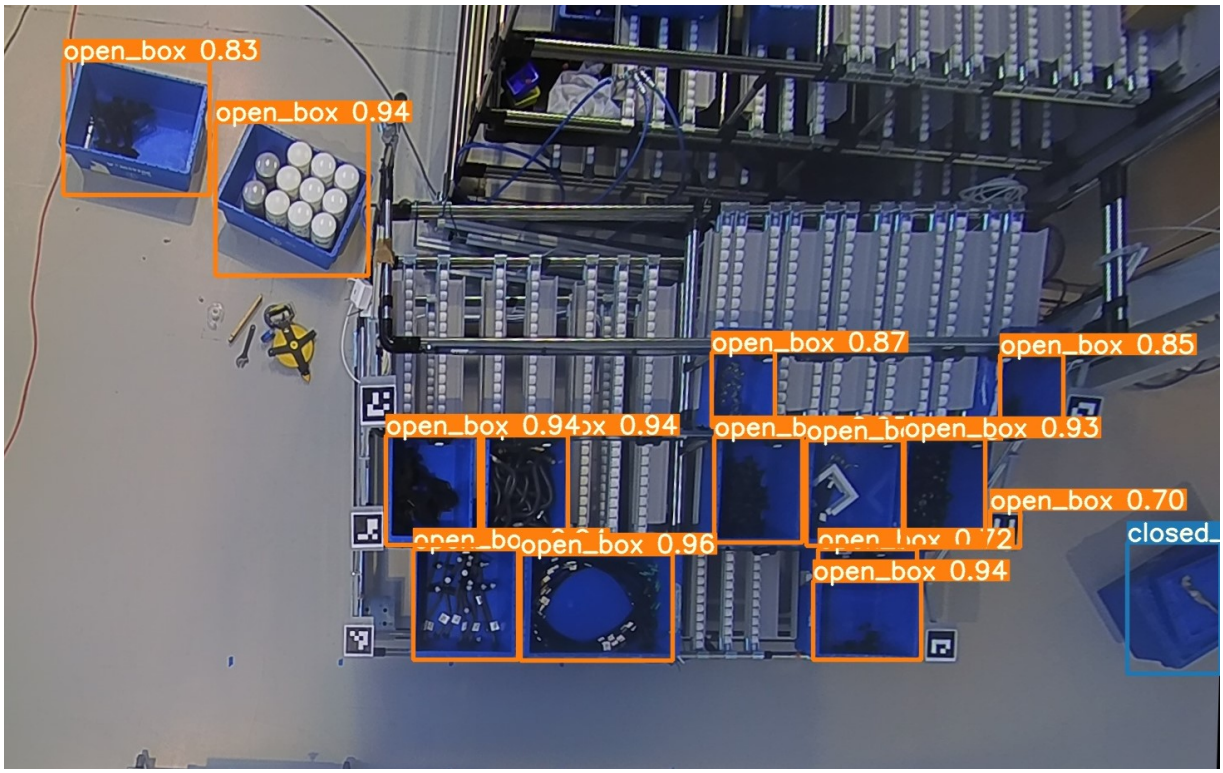
CHALMERS
UNIVERSITY OF TECHNOLOGY



Machine learning to detect the position of storage bins

Bachelor thesis, EENX15-21-19

Alexander Bodin
Ismail Gülec
Iman Shahmari
Khalid Barkhad
Marcus Berg
Gustav Onbeck

Supervisor:
Knut Åkesson

May 14, 2021

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Contents

Abstract **3**

1 Introduction **4**
1.1 Purpose . . . 5
1.2 Delimitations . . . 5
1.3 Ethical aspects . . . 6

2 Approach **7**

3 Detection of bins **9**
3.1 Sub-goals . . . 9
3.2 Object detection . . . 9
3.2.1 Deep learning . . . 9
3.2.2 Convolutional neural network . . . 10
3.2.3 Transfer learning . . . 10
3.2.4 Object detection architecture . . . 10
3.2.5 Regional Convectional Neural Network (R-CNN) - Two staged detectors . 11
3.2.6 One staged detectors . . . 11
3.2.7 Definition of terms . . . 12
3.3 Models . . . 14
3.3.1 You Only Look Once v4 (YOLOv4) . . . 15
3.3.2 You Only Look Once v5 (YOLOv5) . . . 15
3.3.3 EfficientDet . . . 16
3.4 Method . . . 16
3.4.1 Creating dataset . . . 16
3.4.2 Training setup . . . 16
3.4.3 Training . . . 17
3.4.4 Export model . . . 18
3.4.5 Inference . . . 18
3.4.6 Test 1: Unbalanced classes . . . 18
3.4.7 Test 2: Augmentations . . . 18
3.5 Results . . . 18
3.6 Discussion . . . 20
3.6.1 Inference time . . . 20
3.6.2 Training time . . . 20
3.6.3 Ease of use . . . 21
3.6.4 YOLOv5 paper . . . 21
3.6.5 Unbalanced classes . . . 21
3.6.6 Dataset augmentation . . . 21
3.6.7 Occlusion . . . 22
3.6.8 Camera resolution . . . 22
3.6.9 Coordinate precision . . . 22
3.6.10 Classic CV . . . 23
3.7 Summary . . . 23

4 Positioning of bins **24**
4.1 Sub-goals . . . 24
4.2 Theory . . . 24
4.2.1 Fiducial markers . . . 24
4.2.2 Applications of fiducial markers . . . 27

4.2.3 Camera calibration . . . 28
4.2.4 Camera specifications . . . 28
4.3 Bin positioning model . . . 28
4.3.1 Calibrating for lens distortion . . . 29
4.3.2 Fiducial markers . . . 30
4.3.3 Manipulating image . . . 30
4.3.4 Detect boxes . . . 31
4.3.5 Specify plane of each bin . . . 31
4.3.6 Calculate x-axis functions . . . 31
4.3.7 Convert coordinates to WGS84 . . . 31
4.4 Tests . . . 32
4.4.1 Calculating coordinates . . . 32
4.5 Results . . . 35
4.6 Discussion . . . 37
4.6.1 Camera limitations . . . 37
4.6.2 Fiducial markers . . . 37
4.7 Summary . . . 38

5 Evaluation **39**
5.1 Sub-goals . . . 39
5.2 Test . . . 39
5.3 Results . . . 42
5.4 Discussion . . . 47
5.4.1 Fiducial markers not detected . . . 48

6 Conclusion **50**

References **51**

Abstract

Automation in industry can be improved using kitting robots to handle parts in storage bins on leveled shelves. With the final goal of identifying objects in the storage bins and grasping them, the bin localization is a necessary first step. Given that cameras are already mounted in the ceiling, a model that processes the information contained in the images and locates the front center point of each bin is desired. The approach uses machine learning algorithms to determine the bin locations and fiducial markers placed on the front edge of each plane. The markers are used to both convert the distances in the images to real world measurements and to determine a plane associated with the bins. A coordinate for each bin in relation with the fiducial marker with ID 0 can therefore be determined, translated to the WGS84 system and uploaded to a database. The ArUco 4x4 marker was determined to be the best suited fiducial marker for this application when evaluating the detection rate and the error related to distance calculations. The initial comparison of the different computer vision detection algorithms concluded that YOLOv5 was the best fit for use in the continuing stages of this project. Furthermore, YOLOv5 was used for the augmentation comparison, where different augmentations were tested one by one to try and determine which augmentations would be the best for the type of dataset used in a problem like this. The augmentation comparison did not return any conclusive results in general as to what augmentations are potentially better than others, except for cropping augmentation which had a slight increase in mAP. Additionally, a brand new dataset was created, and a new model trained (using YOLOv5) to improve the detection. The evaluation of the model, by inference on test pictures, resulted in a mean absolute error of 12.5 mm for the front center point of the storage bin. Considering this, it can be concluded that the method chosen performs at an acceptable level for the current application of determining front center position of storage bins.

1 Introduction

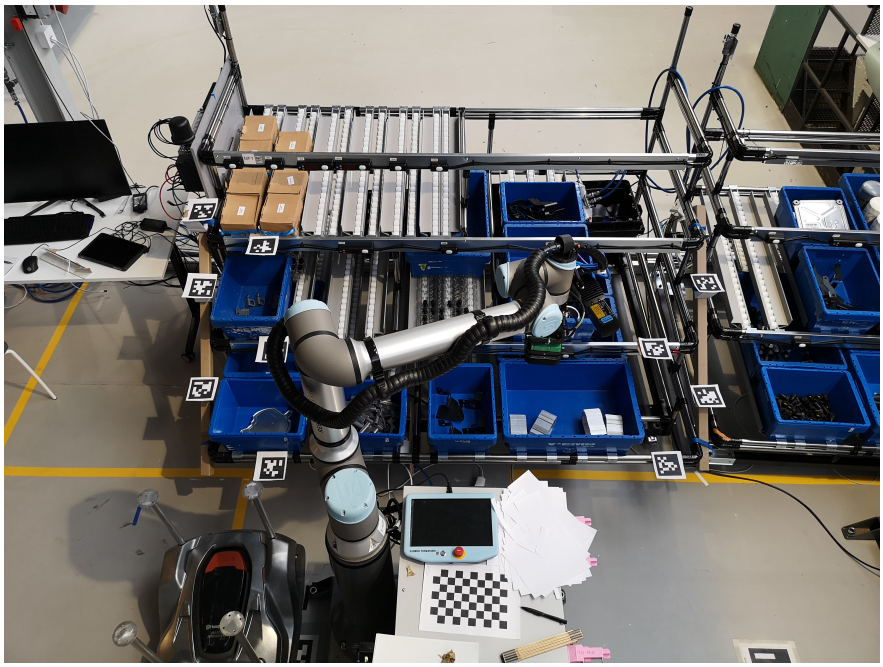


Figure 1: Material rack with associated storage bins used at Volvo. The picture is taken for testing positions of markers.

In the pursuit of increasing productivity in the industry, the automation of production and logistics grows with each passing day. What previously required human workers to accomplish, companies now fulfill using automated robots and artificial intelligence. An example of logistics automation is how Volvo plans to automate the distribution of storage bins. Volvo uses blue bins in the factories to store small details and the goal is to autonomously pick out details and then place them on an ATV (Autonomous Transport Vehicle) that brings the part to the production line. The project goal is to find all the accessible bins on a material rack and determine the position of each bin relative to the shelf. Detection of bins could be accomplished in a variety of different ways but since the end goal is not only to specify the location of the bins but to potentially guide robots, the only accepted method currently is through the uses of cameras. The information contained in a single image could in the future specify the location of multiple bins on a plane, guide a robot arm to pick up parts and place them on an ATV that later use the information in the image to locate obstacles or a desired path.

Translating a location from an image to reality

An image is a two dimensional projection of reality, and to extract information from that projection a certain amount of information has to be known in advance. The proposed method utilises the known dimensions of the shelf the bins are placed on and six fiducial markers, two for each plane. Requiring information puts restrictions on the model, the information chosen is therefore important to consider. Fiducial markers allows for the shelf to be located anywhere in the picture as long as the markers can be read. What fiducial markers will be considered and how they are used will be brought up in Chapter 4.

Computer vision

To allow for a variety of bin locations and bin dimensions, a computer vision algorithm will be used in locating the bins in the image. This algorithm will output a list of coordinates in the image for each bin, from which a real-world coordinate can be calculated. There are a lot of

different methods to choose from when it comes to object detection, in Chapter 3 some state of the art models will be evaluated to create the best detector. The accuracy of a model depends on many factors, of which the dataset quality is a large factor, therefore ways of generating a better dataset will be studied.

1.1 Purpose

From an image locate the storage bins and specify their positions, in both optimal and less optimal conditions. Convert the coordinates to a global coordinate system and transmit the coordinates to a database.

Detection of bins

Acquire an object detection model that detects the storage bins and outputs the pixel-coordinates of them. Evaluate the best model and determine a way to create a better dataset for training.

Positioning of bins

Specify the best fiducial marker for this application. Determine how to calibrate and manipulate the images. Specify the position of storage bins on a shelf.

Evaluation

Determine a position for the markers that are visible from every viable position of the camera while at the same time being safe for workers. Evaluate the model for location of storage bins.

1.2 Delimitations

The task of detecting the position of bins is carried out at and for Volvo, therefore the material rack and storage bins is the type used in Volvo factories (displayed in Figure 1). Material rack dimensions (height, width and length) are known. In the factory the same lighting will be used in testing as in production.

Camera

A dome-camera of type AXIS M3024-LVE is used, which is the one used in the factory. The cameras position will be 4 meters above the ground, centered above the robot. The material rack will be within a radius of 2.5 m from the robot. Position tests will be carried out on a specific type of blue bins used in the factory of Volvo. A portion of the bins top side must be visible. The shelf's position is assumed to be known, a position for the bins will only be specified in a local coordinate system or directly translated from a known point. An illustration of the specified layout (seen from above) is displayed in Figure 2.

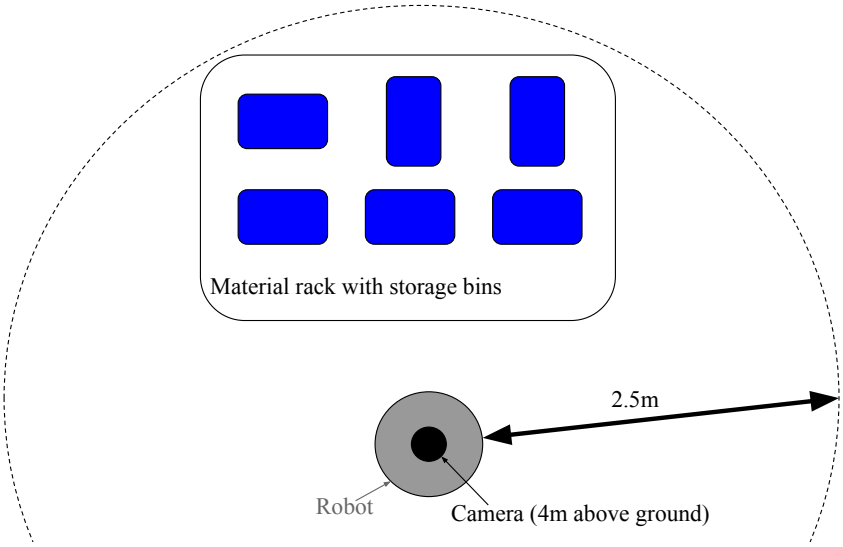


Figure 2: Illustration of the specified layout seen from above (not made to scale).

Detection of bins

There will only be three detection models (EfficientDet, YOLOv4 and YOLOv5) evaluated in this project, these are chosen from the list of state of the art models [1]. The detection needs to have a frame rate of at least one image per second. The input image to the model will be horizontally aligned to the material rack and the image will be taken from the camera position defined in Chapter 1.2 camera position.

1.3 Ethical aspects

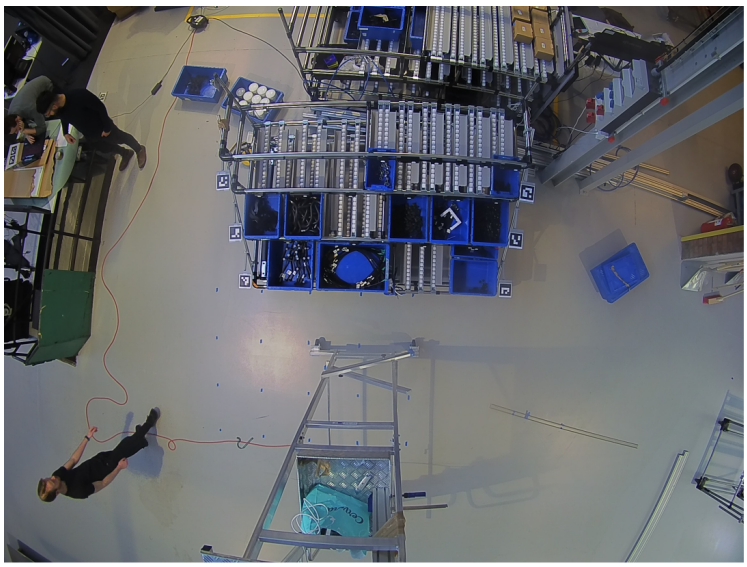
This project involves collaborative robots, object detection using cameras and AI. Despite these technologies mentioned being created with a positive intention of making life easier, they can cause some ethical dilemmas.

One dilemma is the case of object detection. In this project, cameras are used to monitor and detect objects in the factory line. The question that arises is if the technology is capable of tracking objects, isn't it also capable of tracking humans? In this case, as will be thoroughly explained in the report, the object detection model is only trained for the specific purpose of this project, detecting storage bins. However, since cameras are used for this purpose, the privacy and integrity of the people working in view of these cameras come into question. Being tracked by cameras can be considered as a deprivation of ones freedom of privacy for many people. Moreover, object detection as a whole can be used and trained to detect humans and faces as well, also known as facial recognition. The work done in projects like this can potentially in the future pave way for use of the technologies for a more sinister purpose.

Another dilemma is the issue of AI and robots replacing actual humans. From an ergonomic perspective, the robots will likely ease the work done by staff. But at the same time, this facilitation inevitably leads to job loss directly affecting many people whose work can be automated using these methods. This has many implications for the economy and society as a whole, and is something to keep in mind while striving for a more automated future.

2 Approach

The final model for detecting the position of storage bins is presented in Figure 3. The input is an image of a material rack with storage bins and the output consists of coordinates and width for each storage bin on the material rack.




```
{
   "TotalNumOfBoxes" : 10,
   "Shelf1":{
      "NumOfBoxes" : 3,
      "Coordinates" : [57.68897586647415,
11.978731830647634, 62.10000028421304,
0.3904434246639077][57.688976232856405,
11.978706560306426, 62.10000001239002,
0.39044342466390775][57.68897611175851,
11.978714912782877, 62.100000062741124,
0.5872522043055014]
   },
   "Shelf2":{
      "NumOfBoxes" : 5,
      "Coordinates" : [57.68897645621301,
11.978722130830155, 62.20000013713329,
0.3026583480448253][57.68897638056864,
11.978727348204972, 62.20000020929576,
0.29112857578893125][57.6889763059746,
11.978732493120866, 62.20000029571358,
0.28824629875474095][57.68897672761924,
11.978703411167396, 62.20000000313142,
0.31130584326652877][57.68897664882402,
11.97870884591 9, 62.20000002175063,
0.29689379397644433]
   },
   "Shelf3":{
      "NumOfBoxes" : 2,
      "Coordinates" : [57.688976733569746,
11.978733976726295, 62.30000032400538,
0.20024929050229545][57.68897695061422,
11.978719006660022, 62.30000010126793,
0.20551928025813293]
   },
}
```


a) b)

Figure 3: a) Input to the system consists of an image of the material rack. b) The output from the system consists of a formatted text with the coordinates of each bin.

The thesis is presented with an common introduction. Then split up in two parts, Positioning of bins (Chapter 4) and Detection of bins (Chapter 3), with associated theory, method, results, discussion and summary sections. The machine learning algorithm is examined and developed in Detection of bins. The use of camera and fiducial markers as well as merging of the two parts to develop a complete model is examined in Positioning of bins. Testing the model is achieved in the Evaluation (Chapter 5). Lastly a conclusion for the two parts, Positioning of bins and Detection of bins, and the Evaluation is formulated. The split up and approach to the purpose of developing a model that can detect the position of storage bins is illustrated in Figure 4.

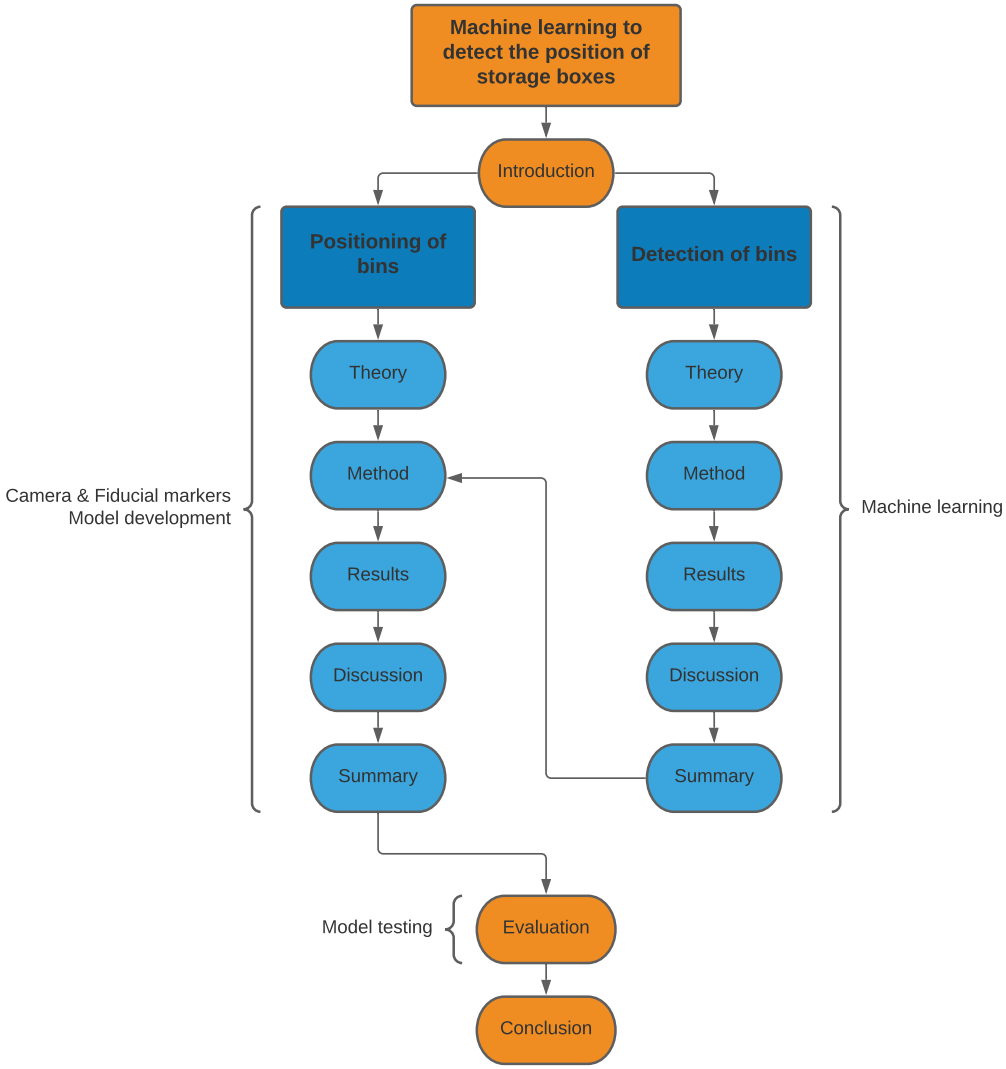


Figure 4: Flowchart of approach used in this thesis.

3 Detection of bins

In this sub-project the focus will be on the detection model. Different object detection models will be tested and evaluated, additionally a comprehensive explanation of how they work will be presented. How to design a dataset to increase performance for the models will also be examined.

3.1 Sub-goals

The goals of the Detection of bins part can be stated as following:

- Train three different models (EfficientDet, YOLOv4 and YOLOv5) for detecting blue bins.
- Given same training material, evaluate top ranked detection models.
- Compare different augmentation methods for the purpose of creating a better dataset for training.

3.2 Object detection

Object detection [2] is a sub field of machine learning and computer vision which deals with both classification and semantics of the input (images). In other words detecting objects and classifying which type of object it is (car, human, train, bin, etc.). There are different approaches for detecting objects in images and these can be divided into two main categories which are neural network-based or non-neural approaches. This project will only focus on the neural network-based methods rather than traditional non-neural methods.

3.2.1 Deep learning

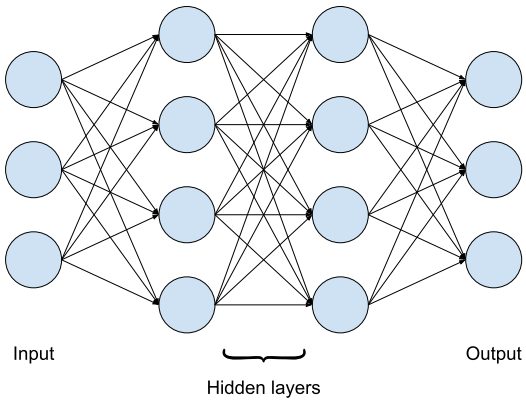


Figure 5: Neural network, consisting of a set of interconnected nodes, called neurons.

Deep learning [3] is a form of machine learning which refers to the process of tweaking the weights in a deep neural network. The term “deep” refers to the amount of hidden layers in the neural network (example of a neural network is shown in Figure 5). Most classical neural networks usually have two to three hidden layers and a deep learning model can have 150 layers of neurons. Because of the large amount of neurons a deep learning model require lots of labeled data to train, but in exchange can handle more complex data, for example images and video. A deep neural network where every node in one layer is connected to every node in the next layer is called a fully connected network (FC) and is a part of what is included in most object detection models.

3.2.2 Convolutional neural network

The FC-layers are often combined with a set of convolutional layers creating a convolutional neural network (CNN) [3]. A convolution is a filter for the data to extract different aspects. In the case of an image, a typical convolution could be edge-detection, where different filters will extract vertical, horizontal, or diagonal lines. In most new models these filters are not predetermined but instead learned by the training process together with the neural network weights. The network itself will learn to find abstract features of the images, which a human would not be able to interpret, but by weighing these abstract features together in the FC-layers the image can be classified.

3.2.3 Transfer learning

Transfer learning can dramatically decrease training time and increase accuracy of a model [4], as well as not require as many pictures for the dataset. This works by freezing some of the "teacher" layers and appending new "student" layers to the end, which are learnable, and used to classify the new categories [4]. Transfer learning works best if the pretrained teacher model solves a similar problem to the specific problem of the student and needs to be chosen carefully. When training a neural network a lot of time and resources are needed to generate an accurate model. Most object recognition models share a lot of general features, for example: lines, shapes or textures. A neural network can be trained on a large generic dataset from which the weights are extracted and used to recognise common features. By using a pretrained base model, called teacher, a new specialized model can be created, called a student, which only need to learn how to combine the features and not train completely from scratch.

3.2.4 Object detection architecture

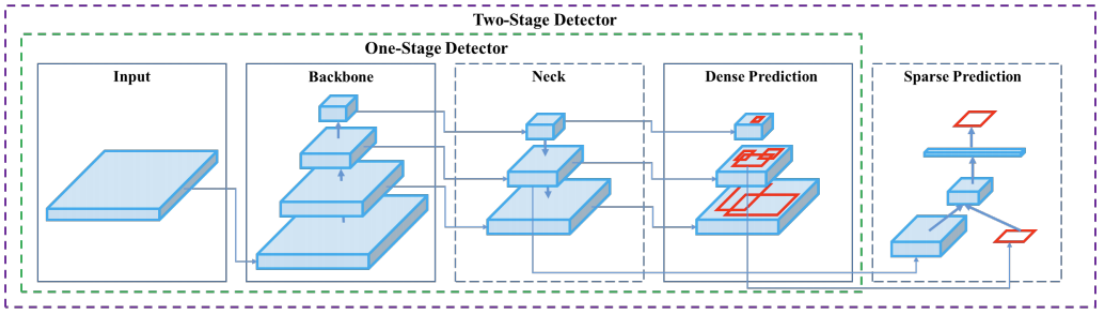


Figure 6: A modern object detector architecture, image from [5].

Modern object detectors are built up from multiple parts (see Figure 6), a backbone, neck and head. The backbone is a network pretrained on the dataset ImageNet, some common backbones are VGG, ResNet, ResNeXt or DenseNet [5]. By using a pretrained model transfer learning is utilized, this extracts essential features which will significantly speed up training and improve inference accuracy by giving the custom model a good starting-point to build upon.

The head of the detector is the part that makes the prediction of the class and bounding box of the object. There are two types of heads, one-stage and two-stage detectors. A two-stage detector works by first finding a region of interest or bounding box and then passes this region into a classifier to find the object class. An one-stage detector does these two things in the same pass, both finding the bounding box and the class. Examples of two-stage detectors [5] are R-CNN or fast R-CNN and of one-stage detectors are YOLO or SSD.

The neck is a part of the model between the head and backbone which serves to collect and fuse features [5] examples of this are FPN (feature pyramid network), PAN (path aggregation network) or BiFPN (bi-directional feature pyramid network).

3.2.5 Regional Convectional Neural Network (R-CNN) - Two staged detectors

The Regional Convectional Neural Network [6] can be categorised as following:

R-CNN: Region proposals stands for 2000 regions extracted from an image. Proposed by Ross Girshick these regions are calculated with the selective search algorithm limiting the number of regions to feed into the convectional neural network (CNN). The CNN extracts the features of the images and outputs a dense layer which will be fed into a SVM to classify the object within the candidate region proposal.

Fast R-CNN: Developed by same author Fast R-CNN is an upgrade of R-CNN. This time instead of feeding regions calculated from the selective search algorithm, images are directly fed in the CNN which generates a convolutions feature map which will be processed. This approach makes it faster because 2000 proposal regions aren't fed in the CNN every time. Convolution operation is done only once per image and feature map is extracted from it.

Faster R-CNN: The selective search algorithm is used in both previous versions. Though selective search algorithm has a huge disadvantage and that is the time it takes to operate. This problem is fixed in Faster R-CNN by letting the Network itself select the regions. This was accomplished using an algorithm written by Shaoqing Ren (2016).

3.2.6 One staged detectors

Single Shot MultiBox Detector (SSD) [7]. As the name suggests the architecture is based on single shot which is the same as it being one-staged so the localization and classification is done simultaneously. Multibox stands for the technique for bounding box regression developed by Szegedy et al. Lastly detector means that the architecture is an object detector.

YOLO: unlike the RCNN [6] family YOLO [8] does not use regions. The name stands for You Only Look Once and the way it works is that it looks at the entire image once and splits it into grids. Within the grids it takes bounding boxes and for every bounding box the network outputs a class probability and offset values for the bounding box (see Figure 7). If the bounding box class probability passes a threshold it gets selected and used for locating the object in the image.

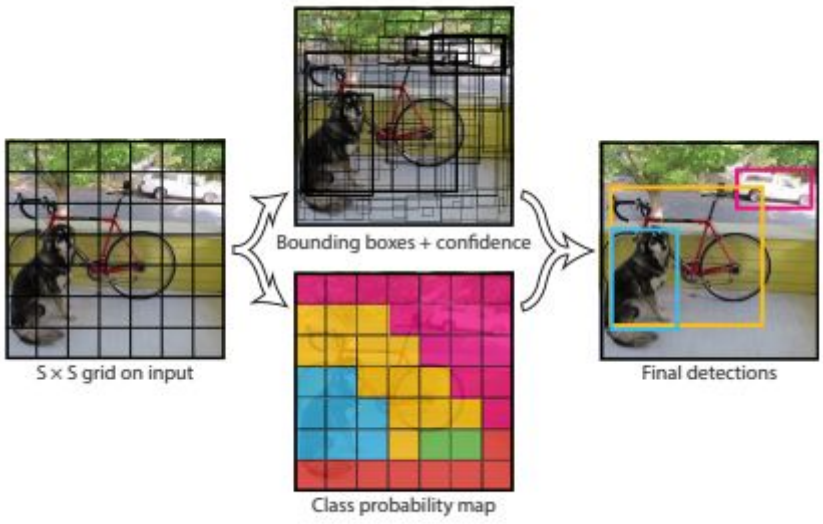


Figure 7: Classification and bounding box prediction of a single shot detector. Image source [8]

3.2.7 Definition of terms

IoU [9], short for intersection over union, is the proportion of the area for which the predicted bounding box overlaps with the ground truth, to the total area between the predicted box and the ground truth. IoU is calculated with the formula 1 and Figure 8 gives a graphical explanation of what the intersection and union is. A common threshold used is IoU of 0.5 or 50%.

$$\textit{IoU (Intersection over Union)} = \frac{\textit{Intersection (Area of Overlap)}}{\textit{Union (Total area)}} \tag{1}$$

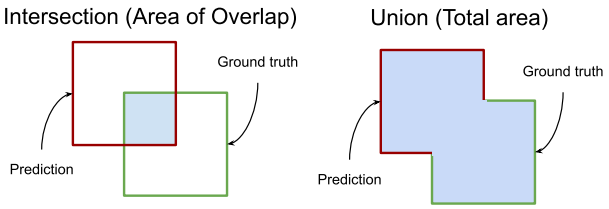


Figure 8: Graphical explanation of Intersection and Union

True positive (TP) is a prediction which coincides with the ground truth, ie a prediction that finds the correct object in the image [10]. This is a binary measurement either true or false determined by the IoU threshold.

False positive (FP) is a prediction which does not coincide with the ground truth, ie a prediction of an object which was wrong [10].

False negative (FN) is an object that should be found but the model misses while predicting [10].

True negative (TN) is not used when discussing object detection as there can be, effectively, infinite amount of bounding boxes that are true negative, and the term is therefore not applicable in this context.

See Figure 9 for an example of how TP, FP and FN can be for a rack of storage bins.

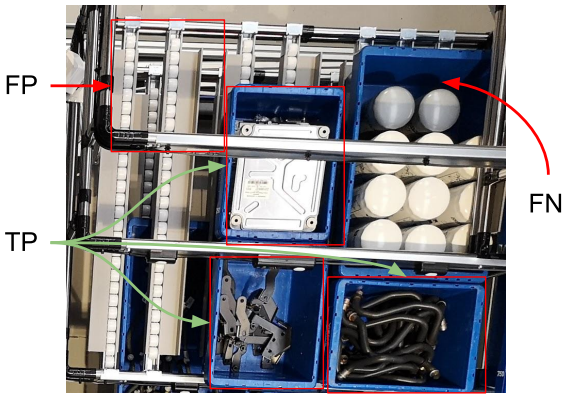


Figure 9: Detection example with true positives, a false positive and a false negative

Precision [10] is a metric that measures the accuracy of the predictions made, i.e. how many of the predictions the model made were actually correct. Precision is calculated by (see Equation 2) the ratio between the true positives and the total number of predictions (true positives and false positives).

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall [10] is a measurement to show how many of the objects are found, in other words how likely is the model to miss an object. Recall is calculated by (see Equation 3) the ratio between how many true positives and how many objects are in a given image (true positives and false negatives).

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Precision vs recall: Precision and recall have a negative exponential relationship ($a - b^{cx+d}$, where $a, b, c, d \in \mathbb{N}$) which means that if you optimise for precision, recall will decrease and vice versa, see Figure 10 for a typical precision to recall relationship. This is due to the fact that increasing sensitivity of detection will decrease the amount of false positives but increase the amount of false negatives and the reverse is also true.

AP, average precision [9] is the area under the curve of the precision-recall diagram, see Figure 10. In some models this value is calculated per each class of objects as is the case for example in YOLOv4, but in other models the AP is calculated across all classes which is the case for the COCO challenge evaluation.

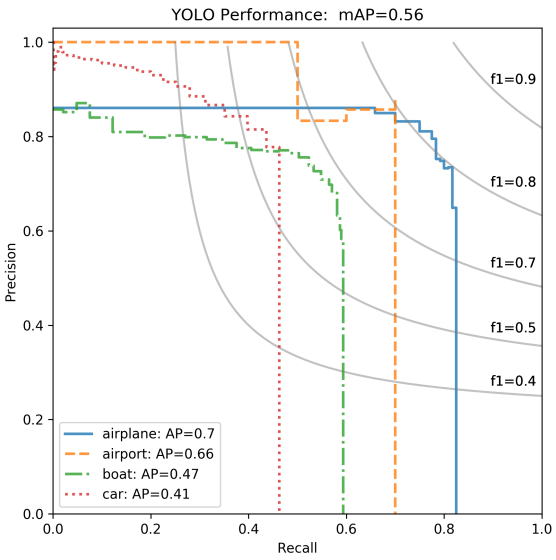


Figure 10: Precision-recall graph example for YOLO, where IoU threshold is 0.5 [11]

mAP, mean average precision [9], is the average AP of all the different classes in the dataset. For example in Figure 10 the classes AP are 0.7, 0.66, 0.47 and 0.41 which gives an mAP = 0.56. In the case that AP is calculated over the total dataset, AP and mAP can be used interchangeably, for example the COCO challenge evaluation.

Threshold for average precision: AP and mAP is dependant on which IoU threshold is used and is denoted as AP@n and mAP@n respectivly, where n is the threshold. For example AP@0.5 where the threshold is 50%. In some cases AP@0.5:0.95 is used this denotes the average AP for the range from 0.5 to 0.95 with 0.05 increments [9]. If no IoU is defined, the threshold can usually be assumed to be 0.5, but in some cases, only writing “AP” can also refer to AP@0.50:.05:0.95 and therefore caution should be taken.

Important to note is that AP is based on binary measurements of TP, TF and NF and therefore the threshold will account for the accuracy in coordinate location of the bounding boxes. In the case for a low threshold the mAP is not a good measurement for if the coordinate-position is accurate. For example the model get all TP detections due to low threshold and therefore very high mAP but the average IoU is very low. Meaning that the xy-coordinates are far from the ground truth, but the object is still counted as detected. To solve this issue a high threshold is needed to get high accuracy coordinates.

Loss: In machine learning, the goal ultimately is to minimize the error of the machine learning algorithm for each successive training example during the training phase. This is where loss and the loss function (s) come in. Essentially, the loss function mathematically represents the price paid for an error in prediction during classification of the object.

There are many different types of loss functions that can be deployed in the neural network. These vary depending on what the function of the deep neural network is.

3.3 Models

There are three models used in this project, EfficientDet, YOLOv4 and YOLOv5. These models are chosen from a list of state of the art object detection models [1], where EfficientDet and YOLOv4 is the top two models and YOLOv5 is a competitive version of the YOLO family.

3.3.1 You Only Look Once v4 (YOLOv4)

The YOLOv4 model [5] is based on an one-stage detector and uses the backbone CSPDarknet53, the necks SPP and PAN and the single-stage head of YOLOv3. Additionally this model implements what is called bag of freebies (BoF), which is methods to improve the results while only slowing down the training and not the inference. An example of BoF is data augmentation, which is performed automatically in YOLOv4. Some basic augmentations are scaling, cropping, flipping, rotating, brightness, contrast, hue, saturation, noise, as well as more complex augmentation methods such as CutMix where you place a cropped image inside another image resulting in obscuring and mixing the data, or mosaic where multiple images are placed side by side to make objects appear in different contexts [5]. Other improvements called bag of specials (BoS) are also implemented where the accuracy is largely improved for a marginal cost of inference time. Examples of BoS are improving layer activation, modifying kernel size of some max-pooling layers in the neck [5].

3.3.2 You Only Look Once v5 (YOLOv5)

Less than 50 days after the release YOLOv4, on June 29th 2020, YOLOv5 [12] was released by Ultralytics. YOLOv5 is the first in the line of YOLO models to be written in the Pytorch framework on release.

The creator of YOLOv5, Glenn Jocher, has said that the goal with YOLOv5 is not to have the best level of detection and highest mAP, but instead focuses on things such as ease of use, speed of training/inference, exportability (to mobile for example), model size etc.

There exists a variety of different pretrained models of YOLOv5, differing in size and complexity, this in turn affecting precision and speed. Generally, the larger the model is, the more precise it is. The drawback however is the speed of training, which is slower the larger the model is. There are four different YOLOv5 models (see Figure 11), in order of increasing size, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x.

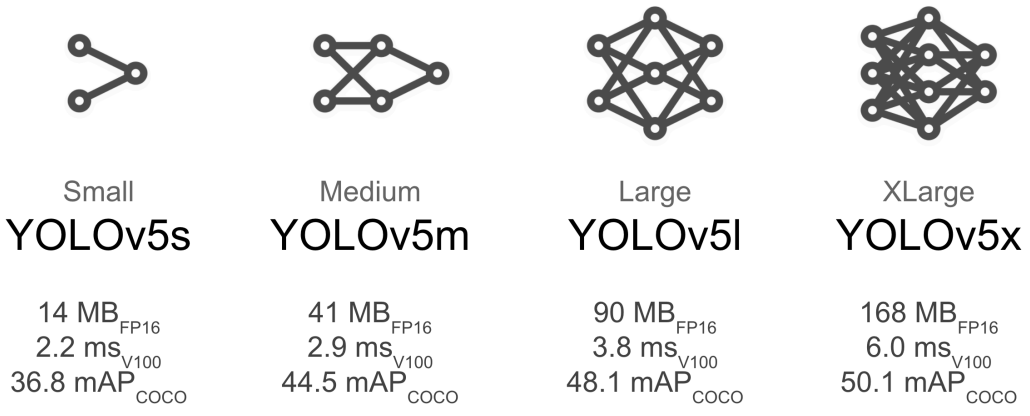


Figure 11: The different pretrained models of YOLOv5, image from [12]

Depending on what one wishes to use the object detection model for, one of these four might be suitable. YOLOv5s, for example, is suitable for projects where having the highest precision isn't of the greatest concern, but rather a small model size and fast training.

3.3.3 EfficientDet

EfficientDet is a family of object detection models developed by Google's team and is considered to be the state of art. EfficientDet is an one-stage detector which is using EfficientNet as the backbone of the model and BiFPN a bi-directional feature pyramid extractor as the neck [13]. Another major advantage this model proposes is the scaling of the backbone which will increase efficiency and accuracy for larger image sizes [13]. The head of this model uses two separate networks, one for bounding box prediction and one for class prediction. Data augmentation that is used when training in this model are for example horizontal flipping, random scaling and cropping.

3.4 Method

For this project Google Colab [14] has been used alongside with running the machine learning models on a computer GPU using CUDA.

3.4.1 Creating dataset

The general approach to create a dataset for object detection is to divide the data into train,validation and test. It is common to set ratio to 70%, 20%, 10% and this was done for all datasets created.

The first step of creating a model is to create the dataset, this was already done in advance for the first part of the project. A small set of images were taken of the objects of interest. These images were annotated with a bounding box around the objects in the image and the coordinates of the bounding box were saved in a text file with the same name as the image. To create a larger dataset of images the initial set was augmented and a set of synthetic images were created. All together a dataset consisting of 1300 images was used in the first part of the project (comparing detection models).

For YOLOv5 the annotation text file was put in a directory called "labels" and the images in a directory called "images". For YOLOv4 the labels and images were placed the same directory. EfficientDet (tensorflow) requires a file called TFrecord which is a sequence of binary records and was created by uploading the dataset to Roboflow [15] and converted into the correct file-type.

During the course of the project, a new dataset was created, this time with the camera height of 4 meters. This was achieved by traveling to ASSAR in Skövde and using their facilities for taking the raw pictures. These pictures were taken using a dome camera and as such needed some processing before use. This included undistortion and calibration of the pictures. These pictures then had to be annotated manually (using roboflow). Following this, several augmentations were applied to the pictures: Hue augmentation (Between -25° and +25°), Brightness augmentation (Between -25% and +25%), and Noise (Up to 5% of pixels, salt and pepper noise)

In addition to the pictures taken in the factory, a number of synthetic pictures were generated using a script and added to the dataset. The script takes blue box cutouts, fills some of them with random artefacts (mimicking an open bin), while others are left empty or has a lid on them resembling a closed bin. These cutouts are then placed against random backgrounds (factory backdrops) and the synthetic image is finished. All in all, including augmentations and synthetic images, the dataset consists of roughly 2000 pictures.

3.4.2 Training setup

As mentioned before Google Colab was used for training the models but some setup needed to be done. First of all GPUs needed to be active during the session and it was done by changing

the runtime in Google Colab to GPU. Second step was to clone the repository for every model in to Google Colab and import framworks for each model. Frameworks used by models were Tensorflow for Efficientdet, Darknet for YOLOv4 and Pytorch for YOLOv5. Next step was to upload the dataset which was prepared since earlier. Last step was to change the config for each model.The config file contains some variables for the training. These variables are listed below and were changed for each model to obtain best performance.

- Number of epochs: Hyper parameter for defining the number of times the training model will go through the entire dataset.
- Batch size: Number of samples of the data-set to be processed in the model in one iteration.
- Number of steps: Depending on the batch size it takes a number of steps to iterate over all data. To iterate over the data that has a size of 100 and the batch size is 10 then 10 steps are needed to iterate over all data one time. To iterate over the data 10 times, it can be done either by setting number of epochs to be 10 or number of steps to be 100 [16].
- Resolution: The default resolution for pictures to be set in YOLO models.
- Choice of model: Specification of which version of model to be used. For example efficientdet-d1 were used for the EfficientDet model.
- Number of classes: Number of different classes the model learns to detect, for example open blue bin, closed blue bin, open pallet and closed pallet.

3.4.3 Training

Starting the training process either by Google Colab [14] or a local machine, the training could fail at any time. A part of this step was to adjust the config file and look up the error to fix it every time the training process failed. Another important task to do was to monitor the process to be sure that the training works correctly. This was done by monitoring TensorBoard and confirming that loss and mAP converged to some value (see Figure 12) and decreased/increased for every step of the training. As mentioned before the dataset was divided to training-evaluation-test and the evaluation is done alongside the training.

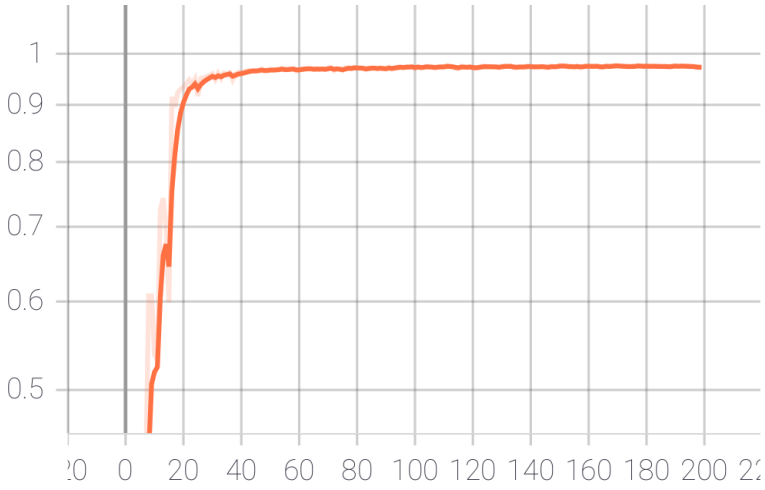


Figure 12: Monitoring mAP@0.5 in TensorBoard during training, the x-axis represents the number of epochs and the y-axis represent the mAP@0.5.

3.4.4 Export model

The training takes about 8-12 hours to complete depending on the model and once it is complete the weights that has been trained can be extracted. These weights are considered to be the custom model that has been trained and can be used as the final product.

3.4.5 Inference

After the models were exported they were used to run inference on same pictures which enabled us to see how well the models were. Evaluation matrices were also extracted from the models to be able to compare them numerical.

3.4.6 Test 1: Unbalanced classes

The pallets were removed from the original dataset (see and a new YOLOv5 model was trained to test the hypothesis of unbalanced classes affecting the result, as described in Chapter 3.6.5. In the original dataset there were four classes of which two were significantly underrepresented (open pallets and closed pallets). Since the main goal was to detect open and closed blue bins, as well as the fact that the pictures of the pallets were few, these were removed.

3.4.7 Test 2: Augmentations

To find out which augmentations improve the models mAP the most and are worth deploying on the dataset, different types of augmentations were tested. First a new set of pictures were created, including 20% real images and 80% synthetic images.

From these images a number of datasets with different augmentations where created:

- Plain dataset: *without augmentation.*
- Hue augmentation: *plain dataset + randomly shifted from -45°to 45°.*
- Saturation augmentation: *plain dataset + randomly changed by -60% to 60%.*
- Brightness augmentation: *plain dataset + randomly changed by -30% to 30%.*
- Noise augmentation: *plain dataset + randomly added salt and pepper noise up to 10% of pixels.*
- Cropping images: *plain dataset + randomly cropped up to 45% of the image.*
- Blur: *plain dataset + randomly applied Gaussian blur up to 7 px.*

Then, for each one of these augmented datasets, a YOLOv5s model was trained and evaluated. This was done by first turning off the inherent augmentations that are applied by YOLOv5, for a more robust comparison. Then the model is trained one by one for each augmentation and the results (mAP, etc.) noted down in a table for comparison. This was done in hope of seeing that some augmentations performed better than others, and as such could be used to construct a new better dataset.

3.5 Results

Training
The results for training of the models is displayed in Table I. Each model (EfficentDet-d1, YOLOv4, YOLOv4 tiny, YOLOv5x and YOLOv5s) is compared using different key figures (mAP@0.5,

mAP@0.75, mAP@0.5:0.95, recall@0.5, precision@0.5, model size, inference time and training time). The data shows that YOLOv5 has the lowest training and inference time and smallest size, while YOLOv4 has the highest mAP, precision and recall.

Table I: Results, training of models

Key figure	**EfficientDet-d1**	**YOLOv4**	**YOLOv4 tiny** [1]	**YOLOv5x**	**YOLOv5s**
mAP@0.5	97.04%	97.38%	52.74%	94.5%	92.9%
mAP@0.75	94.8%	95.42%	44.81%	N/A	N/A
mAP@0.5:0.95	85.9%	77,65%	38.52%	76%	77.2%
Recall@0.5	N/A	0.99	0.95	0.93	0.931
Precision@0.5	0.97	1.00	0.96	0.61	0.694
Model size	101MB	256.1 MB	23.5 MB	168 MB	14 MB
Inference time [2]	0.4203s	0.038s	0.038s	0.018s	0.018s
Training time [3]	8h	12h	4h	3h	1.5h

Test 1: After deleting the underrepresented classes from the dataset, new training results for YOLOv5x were achieved. In Table II all the parameters were higher for the dataset without unbalanced classes.

Table II: Results from Test 1, comparing only bins classes with bins and pallets classes

Key figure	**YOLOv5x bins and pallets**	**YOLOv5x only bins**
mAP@0.5	94.5%	99.2%
mAP@0.5:0.95	76%	89%
Recall@0.5	0.93	0.988
Precision@0.5	0.61	0.844

Test 2: Training results of augmented datasets is shown in the Table III. We can see that cropping augmentation achieves the highest mAP on its own, and blur the lowest. New dataset is there for comparison (with several augmentations applied).

Table III: Results Test 2: augmentations (trained on YOLOv5)

Augmentation	**mAP@0.5**	**mAP@0.5:0.95**	**Recall@0.5**	**Precision@0.5**
New dataset [4]	99.0 %	86.0 %	0.986	0.672
Cropping images	67.2 %	35.0 %	0.686	0.606
Saturation augmentation	65.6 %	34.5 %	0.675	0.510
Hue augmentation	63.1 %	34.3 %	0.614	0.517
Brightness augmentation	62.7 %	32.7 %	0.644	0.489
Plain (no augmentation)	62.3 %	30.7 %	0.678	0.503
Noise augmentation	60.9 %	31.5 %	0.645	0.427
Blur	58.9 %	29.8 %	0.619	0.461

[1] The classes open_bin and closed_bin has AP over 95% but open_pallet and closed_pallet has AP close to 0%

[2] Inference on same dataset for all models, consisting of nine pictures. Inference was run using the graphics card Tesla T4. Inference time was calculated as the sum of each individual picture time divided by nine.

[3] The approximate time it took to train models, different GPUs and epochs/iteration were used and can not be compared directly.

[4] The new dataset is based on the plain dataset with YOLOv5 augmentations enabled and hue, brightness, and noise augmentations applied in pre-processing.

Inference
How YOLOv5 inference on a test picture, using weights from the training of the new dataset, looks like is shown in Figure 13.

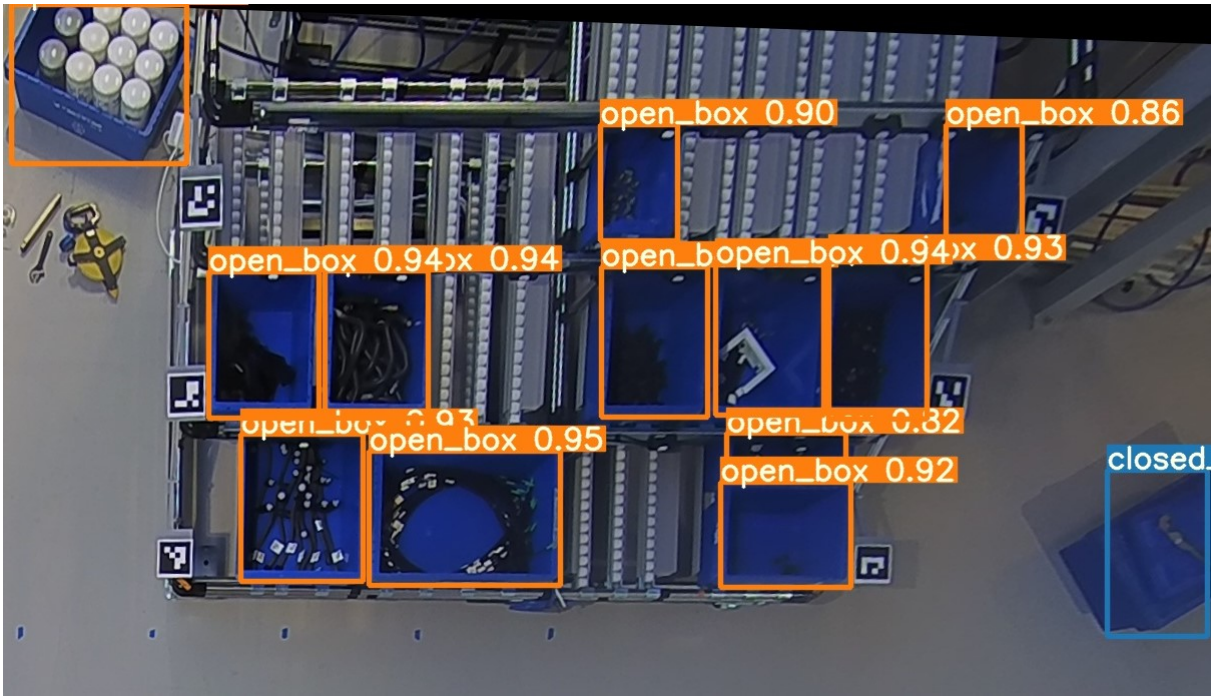


Figure 13: Example of inference being run on YOLOv5 using the exported weights from training.

3.6 Discussion

The result of comparing the different models are shown in Table I. For the accuracy of the model all models performed well with mAP@0.5 over 92% except YOLOv4 tiny, the highest mAP@0.5 was for YOLOv4. The small discrepancy in accuracy between the models can be explained by the training time where longer training resulted in better accuracy. For inference and training time, the best model is YOLOv5 which had less than half the time than the second best model, while still having good accuracy. This is inline to the purpose of YOLOv5 to be a lightweight and fast model.

3.6.1 Inference time

Inference time is the time it takes for the model to do one pass of the image through the neural net. Therefore, when measuring inference time, the time for any other processes like plotting the bounding boxes, starting the CUDA or printing out the results should not be considered. However the time for inference was outputted automatically in the Yolo models but for EfficientDet it was done manually by using the time function in Python. This may have resulted in differences in measurements but all effort has been done to have same circumstances for all three models. This was done by running inference on same pictures for all three models.

3.6.2 Training time

The YOLOv5 model family (YOLOv5s, YOLOv5x, etc.) are all much faster to train compared to model families of YOLOv4 and EfficientDet (see Table I). Of course, training time varies depending on the parameters chosen for the training (epochs, number of steps) that directly affect the length of the training time. However, these discrepancies between the models in mind, YOLOv5

still comes out ahead with an average training time significantly lower than the other models that were tested. One thing to keep in mind is that, no matter the amount of epochs or steps chosen, YOLOV5 saves the best weights from the training automatically. This means that even if too many epochs were initally chosen, the model will still save the best weights and there is no risk of overfitting. This also means that the actual training time, from start of training to the best weights being found, is actually lower than the training time defined by the number of epochs chosen.

3.6.3 Ease of use

When it comes to ease of use, again YOLOv5 comes out ahead. This is no surprise since the creator of YOLOv5, Glenn Jocher, has said that his goal with the model isn't necessarily to achieve the best detection results, but rather things like ease of use. Things like intuitive file folder layout making it clear and easy to change things within the code. Moreover, since YOLOV5 is implemented in the Pytorch framework, it is easier model to train and deploy for inference compared to the other models running on other frameworks, namely Darknet (YOLOV4) and Tensorflow (EfficentDet). YOLOv5 also, as mentioned earlier (see Figure 11), comes with 5 pretrained models of different sizes that can be chosen depending on the purpose the user wishes the model to be used for. These models conveniently all come with the same YOLOv5 Github repo that you clone and switching between them is very easily done as such.

3.6.4 YOLOv5 paper

The model YOLOv5 does not, at the time of writing, have a peer-reviewed paper to accompany the model. This has spurred controversy [17] in the machine learning world, raising questions like "Is this model real?", "Can the model be trusted?" or "Are the claims of smaller and faster model true?". The questions around the performance of the model, for this specific use case, have been addressed and tested in this thesis and seem to be good. But the questions around trust, validity and if it will be supported in the future is still uncertain, although the code is open source so this will most likely not be a problem.

3.6.5 Unbalanced classes

During the creation of dataset it was noticed in the data obtained from Volvo that there were classes for pallets including closed pallet and open pallet. These classes were underrepresented in the dataset and there was only few of them. While training the models a discrepancy was found between the AP of bins and pallets where the bins had a lot higher AP than the pallets.

A hypothesis was formed that the underrepresented classes had a negative effect on the evaluation matrices. Therefore Test 1 was created, in Chapter 3.4.6. This test was only done for YOLOv5x due to the faster speed of training.

When comparing the results of Test 1 in Table II with the original dataset it clearly shows a large improvement in accuracy for mAP@0.5, mAP@0.5:0.95, precicion and recall. The hypothesis was confirmed and it can therefore be beneficial to create a dataset where all classes have an equal occurrence.

3.6.6 Dataset augmentation

To create a better dataset different forms of augmentation can be used. To test the effect of different types of augmentations Test 2 was performed and the result is shown in Table III.

As can be seen from the results in Table III, no obvious conclusions can be drawn in regards to which augmentations are better than others, except for the cropping augmentation, which is the only augmentation that performed significantly better than the others. The increase in mAP for cropping might be explained by the increased resolution for each box, making it easier for the algorithm to see the pattern. This result indicates that having a higher resolution when training or infering is benificial and one way of acieving this is by cropping. The new dataset uses many of these augmentations as well as the built-in in YOLOv5, which indicates that multiple augmentations will have synergies and improve the model more than single augmentations.

3.6.7 Occlusion

To occlude the objects is a common way of augmenting the data, when making the data-set for training a model. Occlusion can be helpful if an object were to be obscured by something in the real world and the model can be trained to “look behind” objects to find the object of interest. But in this project it is important to not find bins that are occluded under other objects, it can be good to find objects obscured from the camera view though. This is because the application for where this project will be used is to give coordinated for a robot to pick up items, therefore a coordinate should only be given if it is accessible by a robot arm. If the coordinate given to the robot were to be below another object or the shelf, a collision will occur which is not wanted. When designing the dataset for this application objects can be occluded but the bounding box should only represent the visible part.

One way to solve the problem of the AI finding bins on the wrong level in the material rack could be to put an opaque material on each shelf. This will prevent the object recognition model to find objects that are on another level in the shelf and therefore not accessible. Another way of disregarding bins found in the wrong place is to compare them to the position of the fiducial markers of each level, if a bin edge is not in the tag-line of the shelf it will be disregarded.

3.6.8 Camera resolution

When using cameras the main constraint is resolution, this will impact the accuracy of the object recognition. An object recognition model will work better when having a larger image of the object to find, this will also be relevant when trying to find small objects or objects far away. This is due to a small change in coordinates resulting in a large loss of IoU-accuracy. When mounting the camera in the ceiling the margin of error will increase further from the camera, resulting in the lowest shelf of the material rack to have the greatest margin of error in the coordinates. When designing the system a tradeoff has to be made between resolution (cost) and accuracy of the output coordinates.

3.6.9 Coordinate precision

The loss of most models are based on mAP, which simplified is a binary measurement of how many objects are found and if the prediction is correct, this does not take into consideration how the quality of the prediction is. In this project the end-goal is to be able to pick up an item from a bin, therefore coordinate accuracy is more important, in other words maximizing IoU. It is not possible to change the loss function of a model without rewriting it, therefore a solution is to increase the IoU threshold for when a prediction is counted as a TP. When using a higher IoU the coordinates will be pushed closer to the ground truth with the trade off being that the recall will suffer and some bins might not be found.

3.6.10 Classic CV

To improve the accuracy of the output coordinates from the model further research is needed, one method is classic computer vision (CV). Where the output from the detection algorithm is used and inside the bounding box classic CV is used to narrow down the error. Example of methods to test are: looking for sharp edges to find the true edge of the bin, convex hull of blue pixels, finding if the storage bin is rotated in the bounding box etc.

3.7 Summary

The most accurate model was found to be YOLOv4 with the other models close behind, the fastest model for training and inference was YOLOv5. The model which is most easy to use when integrating in a Python script is YOLOv5 due to it being based on a Python framework.

It was found that when creating the dataset the number of objects in each class need to be balanced to achieve a better result. Furthermore when creating a dataset it is important to augment the images with multiple augmentations to achieve a good result. The most significant increase from a single augmentation is when cropping images, which suggests that zooming in or having high resolution is beneficial. In the dataset no bounding box should be occluded by something else, the output should only represent the accessible part of the object. When designing the material rack it is important to have opaque shelf's so that no inaccessible object can be found.

To get a higher accuracy of output coordinates, the model needs to use a higher IoU-threshold but might infer lower recall. The object detection model works better with higher resolution which is a trade off for speed and cost. Further testing could be made into combining the detection model with classic CV to improve the accuracy of the coordinates.

4 Positioning of bins

This chapter handles the setup required for the bin positioning model and the functions involved with finding the bin positions. The setup involves fiducial markers and a camera, and the functions are python code. Three fiducial markers (AprilTag, ArUco, and STag) are evaluated to find the best suited for this application. The camera requires no evaluation as it is specified but a calibration matrix is necessary to make use of the images. When the results from Detection of bins (Chapter 3) are merged with the functions developed in this chapter a complete model for bin positioning is accomplished.

4.1 Sub-goals

The goals for the Positioning of bins part are:

- Determine best fiducial marker for the application of detecting the position of bins.
- Obtain a camera calibration matrix that correctly warps an image and retains as much as possible of the source information.
- Merge Detection of bins (Chapter 3) with Positioning of bins (Chapter 4) into a complete model.

4.2 Theory

In the Theory section the parts necessary for the model outside of the object detection algorithm will be introduced. How fiducial markers and the camera is used and exactly what needs to be known to create a model to determine the bin locations.

4.2.1 Fiducial markers

A fiducial marker is something that can be easily distinguished in an image and used as a point of reference for distance calculation or to store information. Fiducial markers are made with an associated programming API that typically calculates the corners of the square bounding the marker and the identification number associated with the marker. With some clever math, a lot can be accomplished by simply knowing the corners of a supposedly perfect square. If the corners given in the image for example is not 90 degrees that would mean the square is tilted in some direction. If two markers of the same size are calculated to have different size a difference of depth is at play. Figure 14 displays the three different fiducial markers used in the project (AprilTag, ArUco & STag).

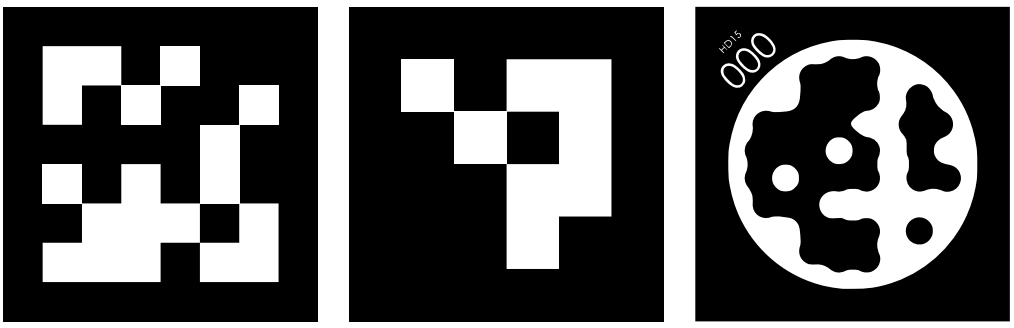


Figure 14: Example of how AprilTag, ArUco & STag looks like.

The AprilTag [18] was presented as an improved fiducial systems in areas such as robustness to occlusion, warping and lens distortion. The fiducial marker is designed to hold a relatively small information payload which is supposed to make it easy to detect and localize it even at low resolutions with bad lighting and somewhat occluded.

The fiducial markers consists of black and white squares which make up a pattern of clear borders between the dark and light regions. The detector takes advantage of the high contrasts to detecting lines in the marker by computing the gradient direction and magnitude at every pixel. The lines then have to form a sequence to form a square, which can be troublesome if the marker is occluded or the image have some other noise in it. To handle this problem the detector search for the lines in different ways. At the first level it searches for all line segments and add the edges to the quads. At the second level it searches for lines that begin close to the previous line in order to try to fix the occluded parts of the image. The robustness is greatly determined by how close the next line segment has to be to the previous one. By increasing this threshold value, bigger gaps in the image should be able to be handled. The implementation described in the paper[18] uses a threshold value of twice the length of a line plus five additional pixels.

The fiducial markers comes in some variety which are suited for different applications. The most common type has a 36 bit coding with a hamming distance (HD) between two codewords of 5 to 11. A marker with a greater bit size will be able to store more information and therefore contains more possible unique fiducial markers. However this also means that the camera needs to be placed closer to the marker in order to detect the smaller quads that are used in the marker. The paper [18] states that only a 25 percent improvement in detection rate should be expected by using 16 bit fiducial markers instead of 36 bit fiducial markers.

The ArUco-tag [19] has a lot in common in appearance with the AprilTag and as seen in Figure 14 the ArUco-tag also consists of black and white quads to make a high contrast pattern. The decoder then uses the high contrast to find the borders between bright and dark areas by comparing the pixel value of the grey-scale image. Then the image of the marker is divided into a grid where each area of the grid gets assigned the value zero or one depending on if the majority of pixels in it are assumed white or black as can be seen in Figure 15.

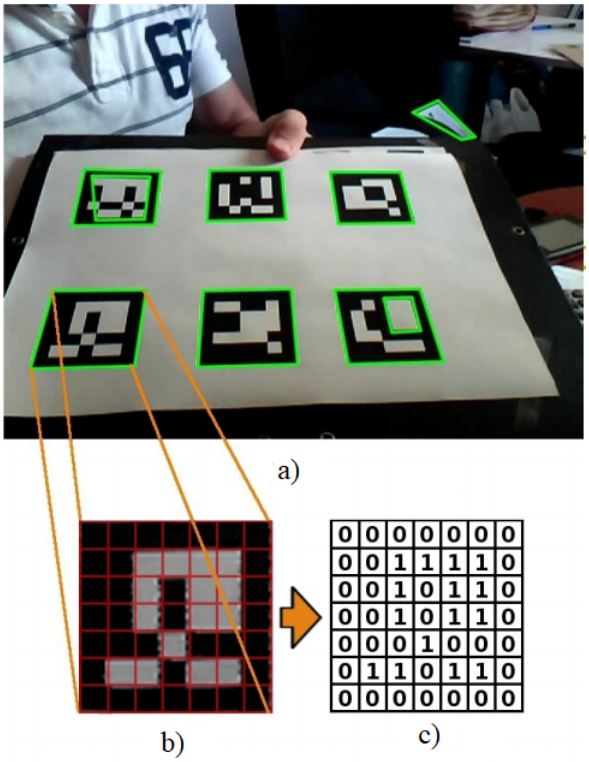


Figure 15: a) Image with detected borders. b) Image divided in a grid. c) Each area of the grid is assigned the value 0 or 1. Image source [19].

To handle occlusion of the fiducial marker a color map of the board is used to compute an occlusion mask by color segmentation. In addition to occlusion the authors uses the approach of making a marker board when using their fiducial markers. A marker board is simply a pattern composed of multiple markers where the position of the markers corners are referred to a common reference system.

The right most marker in Figure 14 shows an example of a third marker type, the STag [20]. Similarly to most other fiducial markers STag also uses a black square for its outer borders, but the border for the marker itself has a circular shape. The authors wanted to improve robustness and stability of the localization. In order to find what shape works best for improvement they tried both squares and circles as borders. To evaluate the result a square and a circle was uniformly sampled at 40 points, preturbed with a Gaussian noise with 0.04 standard deviation along each axis. The test ran 10 000 times with standard deviation values between 0 and 0.04 and the result can be seen in Figure 16. As seen in the first two pictures the square appears as a quad and the circle appears as an ellipse. With the result presented in the graph the authors claim that the circular boarder is a more stable shape to localize and is therefore used as the inner boarder for their marker. The outer black square is used to detect the marker and estimate a rough homography and the circular part is used to refine the accuracy of the previously estimated homography.

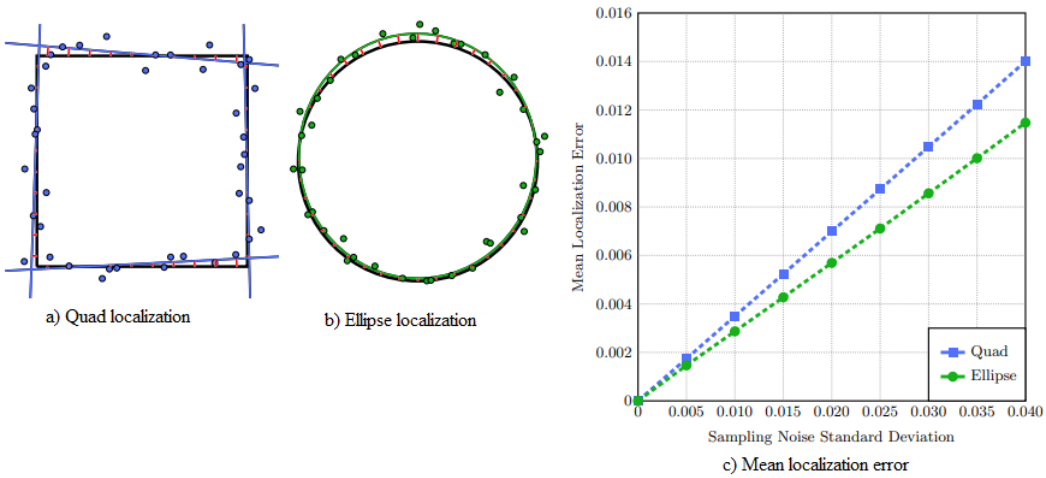


Figure 16: a) Quad localization. b) Ellipse localization. c) Plot of mean localization error. Image source [20].

The detection of the fiducial markers is done in three different steps. In the first step the algorithm tries to find the outer border of the marker as can be seen in the *Candidate Detection* section in Figure 17 by fitting a line to the beginning point of an edge segment and extending as long as the following edge pixels are on it. The *Candidate Detection* approach has the advantage that the border can be detected even if only three corners are visible due to occlusion. In the second part of the detection the decoder perspective transform the markers to eliminate the perspective distortion and prepare the marker for decoding. In the last part of the detection the decoder uses information from the previous part to refine the homography and make a better approximation of the position of the marker.

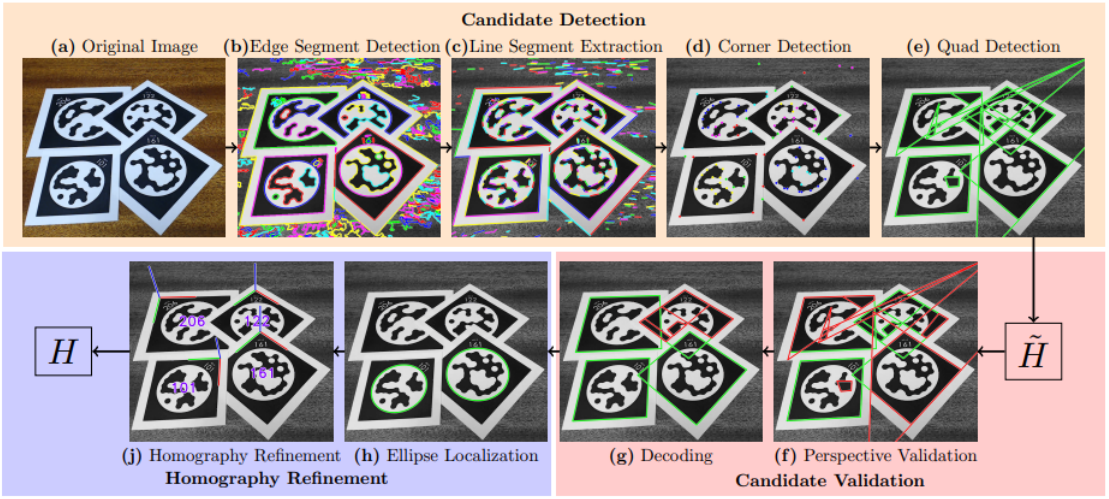


Figure 17: The three steps of detecting the markers. Image from [20].

4.2.2 Applications of fiducial markers

The majority of the fiducial markers on the market was developed to be used in augmented reality or as robot localization. The shape of the marker makes it possible to obtain full six degrees of freedom localization from a single image. The localization can be used to place objects in an augmented reality or get the position of an object in an image to map the position

in the image to real world coordinates.

Image mapping is the action of connecting parts of an image with their corresponding real world equivalents, coordinates for example. The image mapping task is simplified with fiducial markers placed in the image. The coordinates and size of the fiducial markers are predetermined acting as the starting point to potentially map the entire image, the mapping is more difficult if multiple planes are present but that simply requires two additional markers for each plane.

4.2.3 Camera calibration

Due to the anatomy of a camera and the fact that it uses a lens to capture an image, every camera will introduce distortions of various degrees. Some cameras will introduce some significant distortion while others may introduce a lot less distortion. In the every day life, camera distortion will rarely affect us at all due to how well built most cameras are today. If the image is used for calculations however, the distortion may cause some serious problems. The distortion of the image will affect the calculations made based on the image and greatly affect the precision of the calculations.

The two primarily types of distortion are radial distortion and tangential distortion. The radial distortion causes straight lines to appear curved in an image and gets worse the further away from the center of the image. The tangential distortion is caused when the lens is not perfectly parallel to the imaging plane, and will cause some areas of the image to appear to be closer to the camera than they are.

4.2.4 Camera specifications

A camera contains multiple different specifications that in different ways impact the resulting image. Specifically for the project the resolution of specific objects is of interest such as the markers and the bins. The resolution of an object depends on multiple factors concerning the camera specifications, one being the overall image resolution measured in mega pixels. The other relates to the combination of lens field of view (FOV) and the distance away from the object. The lens FOV is the angle at which light can enter and be projected on the digital sensor or photographic film simply put. Since images are generally rectangular in shape two angels are used or one and a ratio specifying the height to length ratio of the image.

For the resolution of objects in an image the image can be viewed as a pyramid with the camera at the top and the objects at various heights at the bottom of the pyramid. The percentage an object uses of the total image can be viewed as the portion the object takes up of the area of the plane in the pyramid where the object is located. To increase the portion the object uses of the total image and consequently the resolution of the object, the area of the plane in the pyramid should be decreased. For a camera this means positioning it closer to the object or to decrease the lens FOV.

4.3 Bin positioning model

As stated in Chapter 1, the result of the project is a library of Python code and a computer vision model that together produces coordinates for all of the blue bins in an image. The flowchart below shows the steps in the process and this chapter explains the steps in detail.

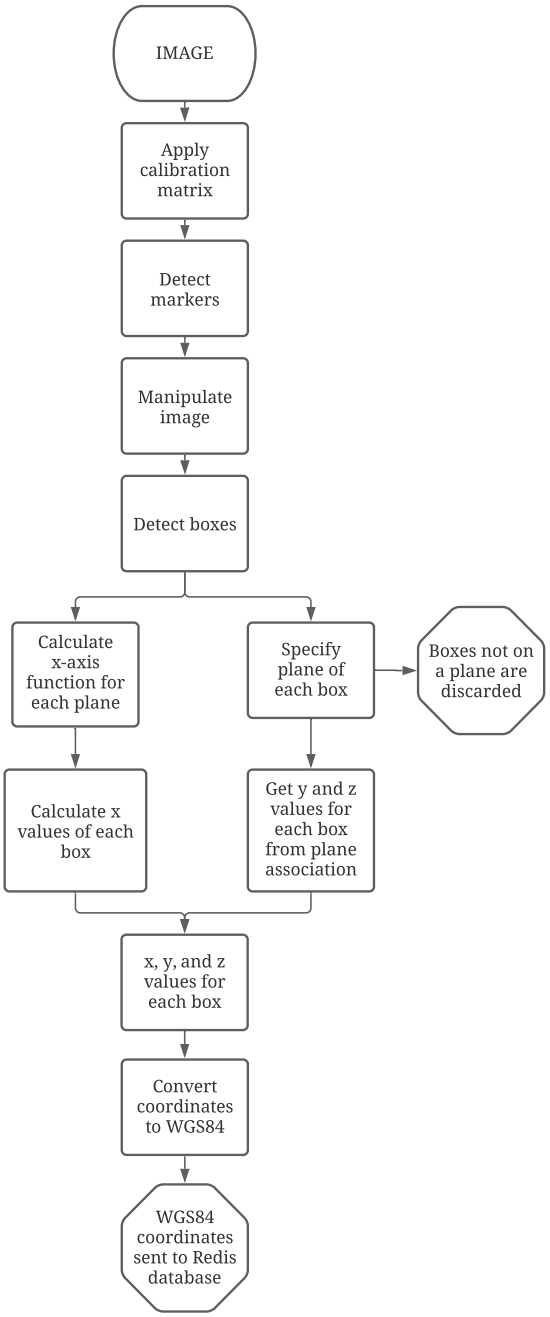


Figure 18: Components of bin positioning model. The flowchart illustrates the process of achieving coordinates from an image.

4.3.1 Calibrating for lens distortion

In Chapter 4.2.3 the effects of lens distortion are described, to accommodate for the distortion a calibration matrix is calculated. Several images are taken of a flat chessboard like pattern, the contrast between black and white square creates precise points that are easy for a computer vision algorithm to recognise. Since the pattern is made with squares of exactly the same size lines drawn through several points should be straight, any deviation is a result of lens distortion. The image is manipulated to the extent at which every line drawn through multiple points are straight, by using multiple images with the chess pattern at different tilts and distances away from the camera any local deviation will be accounted for.

When an image is taken the calibration matrix is simply applied before anything else is done, the calibration matrix is unique for every camera but the calibration images are only required to be taken once for any camera.

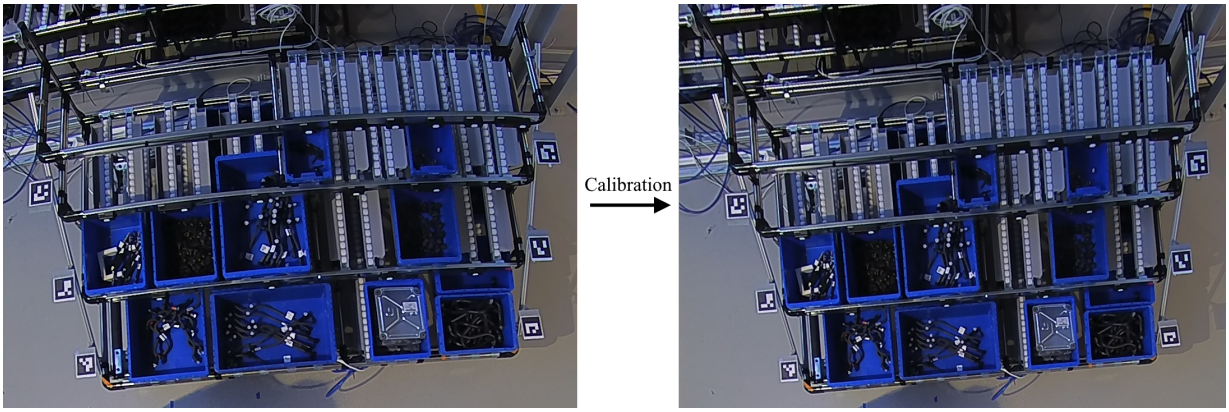


Figure 19: Example of camera calibration applied to a picture.

4.3.2 Fiducial markers

Every fiducial marker the project considers is made with code associated to it for detection. The code delivers pixel coordinates of each corner of the marker square and the markers identification number. From the coordinates of corners and ID of each marker in the image different things can be determined. When the markers are planted on a plane they have the same angle and height as the plane they are associated with, furthermore an edge of the marker is parallel with the edge of the plane. The requirement for same angle for marker and associated plane stores information in the marker that can be used later. Additional information that can be used is the length of the marker square, every markers sides must be the same and known when mounted. To accommodate for varying angles of the camera two markers are required on each plane, by comparing the different length to pixel ratios for the two markers a linear function can be constructed that computes the x axis length values.

A dictionary is included in the code where the id of a marker gives the associated information as described above.

As it is described in Chapter 4.2.1 the code for detecting the ArUco and AprilTag markers does not manipulate the image when it detects markers. In Chapter 5 a fault in the marker detection algorithm was discovered where markers that are clearly visible could not be detected. It was also discovered that rotating the image could improve the detection rate of ArUco marker, the marker detection function utilises therefore image rotation to increase the marker detection.

4.3.3 Manipulating image

The computer vision model is tasked with determining the bounding boxes of all the blue bins in the image. For the computer vision model to work properly the image might need to first be rotated so that the bounding boxes covers the blue bins perfectly. The need for rotation is because the bounding boxes the computer vision model produces are always horizontal and vertical but the blue bins might not be. The image thus needs to be manipulated to the extent that the blue bins are horizontal and vertical as well.

The marker detection algorithm is not limited in the same way the blue bin detection algorithm is. The marker detection algorithm produces coordinates for all corners of all markers detected.

4.3.4 Detect boxes

By running the pretrained object detection model described in Chapter 3, the storage bins on the material rack can be detected. The detection model returns the position in pixels in the image of the bounding box the algorithm finds. The detection model is ran after the image has been manipulated to make sure as good detection is possible.

4.3.5 Specify plane of each bin

As described in Chapter 4.3.4, the bin detection algorithm produces pixel coordinates of every blue bin detected in the image. To define which plane a bin is on, the lower sides y value is compared with the markers lower side y values. Since the image has been manipulated to the extent where the planes lower side is horizontal and the markers are located on the lower edge of each associated plane comparing these value will give accurate results. The local coordinate system originates in the center of the marker with ID 0 (the first planes left marker when aligned as in Figure 19), the x axis is parallel with the edge of the planes, the y axis denotes the depth of each plane, and the z axis denotes the elevation of each plane.

The values for every plane is constant and is stored in a dictionary that can be retrieved with the key 0,1 or 2, plane 0 being the plane closest to the ground. The plane to z and y value correspondence is shown in Table IV. The left x value corresponds to the value of the left side of the plane when the material rack is oriented as shown in Figure 19.

Table IV: X,y,z values for the material rack.

Coordinate	**Plane 0**	**Plane 1**	**Plane 2**
Left x [mm]	0	0	0
Right x [mm]	2220	2225	2230
y [mm]	0	365	722
z [mm]	610	896	1210

4.3.6 Calculate x-axis functions

As has been described earlier the markers placed on either side of a plane acts as a key between the coordinates in the image and those of the real world. When a bin has been deemed on a plane the pixel distance from the bin to each marker is used as weight to create a bias towards the marker the bin is closest to.

To make the x-axis functions for each plane more robust and less reliant on precise marker length detection an optimization algorithm was implemented. The algorithm uses the known length of each plane to modify the values perceived by the computer vision algorithm such that when the modified values are used the calculated length of a plane is the correct value. The marker multiplier is therefore the correct value for the plane divided by the calculated distance.

After optimization the pixel distance from a side of the bin is multiplied by the real size of the markers, the marker multiplier previously explained and divided by the perceived marker size using the weights to get an appropriate average.

4.3.7 Convert coordinates to WGS84

The kitting robot positioned next to the material rack will use the output of the system to locate the front center of a desired storage bin. Since multiple sub systems will be needed to handle the entire process, it is of importance that all parts are able to communicate with each other. In addition to that there will be many similar systems placed in the factory that would have to

communicate with the robot as well. Therefore the output of the positioning system will have to be standardized.

A position expressed in a standardized fashion could be stored in a database and later be used by the kitting robot or any other system whenever the information is needed. For this application the open source database *Redis* [21] will be used to store the positions of each storage bin. To make use of the information stored in the database the positions would have to be expressed in a global coordinate system since there will be many different systems linked up to the database. Therefore the *WGS84* [22] with decimal representation will be used to output the position of each storage bin. The coordinate system expresses the position in latitude, longitude and altitude above sea level. This will be used to represent the position of the front center of the bin in addition with the width of the bin to sum up to four output parameters in order to express all the information the kitting robot will need.

Since *WGS84* is a global positioning system, the position and orientation of the material rack has to be calibrated once it has been located in the desired location in the factory. To make this task as convenient as possible only the global coordinate of the bottom left marker has to be calibrated each time the material rack is moved. In addition to that the orientation of the ceiling mounted cameras will have to be calibrated when they are mounted in order to define what orientation the material rack has.

4.4 Tests

In total, two tests (Different fiducial markers and Material rack simulation) were made. The first test compared the different fiducial markers, both detection rate and length measurement between the marker was used to evaluate the markers. The second test used a simulated material rack to also test markers detection rate, but with a more realistic setup.

4.4.1 Calculating coordinates

Different fiducial markers
The first type of test involved a simple phone camera with a 10 megapixel (MP) resolution. The markers tested are ArUco Original, ArUco 4x4 and AprilTag 36h11. As described in Chapter 4.3.2 calculating the distance using fiducial markers uses the perceived length of the marker to create a function translating image coordinates to real world coordinates, the process require precise detail of the edge of each marker to correctly work. When the detail of the image was lowered the resulting function and consequently the calculated distance should be less accurate. When the distance to a marker is increased the amount of information dedicated to a marker is decreased which should result in a less accurate distance calculation. To compare the markers robustness toward image detail and usefulness the error of the calculated distance will therefore be used.

The tests consisted of altering the distance between the camera and the fiducial markers and angle the camera view the markers while the two markers are mounted on a black board a fixed distance apart. The distance was altered with an 1 meter increment starting at 1 meter until 8 meters. When the angle was altered the distance was kept at a constant 3 meters and incremented with 15 degrees ending at 60 degrees. An example of the resulting pictures from the test is displayed in Figure 20.

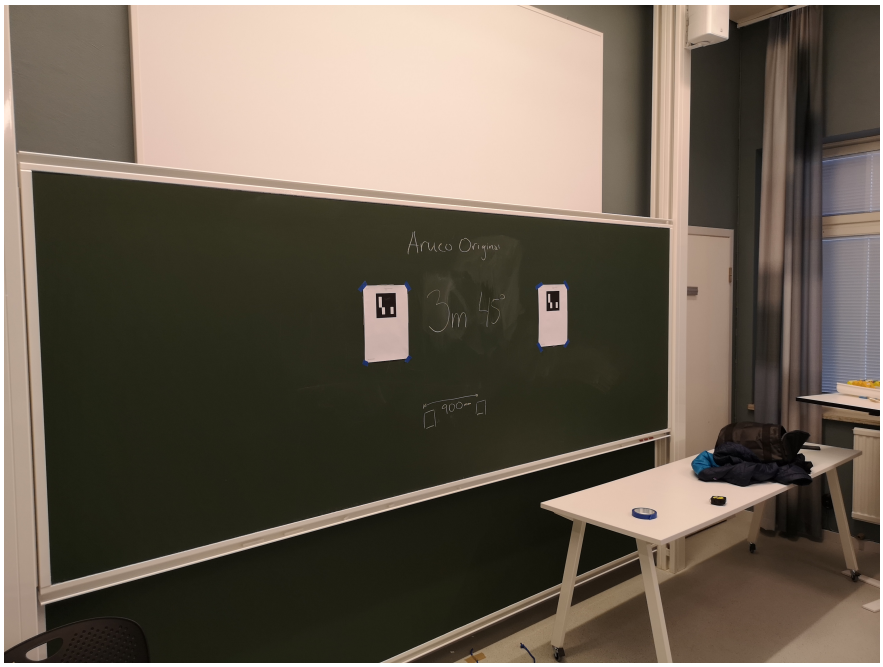


Figure 20: Example from Different fiducial markers test. The presented picture shows two ArUco fiducial markers with 900 mm spacing, camera is located 3 m and 45° relative the middle of the fiducial markers.

Material rack simulation

The second type of test involved a mock-up of the material rack. The mock-up is setup using classroom tables and a blackboard. The blackboard is used as the bottom shelf. The second and middle shelf was then setup by using two overturned tables stacked upon each-other. The third and upper shelf was then made up with an overturned table placed on the ground. Width of the shelves are two tables wide. The table-legs of each shelf touch the shelf beneath to achieve a consistent change in height between shelves. Fiducial markers on the bottom shelf are simply placed on the blackboard. Fiducial markers for the middle and upper shelf are placed on additional tables, placed as the same way for each shelf, with an additional height of a blue bin. The additional height made it possible to have each marker in the same plane as the bins for the present shelf. Bins are placed on the floor for the upper shelf and on top of table-legs for the middle shelf. For the lower shelf, covers are placed on the blackboard. A picture of the mock-up material rack is viewed in Figure 21.

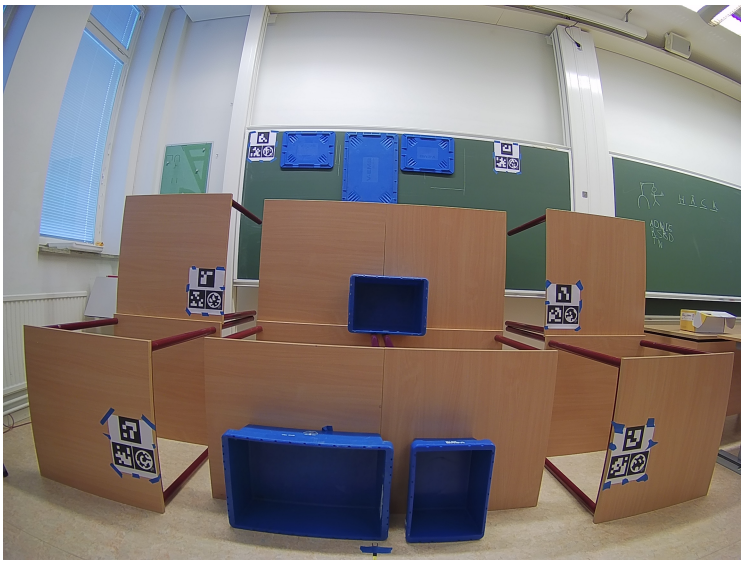


Figure 21: Mock-up material rack setup with tables and a blackboard

Testing includes taking pictures in a grid with the camera placed approximately above the middle of shelf. The grid consists of a picture for every 3m, 4m, 5m, 6m and 7m from the blackboard. For every step in the grid three additional pictures were taken moving 0.5m, 1m and 1.5m sideways (in the same plane as the tables and blackboard). For this test the specified camera was used, more specifically AXIS M3024-LVE, which is the type used in the factory. For each point in the grid, one lower contrast (contrast=50) and one higher contrast (contrast=100) picture was taken. Apart from the contrast, following settings are set for all pictures: resolution = 4MP, no compression, shutter time 1/125, no WDR and IR filter on (with no illumination).
After taking all pictures, all measurements of the mock-up rack are made. Using the detection rates of the fiducial markers and the distance between fiducial markers the best performing out of the three fiducial markers (AprilTag, ArUco & STag) was determined. On top of that, test the final algorithm by comparing the measurements and the calculated distances.

4.5 Results

Distance calculations from the Different fiducial markers test is displayed in Figure 22 and 23. Figure 22 displays the relative error of the calculated distance between the fiducial markers while varying the distance to the fiducial markers.

Different fiducial markers

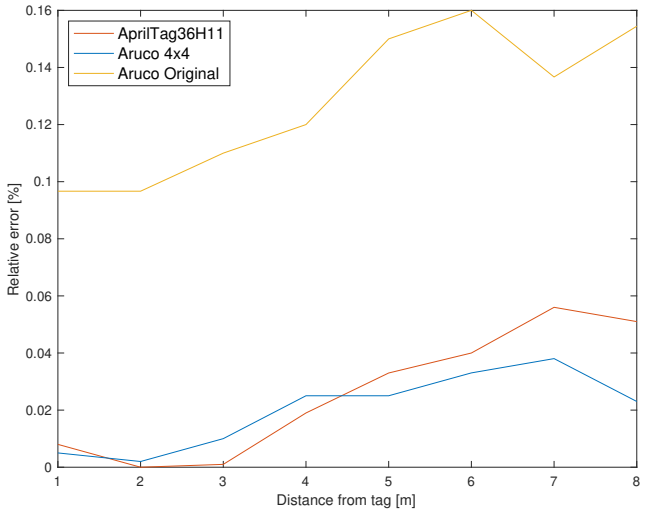


Figure 22: Relative error of calculated distance between fiducial markers while varying the distance to the fiducial markers.

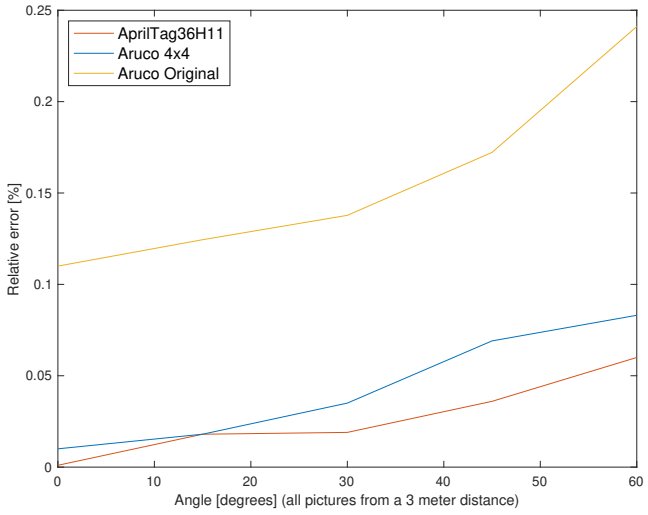


Figure 23: Relative error of calculated distance between fiducial markers while varying the angle and keeping a constant distance of 3 meters to the fiducial markers.

Figure 23 displays the relative error of the calculated distance between the fiducial markers while varying the angle to the fiducial markers (distance of 3 meters to the fiducial markers is held constant). The calculation of the relative error is displayed in Equation 4.

$$\text{Relative error} = \frac{\left|\hat{L} - L\right|}{L} \cdot 100\%, \tag{4}$$

where: $\hat{L}$ = calculated distance, L = real distance.

Material rack simulation

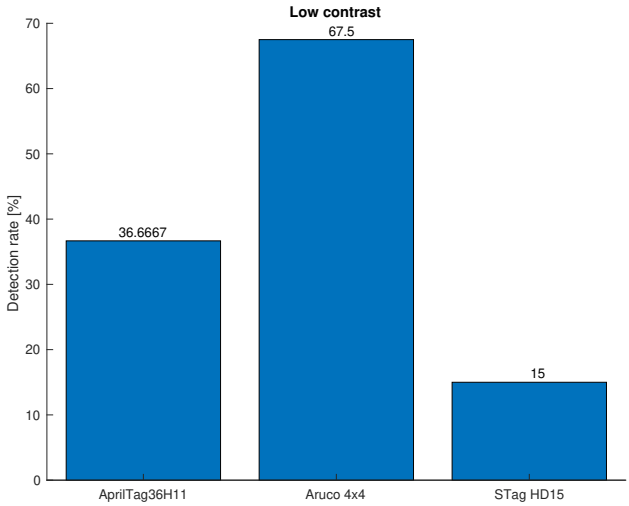


Figure 24: Detection rate of fiducial markers for the rack simulation with a low contrast (=50) setting.

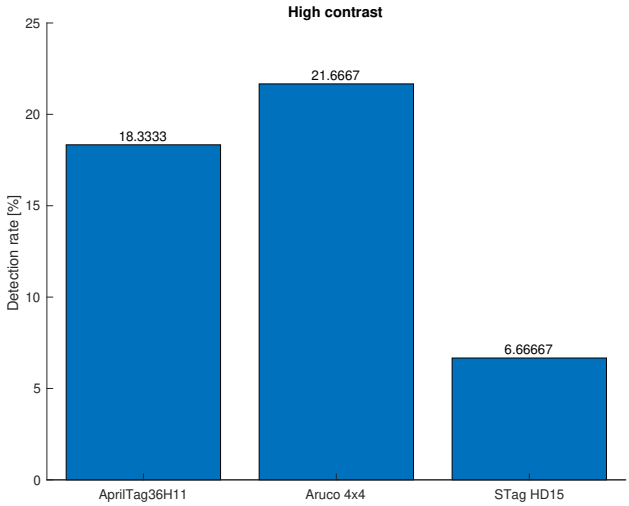


Figure 25: Detection rate of fiducial markers for the rack simulation with a high contrast (=100) setting.

Obtained detection rates (calculation shown in Equation 5) for the fiducial markers in the Material rack simulation test is viewed in Figure 24 and 25. Where Figure 24 displays detection rates for pictures with the low contrast setting (contrast=50) whereas Figure 25 displays the same for

the high contrast setting (contrast=100).

$$\text{Detection rate}_i = \frac{|\text{Detected markers}|_i}{|\text{Markers}|_i} \cdot 100\%,$$

where: $|\text{Detected markers}|_i$ = number of detected markers for marker i (across all pictures),
$|\text{Markers}|_i$ = number of markers for marker i (across all pictures),
i = AprilTag36H11, Aruco 4x4 or Stag HD15.

(5)

4.6 Discussion

A number of aspects have to be considered when deciding how to use the camera and the fiducial markers in the system to position storage bins. How the properties of the camera affect the setup of the system will be discussed in Camera limitations. What type of fiducial markers that will be used and why it is best suited for the application will be discussed in Fiducial markers.

4.6.1 Camera limitations

The first test where different fiducial markers were tested was done by taking pictures with the camera of a mobile phone, however the two later tests were made by taking pictures with a dome camera. These two different types of camera have some significant differences that limits how they can be used. The camera of the phone used in the first test has a much smaller field of view (FOV) compared to the dome camera. The reason a camera with a greater FOV is used is because a relatively big area needs to be covered but the camera has to be positioned relatively close to the material rack due to logistic reasons. A camera with smaller FOV would be able to get a higher resolution from the same distance but would not be able to cover enough surface area.

4.6.2 Fiducial markers

The Different fiducial markers and Material rack simulation tests both combined show that ArUco4x4 is the most optimal fiducial marker for the application.

The first test, Different fiducial markers, enables us to exclude Aruco Original from further testing and keep testing ArUco 4x4 and AprilTag36H11. Even though the test is done using a camera with higher resolution and lower FOV, the results show that ArUco Original contribute to higher error relative to the error from ArUco 4x4 and AprilTag36H11. To be more precise, the MRE (mean relative error) of ArUco Original when calculating the distance in the Different fiducial markers test is $\approx 12.8\%$ while varying the distance and $\approx 15.7\%$ while varying the angle. The MRE of distance calculation for ArUco4x4 and AprilTag36H11 is respectively $\approx 2.0\%$ and $\approx 2.6\%$ while varying the distance, along with $\approx 4.3\%$ and $\approx 2.7\%$ while varying the angle.

The second test (Material rack simulation) is testing how the fiducial markers will perform in a (almost) real world environment, which is achieved by setting up the fiducial markers on a mock-up material rack. By testing different distances to the mock-up material rack it was possible to simulate height differences in the factory. Furthermore the camera was also moved sideways in the test, which simulates the camera moving sideways without changing its height in the factory. The fiducial markers tested were ArUco 4x4, AprilTag36H11 and STag HD15. ArUco 4x4 and AprilTag36H11 are considered due to results from the previous test. STag is included due to the results from other sources, which shows that STag performs well compared to ArUco and AprilTag [23] (high detection rate and low MRE when calculating distances). It was also tested if the appropriate contrast is either high (contrast=100) or low (contrast=50) by taking a picture

for both settings for each camera position. The results show that the detection rates for every marker is significantly lower for the higher contrast setting. For instance the detection rate for ArUco 4x4 was $\approx 68\%$ for the low contrast setting and $\approx 22\%$ for the high contrast setting. With the higher detection rate in mind, we proceeded using the lower contrast setting (contrast=50) for the future. Additionally, detection rates for the low contrast setting were $\approx 68\%$ for ArUco 4x4, $\approx 37\%$ AprilTag36H11 and 15% STag HD15.

The explanation for the lower detection rates (Material rack simulation compared to Different fiducial markers, where all fiducial markers were detected) is probably because the lower resolution of the camera used in the former test, 4MP compared to 10MP used previously in the Different fiducial markers test. But the detection rates from the test done later, Material rack simulation, is more relevant because the test used the camera that is going to be used in the factory. Due to the low detection rate of STag HD15, it is not considered anymore and testing the MRE of distance calculation is therefore not needed.

The higher detection rate of ArUco 4x4 (≈ 1.8 times higher than AprilTag36H11 and ≈ 4.7 times higher than STag HD15) and previous mentioned results which show that the MRE of distance calculation of ArUco 4x4 is similar to that of AprilTag36H11, is simply why ArUco 4x4 is the most optimal marker for the application.

In Chapter 4.3.2 it is described that a step in detecting the fiducial markers of ArUco and AprilTag utilises image rotation to increase the marker detection rate. This is not used for the marker tests due to the fact that this improvement was not discovered at the time. The legitimacy of the test is not worse due to the higher detection rate with rotation. The test where the marker detection improvement was discovered and used is covered in Chapter 5.

4.7 Summary

In order to successfully position the storage bins, a number of different sub tasks has to be solved. First of all the camera needs to be calibrated to remove the natural distortion in the image. Since the image is only a flat two dimensional projection of reality without a sense of scale it is necessary to use some form of reference points in the image. As the reference points the model described in this thesis uses fiducial markers for both location and scale. Each shelf on the material rack is provided with a fiducial marker on each side to be used as references in the detection algorithm. This is used to define on what shelf each storage bin is located on. A number of different fiducial markers are evaluated in order to find what works best on this application since their performance has a high impact on the total performance of the system. To detect the storage bins itself a machine learning algorithm is used that is evaluated on its own in Chapter 3.

5 Evaluation

The main purpose of the project is to find the position of storage bins located on a material rack. In order to successfully detect the position of storage bins every part of the system has been developed and evaluated on its own. To make sure that all parts work well together the system needs to be evaluated on a test setup similar to the environment of the final application. Not only does the evaluation test whether the system works or not, the evaluation must also grade how well the system works on the test setup and what the expected error rate of the entire system will be.

5.1 Sub-goals

The goals for the Evaluation is as follows:

- Determine the acceptable positions of the material rack.
- Find a suitable position for the markers such that they are visible from the camera.
- Determine how the position affects the error of bin detection, for both the middle point and width of the bin.
- Determine overall error for the detection of bins, for both the middle point and width.

5.2 Test

This chapter describes the process in evaluating the performance of the entire system. The accuracy was measured in the test named Material rack at site and the positions of fiducial markers was evaluated in the test named Position of fiducial markers. The specified camera, AXIS M3024-LVE, was used.

Position of fiducial markers
Since the robot has to be able to reach the components in the storage bins it is important that the fiducial markers are not blocking the storage bin (see Figure 26). Therefore a mount for the markers was created to allow them to be accessible for the camera without preventing the robot from reaching the bins. Moreover the fiducial markers was mounted in such a way that they are not causing any danger for workers in the area.

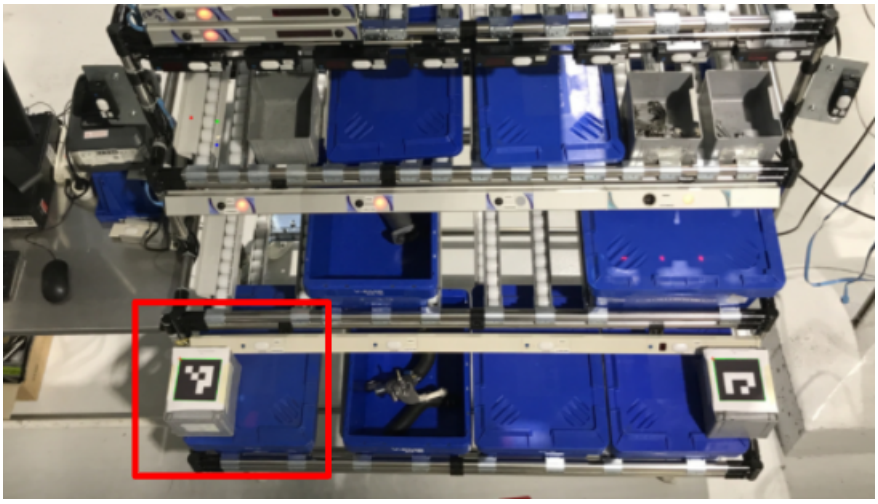


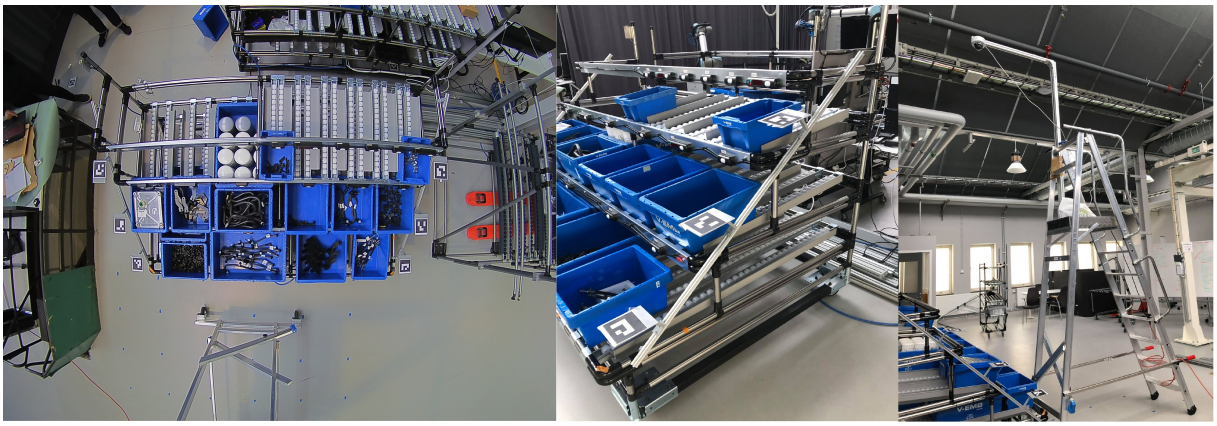
Figure 26: Fiducial markers with bad positioning.

To manage the insufficient positioning of markers, a mount was created for the specific material rack that allows for some flexibility on how the markers are positioned. The mount was constructed of aluminum profiles and plastic shelves which the markers are attached to, the mount can be seen in *b)* in Figure 27.

Material rack at site

To further test the conclusions from previous test (see Chapter 4.4), it was of importance to test the method in a applicable environment. Therefore a test was performed in a development environment with similar appearance as a factory, with industrial lighting and the proper material rack. The goal of the test was to make sure that the position of the markers is chosen in such way that the camera easily sees them, without the position being dangerous to workers in the area. Furthermore the goal of the test was to make sure the algorithm can detect both markers and storage bins with good accuracy. One of the main evaluations from the test was at what accuracy the storage bins can be localized, since the accuracy can estimate how well the method would perform when used in a real factory.

The setup of the test can be seen in Figure 27 and was supposed to simulate a real factory. Each shelf of the material rack was mounted with a marker on each side to be used as references in the detection model. The markers was placed in such way that they are on the same distance from the floor as the top surface of the storage bins. To achieve the same level for markers and top surface of bins, the markers were placed on angled plates attached to the material rack. In addition, the markers were aligned with the bottom line of the bins as seen from above (see *a)* in Figure 27). Only *ArUco 4x4* were used as markers in this test since previous results (see Chapter 4.6.2) had showed that they were superior in this application. YOLOv5 with a confidence threshold of 0.7 was used for all tests.



a) b) c)

Figure 27: a) Material rack used in test and that is used in the factory. b) Position of markers. c) Camera mount to simulate the height of a factory.

Material rack at site, height & optimization comparison

To test the camera height of the application, images were taken in a grid pattern as in Figure 28 with two different camera heights, three meters and four meters above ground. An optimization for calculating the position (see Chapter 4.3.6) was also evaluated, therefore each camera height was tested with and without the optimization. This enabled a comparison of how the performance of the system changes with the camera height and/or the optimization. Since images were taken in a grid pattern the result would also show where the camera can be positioned while still being able to detect all markers and bins.

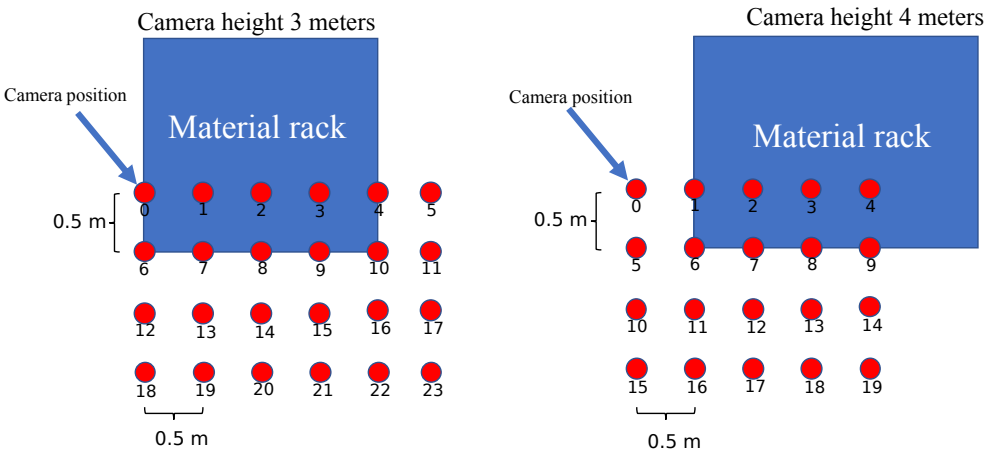


Figure 28: Grid pattern used to take pictures of the material rack for camera height of 3 meters and 4 meters. Each grid point displays the cameras position relative to the material rack.

Material rack at site, dataset comparison

In addition to evaluating the performance with different camera heights, it was important to measure the performance at the different camera positions shown in Figure 29. Note that the positions were slightly different compared to the previous setup, this time the right most positions (4, 9, 14 and 19) were aligned with the center of the material rack. Since the camera in the final application will be mounted on the height of 4 meters, this test was performed on that height. The main purpose of the model is to find the position of the storage bins placed on the material rack, therefore the precision of the estimated position will be the determining factor of the performance of the entire system. To evaluate the precision ten images will be taken for every position shown in Figure 29. The estimated position of the midpoint of every storage bin and their width will be compared to their true values in order to evaluate the performance of the model. It is also of interest to determine the limit of translation allowed for the material rack when the camera is static. By changing the camera position during the test, it is possible to determine the allowed translation of the material rack in the horizontal direction.

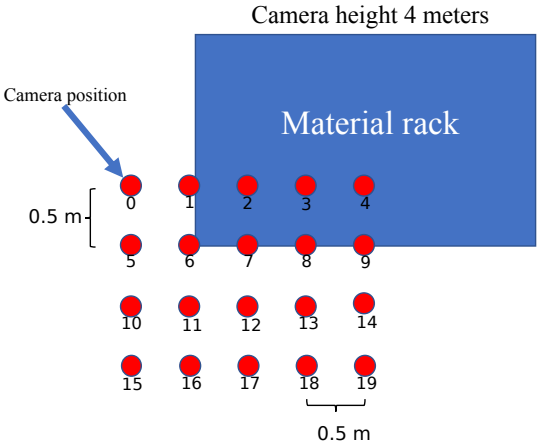


Figure 29: Camera positions used to evaluate the performance at different positions, camera height 4 m.

Material rack at site, vertical translation

A test was conducted to determine the allowed translation in the vertical direction of the image when the material rack is vertically aligned. At a position on the center line of the material rack an image was captured incrementing by 100 mm vertical translation each time, 14 more images were captured.

5.3 Results

The results for the evaluation of the system can be divided in two main parts, Position of fiducial markers & Material rack at site. Position of fiducial markers shows how the fiducial markers are placed. Material rack at site handles the evaluation of the system, precisely the detection rate and error of the bin detection. The detection rate is the quotient of the number of detected bins and the number of total bins for a set of images. The error used in the results is defined as (Equation 6),

$$E_i = \hat{x}_i - x_i \tag{6}$$

where $\hat{x}_i$ is the calculated middle point or width and x_i is the real middle point or width for a storage bin. The mean absolute error (MAE) and standard deviation of the absolute error(σ) is calculated using Equation 7 and Equation 8, n denotes the number of errors for a given set of storage bins.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|E_i| \tag{7}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(|E_i| - MAE)^2} \tag{8}$$

Position of fiducial markers
The position of fiducial markers were chosen in the most convenient way and is shown in Figure 27, note that this position of fiducial markers is the one used in the Material rack at site tests (see Figure 5.2). The fact that the lower part of the markers are aligned with the bottom of the bins (as seen from above) and the bottom of the markers are in level with the top of the bins enables us to utilize the y and z coordinates of the markers.

Material rack at site, height & optimization comparison
As seen in Figure 30 the model was successful at detecting all six markers at every tested camera position from 4 meters height above ground. The model could only detect all markers for a camera position 3 meters above ground when the camera was not moved more than 0.5 meters to the right or left from the center line of the material rack.

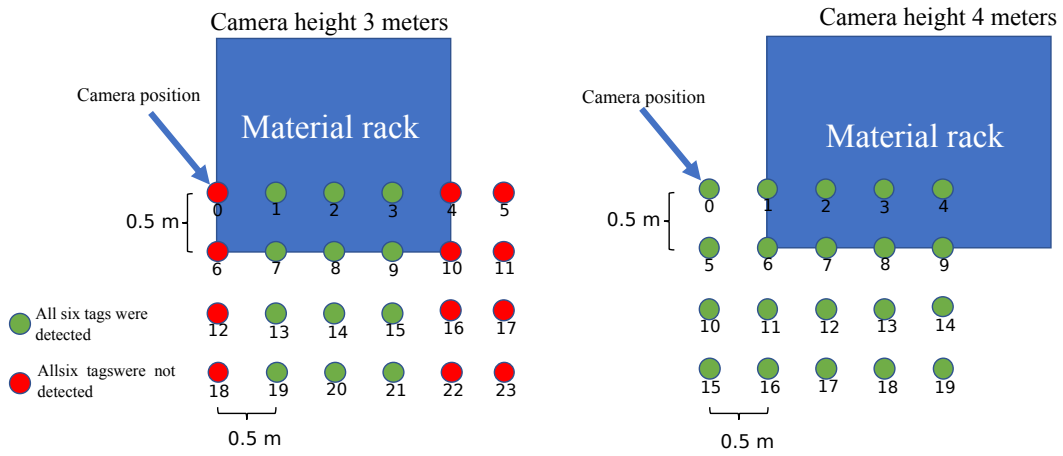


Figure 30: Visual representation of camera positions and if all six markers for a given position is detected. Each position in the grid is presented with an image id. Green grid points represent camera positions were all six markers are detected and red grid points represents camera positions where five or less of the six markers were detected.

The result from the first material rack site test, is displayed in Figure 31 and Figure 32. Figure 31 displays the error of the middle point of storage bins for the detection model trained with the old dataset, Figure 34 shows the same for error of the width of storage bins. The top/bottom of the dashed lines represent the maximum/minimum of the error, the middle red line represents the median of the error and the top/bottom lines of the boxes (blue lines) represent interquartile range of the error for a given position. In addition to the results from different camera heights, the figures also shows the results when using the optimized distance calculations. For the midpoint calculations positive values corresponds to the estimated position being to the right of the true position and for the width calculations a positive value corresponds to the estimated width being greater than the true width.

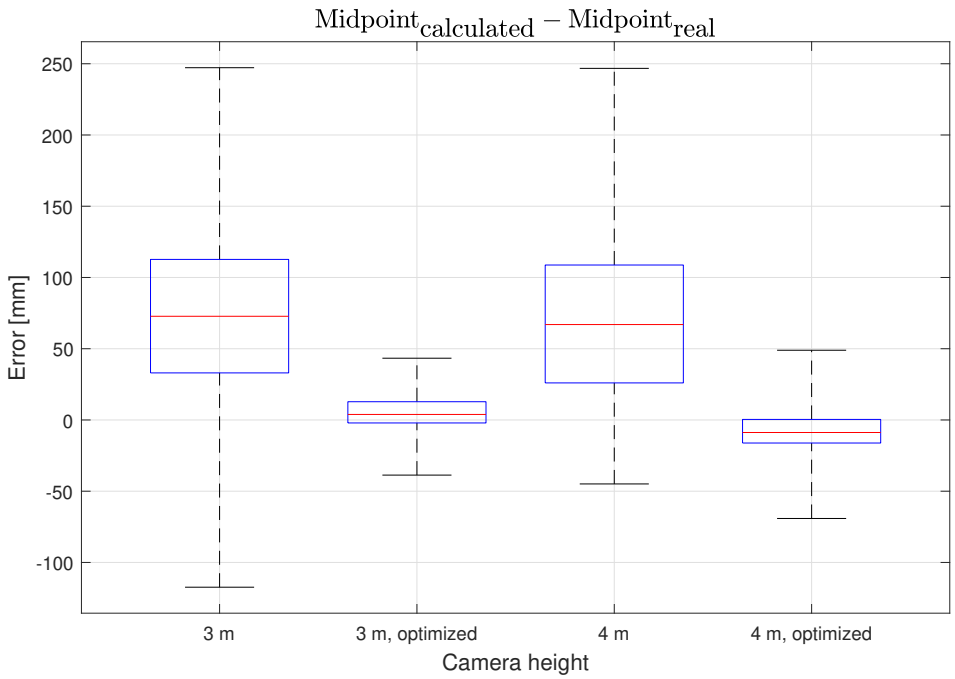


Figure 31: Error for the midpoint of the storage bins, for camera heights of 3 meters and 4 meters with/without optimization.

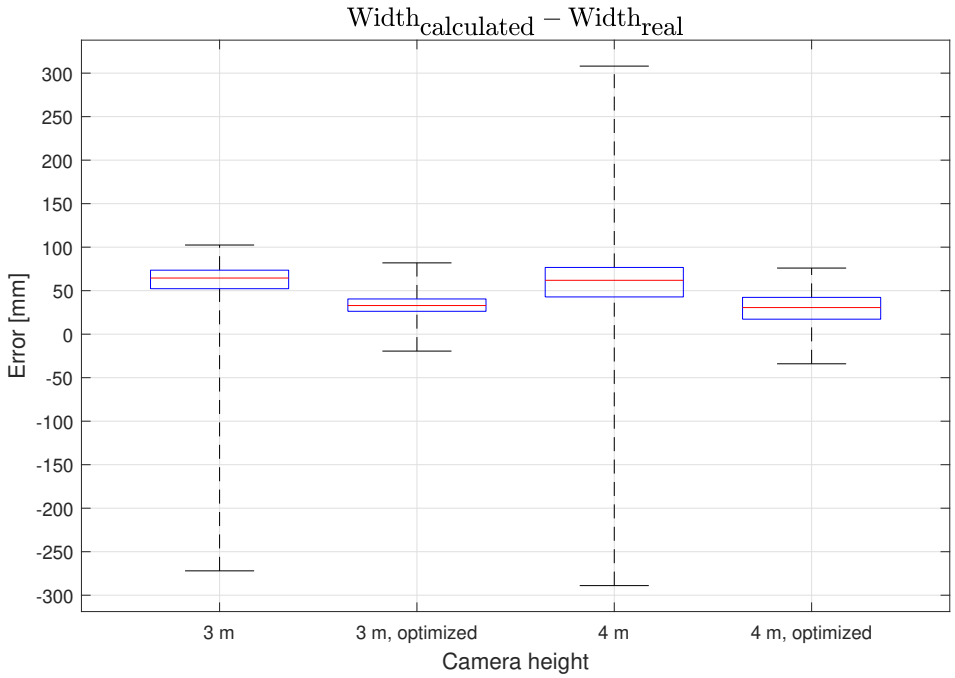


Figure 32: Error for the width of the storage bins, for camera heights of 3 meters and 4 meters with/without optimization.

Material rack at site, dataset comparison

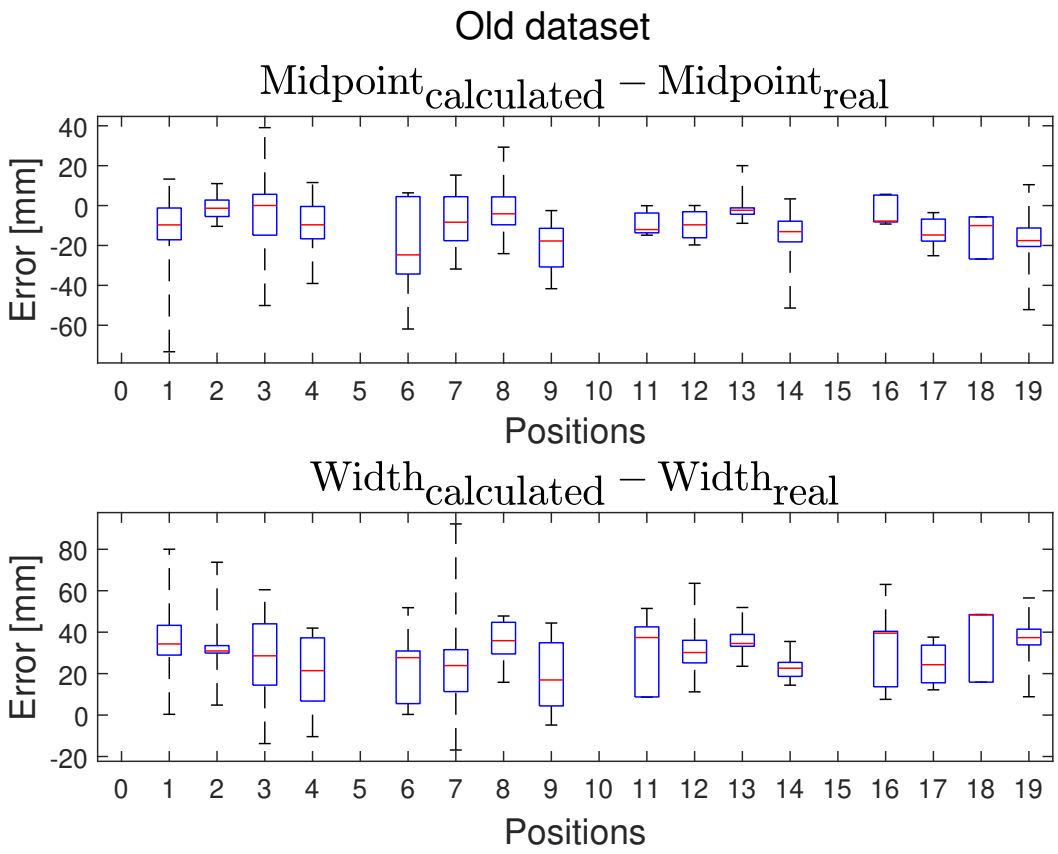


Figure 33: Error of coordinate for the middle point and width of bins. The boxplot shows the error for each position with 10 pictures respectively, the detection model for bin detection is trained with the old dataset.

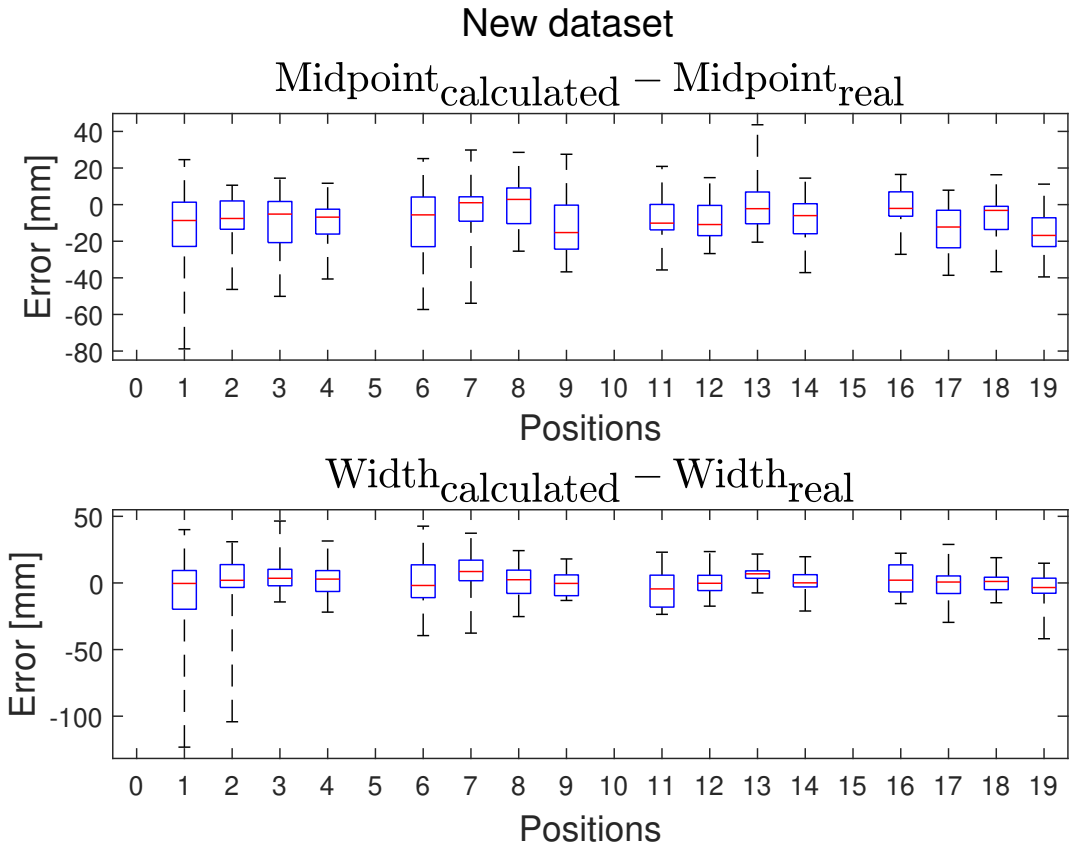


Figure 34: Error of coordinate for middle point and width of bins. The boxplot shows the error of each position with 10 pictures respectively, the detection model for bin detection is trained with the updated dataset

The result from the second material rack site test, is displayed in Figure 33 and Figure 34. Figure 33 displays the error of the middle point as well as the error of the width of storage bins for the detection model trained with the old dataset, Figure 34 shows the same for the detection model trained with the updated dataset. For each position (x-axis of Figure 33 and Figure 34) 10 pictures were taken. The top/bottom of the dashed lines represent the maximum/minimum of the error, the middle line of the boxes (red line) represents the median of the error and the top/bottom lines of the boxes (blue lines) represent interquartile range of the error for a given position. At positions 0, 5, 10 and 15 the six markers were not detected and therefore the detection model of bins is not evaluated on these positions.

When using the old dataset the mean absolute error across all pictures is 13.2 mm for the middle point and 29.9 mm for the width of the bin. The standard deviation of the absolute error across all pictures is 14.4 mm for the middle point and 17.7 mm for the width of the bin.

When using the new dataset the mean absolute error across all pictures is 12.5 mm for the middle point and 11.6 mm for the width of the bin. The standard deviation of the absolute error across all pictures is 11.3 mm for the middle point and 13.7 mm for the width of the bin.

The position considered the worst with the data set was position 1. The mean absolute error for position 1 is 16.5 mm for the middle point and 23.8 mm for the width of the bin. The standard deviation of the absolute error for position 1 is 16.9 mm for the middle point and 29.0 mm for the width of the bin.

The detection rate of storage bins using the old dataset is presented in Figure 35. Each position contains a number representing the detection rate at the specific camera position, with values

averaged for the ten images taken. When running the system on the new dataset all boxes were detected in every image at the positions where all six markers were detected. In other words, the detection rate for storage bins with the new dataset was 100% for all positions except positions 0, 5, 10 and 15 (where all six markers were not detected).

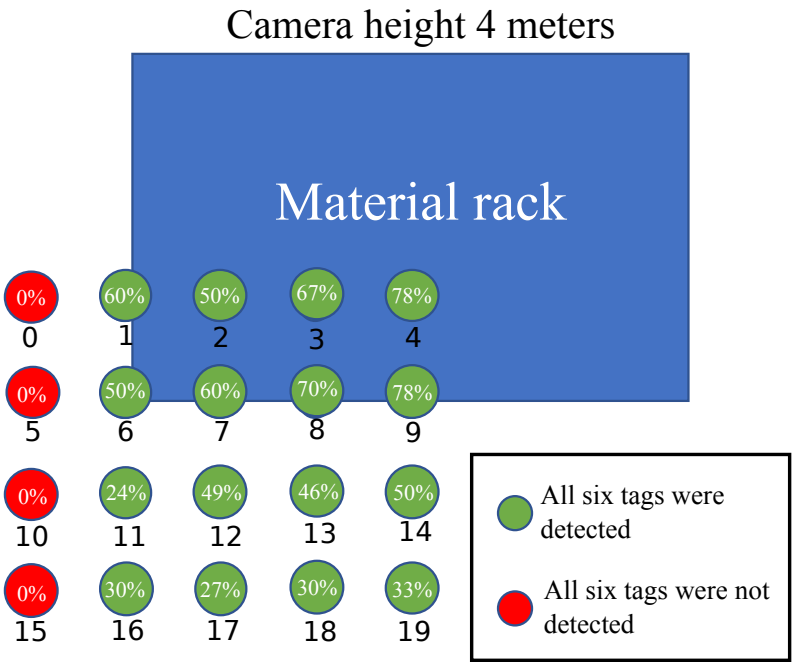


Figure 35: Detection rate of storage bins using the old dataset.

Material rack at site, vertical translation
The vertical translation test showed that fiducial marker ids could be found in images where the vertical translation was less than 1100 mm from the center line of the material rack.

5.4 Discussion

When implementing a system or a model it is important to understand under what circumstances it was designed to be used in and what performance one could expect from it. In camera limitations it will be discussed what the requirements are for the camera position and in accuracy of the system it will be discussed what the expected accuracy of the system is.

Camera limitations, Figure 35 shows the positions where all six markers managed to be detected when the camera was located 4 meters above ground. Detection of all six markers is the first criteria that must be fulfilled for the method to function and is therefore a good indicator of the possible positions for the camera. The markers are also on the edge of the material rack which means they are the first to be located outside of the image when the entire material rack is moved. Positions 4, 9, 14, and 19 are on the center line of the rack as described in Chapter 5.2 and the positions are located 0.5 meters away from one another in the vertical and horizontal directions. The material rack in Figure 35 is horizontally aligned with the images in this test and the translation from position 4 to 0 is also horizontal. From this it can be concluded that the limit for horizontal translation of a horizontally aligned material rack is between 1500 mm and 2000 mm. A second test specifically for translation was conducted to instead determine the limit for a

vertically aligned material rack to be vertically translated. From this test we could conclude that 1100 mm translation was possible but that 1200 mm was not possible.

These tests used the camera calibration matrix discussed earlier in Chapter 4.2.3, the calibration causes information at the edge of the image to be discarded currently which makes the translation limits smaller than they could have been, especially since the pixels at the edge corresponds to the largest distances. The reason why this was accepted is because a balance between an over representation of empty pixels (black pixels) and removed information had to be found. If the usable information is surrounded by too many black pixels, the information would get minimized and the overall performance of the system would be compromised.

Accuracy of the system

The accuracy of the estimated position for each storage bin has possibly the largest impact on the overall performance of the system. The kitting robot relies on the accuracy of the output to safely reach the front center of a desired bin before picking up a component. During the evaluation of the system the accuracy of the estimated positions where evaluated on different camera heights, usage of fiducial markers in the calculations and what dataset used when running the system.

To evaluate how well the system performs on the desired camera height of four meters, the system was compared when running it on both three and four meters camera height. The results can be seen in Figure 31 and Figure 32 where the error of the estimated position of the midpoint and the width of the bin is presented. The figures clearly show that there is not any major differences when using the two different camera height, therefore the system can clearly be used at the desired camera height without loosing accuracy of the estimations compared to a camera mounted closer to the objects.

When making the calculations for the estimated positions of the bins, the impact of how the fiducial markers where used was compared. The initial system used only the known size of the markers as a reference on how to convert the size of an object in the image to the estimated size in the reality. In the evaluation this method was compared to a method where the distance between the markers on each side of the shelf was used as the reference in addition to the size of the marker. This comparison is also visualized in Figure 31 and Figure 32. The figures clearly show that there is a huge improvement of the accuracy when using the markers in this way. Since the size of the marker is quite small, the estimated position becomes very dependent on a precise detection of the marker when only using the marker size.

When evaluating the new dataset it was very clear that it had a huge improvement in detection rate compared to the old dataset as could be seen in Figure 35. With the new dataset every bin was detected in every image which indicates that the previous dataset was not robust enough for the application. Since the old dataset only consists of images of the material rack with the camera being straight above the rack, the model only learns to see the bins from that angle. Therefore the old dataset could not perform as well at the desired camera position since the bins are seen from a different angle. Besides the detection rate the accuracy of the estimated positions is the main aspect that determines how well the system is performing. As visualized in Figure 33 and Figure 34 there was not any noticeable difference in MAE for the midpoint when comparing the two datasets but the MAE for the width however was almost three times as high when using the old dataset compared to the new dataset. In addition the standard deviation was slightly lower for both the midpoint and the width when using the new dataset.

5.4.1 Fiducial markers not detected

When analysing the images from the material rack at site an issue was discovered that fiducial markers were not detected when they were clearly visible. This phenomenon appeared for both

the 3 and 4 meter test, position 2 and 8 for the 3 meter test and 14 and 19 for the 4 meter test. The positions where this occurred is considered to be some of the best positions for object detection as described in Chapter 3.6.8 which made the discovery even more strange. In Chapter 4.3.2 the process of rotating the image is described as a solution to this. Why rotation of the image results in better marker detection in some cases is unclear but the fact remains that every one of the previously undetected images could be detected using this method.

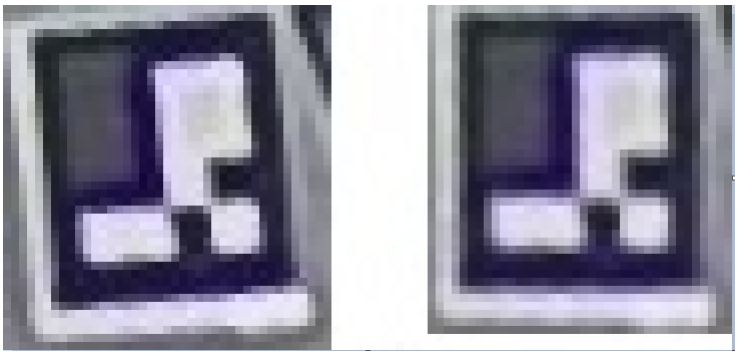


Figure 36: Marker that was initially not detected

Figure 36 shows to the left the marker when it was not detected and to the right when it was detected for image 19 for the 4 meter test. The marker has been horizontally aligned by the rotation when it manages to detect the marker. Image 14 could not detect the same marker without rotation, image 14 needed the same amount of rotation to be able to detect that marker. This might indicate that the detection algorithm for ArUco is not robust against markers not aligned horizontally. This contradicts the belief that the detection code utilises all necessary image manipulation and image analysis techniques to find all markers in an image. It would therefore be of interest to test other forms of image manipulation to see if they have a positive impact on marker detection.

6 Conclusion

Different object detection models were tested and compared in Chapter 3. YOLOv5 was concluded to be the best model for our purpose, due to the fast speed of inference, training time and ease of use. The YOLOv4 model initially had a higher mAP, but when conducting further testing it was shown that YOLOv5 could get higher than 99% mAP with the correct dataset. A trained model based on YOLOv5 was acquired and used in order to export pixel-coordinates.

The testing of augmentations for achieving a better dataset showed that cropping is a good method to increase mAP. Other augmentations on their own did not show a significant increase, although when combining many augmentation techniques, the mAP can be increased drastically. Furthermore, a new dataset was created by taking new pictures with the correct camera setup and more pictures. This new model based on the new dataset achieved a significantly better detection rate (all bins detected) when running inference on test images, compared to the old model based on the old dataset.

Three different fiducial markers (STag, Aruco, and AprilTag) were evaluated for our application in Chapter 4. Results showed that Aruco/Aruco 4x4 was the best marker for our application with a camera resolution of 4MP. The Aruco marker could be detected 1.8 times more than the AprilTag marker and 4.5 times more than the STag marker. In addition, the relative error when using the marker information to conduct distance calculations was also the lowest when using the Aruco 4x4 marker. The camera calibration and manipulation was also managed in Chapter 4.

Given our evaluation of the complete system in Chapter 5 and camera height of 4 meters, it was determined that the camera has to be located at most 1.1 meters from above the middle of the material rack to guarantee detection of all six markers. If the material rack aligned horizontally, the maximum horizontal translation with detection of all six markers increases to 1.5 meters. Further translation is possible in some cases but will depend on both alignment of the material rack and direction of the translation.

The different positions impact on bin location error was also determined from the evaluation in Chapter 5. The worst case for the positions evaluated was position 1, with a mean absolute error of 16.5mm and standard deviation of the absolute error at 16.9mm for the middle point of the bin. The mean absolute error across all pictures was determined to be 12.5mm with a standard deviation of the absolute error at 11.3mm, which implies that all evaluated positions (with all six markers detected) with a camera height of 4 meters is acceptable. From an earlier test with a worse dataset the detection rate suffered greatly when the camera move further away from the middle of the material rack. The higher detection rate and lower error for positions closer to the middle of the rack shows that a camera placement close to the middle of the rack is favourable.

The proposed system can determine the position of all bins located in an image in relation to the marker with ID 0, convert the position to WGS84 and afterwards send the information to a database. This accomplishes the goal for our system but to build a complete system for automated kitting the robot would need to pick up parts and not just find the bins. To pick up parts in the bin, the parts positions would have to be determined and a grasping point from that calculated. Based on the accuracy of the bin positioning, this may not be obtainable with the current setup. A camera mounted on the robot arm is therefore suggested.

References

[1] Papers With Code, “Real-Time Object Detection on COCO.” [Online]. Available: https://paperswithcode.com/sota/real-time-object-detection-on-coco (retrieved 2021-01-20).

[2] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V.-K. Papastathis, and M. Strintzis, “Knowledge-assisted semantic video object detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1210–1224, 2005, doi: 10.1109/TCSVT.2005.854238.

[3] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[4] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” *arXiv*, vol. abs/1911.02685, 2020, url: https://arxiv.org/abs/1911.02685.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv*, vol. abs/2004.10934, 2020, url: https://arxiv.org/abs/2004.10934.

[6] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *arXiv*, vol. 1506.01497, 2016, url: https://arxiv.org/abs/1506.01497.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[9] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 658–666.

[10] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.

[11] A. Van Etten, “Satellite imagery multiscale rapid detection with windowed networks,” in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 735–743, doi: 10.1109/WACV.2019.00083.

[12] G. Jocher, “YOLOv5”, [Software]. Ultralytics, 2020, Available: https://github.com/ultralytics/yolov5 (retrieved 2021-02-08).

[13] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 778–10 787, doi: 10.1109/CVPR42600.2020.01079.

[14] Colab, [Software]. Google, Available: https://colab.research.google.com/

[15] Roboflow, [Software]. Roboflow, Inc, Available: https://roboflow.com/

[16] J. Brownlee, “What is the difference between a batch and an epoch in a neural network?” *Deep Learning; Machine Learning Mastery: Vermont, VIC, Australia*,

2018, url: https://deeplearning.lipingyang.org/wp-content/uploads/2018/07/What-is-the-Difference-Between-a-Batch-and-an-Epoch-in-a-Neural-Network_.pdf.

[17] R. Kanjee. “YOLOv5 Controversy — Is YOLOv5 Real?” *Medium*. 2020. [Online]. Available: https://medium.com/swlh/yolov5-controversy-is-yolov5-real-20e048bebb08 (retrieved 2021-04-21).

[18] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407, doi: 10.1109/ICRA.2011.5979561.

[19] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014, doi: https://doi.org/10.1016/j.patcog.2014.01.005.

[20] B. Benligiray, C. Topal, and C. Akinlar, “Stag: A stable fiducial marker system,” *Image and Vision Computing*, vol. 89, pp. 158–169, 2019, doi: https://doi.org/10.1016/j.imavis.2019.06.007.

[21] Redis, [Software]. redislabs, Available: https://redis.io/

[22] NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY (NGA), “World geodetic system 1984 (wgs 84).” [Online]. Available: https://earth-info.nga.mil/index.php?dir=wgs84&action=wgs84 (retrieved 2021-05-12).

[23] M. Kalaitzakis, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, “Experimental comparison of fiducial markers for pose estimation,” in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 781–789, doi: 10.1109/ICUAS48674.2020.9213977.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY