Development of a method for detecting the orientation of parcels in container unloading processes

Mahdi Islam

Matriculation Number: 3168896

Faculty: Mathematics/Computer Science (FB 03)

Digital Media (Media Informatics)

November 1, 2021

IPS Robotics and Automation department

Primary Supervisor

Prof. Dr Hans-Jörg Kreowski

University of Bremen, Department of Computer Science

D-28334 Bremen

Tel.: +49 421 218 64451

email: kreo@uni-bremen.de

Secondary Supervisor

Prof. Dr.-Ing. Thies Beinke

BIBA - Bremen Institute for Production and Logistics GmbH

Hochschulring 20, 28359 Bremen

phone +49 (0) 421 / 218-50086

e-mail ben@biba.uni-bremen.de

Advisor

Christoph Petzoldt

BIBA - Bremen Institute for Production and Logistics GmbH

Hochschulring 20, 28359 Bremen

phone: +49 (0) 421 / 218-50119

e-mail: ptz@biba.uni-bremen.de

Universität Bremen

Name: Mahdi Islam Matriculation number: 3168896

Declaration of Authorship

Hereby I declare that my Master's Thesis was written without external support and that I did not use any other sources and auxiliary means than those quoted. All statements which are literally or analogously taken from other publications have been identified as quotations.

Declaration with regard to publishing theses

Two years after the final degree, the thesis will be submitted to the University of Bremen archive and stored there permanently. Storage includes:

- Master's Theses with a local or regional reference, as well as 10% of all theses from every subject and year.
- Bachelor's Theses: The first and last Bachelor degrees from every subject and year.
- I agree that for research purposes third parties can look into my thesis stored in the University archive.
- I agree that for research purposes third parties can look into my thesis stored in the University archive after a period of 30 years (in line with §7 para. 2 BremArchivG).
- I do not agree that for research purposes third parties can look into my thesis stored in the University archive.

....

Place, Date Signature

Abstract

On the conveyor belt of an automatic container unloading system, orientation recognition processes must be rapid and accurate in order to retrieve information about the orientation of individual packages. In addition, it is critical to recognize the effects of environmental and parcel-related factors to effectively detect it.

This thesis suggests an algorithm to detect parcel orientation on a conveyor belt in real time, using fast and simple techniques. This algorithm has two stages for the detection process. To obtain parcel orientation results, the suggested technique first identifies the object using a trained YOLOv3 model and then calculates the orientation features with the detection of contours of target features.

For parcel detection on the conveyor belt, the existing YOLOv3 model is trained using both this thesis's own and open source parcel data. The contour-based technique is used to calculate the angle of the parcel's top surface and find the location and direction of arrow marks on parcels to identify the orientation.

This thesis provides consistent orientation information of parcels through real-time detection of parcels by using the rotation angle of parcels and the detection arrow with it's direction. Although the contour-based technique's result shows lower accuracy rate while detecting arrow markers, the detection of parcels model and contour-based angle calculation provides better accuracy rate for detecting parcel orientation.

The orientation detection process, as illustrated in this thesis, is affected by ambient and parcel factors such as lighting, parcel colour, and distance between parcels. It is evidenced by the evaluation of this thesis's algorithm, The bright ambient light improves the accuracy of the orientation outcome. The suggested system detected parcels with a 94% accuracy,detected angle rotation of parcels with a 92% accuracy, and detected arrow marks with a 50% accuracy. The overall accuracy of the method proposed in this thesis is 79%. The orientation detection technique is affected by the lack of arrow indicators in small and big size boxes, as well as the lack of a handle mechanism.

Acknowledgements

When I think back on my academic career, I'm grateful for how far I've come. I'd like to thank Christoph Petzoldt (Head of the IPS Robotics and Automation department) from BIBA - Bremen Institute for Production and Logistics GmbH for his assistance. His door was always open if I had an issue or a question about my study. I'd also want to thank my supervisor, Prof. Dr. Hans-Jörg Kreowski of the University of Bremen's Department of Computer Science, for taking the time to supervise this thesis. Prof. Dr.-Ing. Thies Beinke, scientific staff of the IPS department Robotics and Automation at BIBA - Bremen Institute for Production and Logistics GmbH, has agreed to be my second supervisor and has provided me with constant assistance and direction throughout the composition of this thesis. When he felt I needed it, he offered his assistance and pointed me in the right direction. I thank and extend my gratitude to Lennart Rolfs (Research Associate in the IPS Robotics and Automation Department) at BIBA - Bremen Institute for Production and Logistics GmbH for his assistance with the field evaluation of this study.

Finally, I am eternally grateful and undeniably appreciative of my parents for their unwavering support and encouragement during my years of education, as well as during the research and writing of this thesis.

Contents

List of Figures

List of Tables

1 Introduction

In the realm of image processing, as well as for machines, detecting orientation angles is a difficult issue. Although some current cameras with inertial sensors can adjust image orientation in 90-degree steps, this feature is rarely used [14]. The orientation detection on the conveyor belt is required to know the parcels characteristics to automate the container unloading system. The orientation angle of objects in an image is easy to understand for humans, but an image is a matrix with pixel values for a computer or tool. Many researchers proposed various deep learning methods for detecting object orientation. The thesis provided an algorithm for detecting angle orientation that is quick and simple. The environment has an impact on orienting processes, as this thesis demonstrates. The inspiration for this study, the research question, the scope and limitations of this article, and the outline for this thesis paper are all presented in the following subsections.

1.1 Motivation

Unloading parcels would be significantly more efficient and trouble-free if the operation in the container unloading system was automated. One of the most important industrial operations in automated unloading systems is conveyor belt systems. One of the most challenging phases in this system after unloading the items is detecting the orientation of the parcels. Based on this identification, the automated unloading system can proceed further. Vehicles that transport parcels (piece items) in the parcels-, courier-, and express market segments are manually unloaded due to the current state of technology. Piece items can be found arranged or piled inside containers or carriers, as well as in a chaotic state [15].

The growth of online businesses and markets has increased the volume of B2C products that are delivered directly to customers, by passing the old-fashioned point of sale, where people go to the location and pick the necessary products from the shelves. As a direct consequence, the Courier Express Parcels(CEP) branch distribution centers - with players such as DHL, FedEx, UPS, Hermes, DPD - are overwhelmed by the ever growing stream of parcels that were not expected at the time of distribution. Therefore, automation in the unloading system is required after evaluating the level of automation

of the CEP branch distribution center [16]. The automatic unloading system also has the following advantages:

- Enhanced process efficiency
- Cost efficiency
- improved workplace safety
- versatile systems

In a variety of industrial and commercial scenarios, unsorted parts are often removed from boxes, luggages are unloaded from containers at airports, and shipments are handled in a rather less efficient way. The system that is currently in use for such events is unaware of the position, orientation, shape, and/or weight of the objects that are about to be unloaded [17]. Since larger packages would not fit into the current associated conveyor systems, detecting the orientation of unloaded parcels on conveyor systems is a crucial need for attaining a higher degree of automation in autonomous unloading systems. Because of the necessity of scanning the barcode, item identification and subsequent sorting of the parcels in a correct and efficient manner, the orientation of the packages is necessary to be identified in order to achieve the highest efficiency of this process. Object identification for cargo unloading is a job that is a combination of two things, object recognization and pose estimation. The recognition part has to be done at the instance level with the estimation of each object's pose relative to the sensor [18].

To successfully automate the procedure, the robotic system must recognize the items as well as the various contexts in which they are identified. It is necessary to practice successful identification, unloading, and prudence to avoid injury to the packages. The automated system for unloading items from a container includes robots, a gripping mechanism, and an object detection unit [19].

In such an environment, detection of orientation of the parcels will help industrial applications to create a user-friendly environment by reducing the demand for menial labour. As a result, a reliable orientation detection approach is required to improve the automation process and expand the use of robotic systems for unloading containers. It could therefore result in favourable company growth in such industries as it would save time and improve the efficiency of the current practices.

1.2 Research Question and Objectives

The goal of this thesis is to identify the orientation of the parcels on the conveyor belt in a container unloading system. This challenge has sparked a lot of interest in the autonomous unloading system research community because of the possibility of lower costs and faster unloading times. Due to the obvious conveyor belt speed, it is necessary to detect the parcels' orientation quickly which poses an interesting challenge to the research and development of this technology. Many researchers have presented methods for detecting object orientation, however, the majority of them are sophisticated processes with a large latency.
Aside from that, the unloading system's characteristics have an impact on the orientation detecting processes. The identification of the influencing factors is another purpose of this study. In light of these issues, the following research questions were formulated in this thesis:

- How to extract information regarding the rotation of logistic packages in a real-time autonomous container unloading system using a combination of deep learning and statistical aspects of parcels?
- Is there any effect on the orientation detection processes due to parcel and environmental characteristics?

To make the orientation operations fast and precise, the algorithm first detects parcels on the conveyor belt and then extracts parcel orientation based on statistical information of parcels such as contour, bounding rectangle, and segmentation of detection parcels. The algorithm's evaluation is based on several package and unloading system parameters. This thesis attempts to find the parameters that affect the orientation process by assessing them in various test environments. The detection of the orientation process might improve and deliver more accurate orientation detection by creating necessary knowledge on various parameters that impacts the process efficiency.

1.3 Scope and Limitation

Due to the limited time frame, some constraints must be imposed on this research to ensure that the work is completed on time. To find out the answers to the research questions within this thesis's scope, more focus is put on the topic of detection of parcels and extraction of information regarding its orientation. It is important to explore the current literature regarding the identification and rotation of parcels to understand and extract the existing knowledge on the topic. The state of the art methods and processes are necessary to be studied to successfully pull out parcel rotation data. Besides, researchers have implemented various test methods to their algorithms the results are necessary to identify the effect of various parameters in the orientation processes. To know the performance of the algorithm proposed in this thesis, performing adequate tests is also important for this work. The following components mainly focus on the scope of this work:

- The state-of-the-art chapter will primarily focus on Opencv and deep learning approaches for detecting object orientation
- The conceptuation chapter will analyze the proposed design and attempt to obtain features for this thesis method
- The primary focus of the implementation and integration chapter will be on developing a method to detect the orientation of parcels using dual camera technology and integrating it into a real-world unloading system
- The evaluation and results of the method implemented in various environments and lighting conditions will be observed, and the limitations and future development process will be discussed in the end of this thesis

There is no such thing as a perfect study or one that covers all potential aspects. As a result, increased emphasis was placed in the study on developing quick and simple techniques for detecting the orientation of items on the conveyor belt. The purpose of this thesis is to focus on the orientation detection procedures rather than the camera selection. To save time and to avoid the absence of training data, this thesis employed a pre-trained model in order to train the system with existing data acquired from this model. The visualization of orientation outcomes, such as graphical user interface (GUI), will be less of an emphasis. However, a minimum viable graphical user interface

(GUI) for observing the orientation detection in a real-time container unloading system is provided.

1.4 Methodology and Structure of Thesis

This thesis proposes an algorithm for detecting parcel orientation on the conveyor belt in the unloading system using a design science research approach. This research tries to find an answer to the stated research topics by constructing an algorithm. Design science research (DSR) is the development of a solution to a research challenge and subsequent evaluation of that solution's performance [20].

Phase	Section	Description
Definition	Theoretical Background	This chapter explore the theoretical foundations of this study with a focus on image processing knowledge discovery, machine and deep learning approaches, and OpenCV methodologies.
Identification of the problem	State of the Art	This thesis will describe a pool of possible processes that are relevant to the thesis's topics with finding the problems.
Design and Development	Conceptuation	This chapter will go over design identification, analysis, interpretation, feature extraction, and evaluation.
Demonstration	Implementation and Integration	Following the design selection, this the chapter will go over how to construct and implement the parcel detection design.
Evaluation	Evaluations and Results	This chapter will present the results of the algorithm suggested in this thesis and will explore how the the evaluation process is affected by various parameters linked to parcels and the environment of an automatic container unloading system.
Communication	Discussion and Conclusion	This chapter will go over how to interpret the results, the limitations of the thesis algorithm, and what has to be done in the future to improve the suggested algorithm.

Table 1.1: The methodology of the thesis paper. This thesis chapter covers which phases are presented here to acquire a general summary of each chapter.

The process of design science research (DSR) begins with a problem that is typically application-oriented. This method is common in research disciplines with a strong application-oriented focus. This approach, on the other hand, allows for learning through construction. Peffers (2007) proposed six steps to the design science research (DSR) process, stating that researchers don't have to start from the first step (i.e. identification), but instead should go through all of the processes in some way, working

outward from the research [21]. According to the design science research (DSR) approach, the Table 1.1 outlines the paper in terms of phase, section, and description. Identification of an issue with motivation is a step in the design science research (DSR) process. The motivation for this thesis research is discussed in the introduction portion of this thesis. This thesis identifies the problem and states the research questions and objectives in the introductory part through a review of literature in the state of the art section. The next step is to determine the solution's goals. From the problem definition and knowledge of what is achievable and feasible, infer the goals of a solution. The theoretical background component of this thesis discusses the relative theory of object detection, which aids in the development of an algorithm for detecting parcel orientation on the conveyor belt in the unloading system.

In the section of conceptuation, which is another step of the design science research (DSR) process as the phase of design and development, this thesis discusses the concept of developing an algorithm for orientation detection using a combination of deep learning models and statistical analysis of parcels. The demonstration phase of design science research (DSR) is when the produced notion is used to solve an issue. This thesis discusses each stage of the generated algorithm to detect the orientation of parcels on the conveyor belt in the unloading system in the implementation portion. Another step in the design science research (DSR) process is to evaluate the solution. The evaluation portion explores the evolution of the implemented algorithm, which is divided between lab and field tests. This study also identifies the effects of environmental factors on the orientation detecting process through examination. The problem, the artefact, and its utility are all communicated in the final step. In the discussion and conclusion sections of this research, the findings of the algorithm are discussed, as well as a future study.

2 Theoretical Background

This chapter describes the basic concepts and related theoretical frameworks which has been used in this thesis work. In the first section of this chapter, the fundamentals of image processing have been discussed. In later sections, the basics of machine learning, neural network, and deep learning are explained. Following this, technical documentation on Opencv and Pyrealsense2 is provided for onboarding the readers into the depth of this research work.

2.1 Knowledge discovery in Image Processing

Image processing introduces additional processes in the field of image analysis. In the context of a signal processing algorithm, this depicts all images as 2d signals. The actual work of image processing is to convert an image to a digital format and retrieve the image's contents. Image processing can be divided into several categories. One type of image processing is finding items in the frame. The objects are detected via rearrangement image processing. Sharpening and restoration are two methods of image processing that can be used to upgrade an image. Image processing can also be used to locate the noise of nearby objects.

There are fundamental steps in image processing which is used to feature to analyse the image. Image acquisition is one of the steps that involve retrieving the image from a source, usually a hardware-based source. Segmentation is another way of partitioning an image into its constituent parts or objects. Image segmentation is the most difficult step among the processes es of image [22]. There is also the application of image processing which takes the automation technology to the next level.

Digital image processing is one of the applications that has grown more cost-effective in a variety of domains, including signature recognition, iris recognition, and face recognition, forensics, automotive detection, and military applications [23]. Face detection, image reconstruction, medical image retrieval, and other techniques are also used. Image processing aids in the enhancement of images for human interpretation. Image processing plays a larger role in increasing and decreasing image size. Image processing is used to compress and decompress images for speedier transmission over the network.

2.2 Machine Learning

The machine is learning from input data and then predicting the object in the image frame is the main work of machine learning. This is a type of artificial intelligence (AI). Machine learning helps in many applications such as pattern recognition, computer vision, spacecraft engineering and so on. Three several approaches exist for object detection with machine learning. Viola-jones algorithm, Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) are among them.

Viola-Jones algorithm

This algorithm is designed to detect the object in the vying real-time environment. Robust, real-time and face detection are properties of this algorithm that make it a strong detection algorithm. The true positive rate is high in contrast to the false-positive rate. For the practical information, this algorithm processed at least two frames per second. Besides that this algorithm also differentiates between face and non-face in the frame [24].

SIFT (Scale-Invariant Feature Transform)

SIFT is a fantastic feature detection method for computer vision that can be used to find and describe any local features in images. Locality, distinctiveness, quantity, efficiency, and extensibility are the main advantages of shift. Image stitching, navigation and robotic mapping, object recognition, gesture recognition, 3D modelling, individual wildlife identification, match movement, and video tracking are just some of the applications for SIFT [25].

HOG (Histogram of Oriented Gradients)

This is a feature descriptor and is used in computer vision to detect the object. This is also quite similar to Scale Invariant Feature Transformation (SIFT) and the main technique is to count the image of gradient orientation of localize portion of image [26].

2.3 Neural Network

Neural networks are a popular machine learning paradigm for image identification, audio recognition, and natural language processing. It reflects the human brain's behaviour. The method is based on how actual neurons communicate with one another where the people brain holds roughly 100 billion neurons that are all active at the same

time. Artificial neurons are mathematical functions that are implemented on computers that are more or less serial. The majority of neural network research is influenced by advances in engineering and mathematics rather than biology. Training data is used by neural networks to learn and increase their accuracy over time. It can be divided into several types, each of which serves a different purpose.

- Feed-Forward Neural Network
- Radial Basis Function (RBF) Neural Network
- Multilayer Perceptrontion
- Convolutional Neural Network box
- Modular Neural Network

Convolution Object detection is done using a neural network. A convolutional neural network has several layers, including an input layer, at least one hidden layer, and an output layer. They're great for distinguishing patterns like edges (vertical/horizontal), forms, colours, and textures in object detection. In this sort of neural network, the hidden layers are convolutional layers that operate as a filter, receiving input, transforming it using a specific pattern, and sending it to the next layer. When there are additional convolutional layers, they are altered in different ways each time a new input is supplied to the next convolutional layer. For example, the first convolutional layer may identify shape/colour in a region (i.e. brown), the second convolutional layer may be able to determine what object it is (i.e. an ear or paw), and the last convolutional layer may classify the object as a dog. Essentially, the more layers the data passes through, the more intricate patterns future ones will be able to detect [27].

2.4 Deep Learning

Deep Learning is a Machine Learning sub-field influenced by neural networks. It differs from traditional machine learning in the kind of data it uses and the learning methods it employs. Deep neural networks are made up of numerous layers of interconnected nodes, each of which improves and refines the prediction or categorization. Forward propagation refers to the progression of calculations via the network. The visible layers

of a deep neural network are the input and output layers. The deep learning model ingests the data for processing in the input layer, and the final prediction or classification is performed in the output layer.
Back-propagation is a method of training a model that uses methods such as gradient descent to calculate prediction errors and then modifies the weights and biases of the function by travelling backwards through the layers. Forward propagation and back-propagation work together to allow a neural network to make predictions and fix any errors. The algorithm improves in accuracy as time goes on.
The method 'recognition-by-components' was used in the early days of objection detection. The features that were taken into consideration were distance transforms, shape contexts, and edge-less, among others. This method enabled object detection as a measurement of similarity between the object components, shapes, and contours, and the features that were taken into consideration were distance transforms, shape contexts, and edge-less, among others. Things didn't proceed as planned, thus machine detecting methods were introduced to help solve the situation. Objects of "various sizes" and "different aspect ratios" were to be taken into account when multi-scale detection was to be done. During the early stages of object detection, this was one of the most difficult technical hurdles. However, the problem was rectified after 2014, as a result of increased technological breakthroughs. This led to the second step of object detection, where deep learning was used to complete the tasks [28].

2.5 OpenCV technical Documentation

OpenCV is a big open-source library for computer vision, machine learning, and image processing that is currently used in real-time operations. It can recognize objects, faces, and even human handwriting in photographs and movies. When Python is used in conjunction with other modules like NumPy, it may analyze the OpenCV array structure. To recognize visual patterns and their numerous features, we use vector space and perform mathematical operations on these features. OpenCV is used to solve a variety of problems, some of which are described below [29].

- face recognition
- Automated inspection and surveillance

- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anamoly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

Opencv also has a wide range of capabilities. Object detection, feature detection, geometry-based monocular, CUDA acceleration and computational photography are among them which is this library capable to perform.

2.6 Pyrealsense2

Pyrealsense2 is a Python package that allows your smartphone to communicate with a RealSense camera. Depth and colour streaming, as well as intrinsic and extrinsic calibration, are all supported by the SDK. For streaming sessions, there are additionally synthetic streams (point clouds with depth aligned to colour and vice versa) and built-in recording and playback capabilities. Streaming Depth, displaying depth and colour with OpenCV and Numpy, align and background removal, coordinate system, tensorFlow Machine Learning, and reading bag files are just a few of pyrealsense2's features [30].

3 State of The Art

Detecting the orientation of parcels is not so easy because of its fewer number data and different types of adjustment. Still, this thesis found related work which helped this research to gain its aim. The state of the art is present here by splitting into different methods to detect the orientations of objects.

3.1 Template Matching

One of the common approaches for successfully detecting the object's orientation is template matching. Template matching is an image processing technique for locating tiny portions of a picture that match a template image. Within the template matching process, there are features and template-based techniques. Occlusion, detection of non-rigid transformations, lighting and backdrop changes, background clutter, and scale variations are the primary problems in the template matching task [31]. Calculating scale, rotation, and location of image characteristics within two pictures is the primary approach for detecting the process's object. It also comes with the following advantages:

- Data does not need to be annotated (a time-consuming and mandatory task to train neural networks).
- Bounding boxes provide more precise results.
- There is no requirement for a GPU.
- Python has its API.

The template slides pixel by pixel on the provided picture using OpenCV template matching. A similarity measure is computed between the template picture and the part of the image it recovers for each point [32]. For translation, rotation, and scale-invariant picture registration, there is an FFT-based method. To determine the scale and rotational movement, Fourier scaling and rotational characteristics are utilized. The transnational movement is determined using the phase correlation approach. The Fourier domain technique is used in this registration method to match pictures that have been translated, rotated, and scaled concerning one another. Algorithms in the

frequency domain (such as the fast Fourier transform (FFT)-a based method employed here) are divided into four groups. These algorithms include those that work with pixel values directly as well as those that work with low-level characteristics like edges and corners [33]. For many years, scientists have studied FFT-based methods to picture registration. Using specific features of the Fourier transform, there is a procedure known as phase correction [34].

Edge detection is an important characteristic for detecting the object's orientation. Another technique proposed by LEE and Xiu to increase rotation invariance of the mean absolute difference method is to use the edge feature. They utilized a template matching method in that approach, which entails assembling the template picture and object image into a circle structure [35].

There is also an implementation based on those papers to detect the object's orientation. One method is to use the **`imreg_dft`** python library. The Fourier transform is used here. Given two pictures, this technique can calculate the difference in size, rotation, and location of image features [36]. Additionally, there is **`CV_TM_SQDIFF_NORMED`** implementation for detecting many objects on a scene and finding angles rotation linked to the axis. During the procedure, images were initially obtained from cameras, and then template matching was attempted using the template picture [37]. Though the template matching technique detects the item, it also has the following drawbacks:

- Limited feature to extract with python library.
- Need a lot of data.
- Failed to get the accurate orientation of the object.
- Complicated computation.
- Complicated computation.
- Susceptible to noise interference.
- Need to try several similarity metrics.

3.2 Three Dimension Object Detection and Orientation Estimate (Monocular)

Detection of the object's orientation is likewise feasible in two phases. The first step is to identify objects in the environment, and the second is to extract features from the detected objects in order to assess acclimation. Several researchers have proposed 3d object object as a method for object identification. Three-dimensional object identification may be done directly from a 3D image captured by an Intel RealSense camera, or it can be done monocularly. The following are some of the benefits of three-dimensional object identification and orientation estimation:

- A local image patch can be used to estimate the local (allocentric) orientation.
- With a perspective view, identify 3D orientation.
- It is possible to transform the local orientation to the global orientation using camera intrinsics (principal point, focal length), as well as global information from the picture patch.

Liu and Lu presented in the literature a deep fitting degree scoring network for monocular 3D object identification. They used an anchor-based technique to regress the object's size and orientation. By simply examining the geographical overlap between suggestions and the item, the optimum candidate can be identified [38]. In 2017, Arsalan Mousavian presented a method for detecting 3D objects and estimating their posture from a single picture. The approach employs a new hybrid discrete-continuous loss that outperforms the L2 loss substantially. The KITTI object detection benchmark is used to assess this approach in this proposal [1]. This paper's Figure 3.1 depicts how they determine the object's local orientation. In the left of this figure s car dimension and right is Illustration of local orientation. The local orientation is computed with respect to the ray that goes through the center of the crop. The center ray of the crop is indicated by the blue arrow. Note that the center of crop may not go through the actual center of the object. Orientation of the car ϑ is equal to ϑray+ϑl. The network is trained to estimate the local orientation ϑl [1].

For stereo, optical flow, visual odometry / SLAM, and 3D object identification, the Kitti team has created new computer vision benchmarks. Four high-resolution video cam-

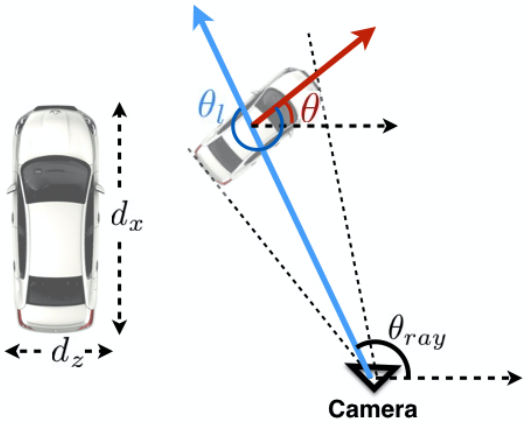

Figure 3.1: Calculated local and global orientation [1]. The local orientation is ϑl.

eras, a Velodyne laser scanner, and a cutting-edge localization system are all part of their platform. This 3D object benchmark focuses on object recognition and 3D orientation estimation using computer vision methods. The number of non-occluded items in the picture, as well as the entropy of the object orientation distribution, are used to determine the orientation estimation benchmark [39].

Another researcher has developed a quick inverse-graphics framework for analyzing 3D scenes. Their technique generates a compact 3D representation that may be utilized in applications such as autonomous driving. They build a deep neural network to learn to map picture areas to every object instances' complete 3D form and position. Orientation is predicted on top of a ROI feature map in the presented technique, which represents object orientation from perspective. The relative camera orientation angles with the camera always pointing towards the center of the object are referred to as viewpoint[40].

In addition, 3D detection prediction was implemented in an Autopilot scenario with a monocular RGB picture, using a faster RCNN as the basemodel and ResNet as the backbone with Python. They utilize Kitti 2d objects as a data set and monoculr RGB picture as input; 2D boxes, dimension, orientation, and placement of objects; inner and outside camera. They forecast the alpha directly for orientation detection, and most of the procedures are based on [1] Arsalan Mousavian's approach. Though it can detect orientation of the object from both a local and global perspective, it has the following flaws:

- Hard to implement because of complex system.

- Need large data to train the model.
- Will be time cost in real time detection.
- Can not detect direction of box.
- Need to add a training model and so need GPU.

3.3 Binary Classifiers

The task of classification necessitates the usage of machine learning techniques. The field of machine learning is concerned with algorithms that learn from previous experiences. In machine learning, there are many distinct sorts of categorization problems to be encountered, as well as specific modeling techniques to be utilized for each [41].The benefits of using a different approach of binary classification to determine the orientation of packages on a conveyor belt were as follows:

- Object detection may be used to extract a variety of features.
- Determine the object's orientation on the conveyor belt.
- Multiple objects can be detected at the same time.

On the 27th International Symposium on Automation and Robotics in Construction (ISARC 2010), Omar Arif and Matt Marshall presents metzhod of tracking and classification objects on a conveyor belt.

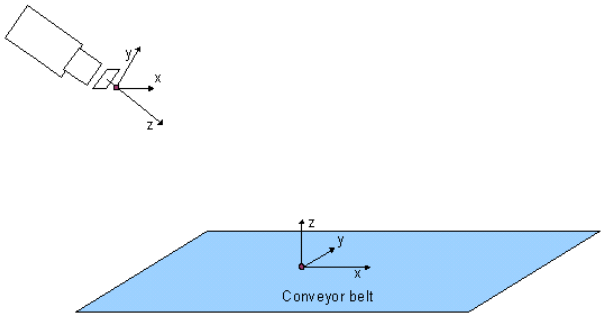

Figure 3.2: Transformation of the range data from camera coordinate system to belt coordinate system [2].

In this process, the camera is placed mounted facing to the conveyor belt with using coordinate transformation. Because the range camera and the robot must agree on a

coordinate system so that the range camera's measurements can lead the robot, the coordinate transformation is required which is show in Figure 3.2.
In the belt coordinate system, the -axis is also normal to the belt plane. Aligning the coordinate axis in this manner allows the extraction of object point clouds on the belt by referencing points above the belt plane and within the limitations of the belt boundaries. This algorithm has two phases: training and tracking. During the training phase, the object is placed in a different orientation so that it can identify different orientations of the object on the conveyor belt during the tracking phase [2].

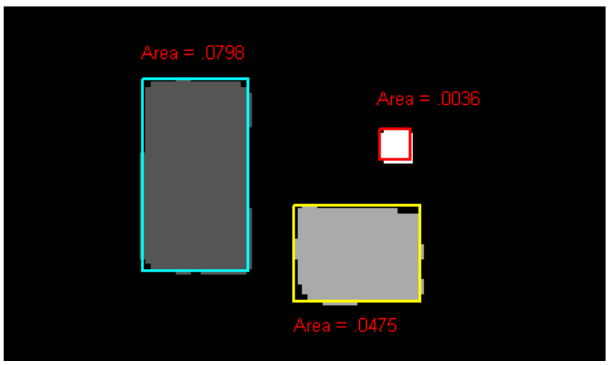

Figure 3.3: Computed statistics of detected boxes [2]. Calculated the bounding rectangle, as well as the area and other factors like orientation.

The point cloud corresponding to each box is projected back to the belt, and a minimal bounded rectangle is fitted to the two-dimensional point cloud, as illustrated in Figure 3.3. This method of binary classification attempts to recognize the item and its orientation, however it has the following drawbacks:

- To train the model, need a lot of data.
- In real-time detection, there is a cost in terms of time
- As mentioned in the study, this may be impossible to do with a standard camera
- It's also necessary to use the camera's exact position
- Unable to determine the orientation of the box

3.4 Central-Based 3d Object detection

Object detection plays a critical part in the stage of object orientation detection. As previously mentioned monocular 3d object detection with the object-orientation feature, this thesis presents a centrally based objection with 3d orientation in this section. While the central-based method detects an object's 3D position, it may also regress other attributes such as velocity, direction, and size, and it benefits from 3d information linked to orientation.

Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl suggested a technique for extracting orientation information using central-based object identification in 2021. They create a bird's-eye-view heatmap and various dense regression outputs, including the offset to centres in the previous frame, using a conventional 3D point cloud encoder with a few convolutional layers in the head. Tracking is a closest-distance matching, while detection is a simple local peak extraction with refining [3].

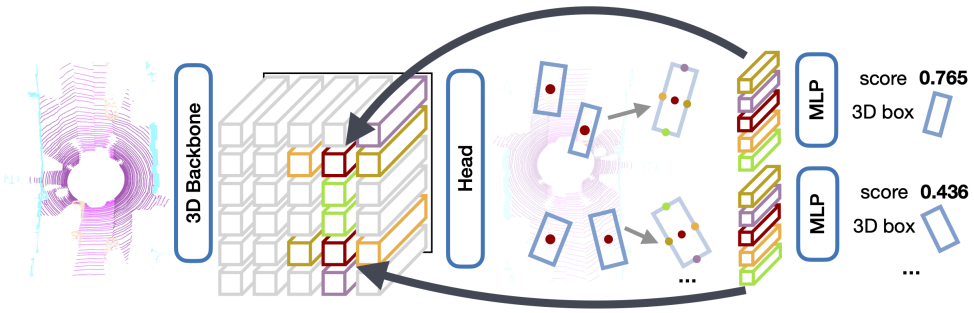

Figure 3.4: Overview of central based framework [3]. By utilze the centre feature of 3d boundign box detect orientation.

The Figure 3.4 illustrates the whole process from a standard 3d backbone to full 3d bounding boxes by utilizing the central feature. A 2d CNN architecture is used to detect the objects' centres, and the Orientation prediction uses the sine and cosine of the yaw angle as a continuous regression goal.

This technique necessitates a significant amount of data to train the data, as well as placing the camera in a bird's eye perspective, both of which might be disadvantageous. Furthermore, it is a very complicated system, and latency will be a factor in determining object orientation. It is also time-consuming to implement in real-time.

3.5 Point-Voxel CNN (PVCNN)

It is also possible to detect things using a 3D deep learning approach. One of these is Point-Voxel CNN, which outperforms other deep learning approaches in terms of object detection accuracy.

In 2019, Liu Zhijian, Tang Haotian, Lin Yujun, and Han Song proposed a method for detecting objects made up of voxels and points that are both quick and efficient in 3D deep learning [42]. Preserved models are also available, which may be used to generate train models, however, their accuracy varies depending on their paper. Following the detection of an item, further features will be estimated to extract the orientation feature. Because this method requires a lot of data to detect the object, it also requires a lot of data to train the model. A lot of GPU memory is required to perform correctly. Detecting the 3D orientation after detecting the object inside this model would be difficult, which is why it would be time-consuming to implement in a real-time conveyor belt to identify the orientation of parcels.

3.6 MediaPipe Objectron

MediaPipe is a framework for creating a cross-platform applied machine learning pipeline for a variety of object tracking and detection tasks, including Object detection, Face detection, Hand detection, Multi-hand detection, and Hair segmentation [43]. The MediaPipe Objectron detects objects in 2D images and expands the size and orientation of those objects in the actual world.

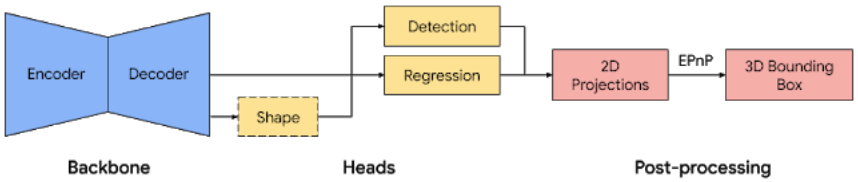

Figure 3.5: Network architecture and post-processing for single-stage 3D object detection [4]. Give Strong relabel result in multiple objects also.

To predict the 3D bounding box of an object, MediaPipe offers two ML pipelines: one stage and two stage. The single stage pipeline (Figure 3.5) is good for multiple object detection, whereas the two stage pipeline (Figure 3.6) is good for single superior object detection. (Figure 3.7) displays an example of the MediaPipe object detection

findings.It shows the original 2D picture with estimate bounding box, object detection using Gaussian distribution on the middle, and prediction segmentation mask on the right [4].

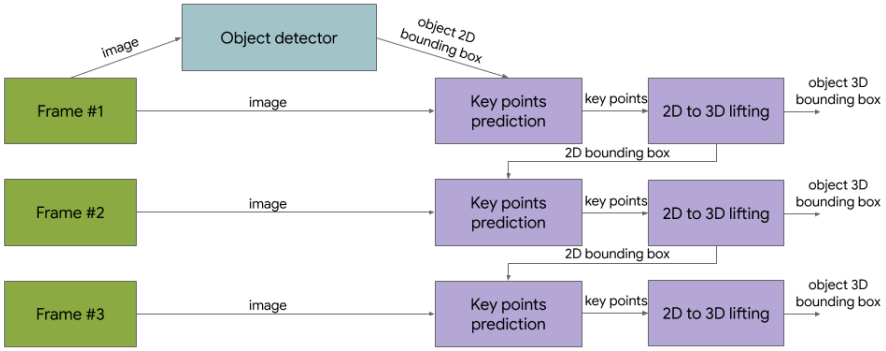

Figure 3.6: Network architecture and post-processing for two-stage 3D object detection [4]. This is good or single objects detection.

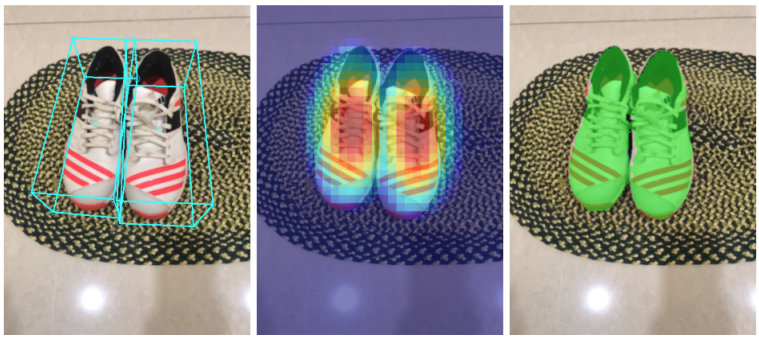

Figure 3.7: Sample results of media pipe detection [4]. Media pipe supports bounding box as well as color segmentation.

Tingbo Hou, Adel Ahmadyan, Liangkai Zhang, Jianing Wei, and Matthias Grundmann proposed a technique for detecting objects from RGB pictures in 2020, and then trained the model using synthetic data to actual data transfers shape learning [44].Segmentation and coordinate map prediction are also shown for their model, and segmentation is a useful feature for estimating the object orientation.

There are 4 million annotated images in 14, 819 annotated videos in the Objectron dataset. By presenting baseline models trained on this dataset, Adel Ahmadyan, Liangkai Zhang, Jianing Wei, Artsiom Ablavatski, and Matthias Grundmann illustrate the dataset's utility. For each item, the data also includes manually annotated 3D bounding boxes that define the object's location, orientation, and size [45].

In a study published in 2020, the researchers proposed an immediate motion track-

ing system to track an object's 3D position, as well as its orientation, translation, and size. When combining the detection and tracking data, the detection network would predict different orientations each time, resulting in failure [46]. Although MediaPipe can anticipate object orientation, configuring it to detect image and orientation, as well as detecting object direction, is difficult.

3.7 Augmented Auto encoder PyOpenGL

Detecting the direction of an item from an RGB picture is extremely fruitful. For object identification and posture estimation, Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, and Rudolph Triebel suggest a real-time RGB-based pipeline with 3D orientation estimation and it is based on a Denoising Autoencoder version that is trained using Domain randomization on simulated views of a 3D model. Compared to previous techniques, this so-called Augmented Autoencoder offers numerous advantages: It requires no real, pose-annotated training data, generalizes to a variety of test sensors, and handles object and view symmetries automatically [5].

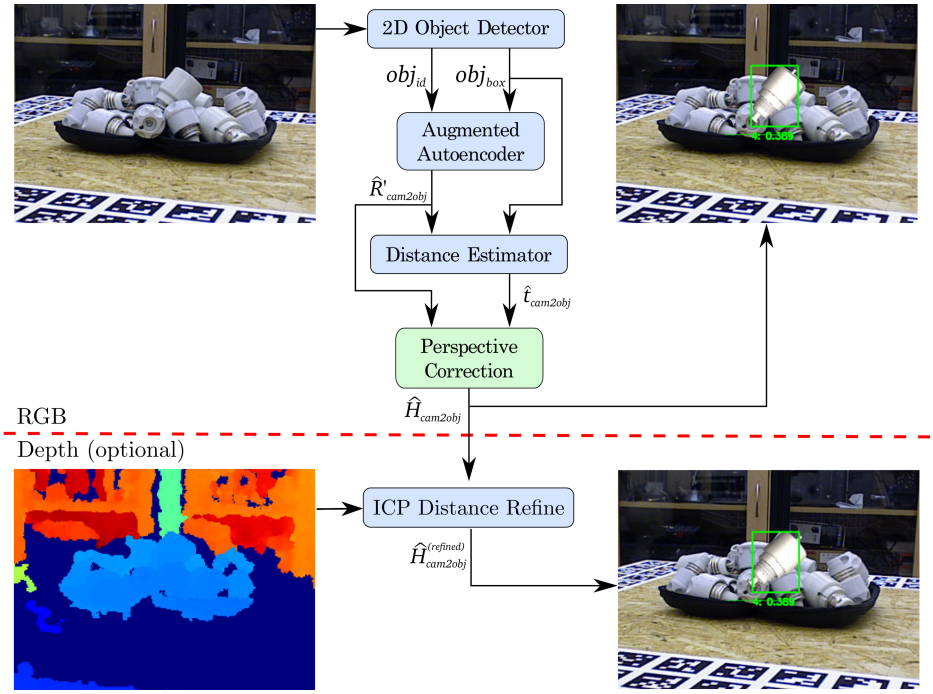

Figure 3.8: 6D Object Detection pipeline [5]. From RGB image this process detect the object. There is also optional depth information of target object.

After identifying an item (2D Object Detector) in the Figure 3.8, the object is quadratically cropped and passed into the suggested Augmented Auto encoder. The bounding

box scale ratio at the anticipated 3D orientation is the next step. There are also the following failure situations in this process:

- Occlusions and object ambiguity caused detect ions to fail.
- This procedure fails due to a strong blockage.
- Because of the shadowing, prediction accuracy suffers.

3.8 Convolutional Neural Network

One of the good features for successfully detecting the orientation of goods on the conveyor belt is detected and categorization of the arrow. A study is being conducted to detect and recognize road markings using an area of interest and a deep convolutional neural network (CNN). To produce the ROI picture, a vanishing point is discovered in the first stage. In the second step, the ROI picture that covers the bulk of the road region is utilized as the input to train the CNN-based detector and classifier [6].

Figure 3.9: Case of correct detection [6]. With hard noise this process can detect the arrow mark on the road.

Figure 3.10: Case of incorrect detection [6]. In case of small arrow mark this process trouble to detect.

Figure 3.9 depicts situations when numerous road markings are correctly recognized, but if the road marking is small or the quality of the marking is poor, the marking is not detected, as seen in Figure 3.10.

3.9 Box Boundary-Aware Vectors (BBAVectors)

For all freely oriented objects, the box boundary-aware vectors are distributed in the four quadrants of a Cartesian coordinate system. Jingru Yi, Pengxiang Wu, Bo Liu, Qiaoying Huang, Hui Qu, and Dimitris Metaxas proposed a technique in 2020 that first detects the object's center key point, based on which the box boundary-aware vectors (BBAVectors) are regressed to capture the oriented bounding boxes [7].

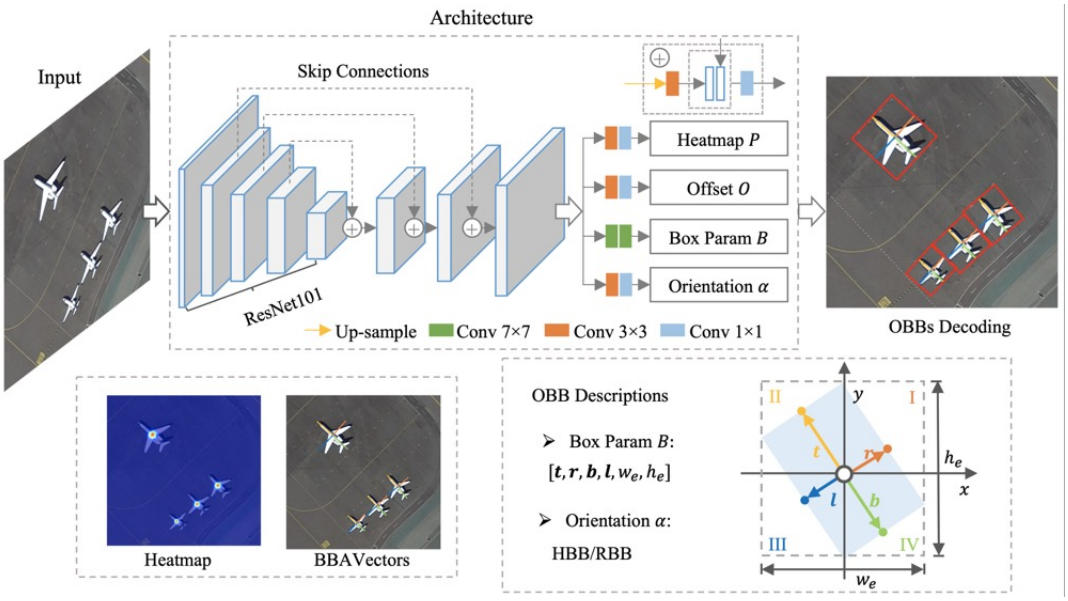

Figure 3.11: Overall architecture with oriented bounding box (OBB)[7]. By calculating the orientation of bounding, this process detect the objects orientation.

The method of determining the object's orientation, as well as the orientated bounding box, is shown in Figure 3.11. Before being sent to the U-shaped network, the input picture is resized to 608*608 pixels. The architecture produces an orientation map, a heat map, an offset map, and a box parameter map.

4 Conceptuation

The automatic unloading procedure adds a new dimension to the manufacturing and logistics systems. An efficient and profitable logistics system requires an automated unloading system. Because of sorting and knowing the various distinctive info connected to packages, detecting the orientation of parcels plays a critical function in the autonomous unloading system. This thesis explains how the design was created to detect the orientation of packages on the conveyor belt in this part.

4.1 Identification of the Conceptual Design

Conceptual design produces an outline solution to a design challenge, and design usually starts with a requirement [47]. To begin, this thesis examines the characteristic information of the target parcels (Figure 4.1) as well as the requirements in order to determine the parcels' orientation.

Attribute No.	Attribute Details
01	Three different package sizes.
02	On two opposing sides, there are arrow signs.
03	The color of the box is approximately Bourbon.
05	The conveyor belt is almost silver in hue.
06	In the case of big size boxes, approximately one box could be placed horizontally on the conveyor belt and more than one may be positioned vertically.
07	The middle-sized parcel may be put horizontally alongside small-sized parcels, and all three kinds of parcels could be arranged vertically on the conveyor belt.
08	For compact spaces, more than one box can be stacked horizontally and vertically.

Table 4.1: Characteristics of the conveyor belt and the packages. Prior to implementation, it is vital to understand all of the targets in depth.

Different demands linked to packages and conveyor belts are identified in the Table 4.1, allowing this thesis to suggest a conceptual design to detect parcel orientation. This thesis focused on two phases to effectively identify the design for the detection of package orientation on a conveyor belt. The first step is to create a pipeline to detect parcel orientation, and the second is to integrate a camera position into the unloading system.

Figure 4.1: The parcel that this thesis is aiming for. On the packets, there is an arrow mark that can be seen in the figure.

To successfully classify the item from extraneous noise in the environment, the camera arrangement is critical. This thesis focuses on simple and quick while designing the conceptual design for pipeline to identify the orient object since the conveyor belt speed would be rapid on an autonomous unloading system.
Jozefowska, Joanna, Pawlak, Grzegorz, Pesch, Erwin, Morze, Micha, and Kowalski, Dawid presented a model for the container packing problem in 2008, as well as several forms of box orientations [8].

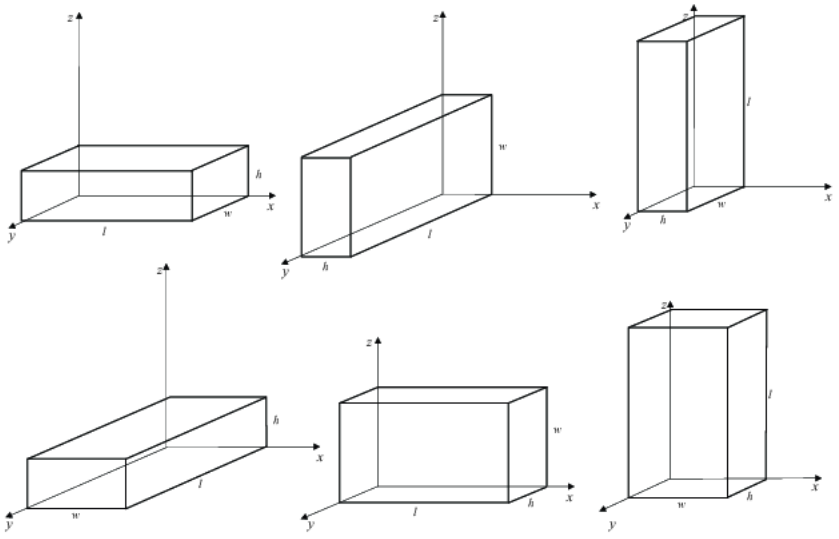

Figure 4.2: Different box orientations [8] suggest in the litterateur. It aids this thesis in determining the orientation kind.

The various sorts of boxes (Figure 4.2) aid this thesis in obtaining the initial stage of the orientation detection pipeline. This thesis identifies the three sorts of box orientations that are recognized on this thesis pipeline to simplify the detection orientation process (Figure 4.3).

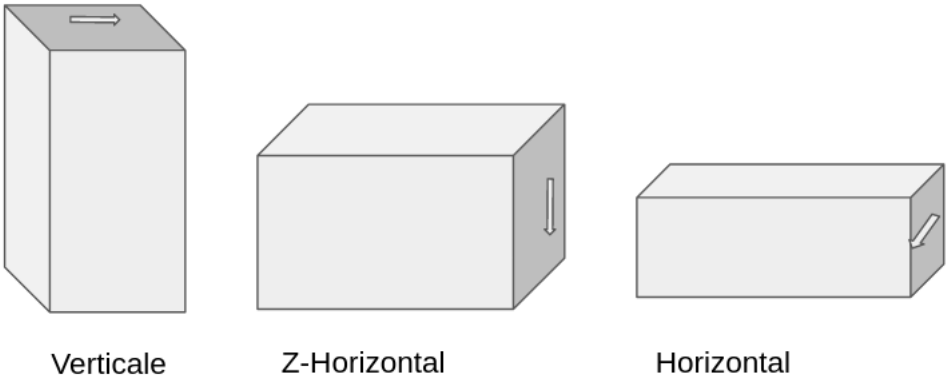

Figure 4.3: There are three different sorts of orientations. On the vertical orientation, the arrow is at the top. It is z-horizontal if there is an arrow on one side and the direction is up or down. Horizontal orientation is indicated by an arrow pointing to the left or right.

The three orientation types in this thesis also indicate the parcels' three-dimensional orientation. This thesis attempts to build a detection system for angel and arrow, which is the most significant characteristic, to correctly categorize the three distinct orientations. But initially, the item must be detected, therefore this thesis includes a process to detect images after receiving photos from the camera. There is a lot of noise in the surroundings when it comes to package detection. As a solution, this thesis adds a second step of selecting an area of interest (ROI) so that identifying an object on a conveyor belt may be done quickly.

The angle is then detected using a feature derived from the detection of parcel bounding boxes. Following that, this thesis discovered that classifying orientations based just on the angle is insufficient. There is another feature arrow in the Figure 4.3, which might be a feature to categorize the orientation. This thesis adds this step detect arrow while calculating the angle. But this thesis also observed that in the Figure 4.4 that, one cam or stage is not enough to detect both angle and arrow. Because of the primary cam on top view then the side part of the parcel is missing and if the primary is inside position, on top arrow could be placed. Besides that, if one pipeline is responsible

Figure 4.4: In the real unloading system, a parcel is on the conveyor belt.

to detect the parcel, calculating the orientation angle and detecting also arrow, then the process could be slow. This thesis determines that two cameras are highly suited to successfully achieve this thesis goal of detecting different types of orientation and making processes light and quick.

Finally, the design for the pipeline to identify parcel orientation is generated, which has been separated into two processes. One process is responsible for just detecting the arrow with relevant features on the side, while another process on the top is responsible for calculating the angle with a detect arrow. This thesis' activity diagram for the first phase is in the Figure 4.5. The design of the overall architecture is examined in the thesis's following part.

The integration of camera position is critical for passing the orientation detection pipeline successfully. For this thesis, the design of integrating cameras on a conveyor belt is depicted in Figure 4.6. The first camera is mounted on the roof to provide a bird's-eye view of the goods on the conveyor belt. From the top, the orientation quickly obtains an image, allowing parcels to be identified and other features to be obtained to compute the first pipeline responsible task. The second camera is positioned at the conveyor belt's side corner, at a distance from the top camera. The detection of objects may be hindered if the parcels are so near to the camera. As a result, the side camera is kept at a safe distance and only focuses on the second step of the orientation pipeline.

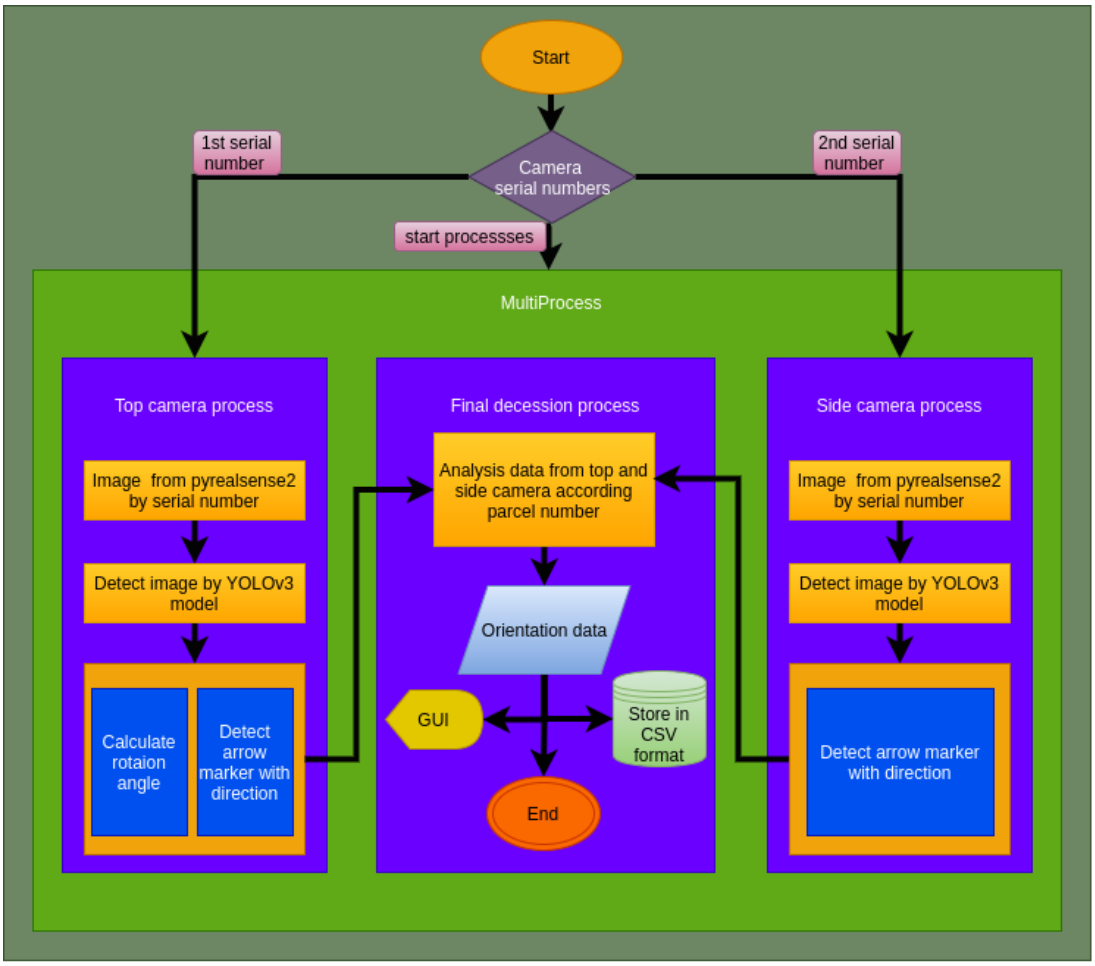

Figure 4.5: A diagram of the suggested orientation detecting architecture can be seen here. After receiving the serial number for the connected camera, three processes run in parallel: top camera, side camera, and final decision. The top and side camera procedures take relevant data for the target and pass it on to the final decision processes, which analyze and create the orientation result.

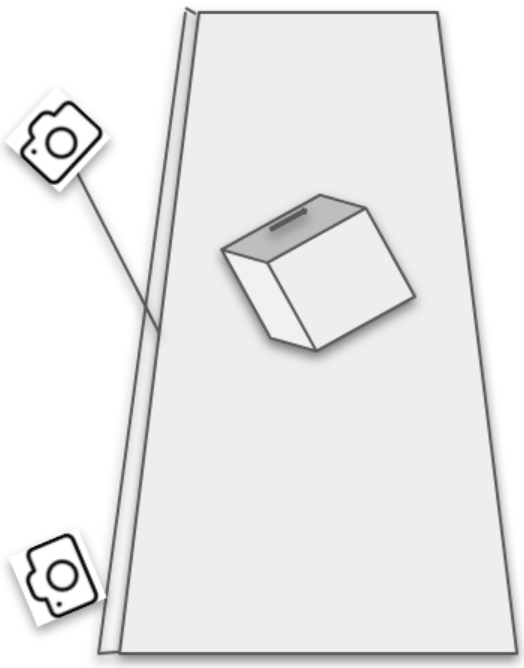

Figure 4.6: Design of a conveyor belt integration camera. To acquire the arrow and angle data of parcels, the top view integrates with the camera stand and captures the surface of the parcel. Another camera is mounted on the side to collect data from parcels with arrow marks.

4.2 Analysis Design

Because of its simple and quick processes, the design of this thesis is feasible. To get the orient information of parcels on a conveyor belt, this thesis uses two different processes. This thesis analysis proposed design in disparate parameter which is show in Table 4.2.

The input of the orientation detection procedure is the variable initial parameter. This thesis contains two procedures for detecting parcel orientation, as well as input data that differs. Intel RealSense camera provided the input images. The pictures from linked cameras are sent into the Intel RealSense camera's own pipeline. Pyrealsense2 is a python package that makes integrating and configuring the frame a breeze. This camera pipeline produces both color and depth pictures, allowing for very accurate and quick object recognition. It is extremely simple to obtain various images from different linked cameras by supplying the serial number, which can also be obtained via pyrealsense2.

The second is object detection, and we have several models and techniques to detect

Parameter	Thesis Proposed Design	Advantage
Input	To acquire the image, use the IntelRealSense camera.	Good quality image Easy to install Easy to get image from multi-connected cameras
Detection Object	To detect the object, use the YOLOv3 algorithm.	Pre-trained model available Easy to train the model Good accuracy with small dataset Fast in real time object detection
Orientation Type	Produce a 3D orientation result from the output of parcels' vertical, horizontal, and z-horizontal orientations.	Easy to understand the actual orientation of parcels
Detection Angle	To detect the angle of the parcels, use segmentation and OpenCV libraries, and produce output with angle type.	Fast calculate angle Good accuracy to detect angle of parcel Easy to understand the angle result type like: straight, acute, right and obtuse
Detection Arrow	To get information about the direction an position of an arrow, use the OpenCV libraries.	Fast and easy to integrate Help to get actual orient info of parcels

Table 4.2: Design with a wide range of features and their benefits. This thesis uses the YOLOv3 model to detect objects and produce orientation detection findings by examining the rotation angle and detecting the position and direction of arrow marks on parcels.

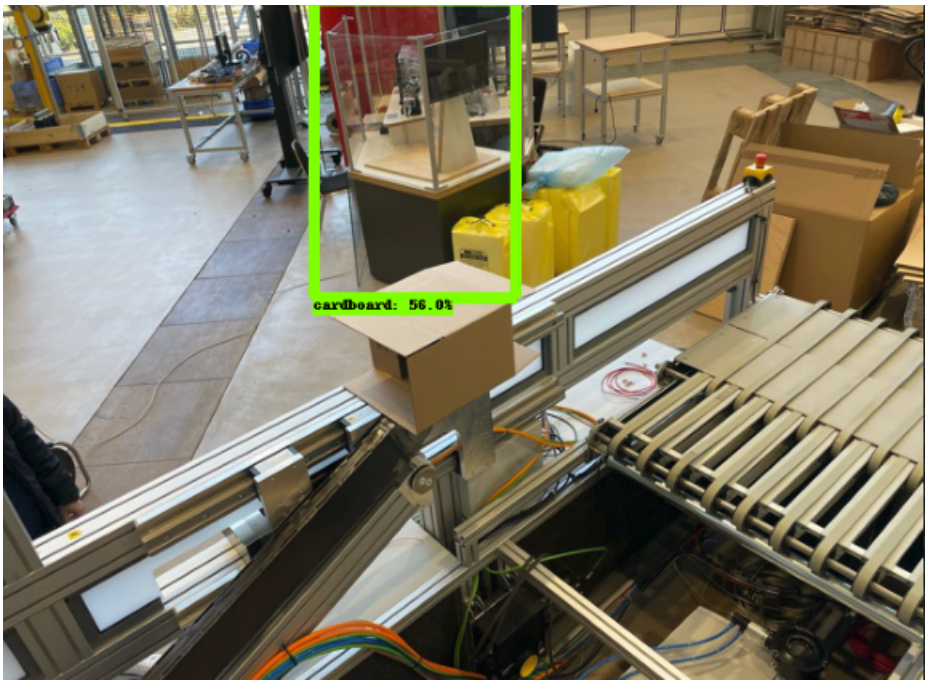

Figure 4.7: MobileNet model for parcel detection with less data. Initially, this research used this approach to detect parcels, but the results were disappointing.

the item from the literature. For the sake of accuracy, many of them require a large amount of data to train the model. For the comparison, this thesis performed a pre-test with limited data in order to identify the model for the examples in Figure 4.7 and Figure 4.8. Furthermore, data processing for the YOLOv3 model is simple. In comparison to other model trains, it also takes less time. The YOLOv3 model is also very quick at identifying images in real time.

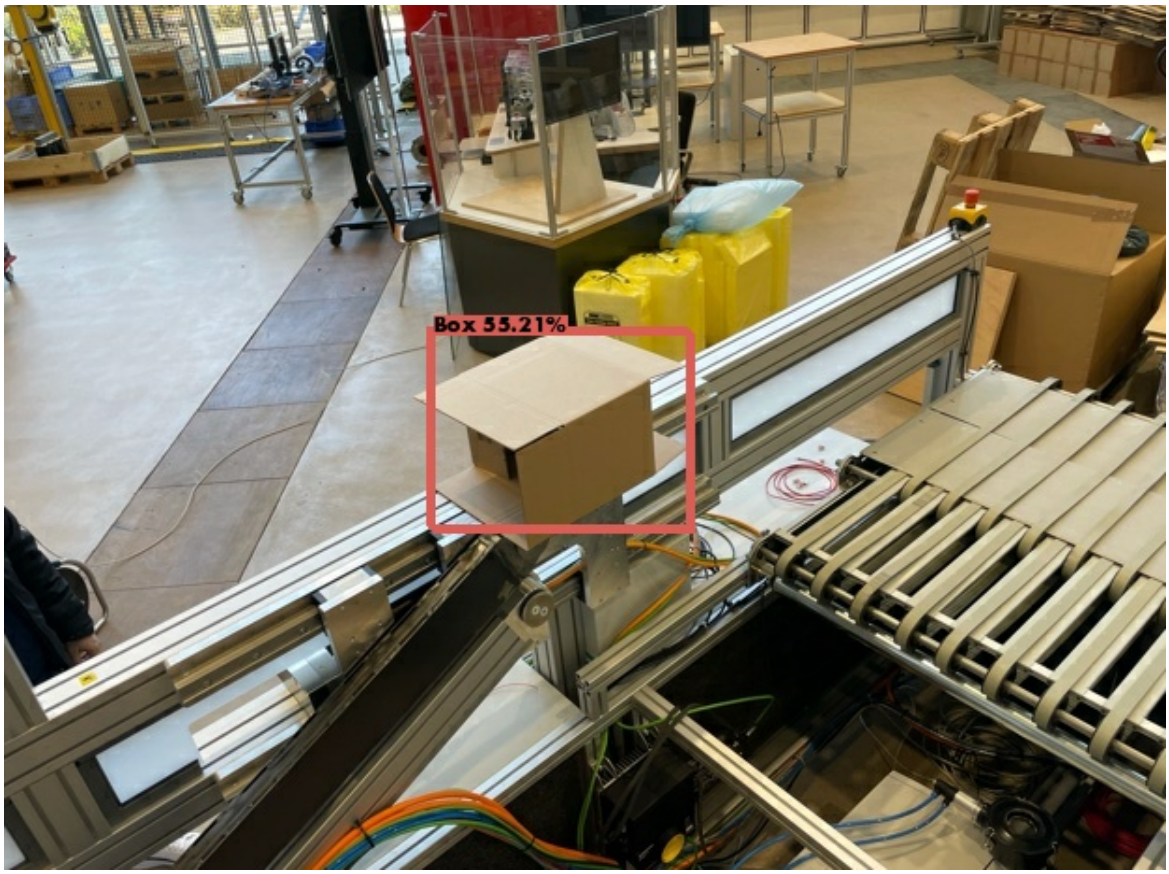

Figure 4.8: Detection of parcels using the YOLOv3 model with less data. To choose a model, run a test, and YOLOv3 gives an excellent result at first.

After obtaining findings from two multi-processes, this thesis calculates orientation type. Vertical, horizontal, and z-horizontal orientation types are divided into three categories for ease of comprehension. Angle and arrow position are used to calculate vertical position. If the arrow is on top, the thesis product will have arrows pointing in the directions of vertical-left, vertical-right, vertical-up, and vertical-down. This thesis discovered that when the arrow is horizontal, it points to the side, but when the arrow points up or down, it has a different orientation. As a result, the horizontal position in this thesis is divided into two parts: 1. horizontal and 2. z-horizontal. The pipeline of this thesis produces the orientation type horizontal-left or horizontal-right in the event

of arrow location in the side, if arrow direction is left or right. Aside from that, the orientation type is z-horizontal-up or z-horizontal-down depending on whether the arrow is pointing up or down. This makes understanding the real orientation and knowing the location of other parcel characteristics for parcel sorting much easier.
This thesis uses image segmentation to compare noise and object surface, which is critical for calculating the angle properly. The angle of orientation of the detecting item was determined by the top camera. It is simple to determine the angle with other parcel characteristics using OpenCv tools. Angles come in a variety of shapes and sizes. This thesis also generates results for the degree of rotation angle and angle type.
Finally, arrow detection has been added to the pipeline, making it more powerful in producing precise orientation results. This is accomplished through the use of a side camera pipeline. This thesis also makes advantage of Opencv's different methods. This arrow detection method is quick and simple to implement. The pipeline computed the direction and location of the arrow when detecting it, which helped to create the orientation type.
With Python's multiprocess architecture, the top and side camera processes are operating at the same time. The outputs of these two simultaneous operations must be combined and analyzed. This thesis does this by including an additional parallel process in the multi-process. So that the output from the top and side camera processes may be combined in this additional step.
Along with the orientation detecting processes, there is an additional process running. This additional pipeline job combines input from two processes and analyzes it to create all parcel-oriented information. This extra step also saves the data as a history so that it may be analyzed later to see how well the real-time orientation detection performed.

4.3 Design Steps Explanations

This thesis concept for orientation detection on a conveyor belt focuses on speed and simplicity. This thesis explains every phase of the thesis in this part to help this thesis better comprehend the design. To complete the whole detection pipeline, there are three separate processes. This section is divided into multiple subsections to ensure that each step is well understood.

4.3.1 Connection Camera

The initial stage of the pipeline is to verify the attached camera after it has been started. One or more cameras will be attached to the device. It's also possible that the camera isn't attached. As a result, pipeline will verify the attached camera in this section.The pipeline is immediately closed if there is no camera connected. This section is also in charge of pyrealsens2 obtaining the serial numbers of all linked cameras. If there are more than two cameras or less, this step will use the serial numbers of the first two cameras. The filtering serial number of linked cameras is the part's output.

4.3.2 Configuration GPU

In this thesis, the YOLOv3 model is used to detect objects. To recognize an item with a model, tensorflow (tf) must be configured so that the model is compatible with the local tensorflow (tf). It checks and creates a list of physical GPU devices initially. Then setup the first GPU device from the list with memory growth.

4.3.3 Store Orientation Results

box_number	rotate_angle	arrow_position	arrow_direction	orientation_type	box_detect_confidence_top_cam	box_detect_confidence_side_cam	total_boxes	angle_type	data_write_time
1	81.0	Top	Right	Vertical-Right	93.12%	98.91%	1	Acute	17:23:39.162005
1	80.0	<null>	<null>	<null>	94.40%	98.64%	1	Acute	17:23:38.132423
1	81.0	Top	Right	Vertical-Right	93.54%	98.78%	1	Acute	17:23:37.094520
1	81.0	Top	Right	Vertical-Right	93.72%	98.83%	1	Acute	17:23:36.066776
1	82.0	Top	Right	Vertical-Right	94.49%	98.66%	1	Acute	17:23:35.036430
1	81.0	Top	Right	Vertical-Right	94.80%	98.88%	1	Acute	17:23:34.007119
1	81.0	<null>	<null>	<null>	93.42%	98.90%	1	Acute	17:23:32.976423
1	81.0	Top	Right	Vertical-Right	94.93%	98.95%	1	Acute	17:23:31.899131
1	81.0	Top	Right	Vertical-Right	94.74%	98.82%	1	Acute	17:23:30.862491
1	81.0	Top	Right	Vertical-Right	93.09%	98.95%	1	Acute	17:23:29.836192

Figure 4.9: Results of orientation in the history. This is stored in the form of comma separated values (CSV) by the orientation detection procedure.

Create a csv file to save the data from the top and side camera operations at this point. It will produce a csv with header (Figure 4.9). if the CSV file does not exist. The saving date can be modified depending on the requirements. The stored information according to headers is given on the Table 4.3.

4.3.4 Top and Side Camera Queue

A queue is a multiprocessing process. It is used to move data from one process to another in a secure manner. Two queues have been created in this step. One pipeline

Header Name	Usage
box_number	For additional headers, add a box number to the information.
rotate_angle	Predict the angle at which a package will rotate in degrees.
arrow_position	The arrow's position on the box (Top or Side)
arrow_direction	The arrow's direction on the box(Right, Left, Up and Down)
orientation_type	Orientation results with direction (Vertical, Horizontal and Z-Horizontal)
box_detect_confidence_top_cam	For top Camera, the percentage of confidence in parcel detection
box_detect_confidence_side_cam	Percentage of package detection confidence for side camera
total_boxes	The number of detected boxes on the conveyor belt at every one time.
angle_type	Different type of angle according to detected angle(Acurte, Right, Obtusa and Straight)
data_write_time	Data was saved on a history csv file at a specific date and time.

Table 4.3: Explanation of the results of the saved history. The most relevant data is saved for subsequent analysis, such as angle in degrees, orientation type, angle type, detection confidence percentage, and so on.

is for top view, while the other is for side view.

4.3.5 Region of Interest(ROI)

There are superfluous items and noise in the frame during real-time detection. To speed up the process, the controller might choose the region where the item would be detected and perform the suitable techniques. There is an initial zone of interest so that the procedures can continue if the controller is not selected.

4.3.6 Intel Camera Pipeline

In this phase, pyrealsense2 is used to construct a realsense pipeline based on the camera serial number input. The frame was generated using the realsense pipeline, which was pre-configured with resolutions of 1280*720. This thesis just requires a color picture, thus the realsense pipeline can only produce a color image, but it can also return a depth image. This thesis design establishes two separate realsens pipelines for two cameras that work in tandem.

4.3.7 YOLOv3 Model

To detect the item on the frame, this thesis use the YOLOv3 model. The data from parcels was used to train this model in this thesis. In this stage, the trained model is loaded. Frames from the realsense pipeline are converted to 416*416 images, which are then utilized to detect with a trained model. The model's output also yielded a bounding box. In addition, the model predicts the score of confidence, classifications for a variety of objects, and the number of boxes.

4.3.8 Detection Angle

The most important is detection of angle of this thesis. With the help of OpenCV libraries, this this detect the angle of detection for detecting objects. Beside that image segmentation also integrate here to classify the top segment of the parcel. After segmentation , the segment part is used for calculating angle with multiple image processing steps. The output of detecting angle is split into two types. One is rotate angle in degree and another one is depend on calculated angle produce angle type.

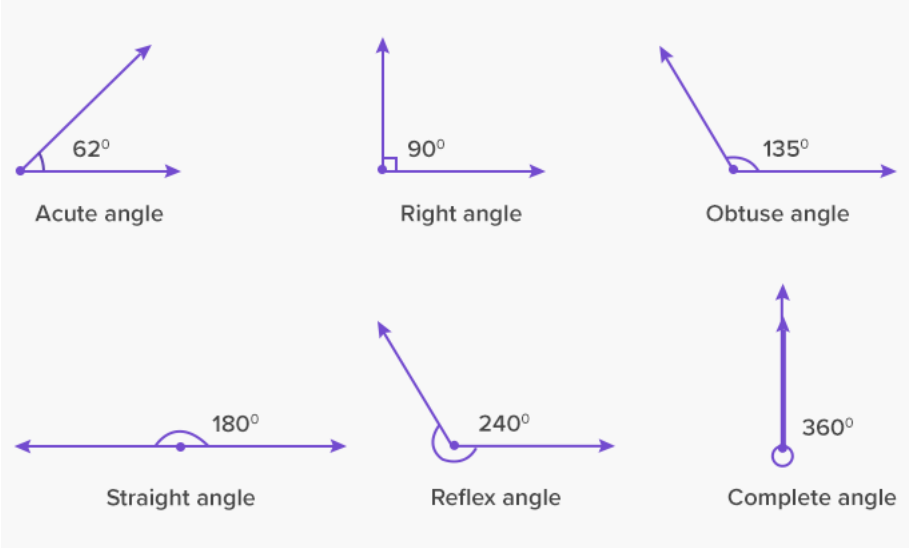

Figure 4.10: Types of angles [9]. To gain a better understanding of rotation angle types, this thesis will provide a simple and stable orientation detection result.

There is a different types of angle which show in Figure 4.10. This thesis consider the four types of angle. This angles type has also different ranges because the detection of angle could not be exact in real time. In the Table 4.4, explained the different ranges for different angles. This angle type helps to compare with different angle and orient of

Angle Type	Ranges
Acute	Greater than 50 and less then 85
Right	Greater than 85 and less then 100
Obtuse	Greater than 100 and less then 130
Straight	Greater than 40 and less then 50 or Greater than 130 and less then 150

Table 4.4: Angle types having a range of values. The orientation results in this thesis are produced using these four types of angle types.

parcel on conveyor belt.

4.3.9 Detection Arrow

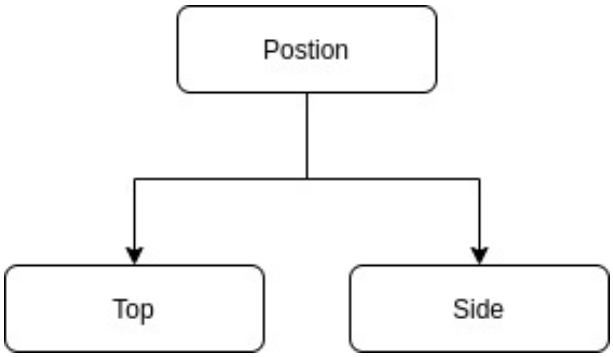

Figure 4.11: Design of arrow position result. The position is divided into two types.

This thesis design also includes the position and direction of arrow detection in order to tackle the challenge of orientation detection. This thesis uses OpenCv libraries to extract the direction and location of arrows on packages, making the procedure quick and easy.

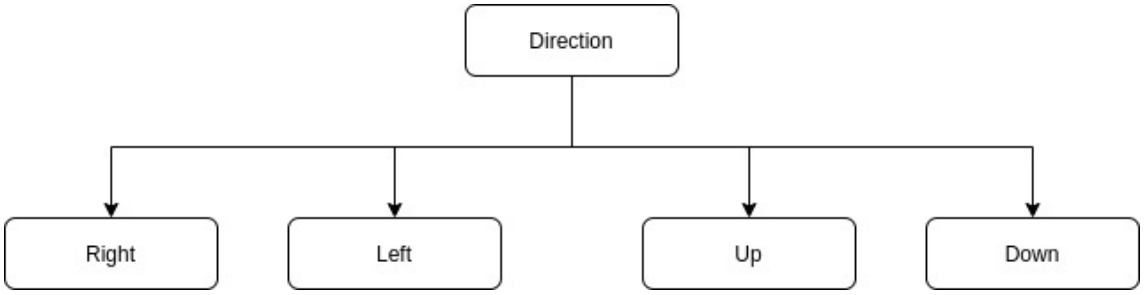

Figure 4.12: Design of arrow direction result. This thesis use this four types of direction to detect the arrow direction.

The detection of arrows is included in both top and side views, allowing this thesis to identify arrows from both perspectives. The first two sections of the detection are

location and direction. According to the camera position, top and side (Figure 4.11) positions are separated. The extract info in the portion of the arrow direction (Figure 4.12) is on the left, right, up, and down which make easy to get the actual parcel orient information. The arrow detection phase adds the information to the final decision process after computing location and direction.

4.3.10 Top Camera Pipeline

This is a top view camera multiprocess pipeline. This pipeline is used to identify arrows and angles. The top camera object queue, camera serial number, and class name are the pipeline's inputs. This pipeline is in charge of putting the extracted rotational angle and arrow information on parcels. This process is also in charge of choosing a ROI and visualizing the detection object with angle data. To visualize the the image and interaction with frame, its use Opencv various method which is explain in implementation section.

4.3.11 Side Camera Pipeline

This is a multiprocess pipeline that runs concurrently with the top camera pipeline. This procedure is in charge of detecting arrows and their associated information, such as their position and direction. The detect arrow on the parcel is also shown using OpenCv techniques in this procedure. In this process, the controller can choose the ROI for detecting the arrow from the frame. The extracted information of arrow position and direction is also stored in the fork by the side camera process, allowing the final decision process to analyze the data and create the orientation result of detect packages on conveyor belt.

4.3.12 Final Decision Pipeline

Final decision pipeline is the third and last process which is also a part of parallel process. The main job is to pull the data from the fork and analysis this data. This process analysis the data and take the decision of how the parcels is orient on the conveyor belt. The top and side view process put the data as this extracted. But the combination job is done by the final decision pipeline. This pipeline also create

object of graphical user interface so that controller easily can overview the information of orientation of parcels on conveyor belt. The data of Graphical User Interface (GUI) every time updated by the final decision pipeline process.

4.3.12.1 Graphical User Interface (GUI)

This thesis offers its own graphical user interface (GUI) that displays the results of parcel orientation on a conveyor belt.

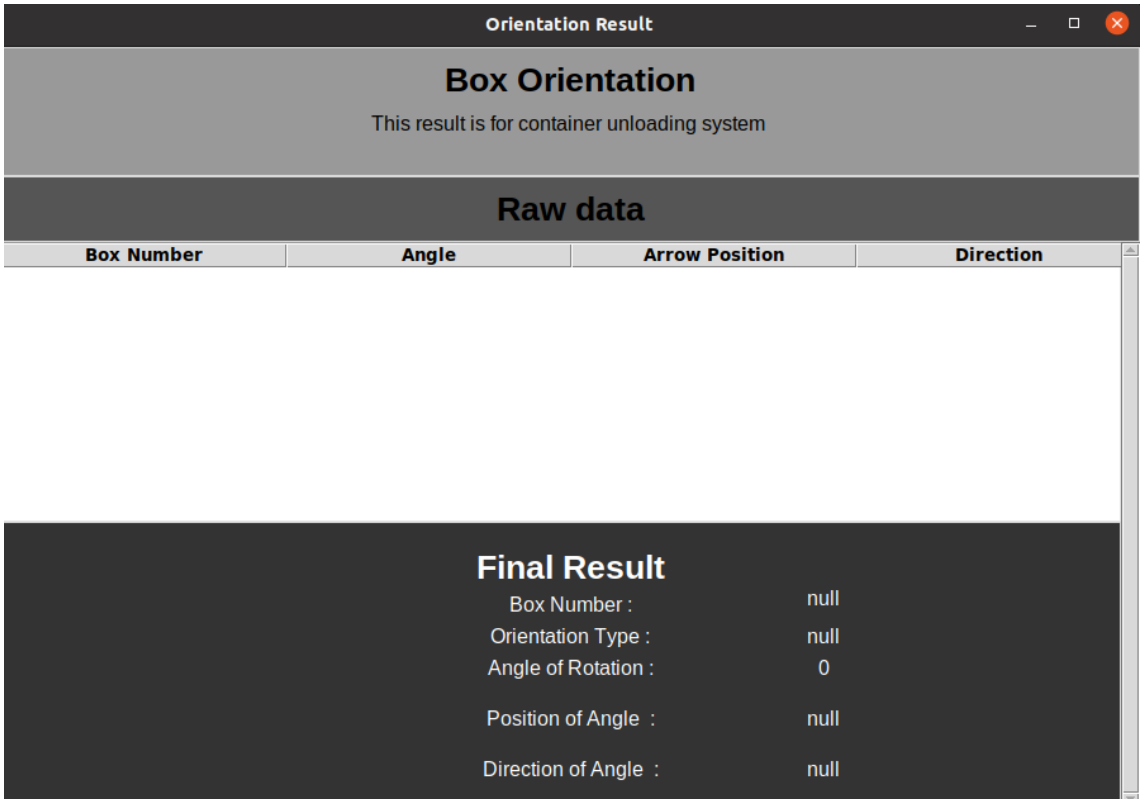

Figure 4.13: Graphical User Interface Design (GUI). This GUI is used to visualize the retrieved orientation information from this thesis algorithm.

The GUI is generated throughout the decision-making process. After analyzing the data, the orientation result is stored and seen using this GUI. The top section of the Figure 4.13is header. The detected orientation result is updated every time in the center section, where raw data is written. The middle section of this thesis displays the box number, angle in degrees, arrow location, and arrow direction. The final result is updated at the bottom of the page with the box number, orientation type, and angle of rotation, as well as the angle of position and angle direction. If there are numerous packages identified on a conveyor belt, the results are shown at the bottom by box number.

4.3.12.2 Data Analysis and Orientation Result

The rotational angle is extracted from the top view camera together with the position and direction of the arrow. Only the position and direction of the arrow are extracted by the side-view camera. The information regarding the parcels on the conveyor belt from these two procedures is sent into the fork, where it is analyzed in real-time by the final decision process. This process started by sorting the discovered data by box number. Then verify the data from the upper camera for arrow location. If the top camera isn't working, look at the side camera. This phase extracts the direction information from the position choosing camera placing information after getting the arrow position. In this phase, the orientation type is also classified. Then, for the box number, create the appropriate orientation information. Finally, save the detected orientation data into a Python data store according to box number for visualization and saving into history.

4.4 Summary of Concept

The goal of this thesis is to identify the parcels' orientation on a conveyor belt. This thesis includes two steps to do this. The first stage is to identify the articles on the conveyor belt, and the second is to determine their direction. This thesis utilized the YOLOv3 pertained model to detect the object. This thesis uses thesis data to train the model. In Figure 4.14, YOLOv3 has its own feature extraction architecture.

In comparison to the original architecture, the network of YOLOv3 now has 53 convolutional layers instead of 24. This improves the item categorization accuracy by more than 30 percentage while retaining the same speed. The categorization accuracy improves even more with higher resolution images [10]. In YOLOv3 use a much deeper network Darknet-53. This model use batch normalization.

YOLOv3 achieved mAP@0.5 with considerably quicker inference time than RetinaNet, as seen in Figure 4.15. Aside from that, YOLOv3–608 achieved 57.9 percent mAP in 51 milliseconds, whereas RetinaNet-101–800 achieved 57.5 percent mAP in 198 milliseconds, which is 3.8 milliseconds quicker.

As shown in Figure 4.16, YOLOv3 is significantly better than SSD and performs similarly to DSSD; nevertheless, YOLOv3 has reasonably excellent performance on `AP_S` but poor performance on `AP_M` and `AP_L`. YOLOv3 outperforms two-stage Faster R-CNN versions employing ResNet, FPN, G-RMI, and TDM in terms of `AP_S`.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4.14: YOLOv3 feature extraction architecture [10]. Its convolutional frame to extract the feature from image frame.

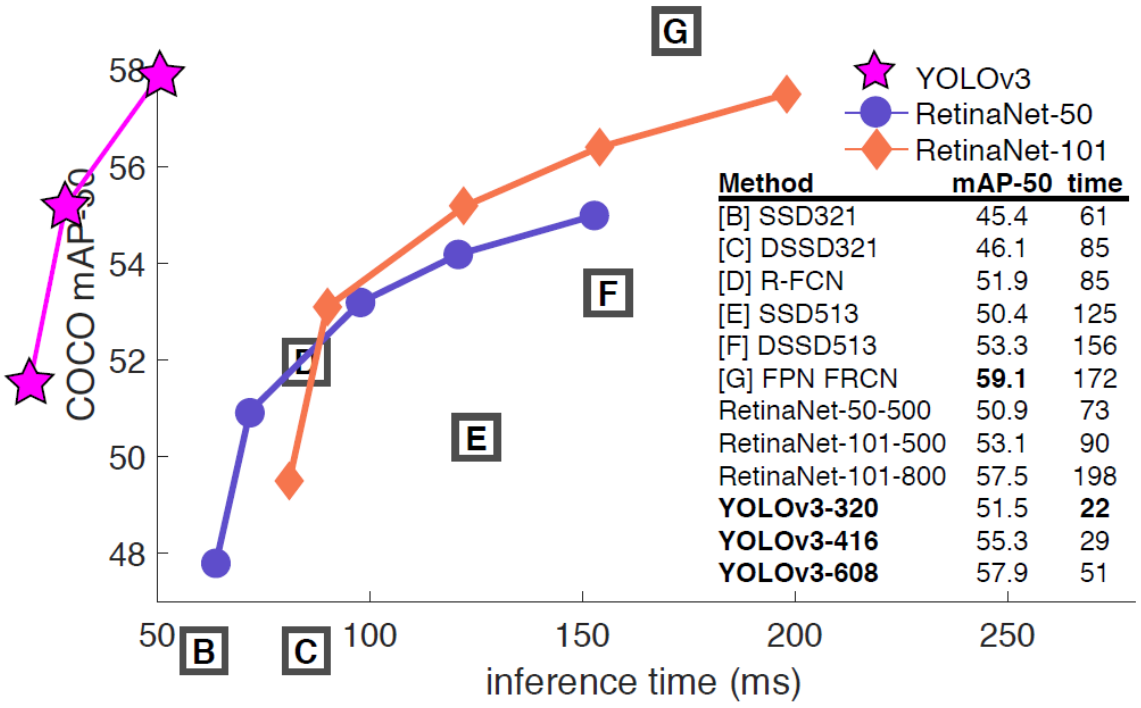

Figure 4.15: Comaprision with other model mean Average Precision (mAP) at IOU threshold 0.5 on COCO [11].

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figure 4.16: Object detection single-model results [12].

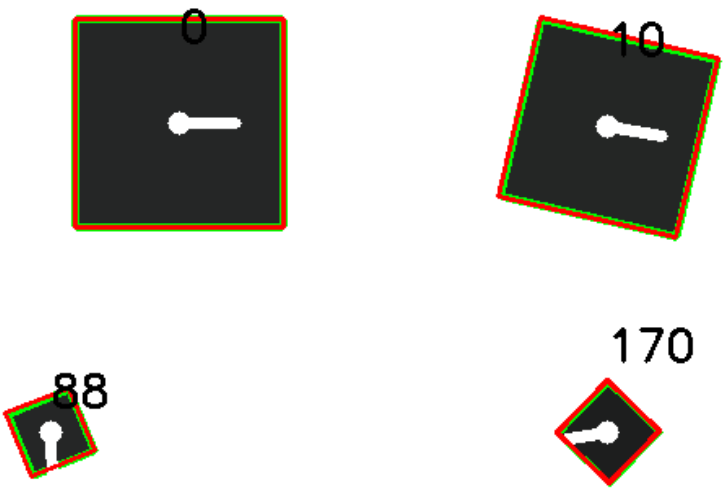

Figure 4.17: Feature evaluation of by calculation angle.

The orientation of the detected items is calculated in the second stage. Calculate angle from above view and identify the arrow sign are the two primary targets from the real parcel. This thesis choose to extract features from the corner ox box and the center of the box in order to identify the angle. A method of converting a picture to HSV may be used to extract this characteristic (hue saturation value). Then, using OpenCV, extract all of the image's contours.

Because of the noise in the image frame, there are several contours in a regular image frame. As a result, filtering contours is required to obtain the target contour so that further calculations can be performed. This thesis uses the canny edge approach and threshold separately for filtering in the early stages. The contours are still a lot after using the canny edge method on the parcels, as demonstrated in Figure 4.18. This image also shows what happens if you use the threshold approach to retrieve the target or if you simply want to get the front section contour from the parcel.

Describe how to acquire the correct parameter of this thesis target contour in the Figure 4.19(left side) and the outcome in the right side of the Figure 4.19 after color segmentation.

This thesis implement the bounding rectangle as well to find the corner. The thesis then

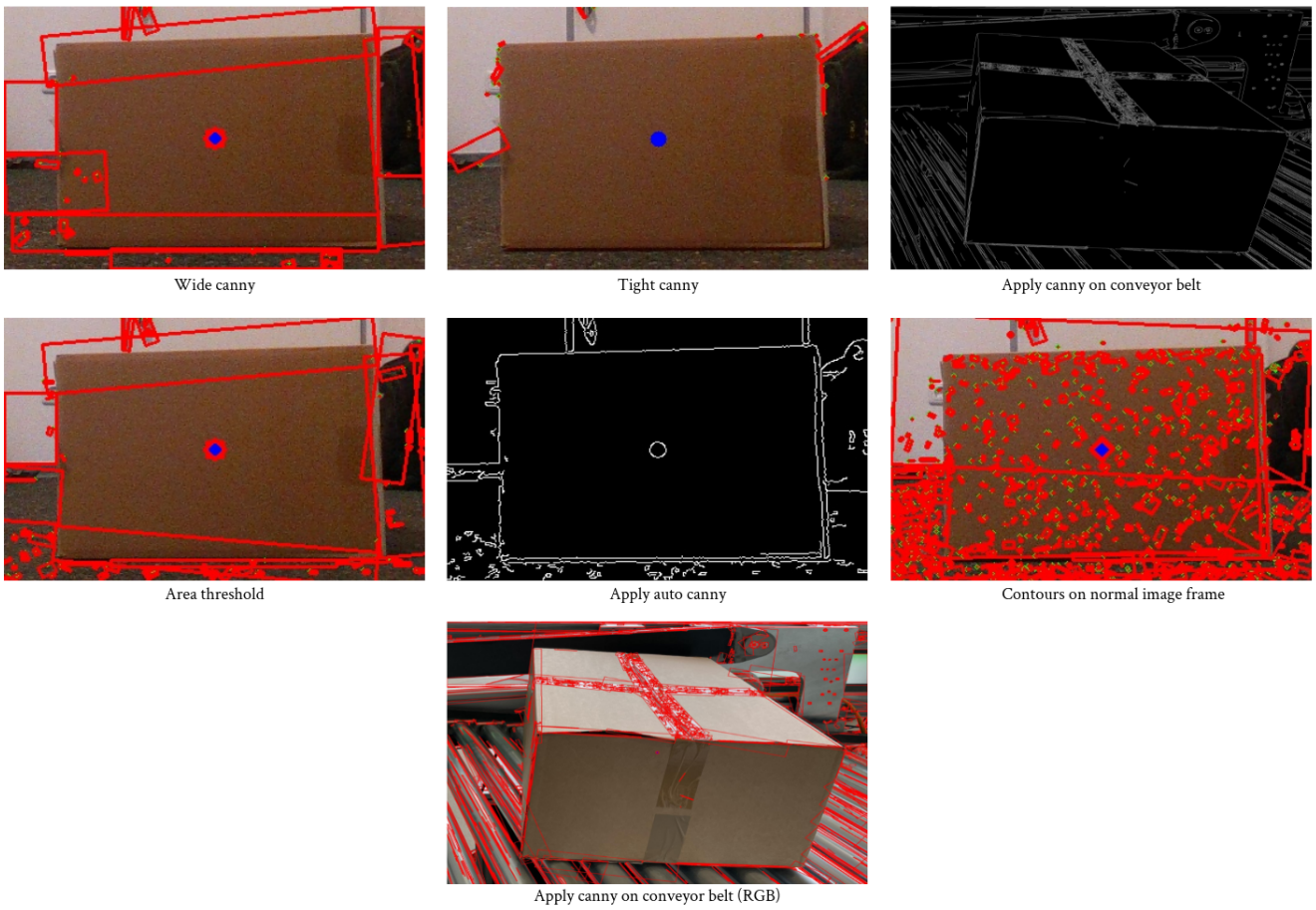

Figure 4.18: Test on initial stage to find the target part of parcel. To detect the desired contour, use a variety of methods, starting with canny. Contours must be filtered due to noise in the image frame's background.

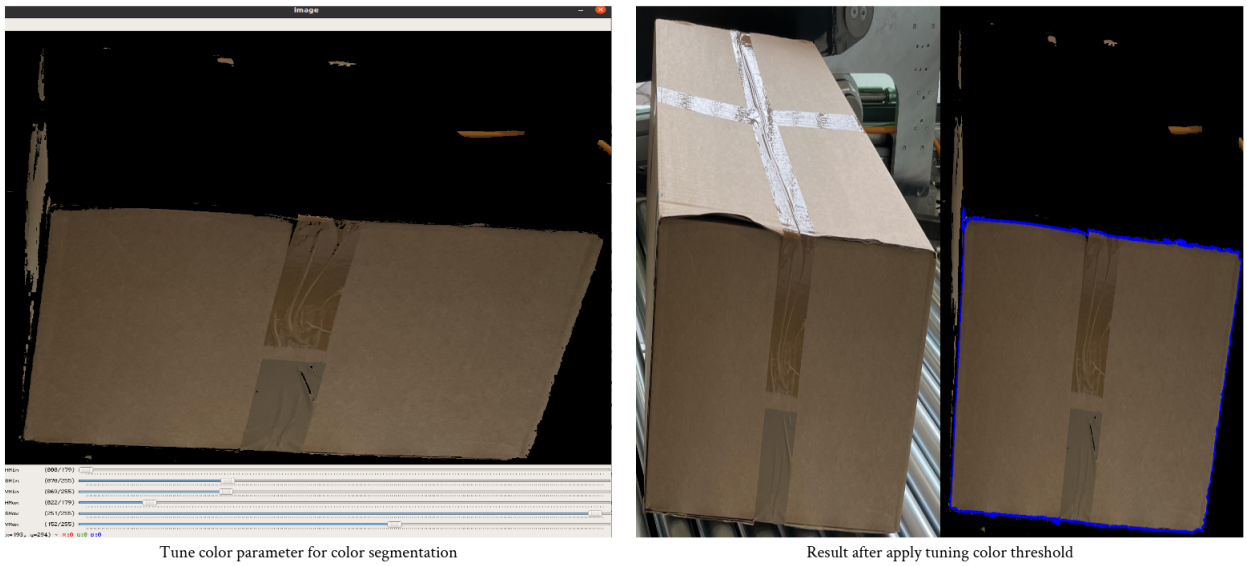

Figure 4.19: Process of color segmentation of parcel. To get the color threshold, this thesis use that to tune the parameter which show on the left and in the right is the result after apply color segmentation with color threshold.

calculates the corner point to obtain the corner. This thesis evaluates this functionality by using a rectangle shape Figure 4.10 to demonstrate it.
The corner of the arrow and the upper section of the arrow are used to detect arrows. It is feasible to determine the arrow position and direction using the corner and top calculations. This thesis evaluates the feature implanting on a typical arrow picture, as seen in Figure 4.20.

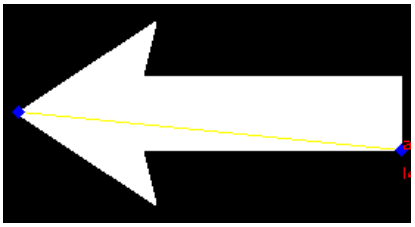

Figure 4.20: Feature evaluation of detection arrow.The arrow tip and the farthest point are calculated. This will assist you in determining the direction of the arrow mark.

OpenCV's canny edge detection technique was also utilized to extract the arrow's corner. Detect the contours also help to remove unnecessary part of the parcel. This thesis use the convex hull approach to extract the arrow's tip, making it simple to determine the arrow's orientation.

5 Implementation

After creating the concept of detecting parcel orientation, the most crucial aspect is implementation. This chapter, look at how to identify parcel orientation on a conveyor belt in practice.

The entire architecture for detecting the orientation of parcels on the conveyor belt in the unloading system is Figure 5.1. This method begins by identifying the connected camera and exporting the serial numbers of the cameras for the following step. Before beginning the real process, the algorithm checks to see if the CSV result file exists and configures the GPU to work with the detection model. This thesis algorithm extracts the orientation information of the observed parcel using three parallel processes. This data is then visualized using a GUI and saved in a CSV file with varied orientation data. In the following part, we'll go over the entire implementation. This section also includes the challenges that arose during implementation as well as the lessons learned.

5.1 Environment and Tools

A Dell Precision 5550 laptop with an Intel® CoreTM i7-10750H CPU running at 2.60GHz 12 and Mesa Intel® UHD Graphics (CML GT2) is used in the research. Ubuntu version 20.04.2 LTS is the operating system (Focal Fossa). This thesis uses Google Colab to train the model and the PyCharm integrated development environment to construct the pipeline. Python is the programming language, and Intel RealSense Depth cameras are used for the cameras. This thesis uses several OpenCV methods to identify the object orientation.

5.2 Objects Detection Model

The initial stage in detecting parcel orientation on a conveyor belt is to detect the object. To detect parcels on a conveyor belt, the YOLOv3 model is trained with parcel data. Prepare the model to detect the parcel by doing the following procedures.

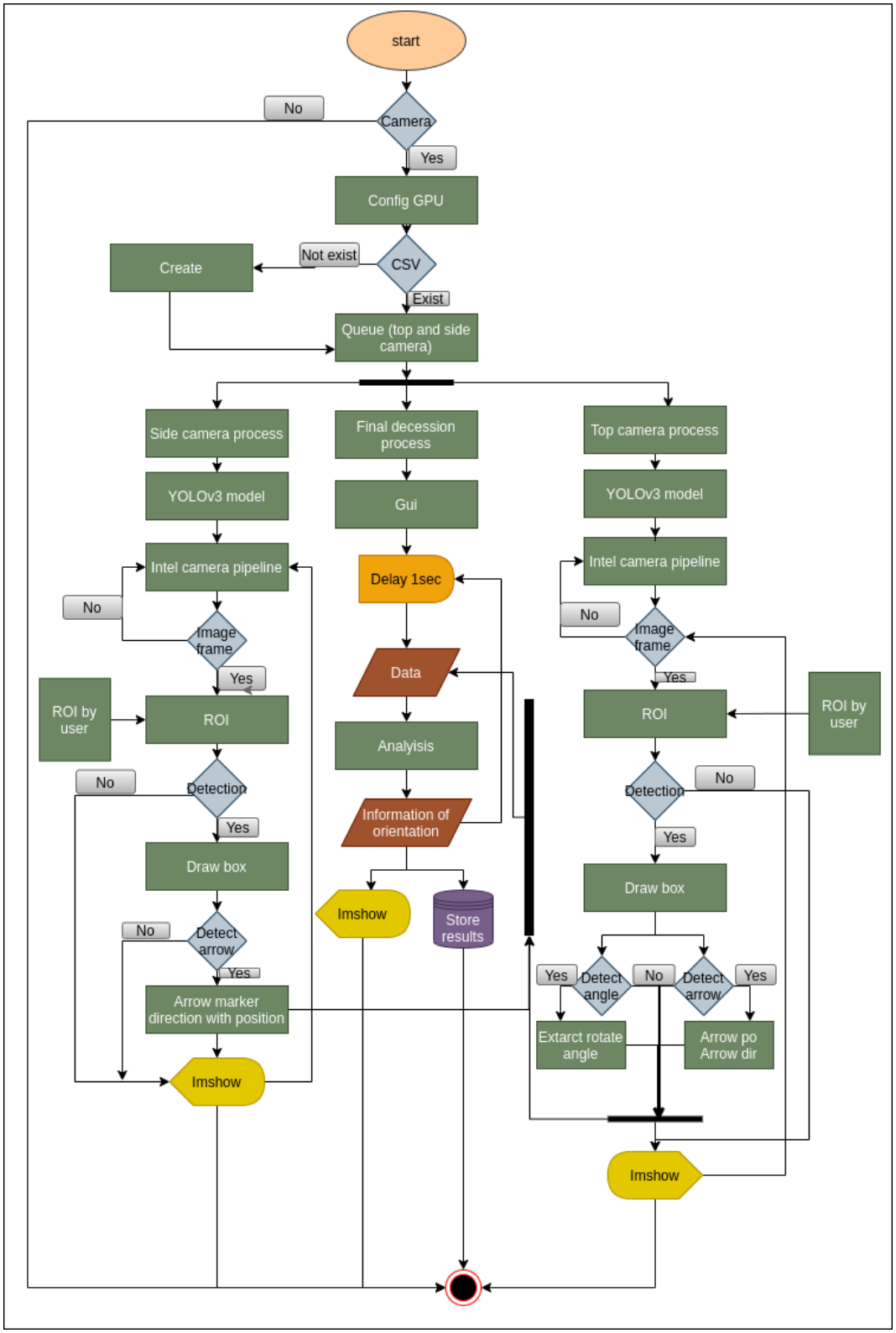

Figure 5.1: Activity diagram of orientation detection pipeline. It shows every step of this research orientation detection algorithm. The steps is describe in the section of design Interpretation. The multi-process of top, side and final decision process help this thesis to get the orientation data f parcels on the conveyor belt.

5.2.1 Data Collections and Annotations

Data collection is a changeable step in the segment of train model. Without data set, it is just a car without diesel we can not move ahead. At the beginning this thesis has very little data its own. With little data, the accuracy of model is not good. Then this thesis search the open source data which can be used to train the data. There is an Open Images which is a data set of approximately 9M images with bounding boxes, objection segmentation masks visual relationships, and localized narratives. In the version of six, the data set is annotated with 59.9M image-level labels spanning 19,957 classes.

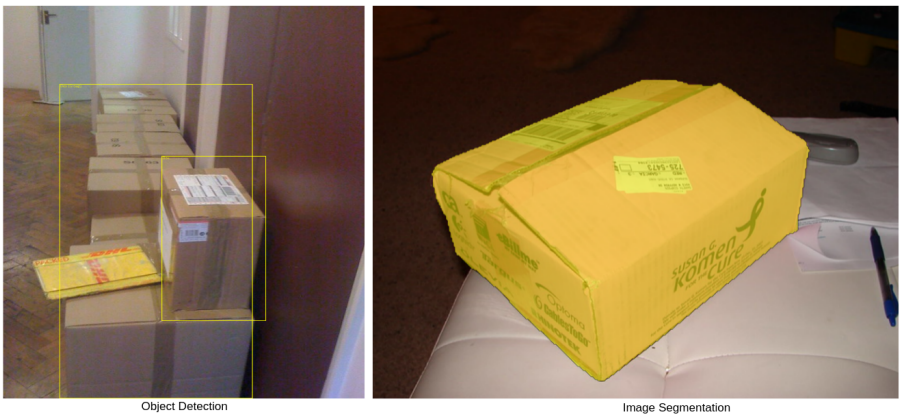

Figure 5.2: Example annotations in Open Images [13]. This open source data store provide the rectangle and segmentation results but in this thesis use only coordinate rectangle data. The segmentation processes delay the whole pipeline.

In the Figure 5.2 shown also the example of annotate image from Open Image for object detection and segmentation. The images have a Creative Commons Attribution license that allows to share and adapt the material, and they have been collected from Flicker without a predefined list of class names or tags, leading to natural class statistics and avoiding an initial design bias. Alina Kuznetsova and Hassan Rom in 2020 validated the quality of the annotations with comprehensive statistics about Open Images data set [48]. There is an another paper in 2019 which describe the method of instance mask used to annotate instance segmentation of Open images and this method is three times faster then traditional polygon drawing tools [49].

This Thesis use approximately 316 image to train the the data. The diagram of data collection with training model shows in Figure 5.3. The train model discuss in the following section. The inputs of download image is class name of desired object, type

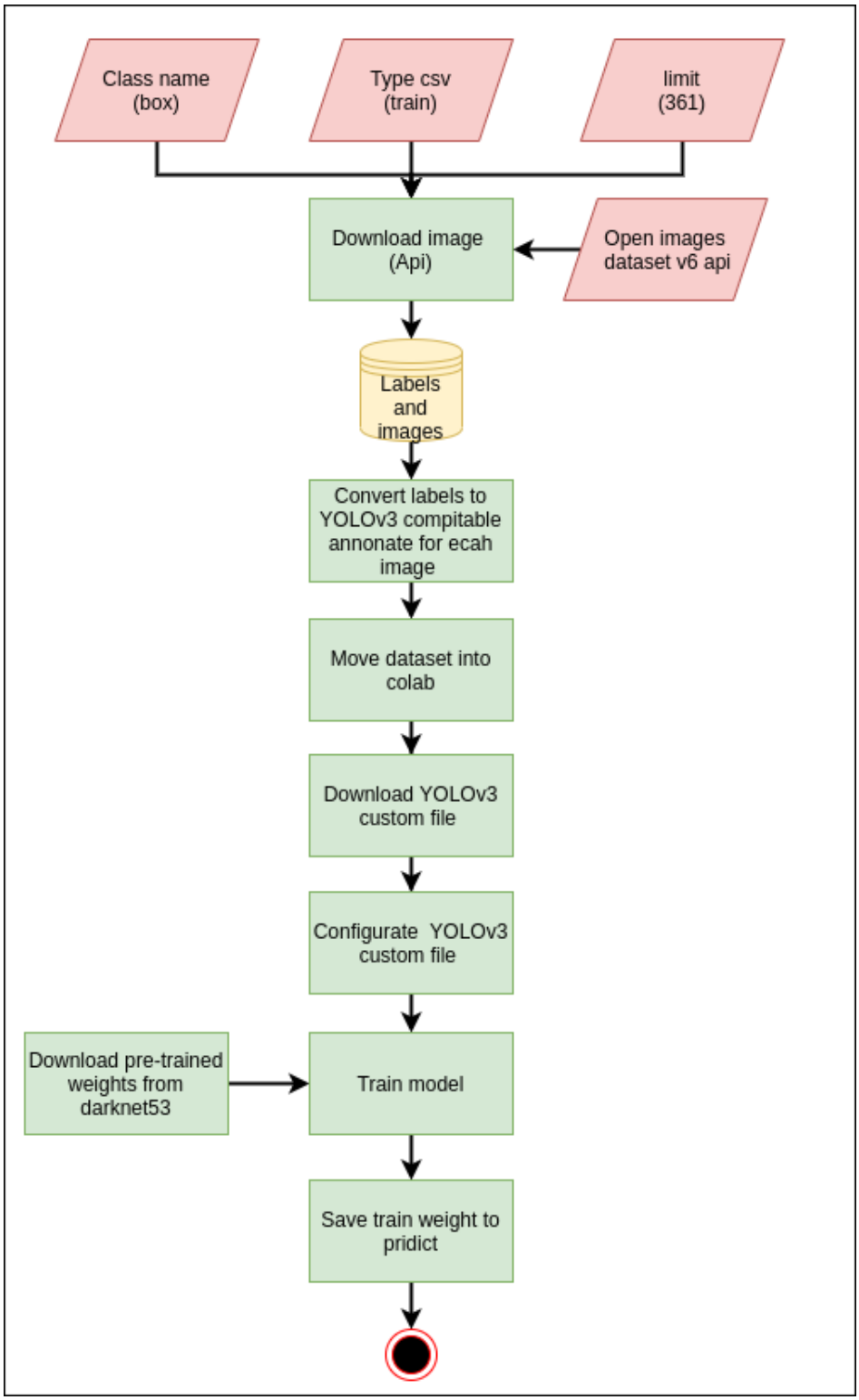

Figure 5.3: The diagram of collection and annotate data with training model. This thesis download data by api from open images data set v6 and annotate data to train the YOLOv3 model.

csv and limit. This is also available to download multi-class objects. The name of class name is Box for this thesis. In open images data sets has two types of csv data. One is train and another is validation with test data. In this thesis, trin is the type of csv. The limit is number of download data. This thesis used 361 data to train the model. The download process is used with the help of api from open images datasets v6. The downloaded imnage store in directory with images and labels. But this labels is not compatible to train the ŶOLOv3 model. The labels are in format of Xmin,Ymin,XMax and Ymax. Papadopoulos (2017) proposed a schema that for object detection only requires the annotators [50]. That thesis annotate the data again according to YOLOv3 model which is shown in the Figure 5.4 [51]. Every image has .txt file which represents the annotation of image object of boxes. Th fist parameter in the txt file is name of the class then left, top, right and bottom. The four different values correspond to the actual number of pixels of the related image. After successfully annotate the data, moved the data to google drive to train YOLOv3 model.

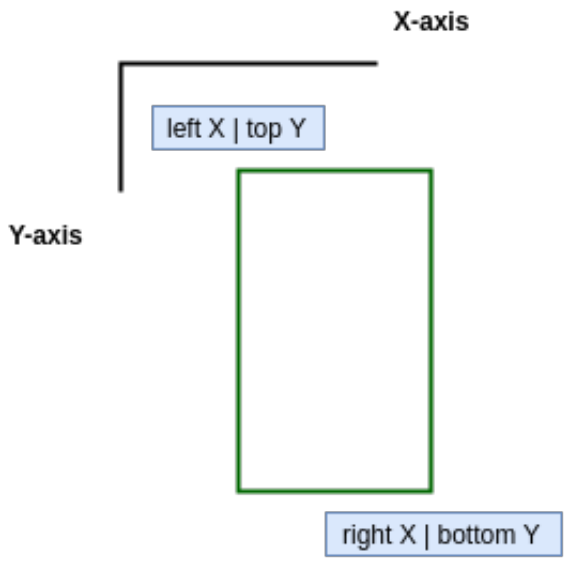

Figure 5.4: Coordinate of bounding box. This information aids in the training of the model in this thesis. Because the YOLOv3 model requires its unique framework for training.

5.2.2 Train and Test Model

This thesis use Google Colab [1] to train the YOLOv3 model to detect the parcels on conveyor belt. Google Colab is a free cloud service hosted by Google to encourage Machine Learning and Artificial Intelligence research, where often the barrier to learning and success is the requirement of tremendous computational power [52]. Its also offer a free GPU. This thesis takes the necessary steps to train the model for parcels detection which shown in Table 5.1.

Step Number	Name of Steps	Details
01	Enable GPU	Using a Colab notebook, enable GPU acceleration so that the YOLOv3 system can process detectors quicker.
02	Data set Upload	This thesis has annotated data sets and at this phase, the data set with annotation files are uploaded.
03	Configurations Training files	This thesis downloads the yolov3 custom.cfg file and use a text editor to change the batch number, subdivision, max_ batches, steps, classes, filters, and random numbers to meet the parcel detection criteria. Then create a file called train.txt that contains the relative path to all of the training images.
04	Download pre-trained weights	This step downloads the weights for the convolutional layers of the YOLOv3 network. By using these weights it helps this thesis custom object detector to be way more accurate and not have to train as long.
05	Train Object Detector	This thesis downloads dark-net from AlexeyAB's well-known repository modifies the Makefile to enable OpenCv and GPU, and then builds darknet. Then, to train the model, replace the data, config file, and pre-download weight with the train data script.

Table 5.1: YOLOv3 model training steps.This thesis uses the Google Colab platform to train the model. It isn't difficult to update the model with new data.

As the curve tends to be closer to zero as the time passes, it will never reach that point since nothing can be perfect or flawless. In general, a total loss value under 2.0 is considered to give accurate results. In the Figure 5.5 also shows the average loss

[1] Google Colab - `https://colab.research.google.com/notebooks/intro.ipynb`.

iterations after trained this thesis model.

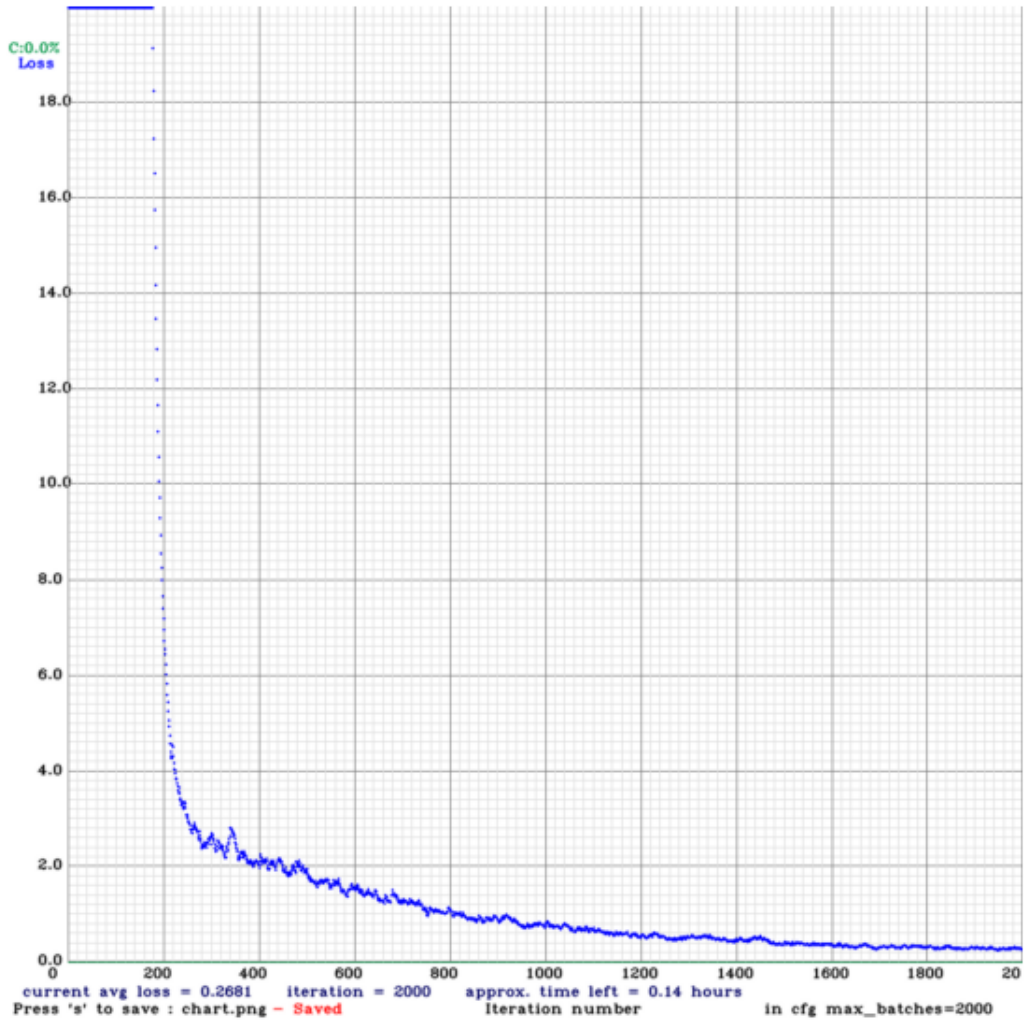

Figure 5.5: Average loss vs iteration from trained YOLOv3 model. The trained model's average loss is less than 2, indicating that it can accurately predict parcels.

As the curve tends to be closer to zero as the time passes, it will never reach that point since nothing can be perfect or flawless. In general, a total loss value under 2.0 is considered to give accurate results. In the Figure 5.5 also shows the average loss iterations after trained this thesis model.

This thesis test the model by different images which give pretty good results on test image which shown in Figure 5.6. This observed that the accuracy is getting low if the parcels are in long distance. The further test with orientation discuss in evaluation section.

Figure 5.6: Test on images by this trained model. After trained the model this thesis tests on various image to predict the parcels with different size and distance between parcels. The small parcels of confidence percent is low.

5.3 Camera Pipeline

Before starting the main pipeline this is thesis set up a separate directory to get the image frame. This thesis has a separate camera pipeline object which is created by a python in the directory of "CameraPipeline". The initial input of this object is the camera serial number and resolution of the frame. By default, the camera resolution is 1280*720. In the init method of this object first create pyrealsense2 object. Then configuration pyrealsense2 object by default value and create "CameraPipeline" object variable. Enable chosen camera with the method of enabling the device in this configuration variable. Besides, that enable stream in the configuration pyrealsense2 object with the value of pyrealsense2 stream depth, camera resolution, rs.format.z16 and rs.format.bgr8. Finally, start the camera pipeline with the configuration pyrealsense2 variable. The Camera pipeline object also has an inheritance to get the colour image and depth image. To use this object first need to create the object and then call the desired object method to get the desired frame. The Figure 5.7 show an overview of the camera pipeline object.

5.4 Orientation Detection Pipeline

The main pipeline to detect the orient of parcels on conveyor belt is orientation detection pipeline or main controller pipeline. After start the program , this main pipeline start. At early stage of this pipeline check the connected camera and extract the serial

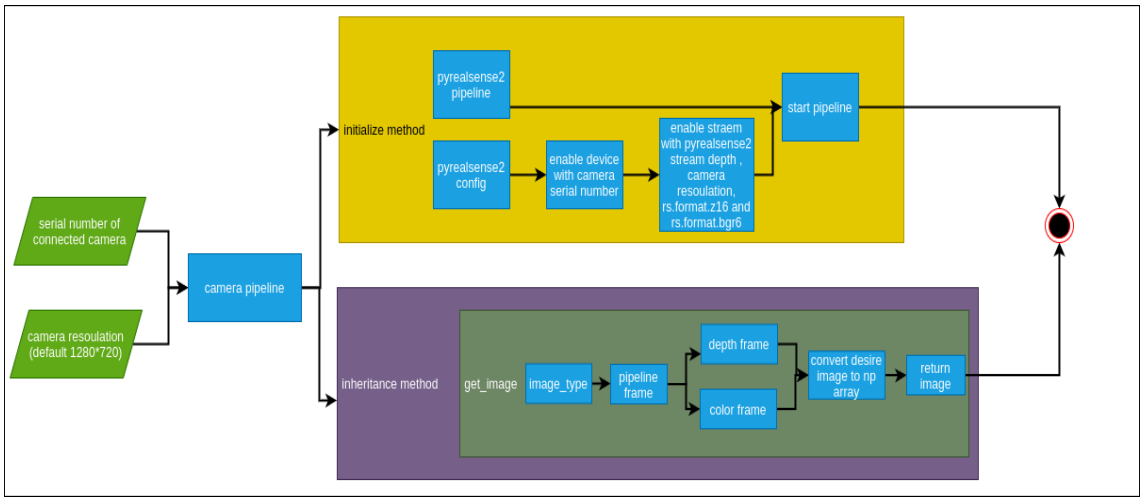

Figure 5.7: Overview of camera pipeline. Its use pyrealsen2 library to get the image from camera by serial number. It has also method to get color and depth image.

numbers. If the connected camera is not none then its run further other else stop the pipeline and close the program. After check and get the serial numbers, this pipeline configure the tensorflow (tf) so that the GPU of the local device compatible with the model. Then it gets the object class name as box in string. Then create the comma separated file (csv) file to save the detection the data if csv does not exist. After that creates two queue object for further two processes (top and Side view pipeline).Then create three multi process object for top view, side view and final decision. Finally start the the processes to detect the orientation of parcels. The top view pipeline are responsible for angle and arrow detection and side view pipeline is responsible for only arrow detection. The final decision pipeline is responsible for combine the data from others two processes (top and side). As arrow detection method is same for top view also , the following sub section split into angle detection pipeline, arrow detection pipeline and final detection pipeline.

5.4.1 Angle Detection Pipeline (Top View)

Detect angle is one of the important features in this thesis implementation. The input of this process is camera serial number, operation name, class name and queue object. From the connected camera serial number list , the first serial number is for this pipeline. Operation name is a string which is used in the imshow to display the frame. Class name is also a string and use for create the model object. Queue is use to put

the data on for final decision that detect by this pipeline.
As this thesis has facility to select the region of interest, at the begging of this process create variable roi is none. Then create the camera pipeline object by giving information of serial number. After that create the YOLOv3 model object and load with this thesis trained model. Then start the while loop which is stop by controller or who is responsible to monitoring the object orientation process. In this while loop, first get color images from the camera pipeline object. If the frame is none then this process neglect the further steps. After that integrate the mouse call back method from OpenCV which is control interaction between the user and the sleeted frame.
After process the selected area from replace the roi variable with selected area. With help of Opencv2, selected are convert to rgb which need to expand dimension for tensorflow. The output from the tensorflow is use for transform image method. The transform method resize the image which need for model. The output from the transform image is use for predict the parcels on frame. The predict method predicts the parcels and return the boundary box, score, classes and number of parcels according to class. As this thesis has one class, classes is not so important in this process.
There is a method in utils which is responsible to draw box for visualize the detected parcel and call the calculation of angle. This method return the the frame after draw the detected parcels and angle of the detected parcels. The utils method draw the boxes according to classes and box numbers. While draw boxes, according to box number also called another method to calculate the angle. The input of the calculate angle method is detected boxes coordinate and number of box. For better overview the process of angle detection shown in Figure 5.8.
The angle detection method extract the frame deepens on input coordinates of detected parcel. Then convert the image to HSV(hue, saturation, value) format with the help of Opencv2 [53]. In the pre-process, this thesis defines the color threshold for top camera view pipeline. With this value, on the HSV format image put the mask. To get the contours of the detected parcel, use the mask image within threshold method from Opencv2 [53]. Using findContours method from OpenCV, this process find the contours and then filter the maximum contour. After that depends on this filter contour this method extract the point of left, right, bottom and top with the help of boundingRect and argument (minimum and maximum) method [53]. From the bounding resct also

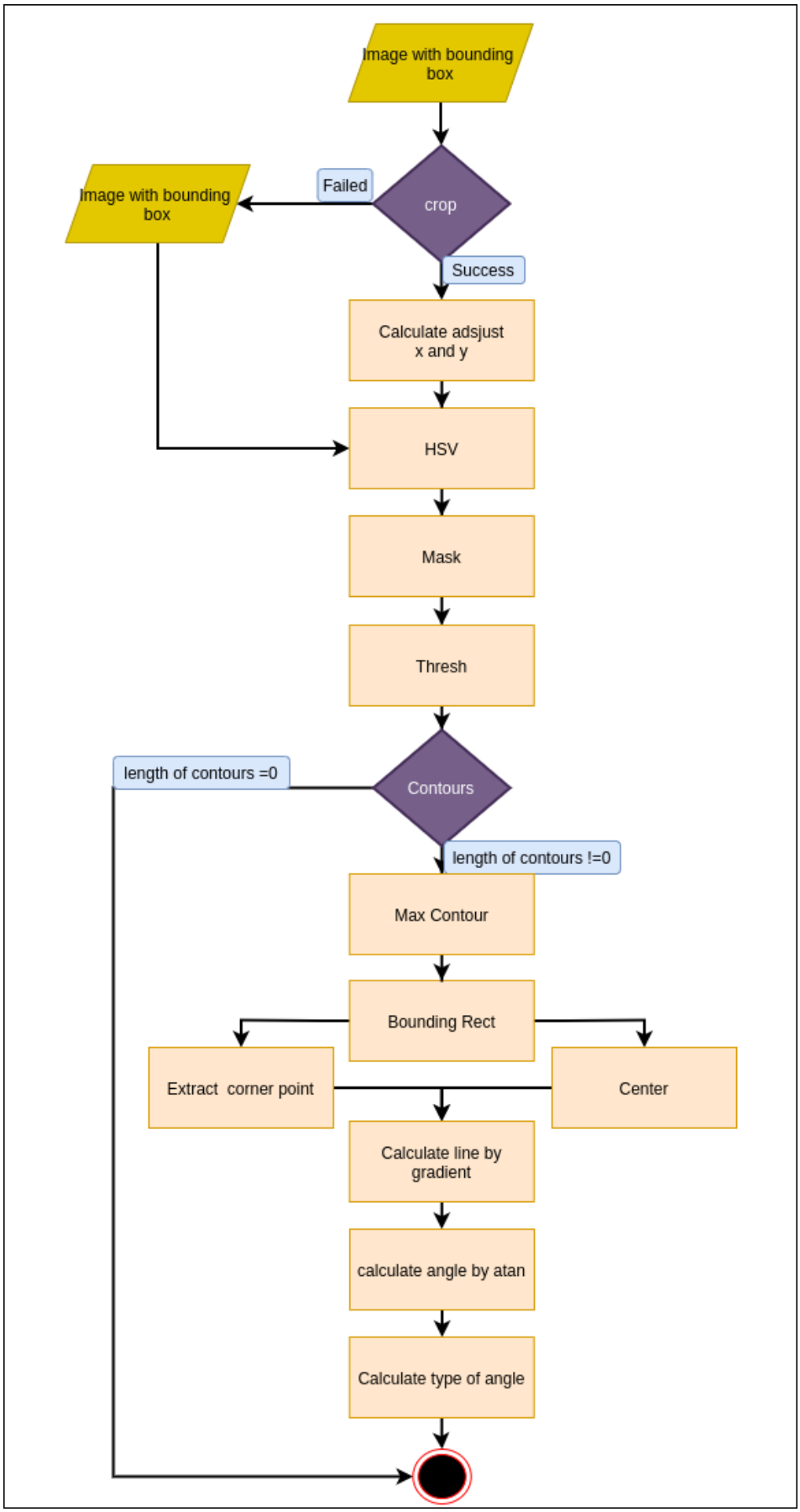

Figure 5.8: Activity diagram for angle detection pipeline. This pipeline examines statistical data from parcels that have been discovered. To detect the orient angle of parcels, this algorithm finds the contour and calculates the bounding rectangle.

calculate the center of this contour. Until now , This method has the corner point and center of detected parcel. To calculate the angle, this thesis use geometric calculation and extract the angle from three point. One is center, another is detected frame middle vertical point.

This thesis draw two line and then calculate the angle. The first line is from center to extra top and the second one is center from middle vertical detected frame point. Firstly this thesis calculate the momentum of two line then calculate the ratio between two momentum. With Math.Atan(ratio) calculate the angle of between three point in radiant. After that the radiant value converted to degree. This thesis calculate only the angle between 0 and 180. That is why after get the angle in degree, if the value is less then zero this thesis minus the value from 180 and the value is the final angle value in degree. Depends on calculate degree, this method also calculate angle type (explain in conception part).

In this pipeline also used same method from arrow detection pipeline to detect the arrow on top view of parcels. The arrow detection method is described in the following subsection side view(arrow detection pipeline). This arrow detection method return the variable of arrow position and direction. With all the information, this angle detection pipeline creates a temporary data store dictionary in python. This dictionary contains calculate angle in degree, angle type, center to visualize in the main frame, bounding box point (use to adjust the angle draw in main frame). The return dictionary value for all detected boxes gathered by utils method and utils method put into the queue. The utils method then return the draw image which is visualize with imshow (from OpenCV). This whole process is run until user shut down.

5.4.2 Arrow Detection Pipeline (Side view)

The side view or arrow detection pipeline is used for detect the arrow with its direction. The method of arrow detection which is used in this pipeline reuse in the top view. The input of the arrow detection pipeline is same as top view but the value is different. The connected serial number is taken the second one from the collected connected serial number list. The class name is same because here also detect the parcel not other object. The queue object is the created for side view pipeline input for put the into queue so that final decision pipeline analysis the data.

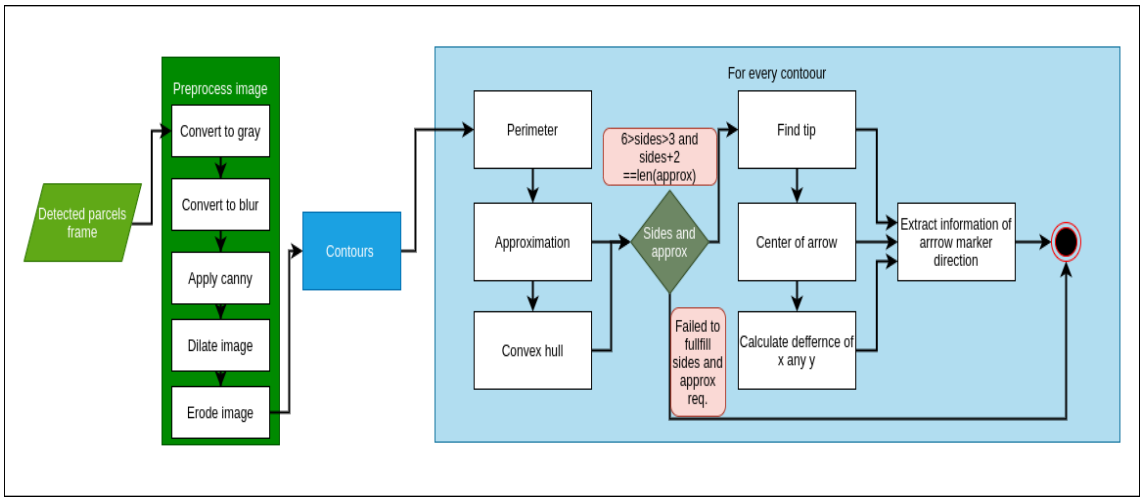

Figure 5.9: Method of arrow marker detection. Used contour based detection to extract the information of arrow direction from every contour.

In this process also first create the camera pipeline object. After that create the YOLOv3 object and load the model with this thesis trained weight. Then start while loop for continuously get the frame by camera pipeline object and detect the arrow.

In the while loop, the first step is get the color image by camera pipeline object from side camera. In this process user has also opportunity to detect the region of interest (ROI). But here also has default frame so that if user is not select, this process can do the further steps. The code of selected the roi is same as top view pipeline. After getting the region of interest (ROI), the image is processed for predict the object by YOLOv3 model. The processes method is same as top view pipeline.

As we use same method to predict the image, the out from model is same as boxes coordinate, scores, classes and number detected boxes. Then this process call the method drawOutputsSideCam which is in the utils script. This drawOutputsSideCam method draw the bounding box around the detected boxes to visualize by the OenCV imshow method. In the drawOutputsSideCam method draw the box according to box number and that time also call another method getArrow to detect the arrow on detected parcel. The getArrow method is responsible to detect the arrow and this method is also used in top view pipeline to detect the arrow on top.

For better overview in the Figure 5.9 show the activity of the getArrow method. The input of getArrow method is relative coordinate and original frame. Then this (getArrow) method crop the detected image. The input of this (getArrow) method can only the detected box area but sometimes in real time bounding box is missing some part of

original box. That is why the original frame is used in this (getArrow) method and then extract the detected parcel.

After that the main steps to detect the contours and its done by another image prepossess called erosion from OpenCV. The function erodes the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken. The function supports the in-place mode and erosion can be applied several (iterations) times. In case of multi-channel images, each channel is processed independently [53].

In the prepossess step, first the detected parcel frame convert to gray and then this grey image convert to blur image with the help of Gaussian Blur. Then on the blur image apply canny edge detection to get the edge. Then apply dilation with canny edge image, kernal(np.ones(3,3)) and iteration number 2. Then do the final step of erosion with dilation of image.

This prepossessed image is used for find the contours on detected image with the help of OpenCV find contours method [53]. There are many contours so that for every contours , this thesis calculate the sides. For the calculation of sides fist step is calculate contour perimeter. The second step is contour approximation which approximates a contour shape to another shape less number of vertices depending upon the precision. Then approximation is used to apply convex hull which is return the hull of the contour. This thesis then calculate the length of hull and called it sides.

If the sides less then six, greater then three and length of approximation is equal to sides +2, then this thesis do the next step, otherwise this getArrow method skip this contour. After passed this condition, next step is to find the arrow tip. This thesis is another method to find the tip which the input is approximation[:, 0, :] and hull.hull.squeeze(). Then find Tip method calculate the tip point by calculate the length and indices depends on the method input.

If the arrow tip is none then also this (getArrow) method skip the contour and continue with next contour. If the arrow tip is not none, then draw the contours and circle on tip because this contour is the target or arrow. Then this (getArrow) method the extract the center point of the arrow or contour to calculate the direction.

To find the direction of the arrow, this thesis use easy calculation of math which also show in the Figure 5.10. If the x point of arrow tip less then x point of center then the

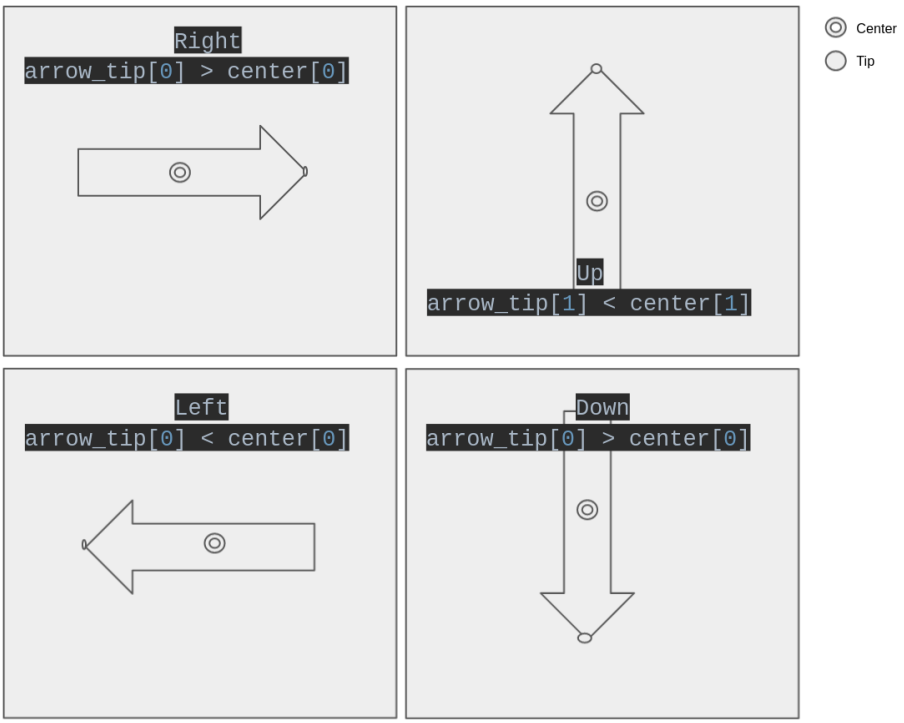

Figure 5.10: Calculate arrow direction depends on center and tip. The X-axis is in charge of detecting left and right. The Y-axis is in charge of detecting the up and down arrow direction.

direction called left and the opposite is right. If the y point of arrow tip less then y point of center then the direction called up and the opposite is down. After that return the direction name as a string to the utils method (drawOutputsSideCam).

The method of drawOutputsSideCam create a dictionary in python and store according all detected parcel with key of position is side and direction is return value from getArrow method. But when the getArrow method use in the top view, the position then top which is define by the angle detection pipeline or top view pipeline. After the final dictionary , this (drawOutputsSideCam) method put that into queue to analysis the final decision pipeline. The while loop is running until its stop by user. In summary, The pipeline of arrow detection or side view pipeline is extract information about the arrow by the following steps: after get the frame inside a while loop from the camera, YOLOv3 predict the parcels, draw the bounding box, detect the arrow direction and put into queue for further analysis.

5.4.3 Final Decision Pipeline

The final decision pipeline is responsible to analysis the data from two others pipeline (top and side) input information. The input of this process is queue objects of top and side camera view pipelines, file path to save the analysis data.
At the beginning of this process, its create the graphical user interface (GUI) object. This object update widow every time in a while loop. In this process also create a while loop so that every time get the data from the queue and analysis the data.
The next is create while loop which is stop only when the main orientation program is stopped. After create this while loop, this process get the data by calling get method of queue object. Then this has separate method to analysis data which is called in this step.
The input of the data analysis method is data from both pipeline (top and side), data save path and graphical user interface (GUI) object. The first step in the data analysis method is get current time from python datetime method. Then create a empty dictionary called finalResult by python to store the analysis data according detected parcels number. The analysis data is for every putted information from both camera pipeline. That is why this data analysis method first calculate total how many box is detected from both cameras. According number of total boxes , extracted information store into finalResult dictionary. For every box, the key of dictionary is rotation angle, position of arrow, direction of arrow, type of orientation, detection confidence percentage for top view or camera, detection confidence percentage for side view or camera, total number of boxes, type of angle and data write time. Graphical user interface (GUI) table also updating while analysis data for every box with box number, rotation angle, arrow direction and position. At this time also write data into input file path with rotation angle, arrow position, arrow direction, orientation type, confidence percentage of top camera, confidence percentage of side camera, total box number, angle type and data write time.
After store all boxes data into finalResult dictionary, update the graphical user interface (GUI) final result which is in bottom part of graphical user interface (GUI). Finally this process update the graphical user interface (GUI) window. If the total box number is zero then this process do not do the loop over boxes number. This process save the by default value which none or zero according to variable type.

5.5 Challenge and Additional Details

There is always challenges objects while giving concept into the real shape. Data collection is the one of the challenge because this thesis has unique object detection which is normally hard to find. At the time of implementation, need to do extra research to find the angle because most of the method is failed and low accuracy. There is another challenge to maintain code flow and arrange the code so that the code is easily under stable with its flow.

All the codes are available `https://gitlab.ips.biba.uni-bremen.de/iris/iris-parcel-orientation-detection` on this repository.

6 Evaluation and Results

Any research project must include an evaluation component. A program's evaluation is a procedure that evaluates it critically. It entails gathering and analyzing data on the actions, characteristics, and consequences of a program. Its goal is to improve program effectiveness, and guide programming decisions [54]. It is also necessary to evaluate the activity in order to understand the story behind the results.

Threshold Type	Angle (Difference between actual and detection)	Arrow Position (Prediction)	Arrow Direction (Prediction)
Good	less equal 5	True	True
Medium	greater than 5	True	True
Bad	greater than 5	False	False

Table 6.1: Threshold type with requirements of calculation overall performance. The importance give to deference between actual and prediction values of angle with arrow direction and position of true and false values.

This thesis evaluates its pipeline to detect and extraction of orientation information of the parcel on conveyor belt in the unloading container system into two steps which is laboratory and field test. Table Table 6.1 shows the three types of thresholds that are used to calculate the algorithm's overall performance in lab and field tests. Three aspects of the method are given priority: angle, arrow position, and arrow direction. The sort of good is determined by an angle difference of fewer than five degrees with arrow position and, if true, direction prediction. The threshold is called medium if the angle difference is larger than five and both the position and direction predictions are correct. If the arrow position and direction are incorrect, and the difference between real and detection is larger than five, the threshold type is bad. The threshold type value is expressed as a percentage and is used to determine how the algorithm's performance responds to the threshold type. The outcomes of this thesis' parcels orientation detecting algorithm are explained in the following subsections.

6.1 Laboratory Test

This thesis at first finalize the parameters of the laboratory test. The parameters are decided by various environment and parcels attributes. The Table 6.2 shows the parameter with its type. The parameters are divided into number of parcels, noises around the parcels, color of parcels, distance between parcels and camera,sizes of parcels and lights which are important to compare the results and calculate the accuracy of the orientation detection pipeline.

Parameters Type	Parameters
Number of parcels	One parcel
	Two parcels
	Three parcels
Noises in environment	Clean
	Medium
	Hard
Parcels color	Own
	Different
Distance between camera and parcels	Less then 50 cm
	Greater then 50 cm
Sizes	Small
	Medium
	Large
Lights	Bright ambient
	Normal ambient

Table 6.2: Experiments parameters for laboratory test. This thesis investigates which factors influence the orientation process based on this parameter.

The results of the orientation detection of the parcel are shown in Figure 6.1. Most of the experiments in the laboratory are done in normal light. To compare with the bright and normal ambient light effects on the algorithm,separately this thesis take experiments parameters of bright and normal ambient light. The algorithm is able to detect the parcels in different environments conditions even the color of parcel is different.

In the Figure 6.2 show the overall orientation result in percentage. This thesis calculated the overall orientation depends on angle, arrow direction and position. If there is a parcel, the percentage of good is better than if scene exists more then two parcels. The distance is also impacts the orientation pipeline. Besides that If in case of bright lighting, the percentage of good is higher than normal light and the most higher percentage

Figure 6.1: Orientation detection in the laboratory. The system detects the angle of parcels in a variety of situations, as illustrated in the image.

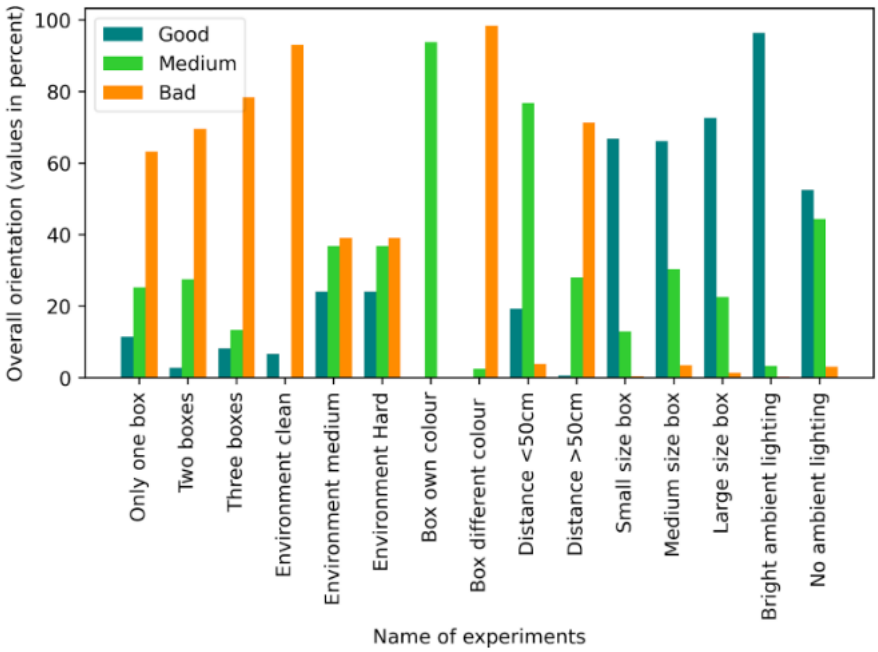

Figure 6.2: In several laboratory experiments, the percentage of good, medium, and bad results for detecting parcel orientation. More good percentage when there is bright ambient light than when there is no ambient light. The ability to determine the orientation of parcels is similarly influenced by distance. When compared to many parcels, the percentage of bad is smaller in exist.

of good is in the environment has bright ambient light.

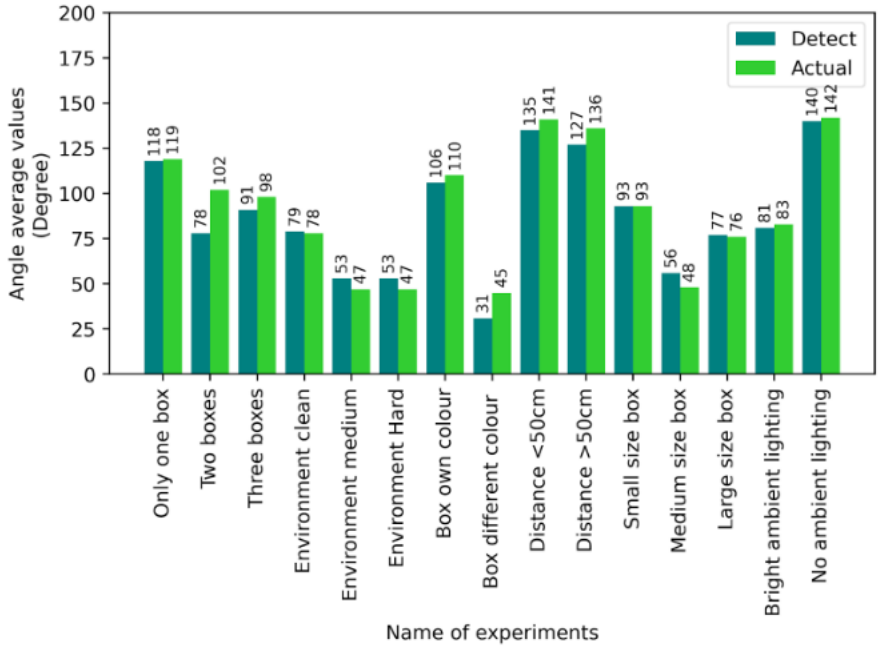

Figure 6.3: On a laboratory test, the actual and average angle of detect parcels were compared. In the medium and heavy noise in the frame, there is a greater difference between the average real and detection angle. The difference between the actual and detection average angle was significantly greater for different colors of parcels.

This thesis also assesses the laboratory test in terms of calculating the standard deviations of detection (std) angles and comparing them to the actual valueFigure 6.4(top). The mean absolute percentage error for all experiments is shown on the downside of Figure 6.4. The total standard deviation values are not very high, indicating that the angle values predicted are not too far from the average value. This thesis compares the standard deviation value with various test parameters to examine how the algorithm detection spreads in different situations. The distribution of detection values is varied in comparison to the number of parcels in one frame, as shown in the image. If one parcel exists in a single frame, the detection values are close to the mean detection values and also to the actual angle value. However, if there are multiple parcels in a single frame, the detection values diverge from the average detection values, and the disparity between the actual and predicted angle values widens. As the number of parcels in one frame grows, so does the mean absolute percentage inaccuracy. When there are fewer parcels in a frame, the angle detection pipeline in this thesis detects

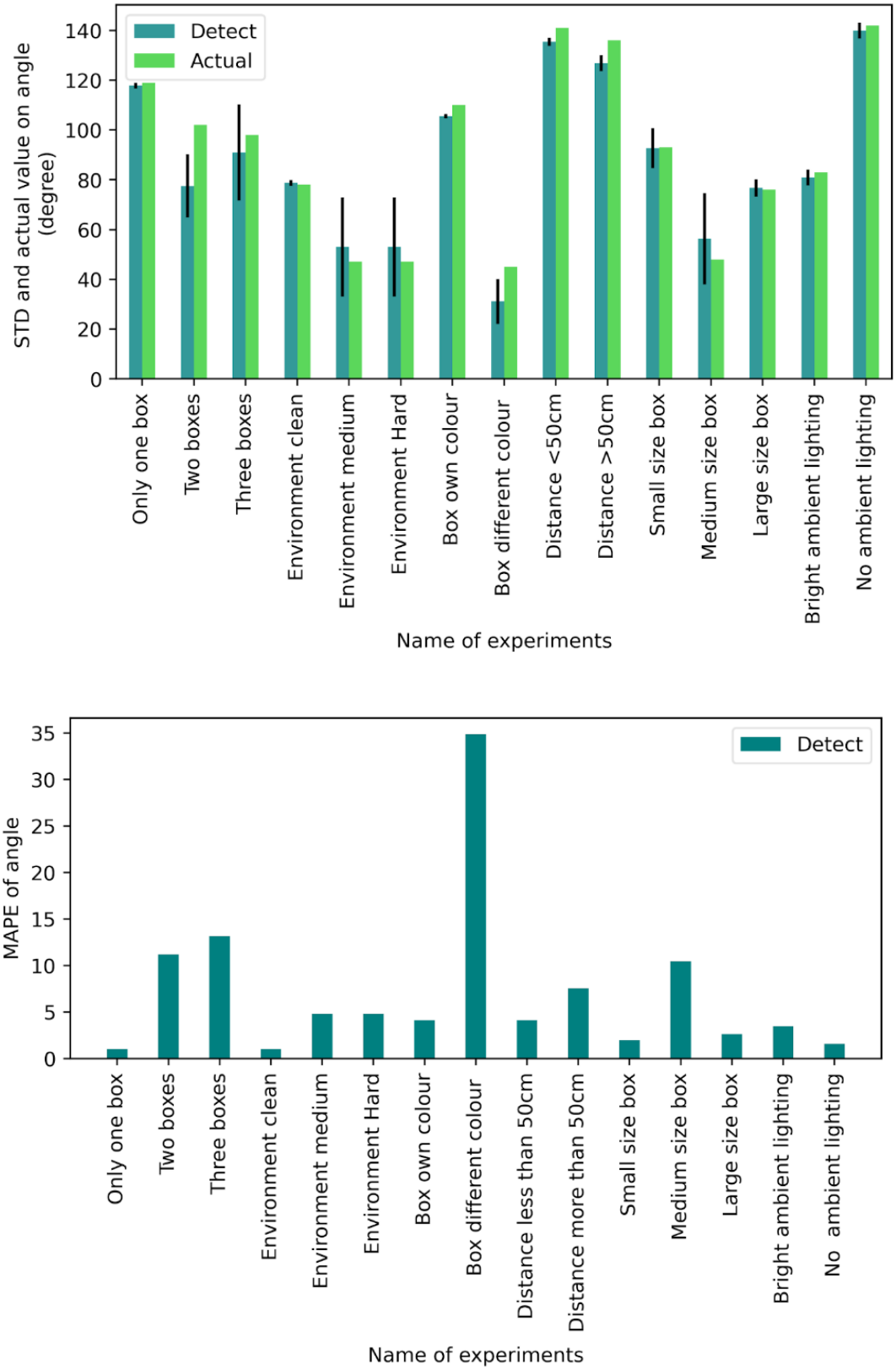

Figure 6.4: Comparison between detecting angle standard deviations and real angle standard deviations on the left, and mean absolute percentage error (MAPE) of detecting angle in all experiments on the right, both for laboratory tests. When the environment is clean and there is only one box in the frame, the value of std is smaller. In the diverse colours of parcels in the image frame, the error rate is larger.

the angle more correctly. The standard values of greater noise in the environment are high as compared to noise in the environment. The angle detection pipeline detects the angle value very precisely with reduced noise in the backdrop of the parcel in the frame. The mean absolute error is likewise low in a clean environment. The difference between the actual and detect angle by angle detection pipeline value has grown significantly as the noise in the picture has increased. The angle detection pipeline is also affected by the colour of parcels. Angle detection for different colours of packages is spread out considerably from the mean detection value, with a large mean absolute percentage error. In comparison to the parcel's colour, the difference between actual and predicted values is similarly large for different colours of parcels. There is no difference in the distance between the parcel and the camera, whether it is a long or little distance. For both long and short distances, the mean absolute percentage error is less than ten. In comparison to small, medium, and big parcels, the detection angle values in the medium size box were spared more. Furthermore, the amount of errors on the frame for medium-sized parcels is also high, indicating that the angle detection pipeline predicts the angle more accurately in small and large parcels. In both bright and typical ambient light, the angle detection pipeline accurately detects the angle of parcels. In the detecting angle, there is little difference in mistake rate between bright and normal ambient lights. According to the findings, more noise in the backdrop of parcels increases as the number of parcels on the frame increases, and medium-sized parcels have a greater impact on the angle detection pipeline's ability to detect the true angle.

The arrow detection pipeline is also evaluated individually in this thesis, as illustrated in Figure 6.5. Lights have a significant impact on determining arrow position and direction. This thesis arrow detection pipeline performs better when there is a bright ambient light present than when there are no ambient lights present. The detection position in several tests is indicated on the left side of this diagram. In bright light, the arrow position pipeline nearly always detects the actual result, but in the absence of ambient light, the pipeline detects various values, such as none, which is not the actual value. In the case of distance, a similar behaviour was seen. The number of detecting real positions is higher when there is less distance between the camera and the parcels than when there is a big distance between the parcel and the camera. In the event

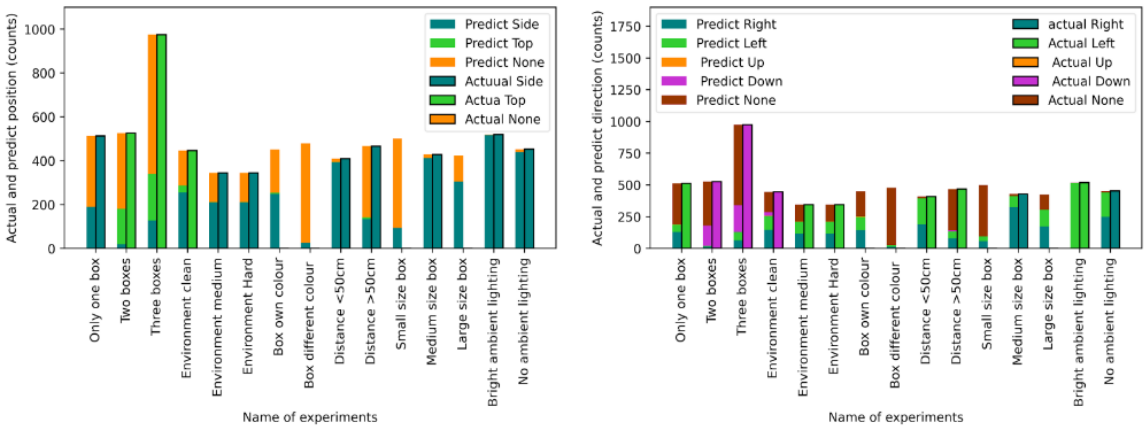

Figure 6.5: Both for laboratory tests, compare the real and detect arrow position on the left, and the actual and detect arrow direction on the right. The arrow detection technique performs well in the presence of medium and hard noise in the frame. Bright ambient lighting aids in detecting the arrow direction more than dim ambient lighting.

of a long-distance arrow detection pipeline, the none value is also detected instead of the real position. The quantity of parcels has an impact on the detection of the arrow's position on the parcel. More than one parcels in one frame makes trouble to the pipeline. On the right of the figure, also shown the bright light in the environment is more strong to detect the arrow direction. In bright light, the prediction of arrow direction is more than without bright light in the environment. The prediction of arrow direction also trouble in case of arrow direction in down on the parcel. The distance and number of parcels makes trouble the arrow direction as prediction position also got problem. The arrow direction is more dependent on the arrow position prediction. Because light, distance, and the number of parcels on the image frame affect both detections illustrated in this figure, if the arrow position prediction is correct, the detection is correct as well. The different color of parcels, small and medium size of parcels actually has not arrow on the parcels. That is why there is no actual bar in the plots. The medium size parcel has arrow and the arrow detection pipeline detect more number of actual value of position and direction of arrow mark than detection of others position and direction. This evaluate also that if there is does not exist none then also pipeline detect different types of values instead of none.

The confusion matrix is calculated in this thesis to demonstrate the performance of the arrow detection pipeline Figure 6.6. The confusion matrix for arrow position is on the

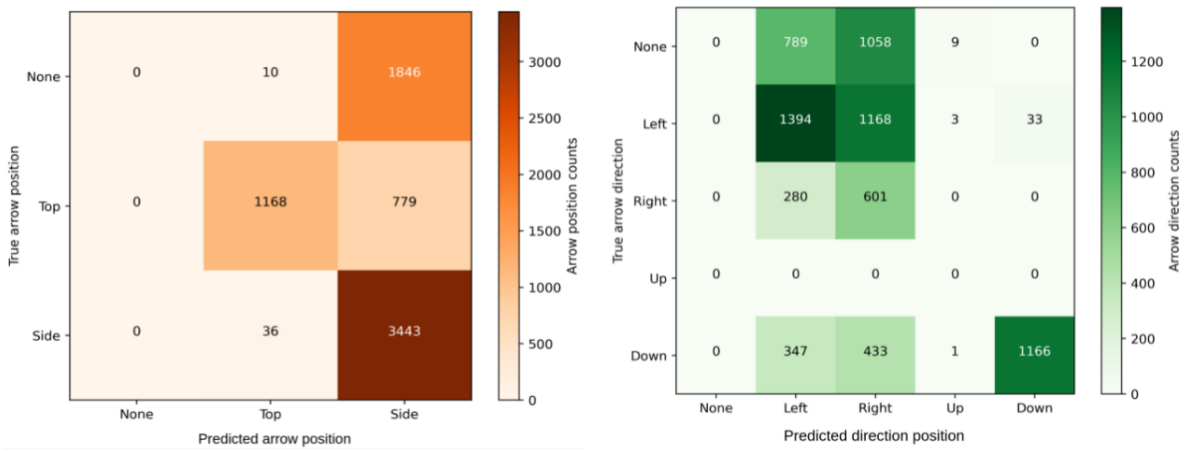

Figure 6.6: For all laboratory experiments, there is a confusion matrix for the arrow detection pipeline. The detection procedure is more accurate if the arrow is on the side. It detects the arrow if the arrow does not exist in both pipelines. The left direction is more difficult to identify than the other directions, but because arrow direction is determined by position, the relevance of arrow position is greater.

left, and the confusion matrix for arrow direction is on the right. The matrix examines the number of true positives, true negatives, false positives, and false negatives in great detail. This thesis algorithm has the highest number of true positives in the position of side, which is 3443. The true positive values for the top and none positions are 1168 and 0. The false-positive values for the positions of side, top, and none are 2625, 46 and 0 respectively. 1178, 5289 and 5426 are the true negative values for the side, top, and none, respectively. According to the side, top, and none positions, the calculation of false-negative values is 36,779 and 186.

This thesis calculates the precision using this statistical data, and the values for the side, top, and none are 0.56, 0.96 and 0. This thesis also calculate the recall the values are 0.98, 0.59 and 0 for the position of side, top and none. The highest precision value is for the position of top and this position correctly predicted positive observations to the total predicted positive observations but the side position predicted high number of positive observations to the all observations in actual position. The position of none makes bad result to this thesis algorithm and indicates as week part of arrow detection pipeline in case of position detection.

This thesis also calculate the statistical values for direction detection of this thesis algorithm. The left direction get highest value of true positive and the value is 1394. The others values of true positive are 0, 601, 13 and 1166 for the direction of none,

right, up and down. The false positive values are 0, 1416, 2659, 13 and 33 for the direction of one, left, right, up and down. This thesis also calculate the true negative numbers that are 5426, 3268, 3742, 769 and 5302 according to the direction of none, left, right, up and down. There is also false negative number for the direction of none, left, right, up and down and that are 1856, 1204, 280, 0 and 781. Based on the numbers of the true positive, false positive and false negative, this thesis calculate the procession and recall for the direction prediction of this thesis algorithm. The precision values are 0, 0.49, 0.18, 0 and 0.97 according to none, left, right, up and down of direction. In the direction of down the algorithm predicts more accurate to the total predicted positive observations. In the direction of right the algorithm also predict more accurate to the all observations in actual direction which is proved by recall calculation. The recall value of the right direction is 0.68 which is okay and the others recall values are 0, 0.53, 0.59 and 0 according to none, left,down and up. The evaluation of this thesis observed that if there is no arrow mark on the parcels, the algorithm is very week to produce the arrow position and direction is none. Further more direction of none also depends on the results of position , because if the position is none then the calculation of direction is not execute. But this algorithm also detect different types of position in case there is does not exist arrow mark and so the calculation method of direction is also execute. Aside from that, the results are acceptable because the small and large size boxes in the test do not have an arrow mark, but this thesis does not discriminate when calculating statistics.

The mean absolute percentage error in parcel detection is shown in the Figure 6.7. Since the side camera uses the same model to detect the parcel on the conveyor belt, this thesis evaluates the top camera detection confidence parentage. In the laboratory, the number of parcels in one frame has little effect on parcel detection because the mistake rate is low and the deference is low (less than five). Also reasonable if there is background noise. When the parcels have varied colours, the model has trouble detecting them. For the varied colours of parcel detection, the mean absolute percentage error is higher than about 20. The distance between the parcel and the camera has no bearing on the ability to detect the parcels depicted in the illustration. In comparison to other parcel sizes, small box detection has a larger error value than medium and large parcel sizes. The error number for both bright and typical ambient illumination is the

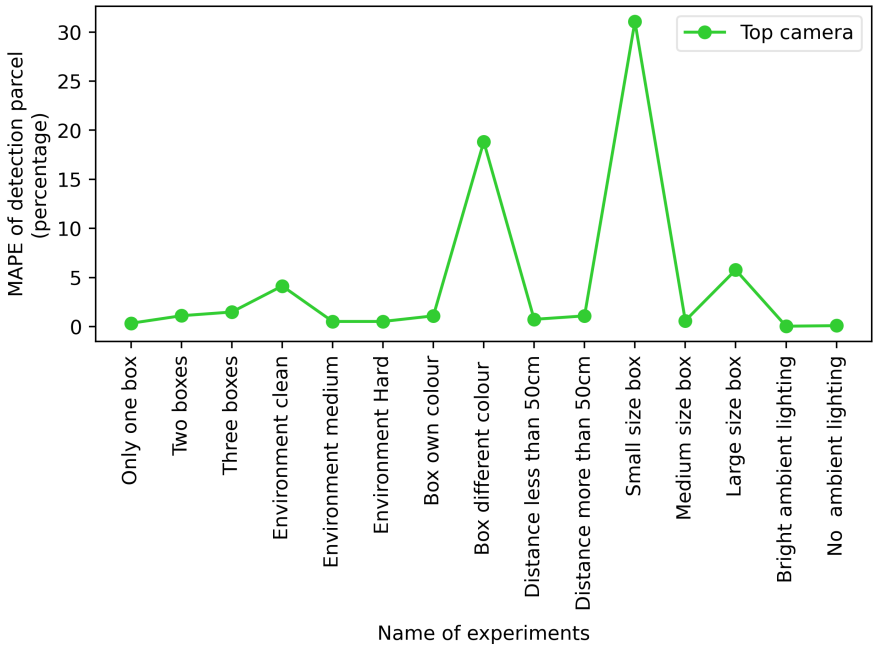

Figure 6.7: In a laboratory test, MAPE of parcel detection. Small parcel detection is less effective with this thesis-trained model. When the colour of the parcel is different, the error rate is also higher.

same, indicating that the model detects parcels accurately in both settings. Overall, the trained model struggles to detect parcels when the colour of the parcels differs and the parcel size is small, according to this thesis.

Finally, this thesis evaluate the laboratory test by calculates overall accuracy of this thesis orientation detection pipeline and separated pipeline accuracy also Figure 6.8. The accuracy of arrow detection pipeline impacts a lot to the overall accuracy of this thesis algorithm. The box detection accuracy is 96 percent which is pretty good compare with other object detection model. The overall accuracy is 80 percent which will acceptable result to detect the orientation of the parcel on conveyor belt.

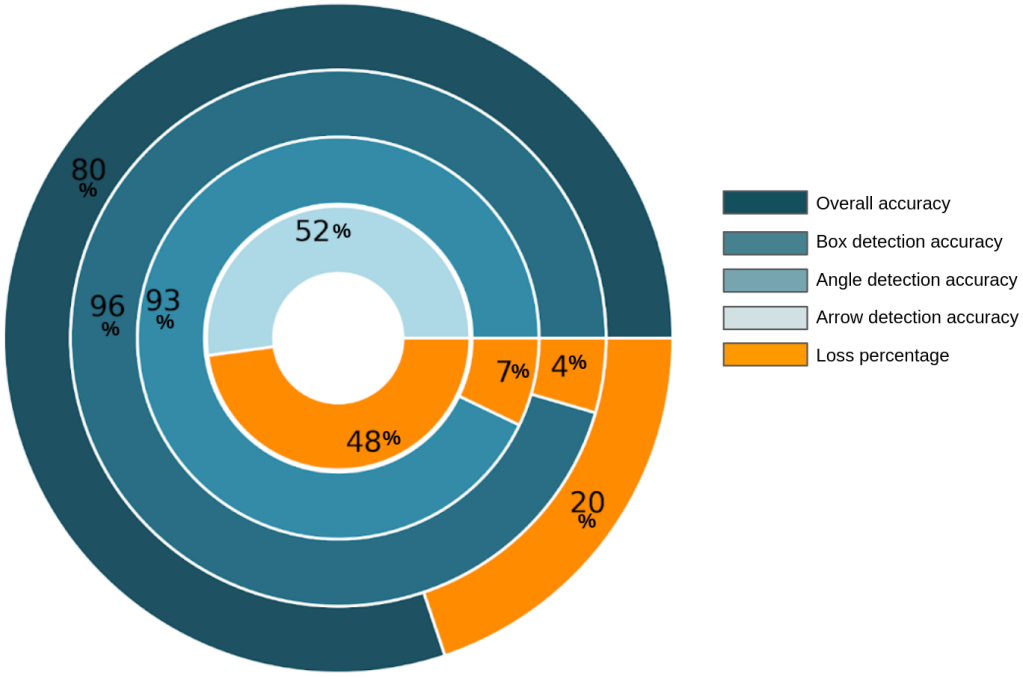

Figure 6.8: Laboratory test accuracy. It reveals that the orientation detecting algorithm is 80% accurate. The entire algorithm is more affected by the arrow detection process.

6.2 Field Test

After doing the evaluation in the laboratory, this thesis evaluate the algorithm in the real environment. In the field test, the parameters are taken slightly different which is important for real time orientation detection. In the field test, this thesis focus on different types of orientations, distance between two parcels, size and speed of conveyor belt to evaluate the orientation detection algorithm Table 6.3.

Parameters Type	**Parameters**
Orientations	Vertical
	Horizontal
	Horizontal-Z
Distance between parcels	Short
	Medium
	Long
	Attach parcels
Sizes	Small
	Medium
	Large
Speed of conveyor belt	Normal
	High
	Slow

Table 6.3: Experiments parameters for the field test. The field test mainly shows the performance of this thesis algorithm depending on orientation type, the distance between parcels, sizes and speed of conveyor belt in a real unloading system.

The orientation types are according to the algorithm detected orientation type which are vertical, horizontal and horizontal-z. This thesis also test on attach parcels, short, medium and long to see how is performance of the orientation detection algorithm depends on distance among parcels. In this test also repeat the sizes of small, medium and large which is important attributes of parcels to test the orientation detection pipeline. At last this thesis test on normal, high and slow speed of conveyor belt to detect the orientation of the parcel. The main evolution and comparison is done without speed of conveyor belt data because the data is not compatible with other tests. But

thesis show the result of the different types of speed result at the end of this section.

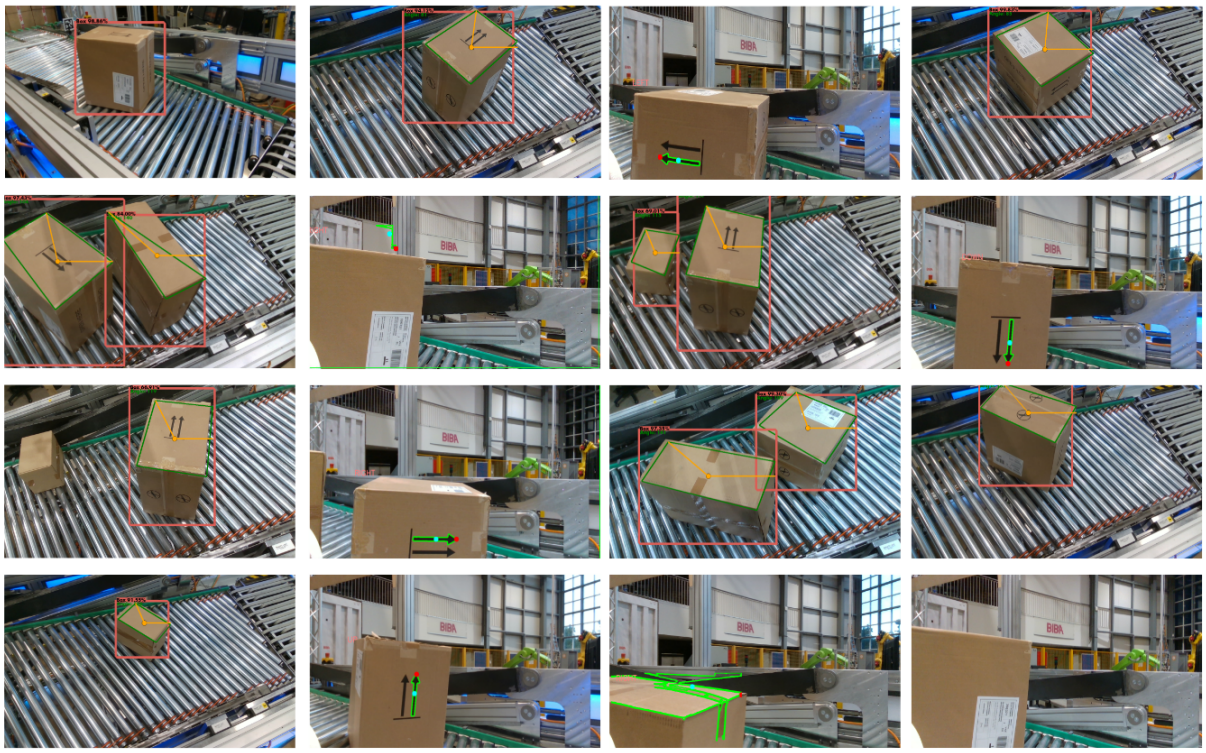

Figure 6.9: Orientation detection on the container unloading system. On the real unloading system, the algorithm for detecting angle and arrow is presented in several tests on the figure.

There is a bright ambient light in the automatic container unloading system. The detection result of the field test shown in Figure 6.9. The tests are depend on various parameter with different size of parcels. The overall orientation detection calculated based on good,medium and bad. Compare to short distance, medium and long distance results are good Figure 6.10. To detect the orientation of parcel in orientation type of vertical and horizontal are good then z-horizontal.

This thesis also shows separately the standard deviation (STD) and mean absolute percentage error in the Figure 6.11 for the detection pipeline of angle. If there is short distance and attach box on the conveyor belt, mean average value is far compared to others position of the parcels on the conveyor belt. In addition to that for the large box is also struggle to detect the accurate angle of the parcel.

The arrow detection pipeline works good on the medium and long distance parcel on the container belt which shows in the Figure 6.12. In the small and medium size box, there is no arrow exist that is why in the Figure 6.12 do not have actual bar.But still it is detected the position and direction arrow. The arrow direction depends on the

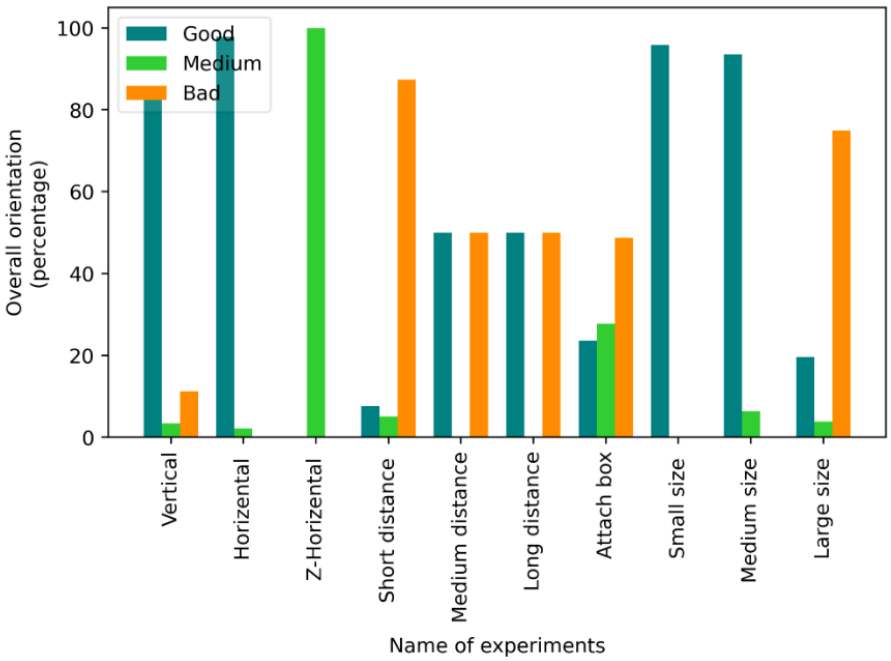

Figure 6.10: Detection of parcel orientation on container unloading system as a percentage of good, medium, and bad. The algorithm is not very good in the case of short distances and large parcel sizes in the unloading system.

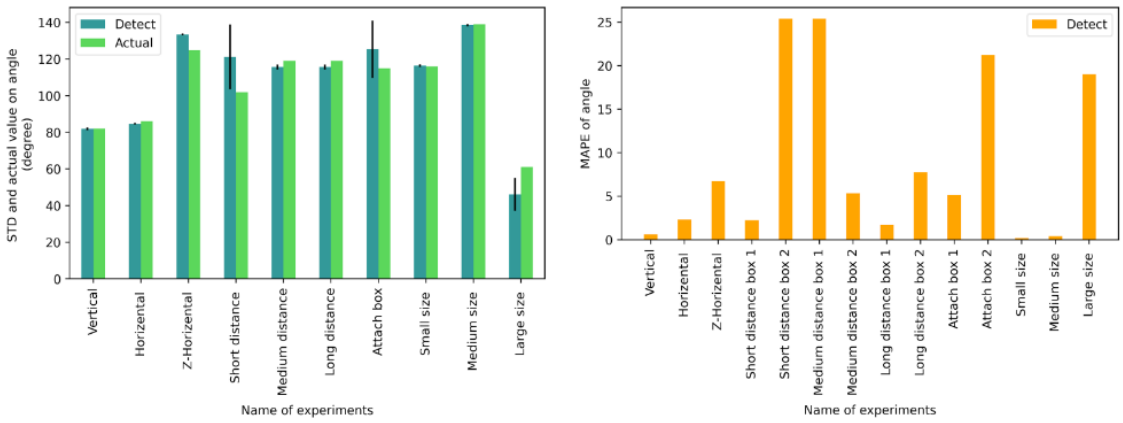

Figure 6.11: Both container unloading systems were used in all experiments: left: comparison of detecting angle and actual angle standard deviations, right: mean absolute percentage error (MAPE) of detecting angle. The value of angle std is very high in the case of a short distance between parcels and attach boxes in the algorithm. Apart from the large box mistake rate, the tiny box error rate is also high.

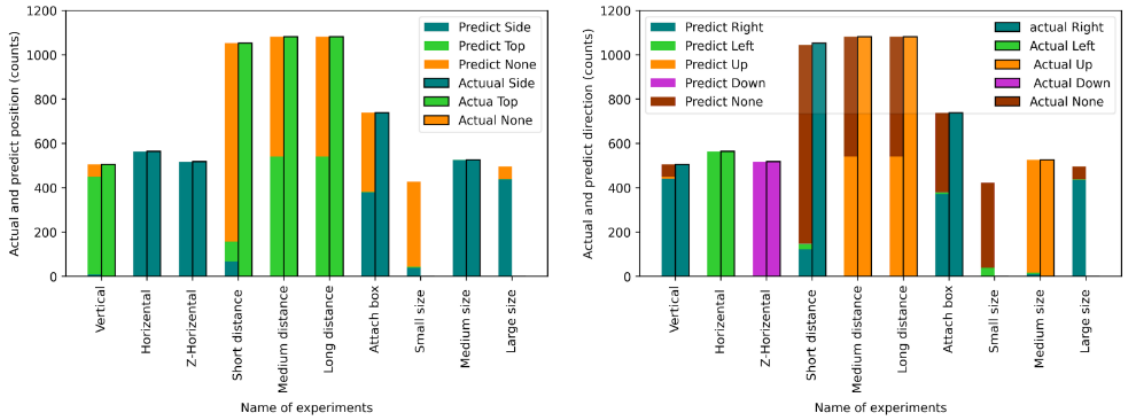

Figure 6.12: Both for laboratory testing, compare the real and detect arrow position on the left, and the actual and detect arrow direction on the right. The process of arrow detection is hampered by a small distance between packets. The arrow detection pipeline's findings are good otherwise.

arrow position because in the algorithm first detect the arrow exist or not then detect the arrow direction.

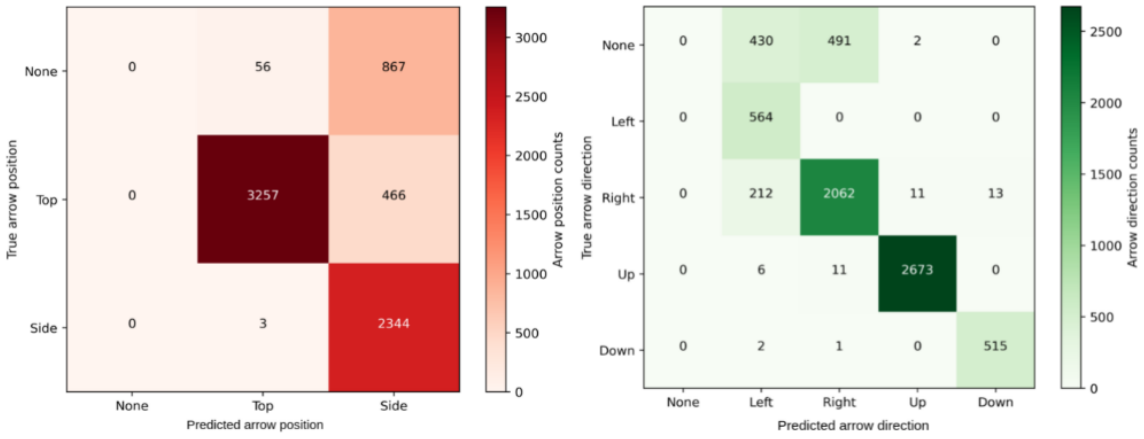

Figure 6.13: For all field tests, there is a confusion matrix for the arrow detection pipeline. The position of the top and side arrows has a higher true positive value than the position of the none type arrow. True positive values are also high in the detection of left, right, up, and down directions.

This thesis also calculates the true positive, true negative, false positive, and false negative, as well as the confusion matrix Figure 6.13 to demonstrate the performance of the arrow detection pipeline technique. The top has the most number of true positive values, with a value of 3257. None and side have real positive values of 0 and 2344. The values of false positive, true negative and false negative for the top position are

59, 3211 and 466, respectively. The values for false positive, true negative and false negative for side position are 1333, 3313 and 3. The values for false positive, true negative and false negative for the none position are 0, 6070 and 923. This thesis calculates the precision and recall based on the data to demonstrate the performance of arrow location recognition in this thesis. According to the position of none, side, and top, the precision values are 0, 0.63 and 0.98. None, side, and top have recall values of 0, 0.99 and 0.87, respectively. In comparison to other types of positions, the top position successfully anticipated positive observations to the whole expected positive observations of position, while the side position correctly predicted positive observations to all observations in actual position, according to this thesis. This thesis, on the other hand, computes the statistical values for the direction. For the up direction, the genuine positive value is higher, and the number is 2673. For the up direction, the false positive, true negative, and false negative values are 13, 4290 and 17. The true positive, false positive, true negative and false negative values for none direction are 0, 0, 6070 and 923. The true positive, false positive, true negative and false negative values for the left direction are 564, 650, 5779 and 0. True positive, false positive, true negative and false negative values in the right direction are 2062, 503, 412 and 236. The true positive, false positive, true negative and false negative values for the down direction are 515, 13, 6426 and 3. The precision for the directions of none, left, right, up, and down is calculated using that data, and the values are 0, 0.46, 0.80, 0.99 and 0.97. None, left, right, up, and down have recall values of 0, 1.0, 0.89, 0.99 and 0.99, respectively. In comparison to other types of positions, this thesis found that the direction of up correctly predicted positive observations to the whole anticipated positive observations of position, but the direction of left correctly predicted positive observations to all observations in the actual position. Aside from that, the up and down directions properly predicted positive observations to all observations in actual position (quite close to the left direction).

This thesis calculates the mean absolute percentage error (MAPE) of the top camera confidence percentage of parcel detection to evaluate the model of parcel detection on the conveyor belt Figure 6.14. The parcel detection model performs well in different orientations on the conveyor belt, but in vertical positions, the model struggles to detect the parcels. The next parameter is the distance between two parcels, and as shown in

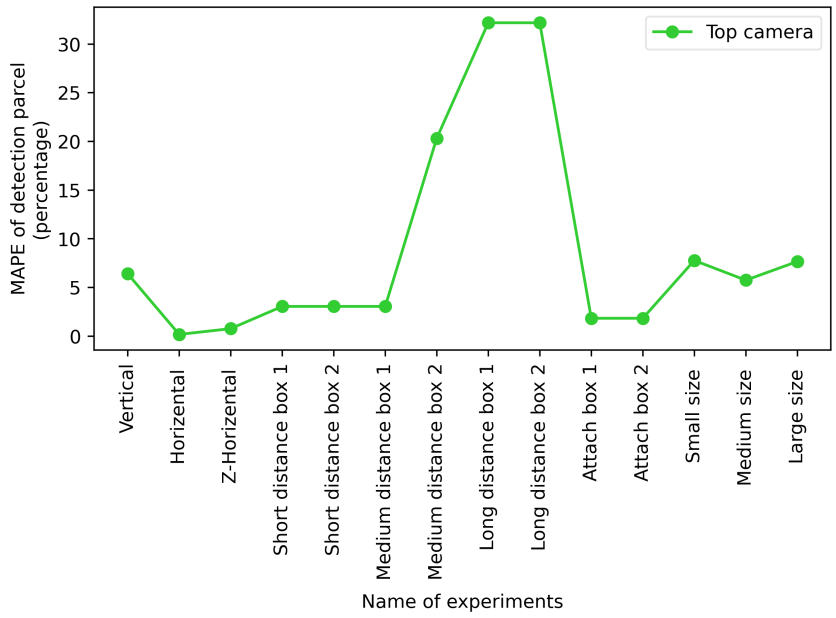

Figure 6.14: On the field test, the mean absolute percentage error (MAPE) of parcel detection was calculated. When there is a large distance between two parcels, the box detection confidence percentage is high. Otherwise, the parcel detection confidence percentage is rather acceptable.

the figure, as the distance value increases, the detection parcels error rate increases as well. When the distance between two parcels displayed on the attach parameter is small, the detection model performs well. When compared to a long-distance between parcels, the mean absolute percentage value in the attached box parameter is low. If the error value for both parameters is low, the thesis tests with two distinct types of attaching box settings. Because the trained model's error values are less than 10 for all distinct sizes of parcels, detecting different sizes of parcels is not challenging. Because of the modest size of the parcels, the thesis' model has no difficulty detecting them. To summarize, except for the great distance between two parcels, the model's performance in detecting the package on the conveyor belt in the unloading system is good.

Finally, this thesis calculate the accuracy of the orientation of detection parcels on the conveyor belt and the percentage is appropriately 77 shows in the Figure 6.15. In the figure also show the separated pipeline of accuracy and show the pipeline is affects to the main accuracy. The lowest accuracy is 47 percent of arrow detection pipeline which affects most to the overall accuracy of this thesis algorithm. The angle and parcel detection accuracy is pretty good which is approximately 91 for both.

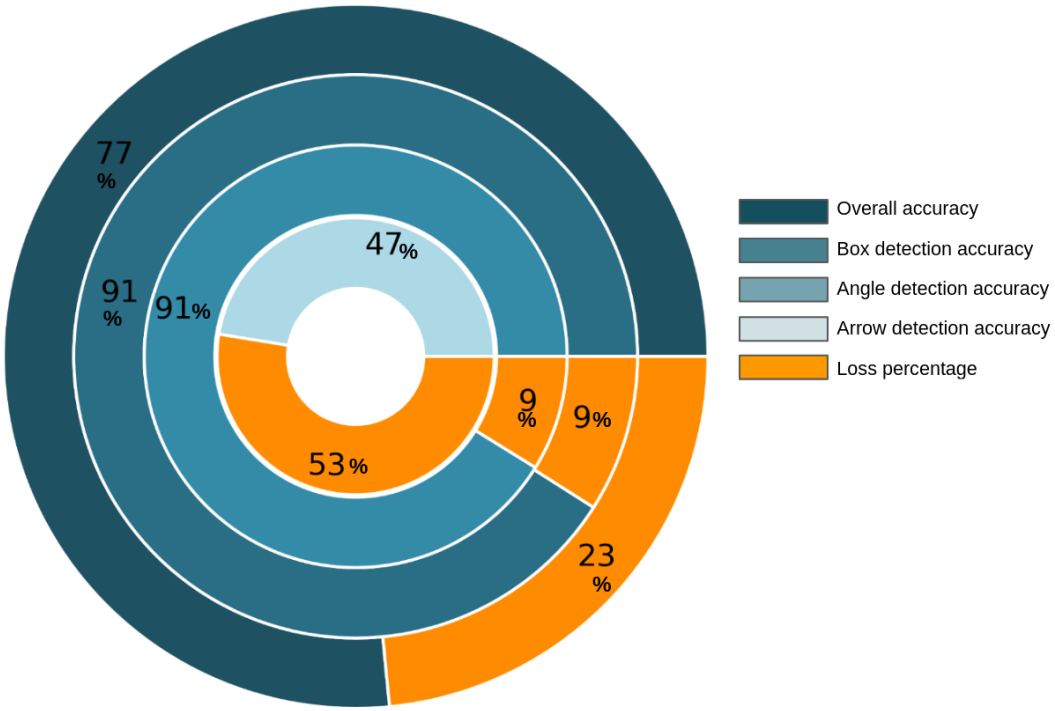

Figure 6.15: The field test's accuracy. The overall accuracy is 77%, with the arrow detection pipeline having the largest impact. The parcel detection model and angle detection pipeline have a strong performance, with a 91 percent accuracy.

7 Discussion and Conclusion

The results have shown that the real-time orientation information of the packages can be extracted by the calculation of the parcels attributes. The number of boxes, lights, sizes, the distance between camera and objects and distance between the multiple parcels on the conveyor belt impacts the accuracy of the orientation detection method which is shown also in the evaluation figure. Although the proposed algorithm is simple, there are tune-able parameters that can impact the performance. In the following subsection, this thesis shows the summary of this work, discuss the meaning of the results and proposed algorithm along with the reliability and limitations of this algorithm. This chapter is finished with a discussion of future work to boost the accuracy of this thesis algorithm.

7.1 Summary

The orientation detection on the conveyor belt in the unloading system boosts the automation of the unloading system to the next level. This thesis proposed an algorithm that detects parcels of different sizes on the conveyor belt and extracts the information of the orientation of parcels. This algorithm trained the YOLOv3 model to detect the parcels and calculate the statistical aspects of parcels to find contours with bounding rectangles to calculate the orientation of detected parcels. The Table 7.1 displayed the laboratory and field test results, as well as the calculation of final results based on the laboratory and field test. YOLOv3 pretrained model is used to train with 316 parcel data to detect the parcels on the conveyor belt and the accuracy of this model is approximately 94% which is reliable for accurate detection of parcels. Then detected parcels on the frame are used to calculate the rotation of angle based on contour-based statistical characteristics of the parcel and the accuracy is also good which is 92%.

On the other hand, this algorithm also detects the arrow marker on the parcels to extract the direction of parcels and the accuracy is 50%. Except for the pipeline of arrow markers, this algorithm is very reliable to detect the orientation of the parcels. This thesis also shows characteristics of the unloading system such as lights and distance affecting the orientation process and suggest orientation detection should need the

Test Type	Overall Acuracy (Percentage round value)	Parcel Detection Accuracy (Percentage round value)	Angle Detection Accuracy (Percentage round value)	Arrow Detection Accuracy (Percentage round value)
Laboratory	80	96	93	52
Field	77	91	91	47
Final (combine of Laboratory and Field Test {(lab+field)/2})	79	94	92	50

Table 7.1: The summary of the algorithm's evaluation findings. The final accuracy of the orientation detection technique is also calculated in this thesis. The final accuracy shows that the arrow detection pipeline has a greater impact on the total final accuracy.

bright ambient light and adjust camera installation on real-time unloading system so that distance is maintained between the camera and parcels.

7.2 Results Interpretation and Limitations of Algorithm

The algorithm of this thesis to detect the orientation of the parcels on the conveyor belt in the automatic container unloading system based on three stages such as parcels detection, rotate-angle detection and arrow detection.Based on this three process, this research work extract the information of orientation of the parcels.

The orientation detection accuracy of this algorithm is approximately 79 percent depends on two different tests (laboratory test and field test). Among the three processes of this algorithm, the accuracy percentage of arrow detection is approximately 50 (laboratory test and field test) which is very low and affects most in the orientation detection processes. The accuracy of the parcels detection is approximately 94 (laboratory test and field test) which is given Strong foundation to this research work. The main and most important process of this thesis algorithm is angle detection pipeline and the accuracy of this pipeline is approximately 92 percent which also make this research algorithm valuable.

The accuracy of the orientation detection algorithm is most affects by arrow detection pipeline.Most of the time is arrow is detected even though there is no arrow exist on the parcels. Besides that if there is arrow on top then this algorithm can detect the

arrow but few times also detect the wrong position. The average detection of the arrow direction results is acceptable for all variants of direction. The lab test results is not so good because of lighting problem in the laboratory but the true positive number is higher then false positive.

The results also vary of this research work algorithm in different environments. The lights are affects to detect the orientation of the parcels. This thesis results also shown that if there is bright ambient lights then the detection of orientation results is more accurate then compare to normal lights. The algorithm is successes in most cases to detect the parcels even more noise in the environment. It is natural that the accuracy is better in the clean environment than more noise in the environment.

The algorithm of this thesis is struggle to calculate orientation of the parcels if there is different parcels color. The performance of detection parcel model is going low while there is different color of parcels.In addition to that the calculation of the angle also struggle if the parcels color is different.

This thesis shown result also for two different distance attributes. One is distance between parcels and another is distance between camera and parcel. The orientation detection algorithm is much affects by the long distance between the camera and parcels in compare to distance between parcel and parcel. The short distance between parcels and parcels is mainly affect the orientation detection algorithm.

It is not so deference of accuracy to detect the orientation of the parcels between small, medium and large box. The accuracy of this orientation detection algorithm also vary to the number of boxes in the frame. The loss percentage is getting high by the increasing the number of parcels in one frame.

Because of lack of parcels, it is not able to test the detection of the arrow position for small and medium size parcels. It can be cause to properly test in all feature of the orientation detection algorithm. It is also affects the overall accuracy of of this thesis algorithm.

Th algorithm can not detect the arrow if is in front part of the parcels. This is one of the limitation of this thesis algorithm. To take care of this part, the algorithm of the orientation detection think none as front arrow position.

In summary, the color of box is the main limitation of this thesis orientation detection algorithm. The carefully handle the position of arrow also affects the accuracy. The

parcel detection model and angle detection calculation pipeline gives the algorithm strength to detect the orientation of the parcels on conveyor belt in the automatic container unloading system.

7.3 Future Work

Detection orientation of objects is challenging work. This thesis proposed an algorithm that depends on the YOLOv3 model and calculation of statistical data of detected objects to extract the information of orientation of parcels on the conveyor belt in the automatic container unloading system. Although the algorithm shows reliable performance to detect the object and calculate the angle, there is still scope to upgrade the performance of this algorithm.

Since the model to detect the parcels on the conveyor belt is performs quite well but it still has scope to improve the model. The increased number of train data will improve the model that can also detect the long-distance parcels on the conveyor belt. Train data can also include more real-time data. More different colours of data can include in train data that can also give the ability to detect more accurate the different colours of parcels. To update the accuracy of angle detection for different colour parcels, automatic colour segmentation can apply at the time of finding the contours to calculate the bounding box.

Finally, the detection of arrow position also need improvements that can contribute to upgrading the accuracy results of the orientation detection. There also can train the model to detect the arrow position so that the orientation detection can also produce three dimensional (front, side, top) information of the direction of parcels on the conveyor belt. However, detection orientation is a complicated process that has not yet hundred percent accuracy. Train model for arrow detection and work with detected data to extract the position and direction arrow in real-time is a challenging task for future work.

References

[1] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, “3d bounding box estimation using deep learning and geometry,” in *CVPR*, 2017.

[2] O. Arif, M. Marshall, W. Daley, P. A. Vela, J. Teizer, S. J. Ray, and J. Stewart, “Tracking and classifying objects on a conveyor belt using time-of-flight camera,” in *Proceedings – The 27th International Symposium on Automation and Robotics in Construction*, T. Brno, Ed. Batislava, Slovakia: International Association for Automation and Robotics in Construction (IAARC), June 2010, pp. 203–212.

[3] T. Yin, X. Zhou, and P. Krähenbühl, “Center-based 3d object detection and tracking,” *CVPR*, 2021.

[4] Jiuqiang Tang and chuoling , “Mediapipe objectron.” [Online]. Available: https://google.github.io/mediapipe/solutions/objectron.html#resources.

[5] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, “Implicit 3d orientation learning for 6d object detection from rgb images,” in *The European Conference on Computer Vision (ECCV)*, September 2018.

[6] T. M. Hoang, S. H. Nam, and K. R. Park, “Enhanced detection and recognition of road markings based on adaptive region of interest and deep learning,” *IEEE Access*, vol. 7, pp. 109 817–109 832, 2019.

[7] J. Yi, P. Wu, B. Liu, Q. Huang, H. Qu, and D. Metaxas, “Oriented object detection in aerial images with box boundary-aware vectors,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2150–2159.

[8] J. Joanna, P. Grzegorz, P. Erwin, M. Micha and K. Dawid, “Fast truck-packing of 3D boxes,” *Engineering Management in Production and Services*, vol. 10, no. 2, pp. 29–40, June 2018. [Online]. Available: https://ideas.repec.org/a/vrs/ecoman/v10y2018i2p29-40n3.html

[9] “Angle - definition with examples,” 2020. [Online]. Available: https://www.splashlearn.com/math-vocabulary/geometry/angle

[10] A. Ivanov and N. Chivarov, “Methods for object recognition and classification for tele-controlled service robots,” *IOP Conference Series: Materials Science and Engineering*, vol. 878, p. 012005, 07 2020.

[11] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” 2018.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” 2018.

[13] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, S. Kamali, M. Malloci, J. Pont-Tuset, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy, “Openimages: A public dataset for large-scale multi-label and multi-class image classification.” *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[14] S. Maji and S. Bose, “Deep image orientation angle detection,” 2020.

[15] C. Landschützer, A. Wolfschluckner, and M. Fritz, “Innovative automated unloading of parcels,” in *TRA 2018 Proceedings*, Apr. 2018.

[16] H. Manne, H. Deecke, J. M. M. Ströh, and W. Symanczyk, “E-commerce und paketdienste" in , hamburg:mru expertise in logistics, pp. 4-7,” pp. 4-7, July 2014.

[17] B. Kaiser, R. Tauro, and H. Wörn, “Automatic and semi-automatic unloading of containers,” *Proceedings of the IASTED International Conference on Modelling, Simulation, and Identification, MSI 2009*, 01 2009.

[18] M. Rudorfer, “Evaluation of point pair feature matching for object recognition and pose estimation in 3d scenes,” 2016.

[19] A. Kirchheim, M. Burwinkel, and W. Echelmeyer, “Automatic unloading of heavy sacks from containers,” in *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 946–951.

[20] J. Iivari and J. Venable, “Action research and design science research - seemingly similar but decisively dissimilar.” in *ECIS*, S. Newell, E. A. Whitley, N. Pouloudi,

J. Wareham, and L. Mathiassen, Eds., 2009, pp. 1642–1653. [Online]. Available: http://dblp.uni-trier.de/db/conf/ecis/ecis2009.html#livariV09

[21] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. [Online]. Available: https://doi.org/10.2753/MIS0742-1222240302

[22] Analytics Insight, “What is image processing : Overview, applications, benefits, and who should learn it,” 2021. [Online]. Available: https://www.analyticsinsight.net/manual-to-object-detection-with-machine-learning/

[23] G. Dougherty, *Digital image processing for medical applications*. Cambridge, UK; New York: Cambridge University Press, 2009. [Online]. Available: http://www.amazon.com/Digital-Image-Processing-Medical-Applications/dp/0521860857

[24] Simplilearn, “Manual to object detection with machine learning,” 2019. [Online]. Available: https://www.simplilearn.com/image-processing-article#:~:text=%20There%20are%20five%20main%20types%20of%20image,Browse%20and%20search%20images%20from%20a...%20More%20

[25] Deepanshu Tyagi, “Introduction to sift(scale invariant feature transform,” 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[26] Mrinal Tyagi, “Hog (histogram of oriented gradients): An overview,” 2021. [Online]. Available: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f

[27] Panth Bhavsar, “Object detection with convolutional neural networks,” 2019. [Online]. Available: https://medium.datadriveninvestor.com/object-detection-with-convolutional-neural-networks-dde190eb7180

[28] Pavan Vadapalli, “Ultimate guide to object detection using deep learning,” 2021. [Online]. Available: https://www.upgrad.com/blog/ultimate-guide-to-object-detection-using-deep-learning/

[29] Ramswarup kulhary, “Opencv – overview,” 2021. [Online]. Available: https://www.geeksforgeeks.org/opencv-overview/

[30] “Intel® realsense™ documentation,” 2021. [Online]. Available: https://dev.intelrealsense.com/docs/python2

[31] Wikipedia contributors, “Template matching,” 2021, [Last edited on 6 May 2021, at 01:22 (UTC)]. [Online]. Available: https://en.wikipedia.org/wiki/Template_matching.

[32] Alexander mordvintsev, “Template matching,” 2013. [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html.

[33] B. Reddy and B. N. Chatterji, “An fft-based technique for translation, rotation, and scale-invariant image registration,” *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 5, no. 8, pp. 1266–1270, AUGUST 1996.

[34] Kuglin, C.D, Hlines, D.C, “The phase correlation image alignment method,” *Proceedings of the IEEE 1975 International Conference on Cybernetics and Society*, pp. 163–165, 1975.

[35] Y. Lin and X. Chunbo, “Template matching algorithm based on edge detection,” in *2011 International Symposium on Computer Science and Society*, 2011, pp. 7–9.

[36] Christoph Gohlke, Matěj Týč, “`imreg_dft`,” 2014. [Online]. Available: https://imreg-dft.readthedocs.io/en/latest/.

[37] Michlvi, “Template matching: find rotation of object on scene,” 2017. [Online]. Available: https://answers.opencv.org/question/179797/template-matching-find-rotation-of-object-on-scene/.

[38] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou, “Deep fitting degree scoring network for monocular 3d object detection,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1057–1066, 2019.

[39] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[40] A. Kundu, Y. Li, and J. M. Rehg, “3d-rcnn: Instance-level 3d object reconstruction via render-and-compare,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3559–3568.

[41] J. Brownlee, “4 types of classification tasks in machine learning,” 2020. [Online]. Available: https://machinelearningmastery.com/types-of-classification-in-machine-learning/.

[42] Z. Liu, H. Tang, Y. Lin, and S. Han, “Point-voxel cnn for efficient 3d deep learning,” in *Advances in Neural Information Processing Systems*, 2019.

[43] Ming Guang Yong, “Object detection and tracking using mediapipe,” 2019. [Online]. Available: https://developers.googleblog.com/2019/12/object-detection-and-tracking-using-mediapipe.html.

[44] T. Hou, A. Ahmadyan, L. Zhang, J. Wei, and M. Grundmann, “Mobilepose: Real-time pose estimation for unseen objects with weak shape supervision,” 2020.

[45] A. Ahmadyan, L. Zhang, J. Wei, A. Ablavatski, and M. Grundmann, “Objectron: A large scale dataset of object-centric videos in the wild with pose annotations,” 2020.

[46] A. Ahmadyan, T. Hou, J. Wei, L. Zhang, A. Ablavatski, and M. Grundmann, “Instant 3d object tracking with applications in augmented reality,” 2020.

[47] M. J. French, *Introduction. In: Conceptual Design for Engineers*. Springer, Berlin, Heidelberg, 1985.

[48] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, and et al., “The open images dataset v4,” *International Journal of Computer Vision*, vol. 128, no. 7, p. 1956–1981, Mar 2020. [Online]. Available: http://dx.doi.org/10.1007/s11263-020-01316-z

[49] R. Benenson, S. Popov, and V. Ferrari, “Large-scale interactive object segmentation with human annotators,” in *CVPR*, 2019.

[50] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari, “We don’t need no bounding-boxes: Training object class detectors using only human verification,” 2017.

[51] A. Vittorio, “Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset,” https://github.com/EscVM/OIDv4_ToolKit, 2018.

[52] kaba Bisong, “Google colaboratory. in: Building machine learning and deep learning models on google cloud platform,” https://doi.org/10.1007/978-1-4842-4470-8_7, 2019.

[53] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[54] M. Q. Patton, *Qualitative research evaluation methods / Michael Quinn Patton.*, 3rd ed. Thousand Oaks, Calif.: Sage Publications, 2002.