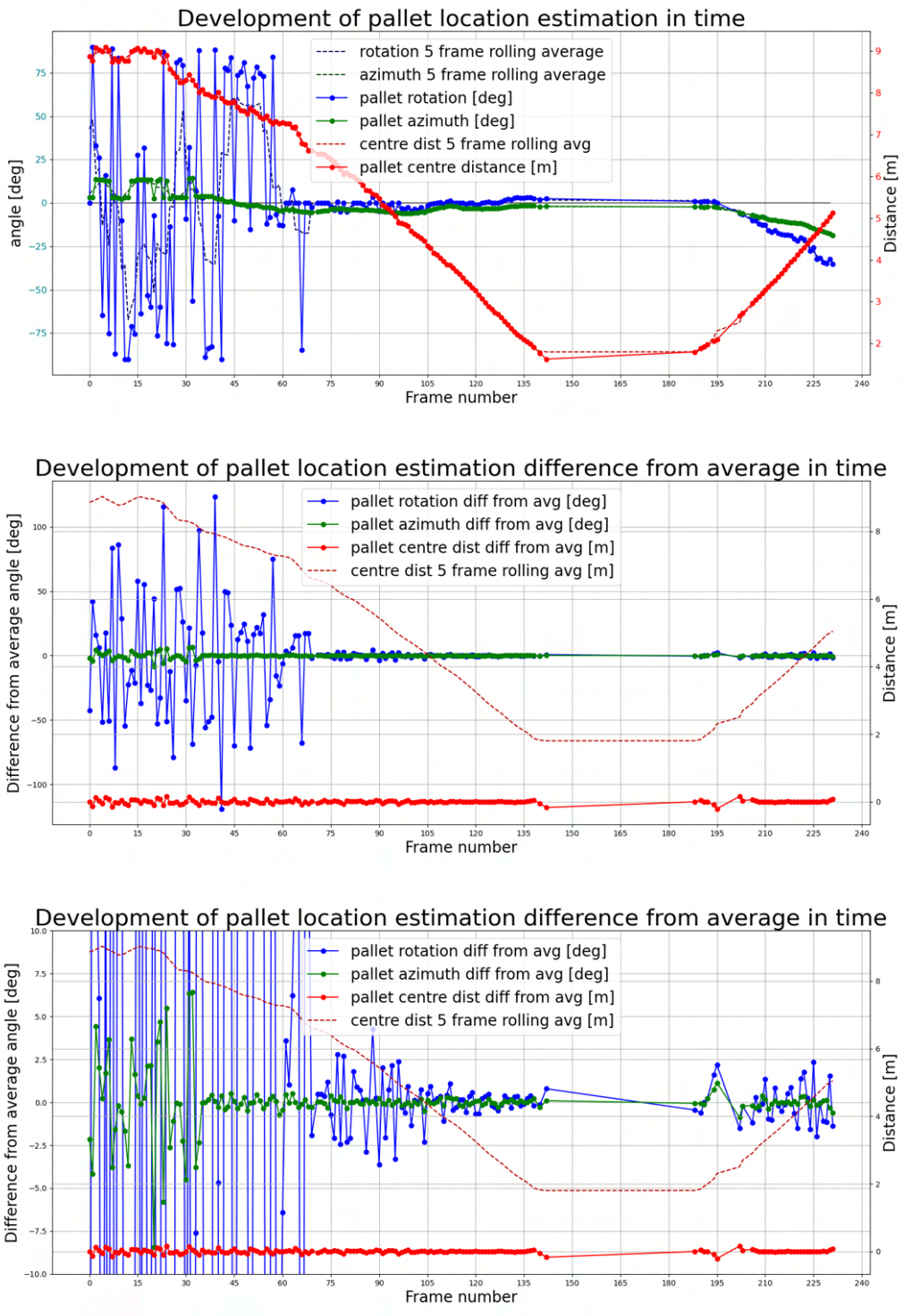


Figure 6.17: Evolution of pose estimations in scene seen in Figure 6.8

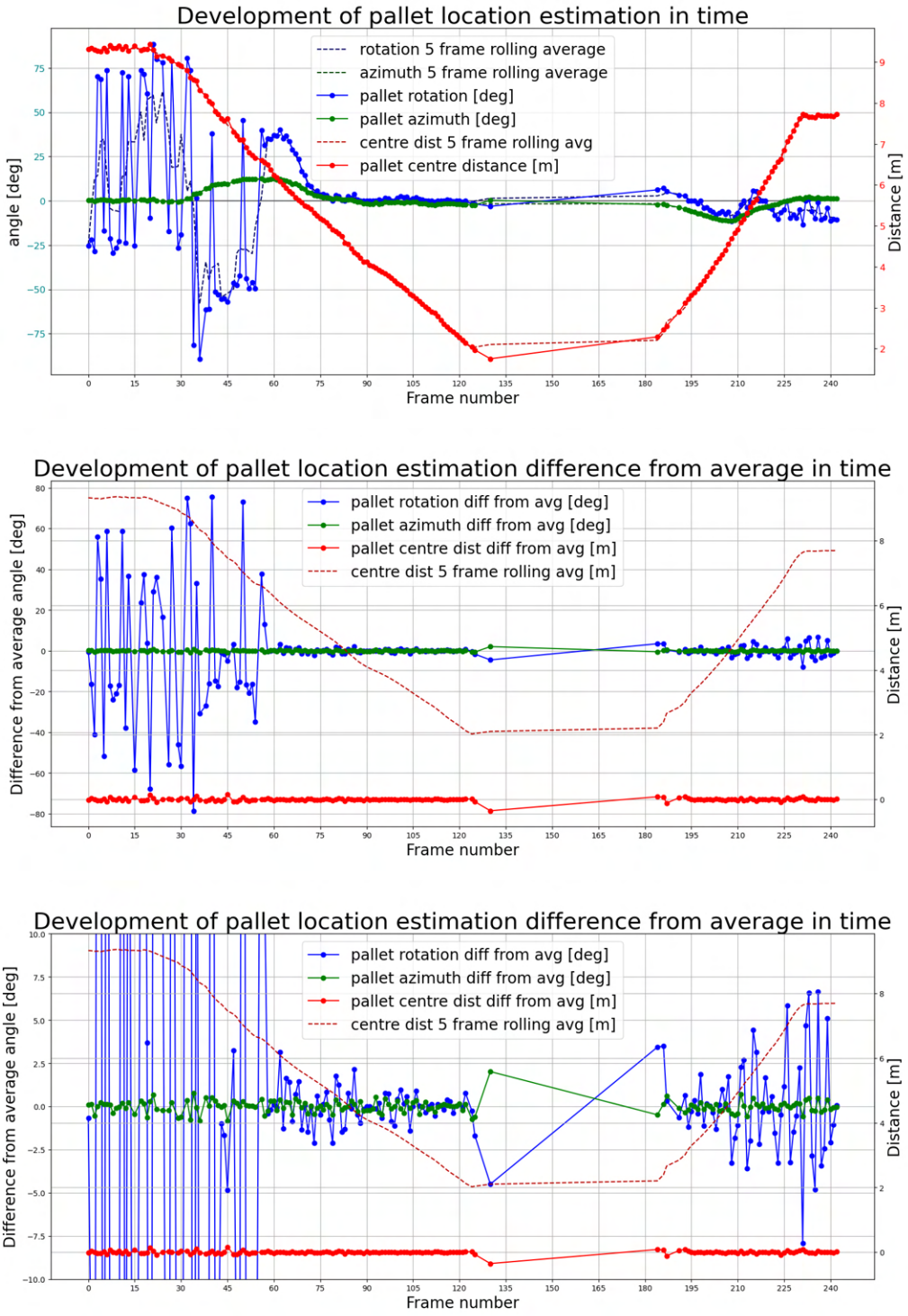


Figure 6.18: Evolution of pose estimations in scene seen in Figure 6.9

6.3 Discussion of Results

Detection

Our trained YOLO NN detected pallet sufficiently well. We have seen a great improvement from the original version we started with, but more training is still required. The bounding boxes still do not precisely fit the pallet edges and when we exposed the NN to a chaotic situation like the one seen in Figure 6.3 it handled itself poorly.

It also surprised us that there was a visible difference between the different versions of YOLO used. The conversion onto the stereo camera possibly created a slight difference in the NN, but all of the versions worked sufficiently well enough anyway.

Per Frame Noise

We see minimal noise in the pallet location estimation, unlike the noise in the pallet angle. The angle was measured only by the RANSAC. Here we ran into the issue of balancing the requirements between accuracy and speed and balanced these issues well.

Per Recording Detection Noise

Our tracker skilfully paired any newly detected pallets to their previous detections. We see that at long distances, before we started refining using QR codes, the pallet rotation angle moves a lot, unsurprisingly by about 90°. And as we get closer to the pallet the deviation from our rolling average quickly decreases. Unlike the pallet rotation both the pallet azimuth angle and centre distance follow a smooth path. We can also see, at close centre distances, we are no longer able to detect the pallets.

Of course, we did not have access to any ground truth data and as the issues raised in Section 5.3 are quite substantial we dare not pronounce even our rolling average detections to be completely accurate. But the measured noise represented in Figures 6.16 to 6.18 is still rather useful.

Time Complexity

As described above our code had to proficiently balance between achieving good accuracy and sufficient speed. The control module described in the sister thesis [1] delves deeper into its delay requirements, but we were able to decrease our time delay to under 0.6 s at 5 fps. The most resource-intensive parts of our solution were the two used NNs. The time they required could be greatly decreased by using more specialised hardware, but this would bring other issues such as higher power consumption, larger cost and weight.

It is our opinion that the performance of the vision module is satisfactory for our given task.

7 Conclusion

In this thesis, we designed a pallet pose estimator that was able to communicate with ROS.

We mounted our chosen sensor, Luxonis OAK-D Pro, onto our development pallet truck. We learned how to control it and communicate with it. We gathered real data at the premises of KM Robotics s.r.o. We retrained a neural network to better detect pallets of bricks in the gathered data. We converted our measured depth data into a point cloud and optimised the fitting of planes onto it. We also improved the fitting of pallets onto the detected planes. We created logic for the utilisation of a tracker and the logic for establishing a track. Additionally, we created many visualisations of our measured data. We incorporated an existing QR code detector and decoder into our code. Finally, we created a pose refiner using the detected QR codes. Thus we have created a working prototype of a pallet pose estimator and converted it into ROS.

Our vision module works sufficiently well for our given task and should allow us to complete the larger task of automatic pallet loading and unloading.

We were not able to unite our vision module together with the control module described in the sister thesis [1]. We plan to test them together during the upcoming summer.

8 References

[1] A. Basl, *Pallet truck trajectory design and control*, 2024-05.

[2] J. Guan, Y. Hao, Q. Wu, S. Li, and Y. Fang, "A survey of 6DoF object pose estimation methods for different application scenarios," *Sensors*, vol. 24, no. 4, 2024, ISSN: 1424-8220. DOI: 10.3390/s24041076. [Online]. Available: https://www.mdpi.com/1424-8220/24/4/1076.

[3] OpenCV. "Camera calibration with OpenCV." (2024), [Online]. Available: https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html (visited on 02/08/2024).

[4] OpenCV. "Detection of ChArUco boards." (2024), [Online]. Available: https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html (visited on 02/10/2024).

[5] Wikipedia contributors, *K-nearest neighbors algorithm — Wikipedia, the free encyclopedia*, [Online; accessed 23-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=1212348037.

[6] J. W. Anders Grunnet-Jepsen John N. Sweetser. "Tuning depth cameras for best performance." (2023), [Online]. Available: https://dev.intelrealsense.com/docs/tuning-depth-cameras-for-best-performance#section-use-sunlight-but-avoid-glare.

[7] BOP. "Datasets." (2023), [Online]. Available: https://bop.felk.cvut.cz/datasets/ (visited on 05/23/2024).

[8] CVUT, *Pallets_and_bricks_2 dataset*, Open Source Dataset, 2023. [Online]. Available: https://universe.roboflow.com/cvut-nnle8/pallets_and_bricks_2 (visited on 02/09/2024).

[9] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[10] KM Robotics.sro and CIIRC. "Internal github repository." Available: https://github.com/mykkro/rovozci-dev, https://github.com/mykkro/rovozci-dev/tree/rgbd-sensor/rovozci_vision. (2023), (visited on 05/24/2024).

[11] LUCID. "Understanding the digital image sensor." (2023), [Online]. Available: https://thinklucid.com/tech-briefs/understanding-digital-image-sensors/ (visited on 05/23/2024).

[12] Luxonis. "Robotics vision core 2 (RVC2)." (2023), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/rvc/rvc2/#rvc2 (visited on 05/07/2024).

[13] V. Petrík. "Robotics: Introduction to perception." (2023), [Online]. Available: https://data.ciirc.cvut.cz/public/projects/2023CTURoboticsData/05_perception_en.pdf (visited on 02/10/2024).

[14] M. Sundermeyer, T. Hodan, Y. Labbe, *et al.*, *Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects*, 2023. arXiv: 2302.13075 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2302.13075.

[15] SuperAnnote. "Image segmentation detailed overview." (2023, updated 2024), [Online]. Available: https://www.superannotate.com/blog/image-segmentation-for-machine-learning (visited on 02/08/2024).

[16] Lenovo. "Lenovo legion Pro 5i (16", 8)." (2022), [Online]. Available: https://psrefstuff.lenovo.com/syspool/Sys/PDF/datasheet/Legion_Pro_5_16IRX8_datasheet_EN.pdf (visited on 05/07/2024).

[17] SICK. "Automation in intralogistics: Perfecting the automated forklift truck." (2022), [Online]. Available: https://www.sick.com/ag/en/sick-sensor-blog/automation-in-intralogistics-perfecting-the-automated-forklift-truck/w/blog-intralogistics-automated-forklift-truck/ (visited on 02/10/2024).

[18] T. Yang, Y. Li, C. Zhao, *et al.*, "3D ToF LiDAR in mobile robotics: A review," *ArXiv*, vol. abs/2202.11025, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247025690.

[19] V. Kilic, D. Hegde, V. Sindagi, A. B. Cooper, M. A. Foster, and V. M. Patel, *Lidar light scattering augmentation (LISA): Physics-based simulation of adverse weather conditions for 3D object detection*, 2021. arXiv: 2107.07004 [cs.CV].

[20] J. A. M. Ulloa, *Investigation into pallet durability throughout the hazards that pallets experience during regular use and handling*, 2021. [Online]. Available: http://hdl.handle.net/10919/108242.

[21] S. Dandrifosse, A. Bouvry, V. Leemans, B. Dumont, and B. Mercatoris, "Imaging wheat canopy through stereo vision: Overcoming the challenges of the laboratory to field transition for morphological features extraction," *Frontiers in Plant Science*, vol. 11, p. 1, Feb. 2020. DOI: 10.3389/fpls.2020.00096.

[22] APC. "User manual Easy UPS." (2019), [Online]. Available: https://download.schneider-electric.com/files?p_File_Name=AHUG-ASV5RQ_R1_EN.pdf&p_Doc_Ref=SPD_AHUG-ASV5RQ_EN&p_enDocType=User+guide (visited on 05/07/2024).

[23] Intel, *Intel® RealSenseTM D400 series product family*, 2019. [Online]. Available: https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf.

[24] Velodyne Lidar, Inc., *Alpha prime datasheet*, 2019. [Online]. Available: https://www.goetting-agv.com/dateien/downloads/VelodyneLidar_AlphaPrime_Datasheet.pdf.

[25] J. Lambert. "Stereo and disparity." (2018-12-27), [Online]. Available: https://johnwlambert.github.io/stereo/ (visited on 05/19/2024).

[26] Quanergy Systems, Inc., *S3-2 solid state lidar sensor datasheet*, 2018. [Online]. Available: https://sitekcaspian.com/static/documents/S3-2NSI-H60%20DataSheet.pdf.

[27] F. Bergamasco. "Computer vision the pinhole camera model." (2016), [Online]. Available: https://www.dsi.unive.it/~bergamasco/teachingfiles/cvslides/11_pinhole_camera_model.pdf (visited on 05/23/2024).

[28] J. Castorena, U. S. Kamilov, and P. T. Boufounos, "Autocalibration of lidar and optical cameras via edge alignment," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2862–2866. DOI: 10.1109/ICASSP.2016.7472200.

[29] D. Lawrence. "Puddle detection." (2016-04), [Online]. Available: https://daniel.lawrence.lu/programming/puddle/ (visited on 01/27/2024).

[0] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV].

[30] P. Sluka, *System for stereo reconstruction of transparent objects using active lighting*, 2015. [Online]. Available: https://is.muni.cz/th/k996k/text_final.pdf.

[31] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[32] Dell, *Dell™ Latitude™ E6440 technical guidebook*, 2013. [Online]. Available: https://i.dell.com/sites/csdocuments/Shared-Content_data-Sheets_Documents/en/au/APJ-Dell-Latitude-E6440-Technical-Guidebook.pdf (visited on 05/07/2024).

[33] L. Barazzetti, L. Mussio, F. Remondino, and M. Scaioni, "Targetless camera calibration," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII-5/W16, Sep. 2012. DOI: 10.5194/isprsarchives-XXXVIII-5-W16-335-2011.

[34] M. Kytö, M. Nuutinen, and P. Oittinen, "Method for measuring stereo camera depth accuracy based on stereoscopic vision," vol. 7864, Jan. 2011, pp. 78640I–1. DOI: 10.1117/12.872015.

[35] G. Reina, J. Underwood, G. Brooker, and H. Durrant-Whyte, "Radar-based perception for autonomous outdoor vehicles," *Journal of Field Robotics*, vol. 28, no. 6, pp. 894–913, 2011. DOI: https://doi.org/10.1002/rob.20393. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20393. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20393.

[36] G. Antonelli, F. Caccavale, F. Grossi, and A. Marino, "Simultaneous calibration of odometry and camera for a differential drive mobile robot," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 5417–5422. DOI: 10.1109/ROBOT.2010.5509954. [Online]. Available: https://ieeexplore.ieee.org/document/5509954.

[37] G. C. S. Nalpantidis Lazaros and A. Gasteratos, "Review of stereo vision algorithms: From software to hardware," *International Journal of Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008. DOI: 10.1080/15599610802438680. eprint: https://doi.org/10.1080/15599610802438680. [Online]. Available: https://doi.org/10.1080/15599610802438680.

[38] botland. "RPLidar A2M12." (n.d), [Online]. Available: https://botland.cz/laserove-skenery/22352-laserovy-skener-rplidar-a2m12-360-stupnu-12m-6959420910274.html (visited on 05/24/2024).

[39] E. Cañas. "Qreader." (n.d.), [Online]. Available: https://github.com/Eric-Canas/qreader (visited on 05/10/2024).

[40] DW Forklift. "PTE20N." (n.d.), [Online]. Available: https://www.dwforklift.cz/elektricky-paletovy-vozik-pte20n/ (visited on 05/07/2024).

[41] Global GPS Systems. "LS Lidar C32W." (n.d), [Online]. Available: https://globalgpssystems.com/ls-lidar-c32w/ (visited on 05/24/2024).

[42] GPS.gov. "GPS accuracy." (n.d.), [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/ (visited on 05/23/2024).

[43] KM-Robotics. "Construction robot." (n.d.), [Online]. Available: https://www.km-robotics.cz/en/construction-robot/ (visited on 05/07/2024).

[44] Luxonis. "Configuring stereo depth." (n.d.), [Online]. Available: https://docs.luxonis.com/projects/api/en/latest/tutorials/configuring-stereo-depth/#depth-from-disparity (visited on 05/19/2024).

[45] Luxonis. "Depth accuracy." (n.d.), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/guides/depth_accuracy/#p-75mm-baseline-distance-oaks (visited on 05/13/2024).

[46] Luxonis. "Depthai-ros." (n.d.), [Online]. Available: https://github.com/luxonis/depthai-ros/tree/humble/depthai_ros_driver/config (visited on 05/07/2024).

[47] Luxonis. "OAK-D Pro." (n.d.), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/DM9098pro/ (visited on 05/07/2024).

[48] L. Mariga, T. Karella, and S. Jungerman. "pyRANSAC-3D." (n.d.), [Online]. Available: https://github.com/leomariga/pyRANSAC-3D/tree/master (visited on 05/13/2024).

[49] W. Muroń and M. Budyś. "Motpy." (n.d.), [Online]. Available: https://github.com/wmuron/motpy (visited on 05/10/2024).

[50] Open Robotics. "Understanding actions." (n.d.), [Online]. Available: https://docs.ros.org/en/iron/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html (visited on 05/18/2024).

[51] rfidhy.com. "How AGV car achieves intelligent RFID positioning?" (n.d.), [Online]. Available: https://www.rfidhy.com/how-agv-car-achieves-intelligent-rfid-positioning/ (visited on 02/10/2024).

[52] roboflow. (n.d.), [Online]. Available: https://roboflow.com/ (visited on 05/10/2024).

[53] Webster Griffin. “Agv robot fork truck - faq’s.” (n.d.), [Online]. Available: https://www.webstergriffin.com/frequently-asked-questions-agv/ (visited on 05/23/2024).