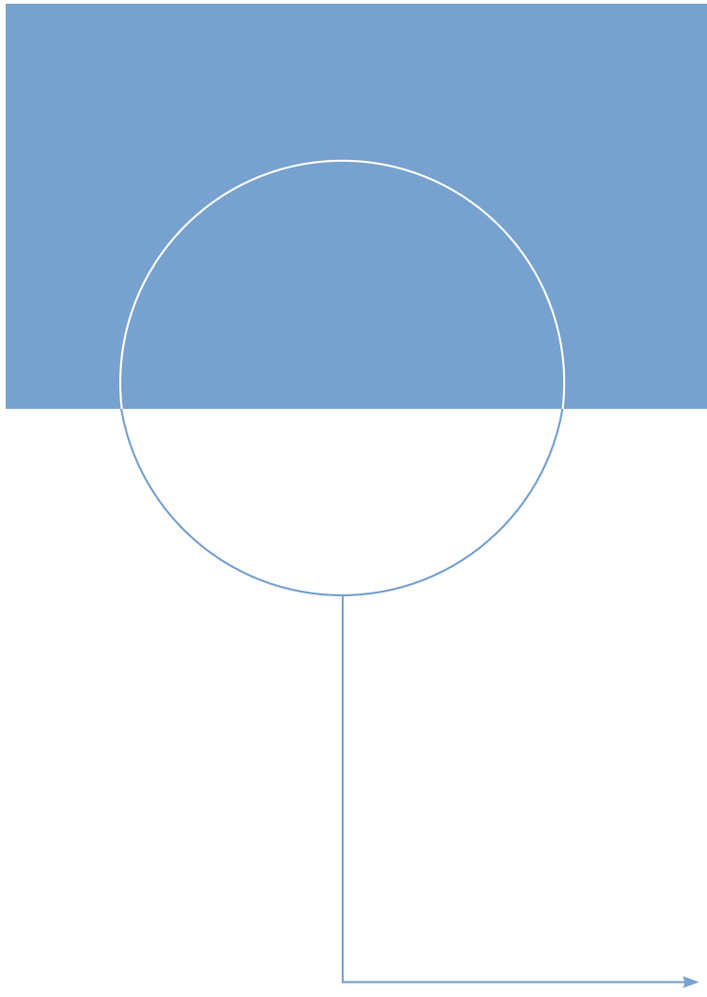


Master's thesis

2022

Dipanjan Pokharel

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Ocean Operations and Civil Engineering

Dipanjan Pokharel

Object Detection with SSD for Dimension Extraction in a Manufacturing Facility

June 2022

NTNU

Kunnskap for en bedre verden

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Object Detection with SSD for Dimension Extraction in a Manufacturing Facility

Dipanjan Pokharel

Product and Systems Engineering
Submission date: June 2022
Supervisor: Ola Jon Mork
Co-supervisor: Robert Skulstad

Norwegian University of Science and Technology
Department of Ocean Operations and Civil Engineering

NTNU
Kunnskap for en bedre verden

Department of Ocean Operations and Civil Engineering

Product and Systems Engineering

IP501909 - MSc Thesis

Object Detection with SSD for Dimension Extraction in a Manufacturing Facility

Author:
Dipanjan Pokharel

Supervisors:
Prof. Ola Jon Mork
Robert Skulstad

Industrial Partner:
Hallvard Tyldum
Vello AS

June, 2022

Abstract

Like in many other industries, Vision System has made significant impact in manufacturing industry. It is however understood that same vision system approach cannot be installed for different manufacturing facility with different circumstances and producing different products. At Vello AS, glass/door profiles are manufactured. The fibre creels used as a raw material can run out and needs to be replaced. Right now, inspections and replacements are manual. Due to the human factor, come fibre rolls can be missed out during the inspection and this can lead to non-uniformly structured final product. If the inspection could be automated with the help of robust vision system however, this can be prevented and can impact the overall production quite significantly. So, in this paper, different vision systems are looked upon and the one which looks the most promising for our case was selected. Vision system powered by deep learning, SSD network architecture, is selected and implemented to extract the dimensions from the fibre rolls. The aim is to make the facility independent of manual inspections required to know the fibre roll status. This way, it can prevent running out of fibre materials and ultimately, eliminate negative effects on the production quality. After number of experimentation, it was concluded that SSD network architecture can deliver a robust vision system for the facility. Object detection is capable of extracting the dimensions in millimeters once the pixel dimension is calibrated. However, there are many factors such as camera quality, lighting, camera strategy, training image quality etc. which are to be improved for a vision system. Once these things are improvised, the vision system can be installed at the facility for dependable results.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Motivation

In this report, a suitable vision system is to be studied which is able to feed-in the status of the raw material being used for the production of window or door profile in the manufacturing facility at Vello AS. Usually, the vision systems are used to inspect the final product[8]. Here however, we will try to inspect the raw material, which is the roll of glass fibre, stacked on a multi-layer cabinet and provide the necessary information so that the operator does not need to perform manual inspection every once in a while. Although it might sound pretty straight forward, there are a lot of challenges which are to be overcome.

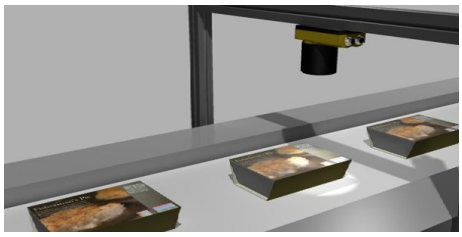
Figure 1: Rolls of Glass Fibre at Vello AS

As shown in Fig.1, the creels are placed on the shelves with multiple floors. Each production line has two racks, each rack consists of 168 creels. There are 8 different production lines which means there are 16 shelves and 2688 creels in total. The fiber from the creels with the help of the metal guides are fed, as shown in Fig.2, into a resin bath where they fuse together and go further to be shaped into a desired profile.

Figure 2: Fibre being fed into a hot resin bath

Currently, an operator does the inspection and manually replaces the skein. It is tedious. If the operator misses out on an empty skein and is not able to replace it in time, then the produced profile might have uneven fibre distribution. Hence, it seems like an effective vision system would be able to positively impact the production line at Vello AS. However, there's a catch to this. The fibre from the skein are being pulled out from the center and this makes it difficult for the vision system to recognise the changes in outer dimensions because there are none. Therefore, it is necessary to keep an open mind and come up with a clever strategy.

Given the circumstances in the manufacturing facility at Vello, attention immediately goes to a computer vision system. Computer Vision System is a tool which helps solve industrial problems, usually by eliminating the need of manual inspection and using a Vision System for better productivity is not a new concept. From Agricultural industry to medical industry, it is only becoming more and more popular[24]. Vision Systems are often used to perform the dull tasks involved in the process that can be fatiguing to a human, which consequently might lead to blunders. For example, if a person has to check the quality of potatoes produced in a farm all day long, there is a high probability that he/she is going to become inconsistent. Additionally, smart vision systems can substitute or even outperform skilled workers when it comes to performing dangerous and demanding works. Marriage of Vision System with Machine Learning has made it even more sophisticated and trustworthy. Use of AI in the vision system has improved to the capabilities that are closer to that of human vision.



(a) Vision System in food industry

(b) Vision System in Agriculture

Figure 3: Vision System in different industries

The development in vision system over the years has proven it to be a robust tool in the manufacturing industry. It is cutting costs, lead time, wastes and moreover, improved the product's quality by reducing errors. This does not necessarily mean that installing a vision system is always fruitful. There are vision systems which follow different strategies and one strategy might or might not be suitable for the given process. Hence, it is necessary to study what type of vision system, if needed, is the most compatible with the process associated. For example, there is a vision system that uses Deep Learning and there is a traditional computer vision system. A traditional vision system could be the answer here as there is basically only one type of raw material to be inspected[25]. However, deep learning vision system will be looked into thoroughly as there are different factors which might play against the results if the vision system is not smart enough.

1.2 Industrial Objectives

While looking into possible vision system solution with respect to the manufacturing environment, it is important that the system we choose to work with must satisfy the following objectives.

⇒ **Vision System must be able to replace manual inspection**
As mentioned earlier, the primary objective is to replace the manual inspection with a vision system which is smart enough to notify the operator about the status of raw material. The operator should be able to know when a skein has approximately 100 m of fiber left so that it gives him/her the time to replace it without any rush.

⇒ **Vision System must be flexible**
While replacing a skein with a new one, the operator needs to manually glue the ends of the fibres. Vello AS aims to automate this as well in the future. Hence, the vision system we decide on installing must be flexible enough that it is compatible with the automated splicing technique Vello AS might use in near future. If required it should also be able to communicate with the mobile robots as the company is considering to automate more and more processes involved in the manufacturing.

⇒ **Monitoring System must be economically feasible**
Most importantly, the vision system should be worth the impact it can make. The system must have a cheap unit price to cover all the creels.

2 Methodology

The thesis will briefly look into different vision system strategies used in the similar industries. Different vision sensors/equipment normally used will also be looked upon and the relevant parameters will be compared. The comparison might include hardware components along with the software components. Variety of strategies to maximise the accuracy along with the promising vision technologies will also be discussed. State-of-the-Art and future direction in the field of vision system will also be included.

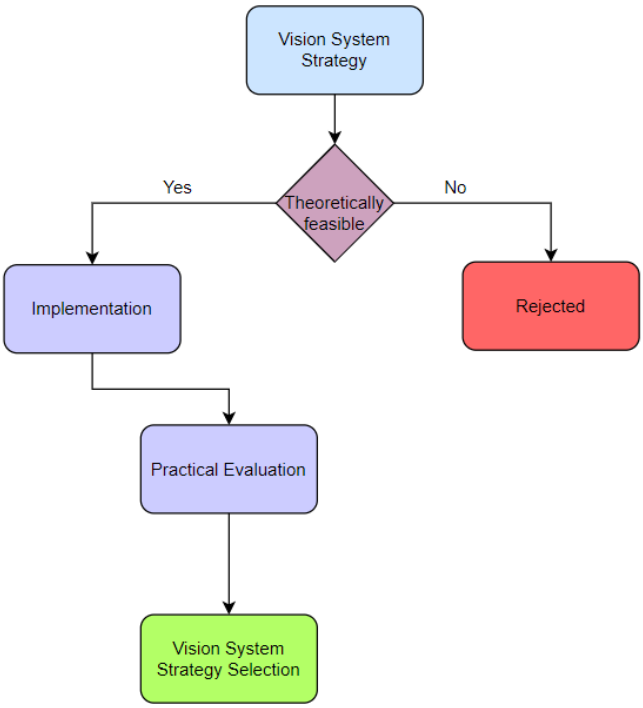


Figure 4: Simple decision-making flowchart

The vision system technique which looks most promising for our case, will be discussed in detail. Profound knowledge about these systems will be gathered from the relevant resources. The system will be experimented on the images acquired from the manufacturing facility. The results obtained from the experiments will be displayed. They are evaluated on the basis of precision they deliver and the final recommendation will be made to Vello AS along with the relevant guidelines. All the factors that have significant impact on the results' precision along with the ways to improve those shortcomings will also be thoroughly discussed.

3 Research Questions

The questions this report tries to answer are listed and briefly elaborated below:

1. What are the vision system concepts which can contribute to a robust vision system?

   Different strategies normally utilised by different vision systems will be discussed. How are these techniques being or can be implemented to build a robust vision system?

2. Can the selected concept be implemented at the manufacturing facility?

   Does the vision system concept satisfy the industrial objectives? Also, is it or will it be able to deliver results with required precision and accuracy?

3. How can the selected concept impact the overall production line?

   If the vision system is able to deliver or will be able to deliver, how can it prove itself to be a valuable asset? What are the problems it is able to solve or minimise at the manufacturing facility? Why should Vello have this system installed?

4. What are the external factors that can influence the vision system's quality?

   What are the different factors that can affect the quality of results obtained using the selected vision system concept?

5. How influential can these external factors be?

   How significantly can these factors affect the quality of our results? Can they be ignored or are they a priority?

6. What are the relevant guidelines?

   What are the different techniques that can be implemented in order to negate the negative effects of the external factors in order to get results with better accuracy?

4 Vision System Concept Research

All the relevant information on vision system are collected from different sources and are discussed in this section. Basically, different components that make up a vision system strategy and the theories that connect these components with the scope and the objective of the thesis are put under the spotlight.

Over the years, different strategies have been implemented in different industries in order to have an efficient vision system that can actually make a positive impact. The strategy does not have to be same just because the vision system is being used in the same industry at the similar point of the product's life-cycle. For example, in our case, the change in dimension cannot be recognised with a horizontal view because the thickness of the skeins changes from the inside. Hence, vision system that are already in use may have to be improvised. However, information on different computer vision strategies from available literature are discussed below.

4.1 Embedded Smart Camera

Embedded smart cameras are the ones with an inbuilt computing system. Among the embedded smart cameras, Open Machine Vision Camera is under focus. It is a low-power smart camera that lends itself naturally to wireless sensors and Machine Vision applications. It makes use of Python 3 interpreter which makes it desirably flexible. However, some of the other embedded cameras are compared in the table below.

Sensor	eCam	CMUCam3	Pixy (CMUCam5)	MeshEye	MicrelEye	OpenMV
Power	230mW	500mW	700mW	260mW	N/A	500mW
External Storage	No	SD Card	No	MMC	No	uSD/Flash
Scriptable	No	Yes	No	No	No	MicroPython
Image Processing	No	Frame Differencing	Color Tracking	Object Detection	Image Classification	Face Detection
Commercial avail.	No	No	Yes	No	No	Yes
Dimensions(cm)	2.0x2.8	5.5x5.7	5.3x5.0	N/A	5x5x3	3.0x3.5
Open Source	No	Yes	Yes	No	No	Yes

Table 1: Comparison of different embedded cameras [15].

Monocular camera system can achieve real-time measurements of an object . It is a camera which can detect distant objects and lane boundaries and when the system adopts open MV camera, Python scripts can process the image quickly[16]. When the image captured are subjected to processing such as Gaussian Filter and Canny Edge Detection, target distance and dimension can be obtained.

4.2 Digital Camera System

In this vision system, a simple digital camera is used to consistently feed in the image of the target to a computer. The image then is processed with the help of Python script and the necessary dimensions are measured. As shown in Fig.5, a reference object(with known dimensions) is used in order to calibrate, then the target objects are measured accordingly. It is accurate enough when the camera is parallel to the face of the objects [29] and this feature makes it tricky for us because, as mentioned earlier, it is impossible to realise the change in dimensions from the side view.

Figure 5: Calibration with reference object(left), Measurement(right) [29]

4.3 Stereo Vision Camera

It is a type of camera with two or more image sensors which allows the camera to perceive depth. Two different 2D images taken from different positions(right and left) are used to find stereo disparity[7]. The disparity is then used to create a depth image. In a production industry, stereo vision cameras are used to acquire two 2D images. From the contours detected on these 2D images, 3D contours are reconstructed and the required dimensions are obtained[10]. They are cost-effective and possibility of real-time mapping makes them a valuable vision technology.

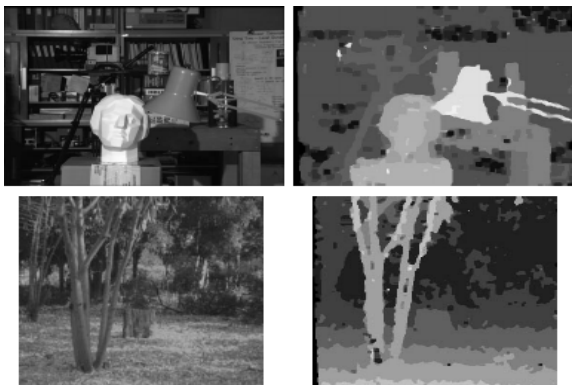


Figure 6: Example of depth image from stereo image[17]

4.4 Image Processing

Images obtained from a camera might be usable but are prone to higher uncertainties. Processing the image using relevant algorithms has the potential to substantially improve the quality of a vision system. Moreover, to find intricate details in an image can be made possible after image processing. For example, maybe distance between edges of an object in an image are to be determined but it might happen that the edges are not quite clear. Here, image processing plays a vital role and prepares the image for the aimed operation.

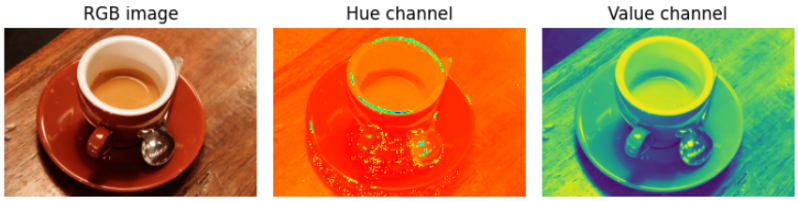


Figure 7: Image processed via Hue and Value channel [3]

Scikit-image is such an image processing library. It is easily accessible, powerful and has been consistently evolving[31]. The algorithm from the library can be implemented on RGB image to acquire a totally different perspective as shown in Fig.7. It can significantly minimise the effect of environmental uncertainty factors such as light intensity. As it is an open source library, there is no need of spending time and energy in other similar commercial packages. The development team of scikit-image collaborates on GitHub where we can look for support if an issue arises.

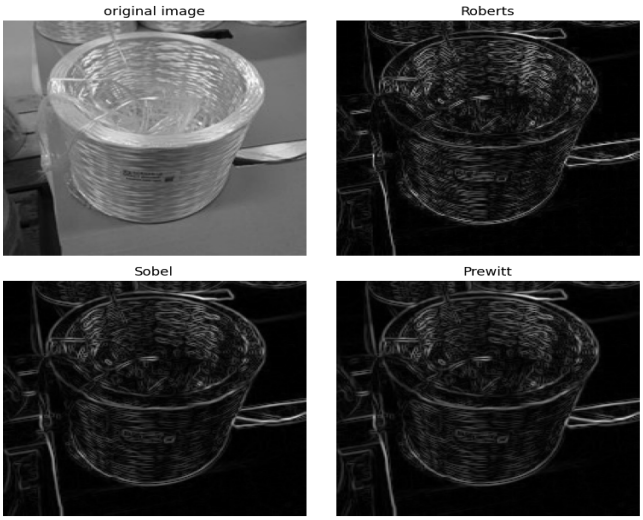


Figure 8: Application of different edge detection filters from scikit-image on the glass fibre skein image [A.1]

4.5 Metrology

Using a vision system to obtain different dimension is, as expected, not very straight forward as a conventional contact-based measurement. When using a vision system to measure features of the target object, calibration is the most important stage. Coefficients are calculated using the reference object which dimensions are known[12]. The coefficients are the numbers that relate the dimension in units with number of pixels in the image.
Mathematically,

$$Pixels\ per\ metric = Object\ Width\ in\ pixel / Known\ width$$

Once the calibration done and the coefficients evaluated, number of pixels are used to obtain the actual dimension. Lens distortion and camera view angle play a vital role in determining the accuracy though. Also, it should be such that the reference object is constantly present in the camera view.

4.6 State-of-the-Art

It is always wise to know the heights the technology has reached so that it gives a sense of possible sophistication that can be incorporated for a given system. Deep learning, a type of machine learning, contributes to the modern and highly sophisticated computer vision system. It is realised using Artificial Neural Network, which can be defined as a matrix of interconnected nodes with flow of data from input to output as shown in Fig.9. ANN tries to imitate a human neuron system to accomplish decision making. The system can be trained by using the data available and since it is a autonomous system, there is no need of manual programming.

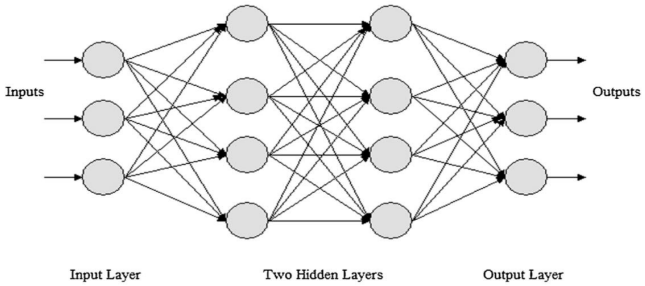


Figure 9: Simple Structure of Artificial Neural Network [2]

For example, let us say there is a need of a computer vision system to identify different plants out there. This is going to be a lot of work if an engineer has to manually program all the recognisable features. Instead, CNN(Convolutional Neural Network), ANN optimal for image classification, can be implemented for autonomous feature extraction and classification. The computational cost however is very high as compared to that of a classical Machine Vision System. Also, it is understood that use of Deep Learning in a vision system is fruitful where it is difficult to hand-craft the design of algorithms due to limited uncertainties involved in the process.

5 Concept Selection

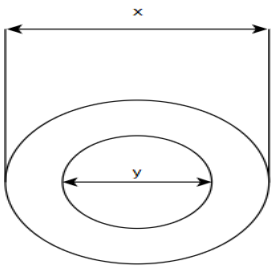
The most promising vision system strategy or combination of strategies will be selected. In order to find out which strategy works best at the facility, it is important to consider the different influential factors in the manufacturing facility at Vello AS. If a strategy is backed by strong reasons, has promising features and satisfies the objectives listed earlier, we can proceed further and start experimenting by utilising relevant data and analysing the results obtained.

5.1 Circumstances at the manufacturing facility

Before choosing a right Vision System, there are several things to be considered [6]. Some of the relevant factors to our case are listed as following.

- Number of Inspection per Second
- Attributes to Inspect
- Size of area to be inspected
- Actions to be taken as a result

(a) Individual creel



(b) Schematic top view of the creel

Figure 10: Fibre creel

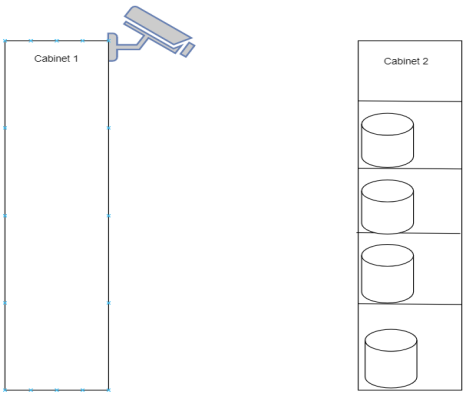
The roll of fibre glass used by Vello is opaque as shown in Fig.10(a). and the single glass fibre thread is translucent. So, object detection is not much of a challenge. The biggest challenge, as mentioned earlier is the fact that fibre is pulled out via metal guides to the resin bath from the center of the creel. As shown in Fig.10(b), y is the inner diameter and x is the outer diameter. Among these, only the y value changes. Hence, the change in dimension can be most effectively realised only with the top view. Unfortunately, the creels are stacked on the shelves which limits the view. So, workable view we can acquire would look something like Fig.11.

Figure 11: Image acquired with Ubiquiti UniFi

5.2 Approach

It was realised that, in spite of the cabinet structure making it difficult for acquiring good view of the individual rolls away from the camera, part of the fibre rolls where the thickness can be realised are still visible as demonstrated in Fig.12(a).

(a) Image from an angled view



(b) Schematic of a possible camera position

Figure 12: Approach

So, a camera can be placed on one cabinet to acquire images/videos of the rolls on the adjacent cabinet as shown in Fig.12(b). Then the data can be processed accordingly to extract the thickness. The question now is that which of the strategies is the most promising.

5.3 Strategy Selection

Flowchart shown in the methodology section was implemented. Among different concepts discussed earlier, OpenCV and Deep Learning vision system were theoretically promising and satisfied the industrial objectives. However, this does not mean that other concepts mentioned earlier are eliminated as both OpenCV and Deep Learning vision system include combination of different concepts such as image processing, data augmentation, calibration etc. Initially, utilising simple OpenCV script to extract dimensions was attempted[A.12]. The aim was to implement the script which is able to detect circular/elliptical feature on one of the images acquired. However, it was immediately realised that the script was not powerful enough to detect the necessary features from the real images as only a part of the rolls in the images were visible. Maybe, the position of the fibre rolls could be improved for the OpenCV to work. The plan was to place the rolls horizontally as the script had good results when the shape was completely visible as demonstrated on Fig.13.

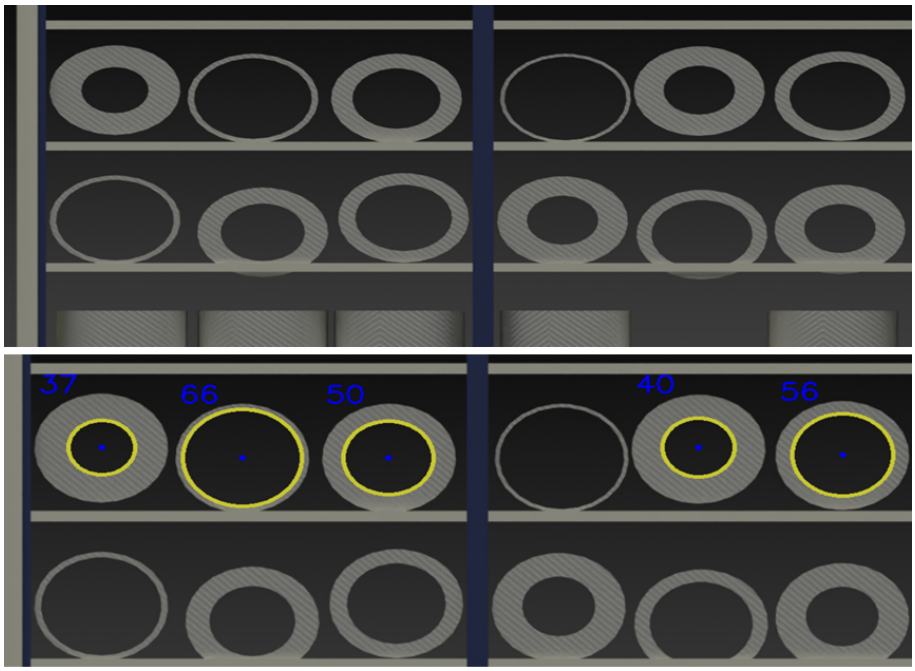


Figure 13: Open CV implemented by altering the rolls positioning in the 3D model

As it was not feasible to actually do this in the facility, a 3D modeling was done on Siemens NX to replicate the cabinets. Although majority of the circles in the image was not detected, it was only the matter of tweaking the parameters. However, the bigger problem is, it is not practical to place the rolls horizontally as when the thickness of the roll is less, it would not hold the structure and collapse, which would negatively impact the production quality. Hence, it was clear that this would not be the right strategy.

Using vision system that incorporates deep learning techniques seemed viable and more promising as they are way more efficient. Machine learning is able to do wonders now and is evolving exponentially. So, it was realised that using deep learning could be a solution given the challenging circumstances in the manufacturing facility. Also, if successful, they seem to satisfy the objectives stated in one of the earlier sections. A good camera positioning with a good camera can capture significant number of rolls in a frame which makes it very economical even though the computational cost can be quite high. Therefore, going forward, a script that incorporates deep learning will be implemented and the results will be analysed.

6 Deep Learning for Vision System

It is important to have a profound understanding of the working principle of deep learning to be able to implement it in the most efficient manner. It is understood that Convolutional Neural Network is a type of neural network which delivers when it comes to classifying images[2]. In health sector, they are being used to diagnose lung cancer from the X-ray reports[1] and in automobile industry, they are being used to develop vision system for autonomous cars[4].

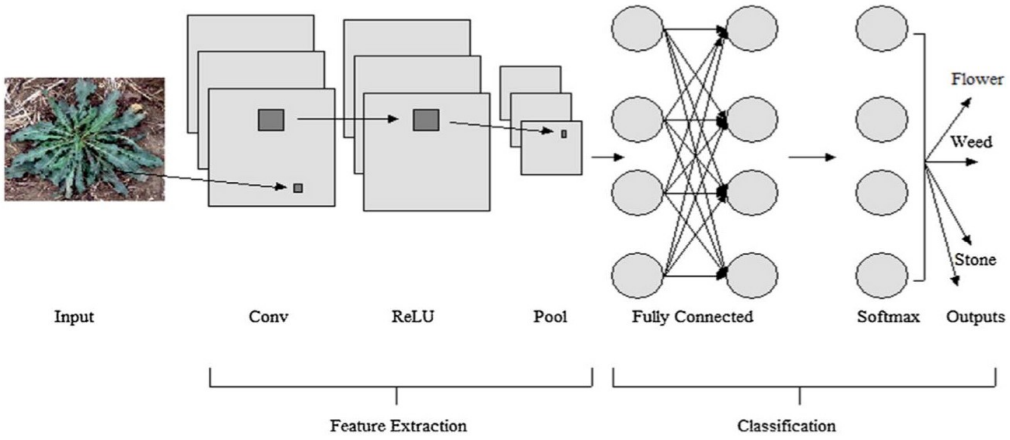


Figure 14: Data and information flow in CNN [2]

Fig.14 displays different components a CNN is comprised of. They are typically composed of the feature extraction part and the classification part.

6.1 Feature Extraction

Feature extraction part has a series of convolutional layers. This is where the network extracts individual measurable property from the input image[23]. Image has its height, width and depth. Depth in case of 2D images is the color channel. As an image goes through different layers, dimension decreases and the depth increases. This is because in the later layers, the depth are feature maps that represents the feature extracted from the previous layers. Convolutional layer makes use of filters called kernel to extract the features as shown in Fig.15. Hence, the depth of the layer depends on the number of kernels used in the layer whereas the decrease in the image dimension depends on kernel size.

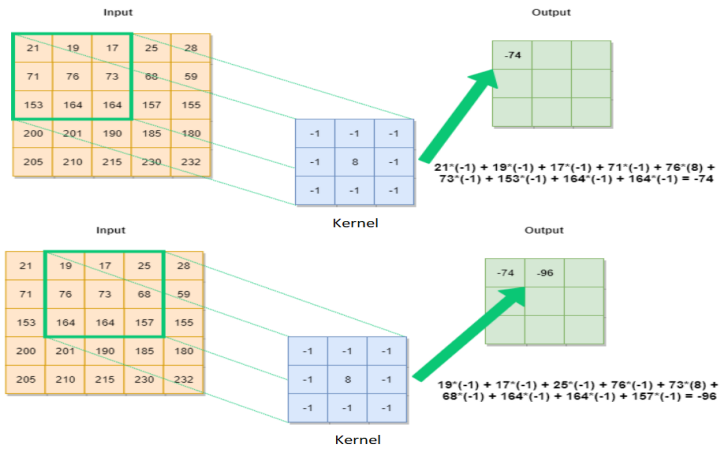


Figure 15: Kernel in action [23]

6.1.1 ReLU as an activation function

Then there is ReLU(Rectified Linear Unit), a type of activation function. The activation functions are used to achieve non-linearity. They decide if a particular neuron is to be activated or not. ReLU is one of the most common activation function because it works well in many different situations. It also trains better. The ReLU activation function activates only when the output is greater than zero. Otherwise, it just outputs zero. They are present in both feature extraction part and classification part. Some of the other activation functions are Sigmoid, Leaky ReLU, Hyperbolic, Linear etc.

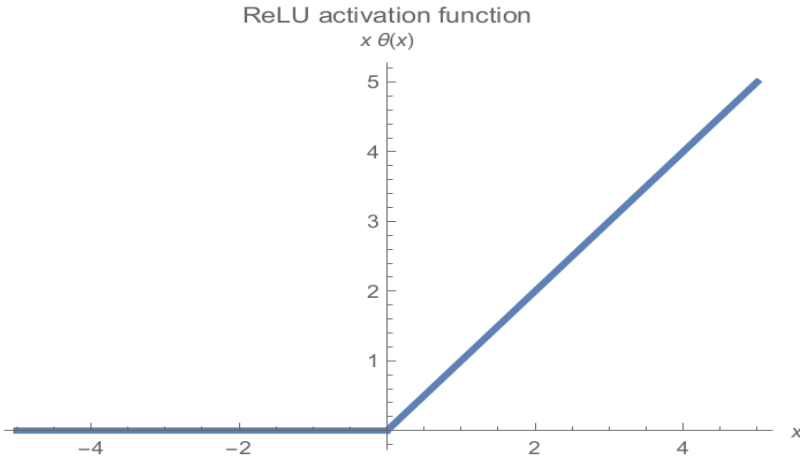


Figure 16: ReLU activation function

6.1.2 Pooling Layers

Pooling layers are used so that the depth of the output layers do not increase because this makes learning for the network more complex. Pooling simply downsamples the parameters before being passed on to the next layer. We can then get feature maps with smaller dimensions and same depth.

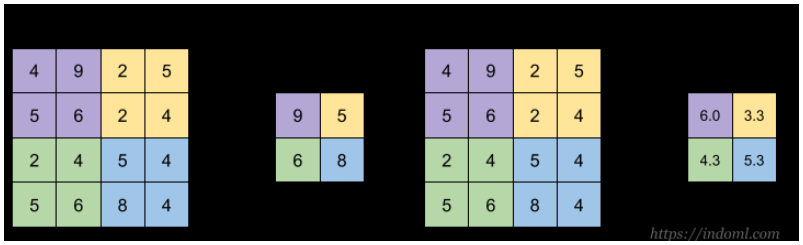


Figure 17: Two types of Pooling

6.2 Classification

Classification part consists of fully connected neural networks and a activation function. The output from the feature extraction part becomes the input for the fully connected network. The matrix form is however flattened so that the output from the CNN which is represented in a matrix form is transformed to array as they are not compatible with the fully connected layers.

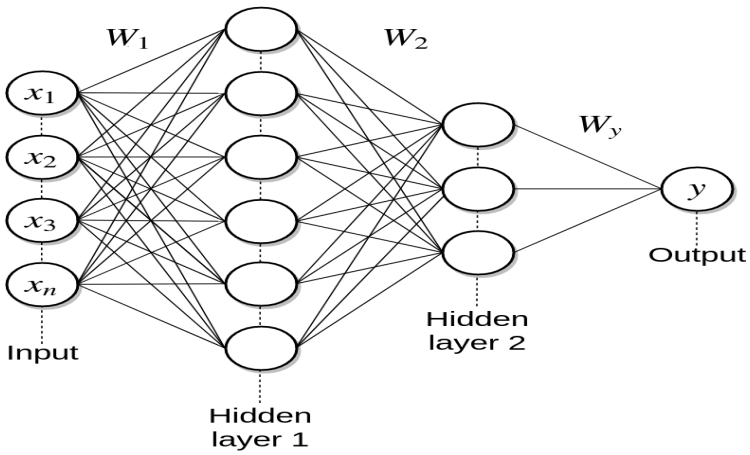


Figure 18: Classification part of the network [13]

6.2.1 Hidden Layers

In a network, hidden layer is where features are learnt. The first layer detects simple pattern like straight line. The second layer learns pattern the straight line and this goes on and on until the network is able to learn complex patterns. In a fully connected neural network, each node is connected to all the nodes in the previous layer. For each node assigned with a weight value which represents how important it is, to determine the output. Initially, the weights are randomly distributed, but as they are trained, they will update the weights in a way that gives the correct output.

6.2.2 The Feedforward Process

The information flows from the input layer through the hidden layers to the output layer. Weighted sum is calculated for each of the connections present in the network and activation function is implemented.

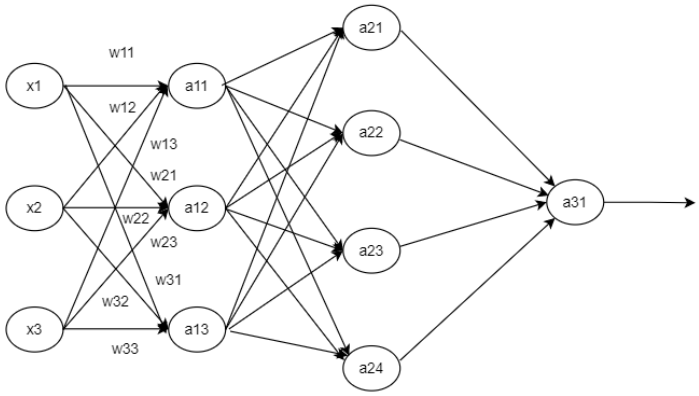


Figure 19: Node connections with weight values

The feedforward calculations for layer 1 of the network in Fig.17 is demonstrated below:

$$a_1^{(1)} = \sigma(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3$$

Same calculations will be done for the nodes in the other layer including the output layer.

6.2.3 Error Function

Error functions or loss functions help the network measure how far away the output is from the right output. It evaluates the network internally and is always greater than zero. Hence, greater the loss, more the network needs to be trained in order to achieve better accuracy. The most commonly used error function for classification problems is cross-entropy. It quantifies how different are the probability distributions. Mathematically,

$$E(W, b) = -\sum_{i=1}^{m} \gamma_i log(P_i)$$

where γ is the desired probability, P is the predicted probability and m is the number of classes.

6.2.4 Optimisation

The network has obtained the error value, but it still needs a way to use that value to update the weights to minimise the error. This is when optimisation comes to play. The most common optimisation algorithms are batch gradient descent, mini-batch gradient descent and stochastic gradient descent.

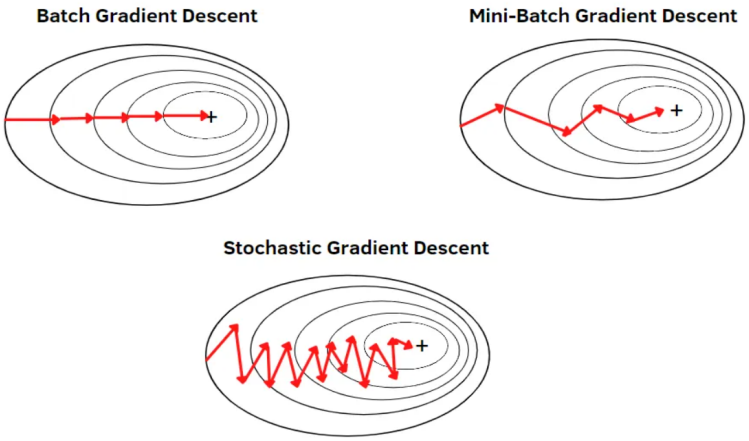


Figure 20: Different optimisation algorithm [14]

The network starts with random weight and the goal is to obtain weight value/s so that the algorithm can reach the shallowest part of the error curve to produce the minimum error. The Mini-Batch Gradient Descent is a hybrid of BGD and SGD. It has good features of both Gradient Descents. Hence, usually Mini-Batch GD can be the best compromise a deep learning engineer can make.

6.2.5 Backpropagation

The weighted sum and activation functions are used to get an output. The error function will give network the difference of the acquired output from the target output. Gradient descent optimisation then computes the Δw which minimises the error value. There must be a system which updates the weight value that incorporates the change responsible for minimising the error. This is where Backpropagation comes to play. Mathematically,

$$W_{new} = W_{old} - \alpha(\delta Error/\delta W_x)$$

where α is the learning rate.

So, through backpropagation the change in the weight value is passed on until the neurons of the very first layer. Now, this new weight, using the feedforward process, generates the output. And, this goes on and on until we have weight value that is able to give us desired output.

6.2.6 Softmax as an activation function

The Softmax function changes the input values to probabilty values so that the sum of the output is 1. This feature makes it the most common activation function to be used in the output layer, specially when we need to predict the class between two or more classes. It is a generalisation of the sigmoid function.

6.2.7 Hyperparameters

Some of the most common hyperparameters are learning rate, loss functions, optimisers, number of layers and activation functions. These can be used to tune the network and influence the output quality. It is important to tune them to a balanced value to get about the desired quality of output though. For example, when tuning number of layers, it is understood that more the number of layers, more learning for the network is possible. However, this does not mean having the maximum number of layers will necessarily train the network most efficiently. There is a good chance that the network will have very good result with the training data set, but cannot perform when subjected to a new data. This phenomenon is also called overfitting. This is the way with most of the hyperparameters.

7 Literature Review

As it was decided to use a vision system that incorporates deep learning, it is necessary to know how it has been or could be used in situation that is similar to the one we have. This way, we can get inspiration and at the same time make sure that we do not end implementing something that has already been done in the exact same manner. Hence, in this section, we will look into implementation of the different vision systems.

Image preprocessing and edge detection are one of the techniques that can be used for measuring dimensions. Image preprocessing will make the image more suitable for edge detection usually by removing the unwanted objects in the images which will intensify the edges' contrast. Same steps were taken to measure dimension of shaft workpiece[20]. To avoid the edge detection from becoming influenced by the interference region, Fully Convolutional Network was used along with texture repair method. The edge was obtained with Holistically-nested Edge Detection and to make that more efficient, canny operator was used. Outer diameter of the shaft workpiece was measured and accuracy of 0.02 mm was achieved.

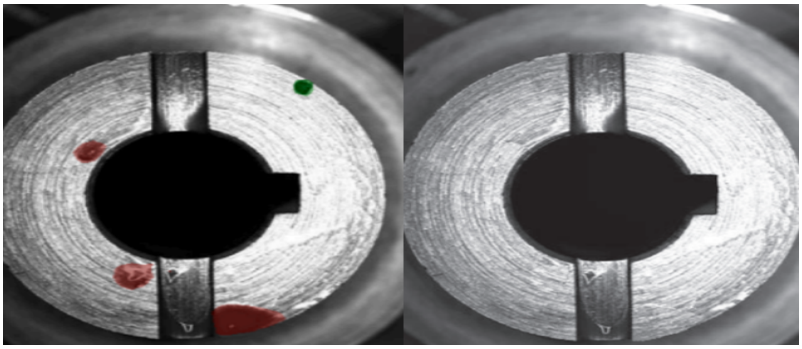


Figure 21: Shaft workpiece [20]

Deep Learning Vision System are also often used just to find out the defective products in a production line. They are usually trained using the images of products containing defects and of products that do not contain defect. Such a vision system was designed to inspect fiber interlock armor cables[32]. The resolution of the images were determined such that the defects size on the order of tens of microns could be detected. Accuracy of 78.9 % was achieved when the network was trained with 10000 samples. It was also realised that this vision system solution was economically feasible.

Figure 22: Fibre interlock armor cables [32]

Similarly, vision system with convolutional neural network was used to detect defect in shapes of crankshaft[18]. Use of CNN made the inspection more efficient compared to the traditional methods. However, the inspection system was designed specifically to replace the manual inspection of the crankshafts.

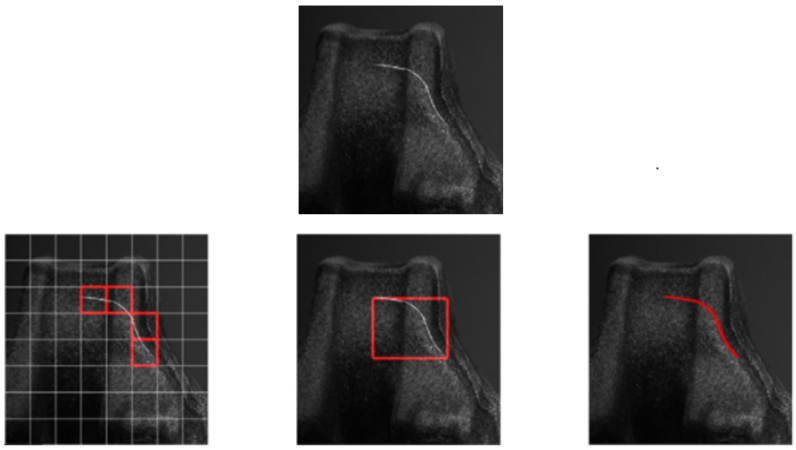


Figure 23: Crankshaft with defect [18]

Deep learning vision systems can also be used in the field of agriculture. For example, they can be used to estimate yield of fruits or vegetables. A study was performed using faster R-CNN to detect, count and estimate size of citrus fruits[9]. Drone camera was used to acquire images which would then be preprocessed before they are in the network. The flight altitude for the UAV was determined to make sure that the ratio of metric dimension to pixel dimension is 0.26 cm/pixel consistently. Deep learning was used for object detection and openCV was used for dimension extraction. Estimates from the trees were compared in terms of weight and turns out that the vision system was able to deliver 7.22 % which was significantly more accurate than the yield estimated by an expert.

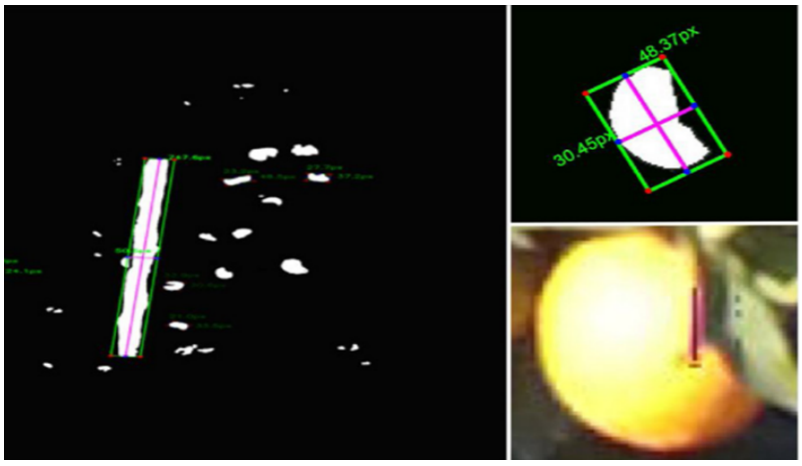


Figure 24: Object detection and size measurement of citrus fruit [9]

A survey was conducted to provide an overview on traditional machine learning, deep learning methods and machine vision techniques which can be implemented in a food processing industry. Food items like soybean, walnut, tomato, banana, shrimp and cod fillets all had grading accuracy more than 95 %[21] when different types of neural network were utilised. It was also concluded how much more room for improvements there is when neural networks are used to perform food processing operations.

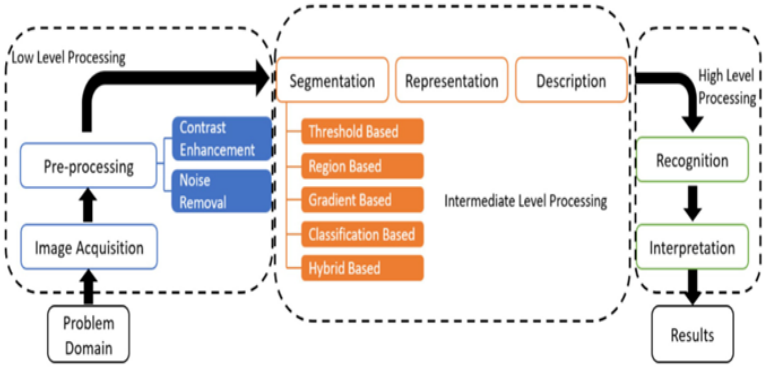


Figure 25: Vision system flow chart [21]

Vision system can also use segmentation technique to measure the target object. An automatic segmentation using deep learning was implemented to measure fish size. The aim of the paper was to show how effective automated vision system would be in minimising number of undersized fish. The image system was placed in the trawl and Masked R-CNN was used to localise individual fish. The system was even able to distinguish overlap and overcome it. Cubic polynomial for the main skeleton axis is estimated using Random Sample Consensus in order to measure the length. 2600 manually labeled fish image was used to train the network along with augmented images. Accuracy for single fish was 99.4 % and accuracy for overlapping fish was 98.4 %[26] which seemed very promising.

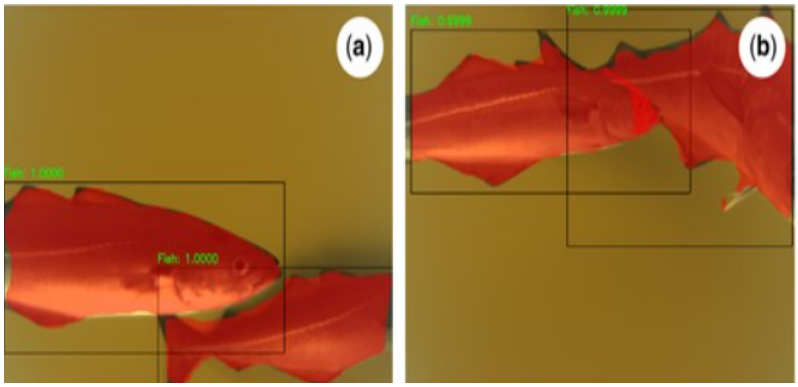


Figure 26: Segmentation of fish using Masked R-CNN [26]

Similar segmentation technique along with object detection was implemented to estimate the area of shuck, shell, and embryo on pecans in different growth stages. Masked R-CNN was used to make object detection and segmentation possible. The network was able to achieve 95.3 % to 100 % on the models for the object detection task and area estimation achieved a mean absolute percentage error of 10.14 % to 28.06 %[22].

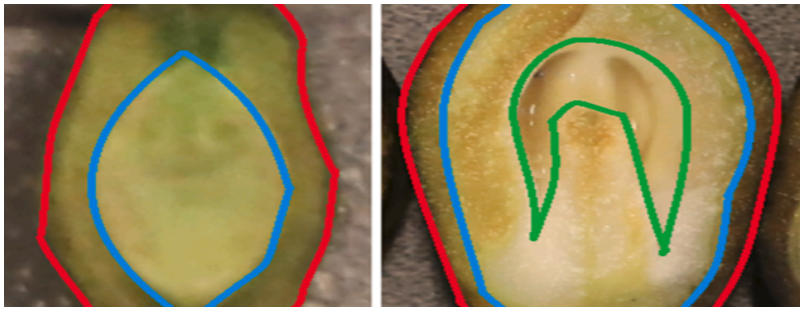


Figure 27: Area estimation of the embryo of pecan nut [22]

7.1 Conclusion from the review

Studying all the deep learning vision system implementations mentioned above, it was realised that the robust vision system powered by deep learning definitely has the potential to deliver desired functionality with good accuracy at the manufacturing facility. The different techniques used for vision system which were subjected to varying conditions was, without a doubt, inspiring.Keeping the objectives and the influential factors at the manufacturing facility in mind, approaching the challenge with the help of object detection seemed very much viable. It was also clear that convolutional neural network is the best option when it comes to working with images. We then know that the bounding box used for the object detection returns its coordinates. Hence, the plan is to test a vision system that utilises convolutional neural network for image processing, object detection and the coordinates to acquire thickness of the glass fibre rolls.

8 Implementation

In this section, we will look into how different parts of the vision system is made to work with the target object so that the desired result is obtainable, desired result being successful detection of fibre rolls and accurate dimension extraction. As the plan is to use coordinates of the bounding boxes, it can be understood that accuracy of the extracted dimensions are going to depend on how accurately the target objects are detected. And, accuracy of the detection will depend on the network architecture chosen to work with and how it is trained.

8.1 Training Data Acquisition

Collecting training data is one of the most important part of building a vision system. Quality of all the following phases directly or indirectly depends on the quality of training data collected, quality not only in terms of resolution. Keeping all that in mind, the pictures were collected using Ubiquiti Unifi G3 dome camera. The camera has maximum digital video resolution of 1920 x 1080.

Figure 28: Some of the pictures acquired at the manufacturing facility

We could use a better camera than Ubiquiti Unifi G3 dome to capture images for training. However, it was already available and it could be used to experiment the vision system. It would not make sense if we spend a significant amount on a better camera and the network is not able to produce quality result as desired. Hence, using Ubiquiti Unifi G3 was the most economically feasible option at the moment.

The plan was to train the neural network manually on images of individual fibre roll images so maybe taking pictures of individual images separately sounds like a good idea. However, if we were to acquire one image for one fibre roll, it would be difficult to eliminate other roll in the image as they stand very close to each other. It is possible to get the pictures of individual rolls with different thickness in a different location but then, it would miss the essence of the cabinet and other factors such as lighting, overlap etc. Since the trained network was to be implemented upon the rolls in the cabinets, it was decided that we take images as displayed in Fig.26.

We can see the images, because of its perspective nature, are distorted vertically. Horizontal distortion however, was eliminated as we had the option to do that with the Ubiquiti software. Although just acquiring those images was not enough as the plan is to train the network using individual roll images. So, that would be done by simply cropping out individual images as shown in Fig.27. With the vertical distortion along with poor lighting going downwards, it was difficult to acquire good quality images for training specially for the rolls in the bottom shelf. Hence, low resolution and low quality images were obtained for the rolls in the bottom shelf. Anyways, as much images with lighting and position variations were collected so that the network once trained, could be able to detect all those rolls subjected to different positions, overlaps and of course lighting. Total of 100 images were acquired for training and testing.

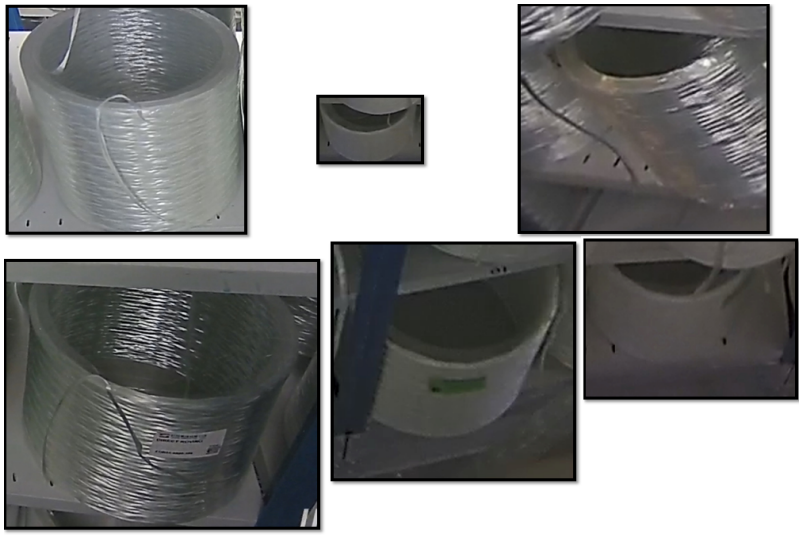


Figure 29: Cropped out images of individual rolls

8.2 Network Architecture

With the training data available, it is time to select a network architecture which is ideal to the situations we have in hand. It was decided that CNN would be preferred as they deliver when it comes to working with images. However, there are so many different architectures that incorporates CNN and is able to perform object detection task. Some of such object detection systems are R-CNN, Single-shot Detector(SSD) and You Only Look Once(YOLO). So, it is necessary to consider all the factors and select one of the most desirable neural network architectures. For R-CNN, training happens in multiple stage and because of that, it takes longer time to train the network. Whereas SSD and YOLO are both single-stage detectors and are faster, but not necessarily more accurate than multiple-stage detectors such as R-CNN. However, comparing mAP of 74 % achieved by a SSD family and mAP of 57.9 % achieved by YOLO on standard datasets, SSD seemed more attractive for now. Mean Average Precision(mAP) compares the ground-truth bounding box with the detected box. Higher the mAP, more reliable the detection is.

8.2.1 SSD

The SSD network released in 2016 by Wei Liu[5] was able to reach new levels for object detection tasks. Since SSD is a single-stage detector, convolutional layer is responsible to predict the objectness score and classification probability. Objectness score is a way of quantifying how accurate is the predicted bounding box. The manually constructed bounding box is compared to the detected bounding box and the level of overlap between them is measured. When the overlap is 100 % then the objectness score of 1 is returned. Along with the objectness score, score for the target object being inside the bounding box is also returned.

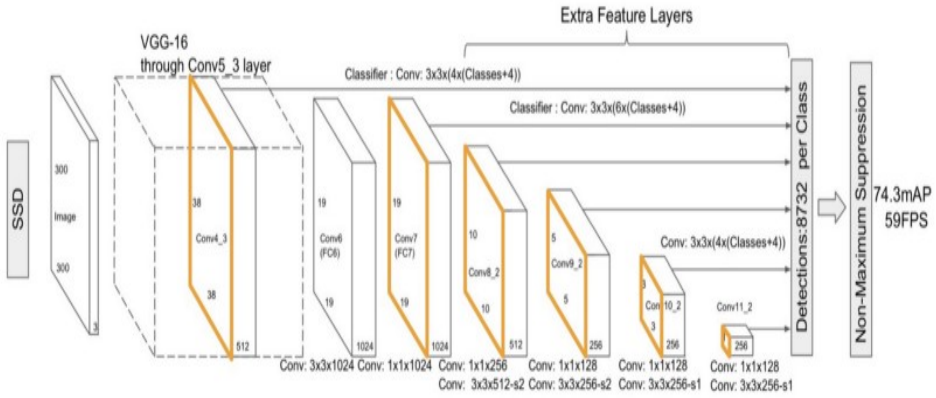


Figure 30: SSD300 Architecture [19]

SSD architecture normally has three different parts and they are **Base network**, **Multi-scale feature layers** and **Non-maximum suppression**. We can see that VGG16 is used as a base network for the SSD300 architecture because it performs well in image classification. VGG16 is a type of SSD which has 16 concolutional layers. Base network segments every bounding box and treats it as individual images and checks if it was able to extract any feature of the target object there. Multi-scale layers is comprised of convolutional layers that decrease in size. These are used to make it possible for the network to detect features that are larger in size or which are closer to the camera when the image was captured. Now, NMS lets only one box with the highest overlap with the ground truth box stay and eliminates all the other boxes. It is important because for example, in the SSD architecture displayed above, for one class there will be 8732 boxes which is taken care of when fed to the NMS layer.

Model name	Speed (ms)	COCO mAP	Outputs
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes

Figure 31: Different SSD architecture available [28]

Some of the SSD model available in the TensorFlow 2 Detection Model Zoo are displayed above. Among them, SSD MobileNet V2 FPNLite 320×320 was chosen to be implemented. On the list, the chosen model is only the second best in terms of speed and accuracy. However, it was the best compromise when it came to computational cost. If the implementation goes in the right direction with this model, then we always have the option to use a faster and more accurate model. In this model, Feature Pyramid Network is used for feature extraction. It basically outputs feature maps of different resolution. So, we can say that FPN is the base network here. MobileNet V2 means that the network is built on depthwise separable kernels where the input channels are kept separately. The model is designed to be trained on image of resolution 320×320 and the model is created using the TensorFlow Object Detection API which is an open-source framework. It is able to output bounding box coordinates in the form [ymin, xmin, ymax, xmax] which can then be used to obtain required dimensions.

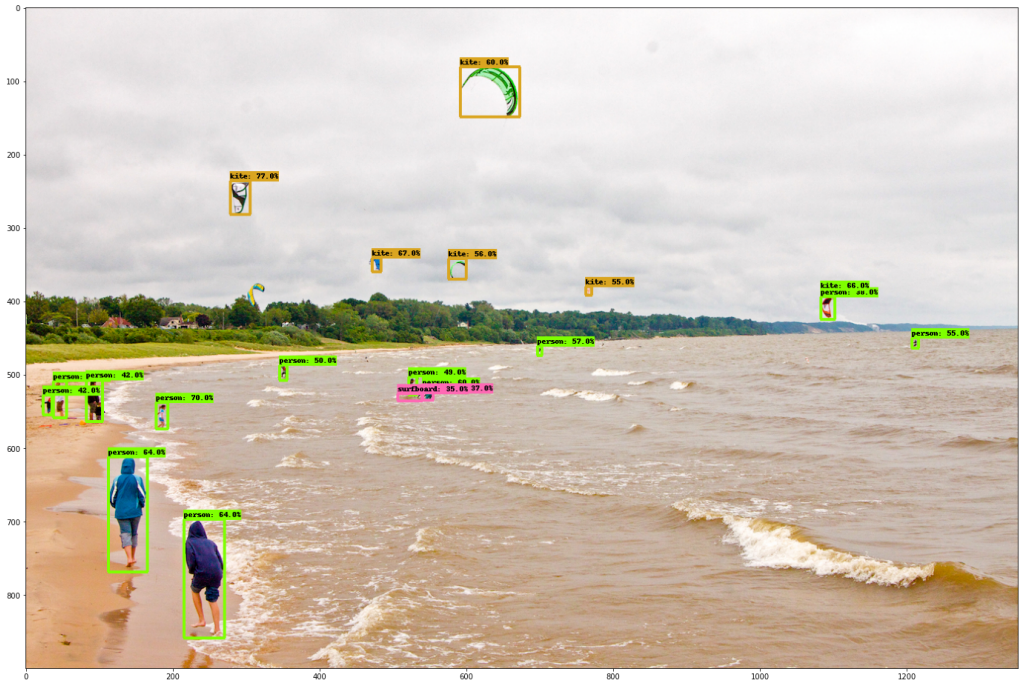


Figure 32: Example of a detection result with SSD MobileNet V2 FPNLite 320 × 320 [30]

8.3 Labeling Image

The network architecture chosen here utilises supervised learning as manually labelled data are used to train the network. Hence, it is necessary to label the images acquired earlier so that the SSD can learn. To label the images, we use LabelImg[11] as an image annotation tool. It is written in python and uses Qt for graphical interface.

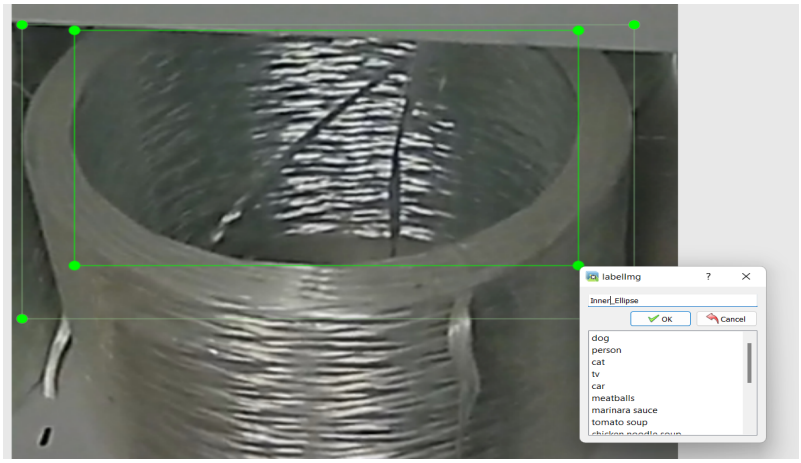


Figure 33: Labeling a single image

Images like Fig.26 were cropped to acquire images like Fig.27. The resolution of the cropped out images varied. So, they were all made to have a uniform size of 480 × 640. Then the LabelImg platform was used and the object of interest was manually labeled. A total of 100 different cropped images was labelled with two object classes, Outer Ellipse and Inner Ellipse. The quality of labeling is something that can significantly influence the precision object detection can deliver. It can be challenging sometimes though, when the edges are not very clear for our eyes specially because of the poor resolution obtained by making the cropped out images have uniform size.

Once we label the target objects in the images successfully, the platform will create an XML file which has all the information. As shown in Fig.32, it has the coordinates of the bounding boxes that was manually constructed around the target objects. Both the images and the XML files are stored in the training folder and will be accessed by the network to learn. Some of the pairs are also stored in a testing folder which will be accessed by the network to evaluate the precision of the detection. The detected bounding box is compared to the manually constructed bounding box, in terms of overlap, in order to obtain the precision the network is able to deliver.


```
<annotation>
  <folder>inner_ellipse</folder>
  <filename>v_81.png</filename>
  <path>C:\Users\Dell\TensorFlow\TFODCourse\Tensorflow\workspace\images\collectedimages\inner_ellipse\v_81.png</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>640</width>
    <height>480</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Inner_Ellipse</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>114</xmin>
      <ymin>44</ymin>
      <xmax>492</xmax>
      <ymax>171</ymax>
    </bndbox>
  </object>
</annotation>
```


Figure 34: XML for a manually labeled individual roll

8.4 Running the script

Once the training and testing data is prepared, we can now begin the learning process for the network. The script starts by having the selected SSD network in place and creating the label map for the object classes we have. Then the script is cloned and TFRecords are created for efficient storage. Since the SSD network we are using is trained already, it already has its architecture defined. However, since not all the parameters match, the script is configured on the basis of components required. Things like number of classes, number of layers, size of filters etc. are configured as per our needs. Finally, we can run the training cell followed by testing cell where the detection is evaluated. When we are done with this, we will have downloaded a good number of python modules.

Parameters	Values
Object classes	2
Minimum depth	16
Weight	$3.9999998989515007\ e^{-5}$
Activation	$RELU_6$
Decay	0.996999979019165
Epsilon	0.0010000000474974513

Table 2: Feature Extractor parameters

The object classes are Outer Ellipse and Inner Ellipse. The feature extractor part of the architecture has 16 convolutional layers. Initial weight value mentioned above is the weight value obtained from training the network on COCO, a large-scale segmentation, object detection and captioning dataset. RELU 6 is the activation function being used here. It is a modified RELU for which the limit for activation can be set. Epsilon value here refers to the probability of choosing to explore and the weight decay is regularisation technique in deep learning. Weight decay penalises a cost function which is likely to shrink the weight values, discouraging the network to activate the weights during backpropagation.

Parameters	Values
Kernel Size	3 ×3
Depth	128
Weight	3.9999998989515007 e^{-5}
Activation	$RELU_6$
Decay	0.996999979019165
Epsilon	0.0010000000474974513

Table 3: Parameters for Box Predictor

Parameters	Values
Minimum Level	3
Maximum Level	7
Aspect Scale	4.0
Aspect Ratios	1.0, 2.0, 0.5
Scales per Octave	2

Table 4: Parameters for Anchor Boxes

Parameters	Values
Kernel Size	3 ×3
Depth	128
Function	Sigmoid
Gamma	2.0
Alpha	0.25
Class Prediction Bias	-4.599999904632568
Batch Size	5

Table 5: Parameters for Post Processing

Kernel size refers to the dimension of a kernel filter being used to extract the features. The box predictor here has 128 layers. Weight, activation function, decay and epsilon however, have the same values as the feature extractor.

Another major feature of an SSD network is that it makes use of Anchor Boxes. They are the key to quality object detection. The network actually makes thousands of prediction, but only shows the ones that is closest to accurate. Thousands of anchor boxes are created for each images. The anchor box with highest value of overlap divided by non-overlap is ,for the network, to be learned from. This value is also called Intersection Over Union. For example, if IOU is more than 50%, the network is instructed to learn from it, but if it is 40%, then the network understands that the object in the box is ambiguous and does not rely on it. The parameters used for the anchor boxes in the network we have implemented are displayed in Table 4.

9 Results

The network was trained with a different strategy each time and three different strategies were implemented.

9.1 3D Model

The 3D model was already available. So, it would be wise to make use of them as we could control a lots of factors that can influence the image quality like lighting, camera angle, frame width etc. Hence, in Siemens NX, a camera was positioned justifiably. The camera view was then rendered and every individual roll was cropped out. A total of 30 images was acquired and labeled only for inner ellipse making a total of 30 images available for training and testing.

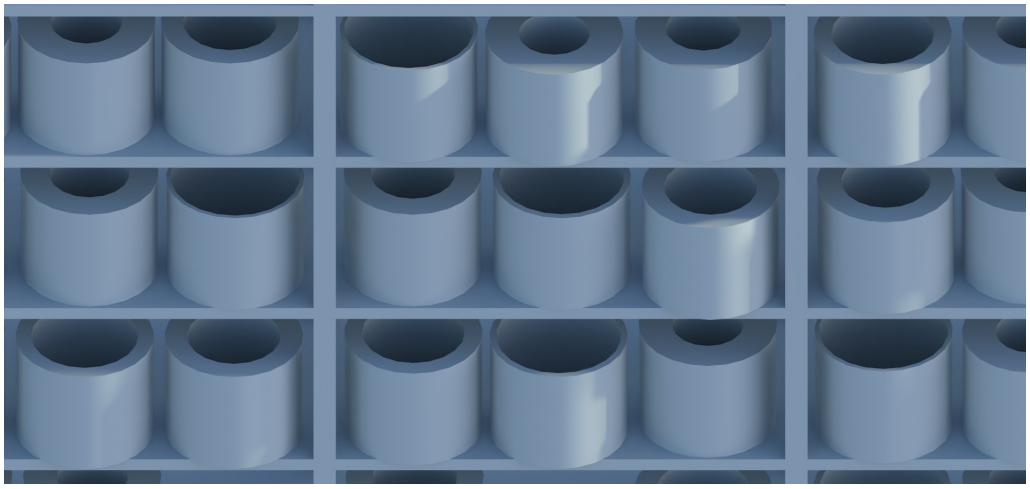


Figure 35: Image from the 3D model

The labeling of images acquired from the 3D model is demonstrated in Fig.34. Once labeled all the images of individual rolls, 80 % of the image and XML pairs was used for training and the remaining was used for testing.

Figure 36: Labeling 3D modeled roll

Finally, the training was initiated and the result was evaluated by using the data stored inside the testing folder. Detection on one of the images used to test the script is shown in Fig.35. The network was trained only to detect the inner ellipse as this was one of the first experiments. The quality of the result obtained was however very encouraging.

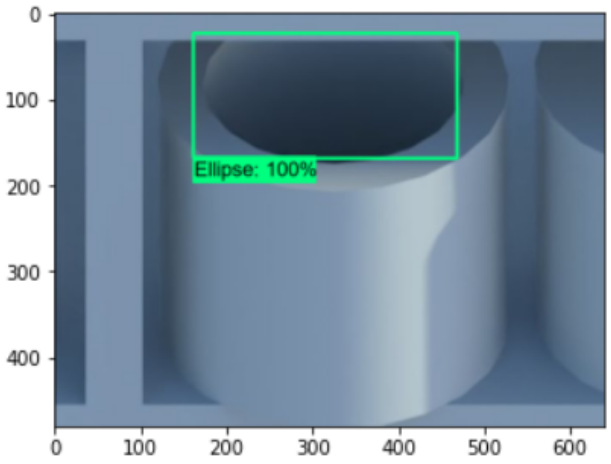


Figure 37: Result obtained on 3D model image

9.2 Individual roll

After successfully performing the test on the 3D model images, it was time to test the network on the real images where not a lot of environmental factors are not or cannot be controlled. Similar strategy was implemented to acquire images of individual rolls. Cropped out images of size 480×640 was then labeled manually as shown in Fig.31. Some of the pictures, specially ones of the rolls in the bottom cabinet, had very poor resolution due to vertical distortion. Anyways, total of 100 different images were labeled separately for two object classes, inner ellipse and outer ellipse. That made a total of 200 images available for training and testing. Like before, 80 % of the image and XML pairs was used for training and the remaining was used for testing.

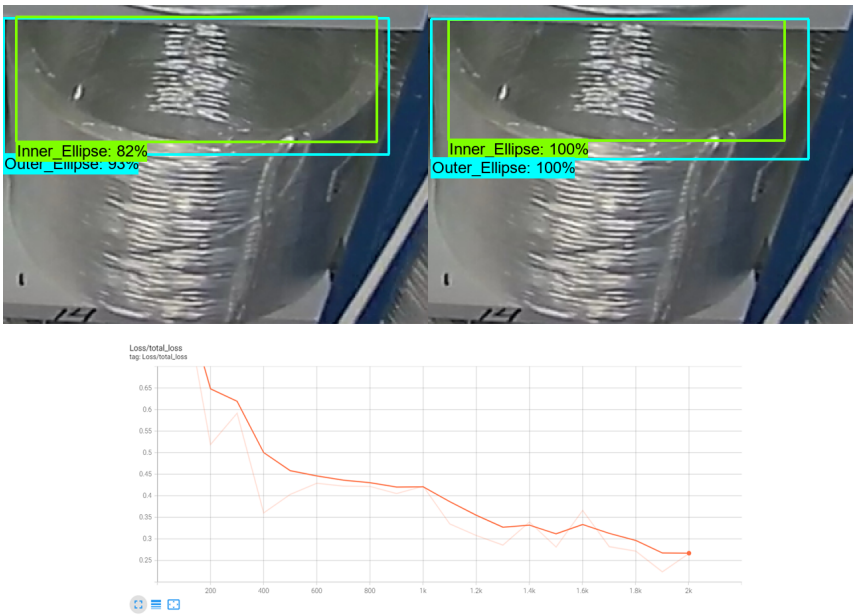


Figure 38: Result obtained on real image (top), total loss vs epochs (bottom)

```
INFO:tensorflow:Step 2000 per-step time 0.188s
I0604 16:29:49.812438  4120 model_lib_v2.py:705] Step 2000 per-step time 0.188s
INFO:tensorflow:{'Loss/classification_loss': 0.10023539,
 'Loss/localization_loss': 0.05735116,
 'Loss/regularization_loss': 0.1443187,
 'Loss/total_loss': 0.30190524,
 'learning_rate': 0.07991781}
I0604 16:29:49.814471  4120 model_lib_v2.py:708] {'Loss/classification_loss': 0.10023539,
 'Loss/localization_loss': 0.05735116,
 'Loss/regularization_loss': 0.1443187,
 'Loss/total_loss': 0.30190524,
 'learning_rate': 0.07991781}
```

Figure 39: Training results for the $2000^{th} iteration$

While the results were mesmerising for some of the images as shown in Fig.38, it was not the same for all. It was quickly realised that the images with poor resolution, as expected, did not perform too well. At the same time, the kind of result obtained in most of the images proved that it can work on real images which are subjected to the actual environmental influences.

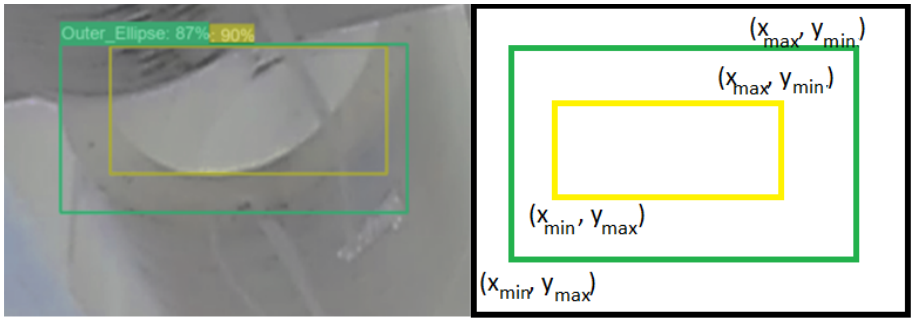


Figure 40: One of the images used for dimension(in pixel) extraction

For the bounding box that is detecting an inner ellipse and an outer ellipse with 87% and 90% accuracy respectively, the dimensions in pixel was extracted by utilising the output coordinates as shown in Fig.40. For fibre roll in the image, the thickness in pixels can be obtained by simply subtracting y_{max} for the bounding box that is detecting the outer ellipse from y_{max} for the bounding box detecting the inner ellipse. It would have been challenging if the output was not in terms of coordinates but in terms of length and width. The reason being that the starting point for the bounding boxes might not be the same, which is the case for the image displayed above. So, as long as the bounding box detects the edges of inner and out ellipse that is closest to the camera, thickness can be precisely measured.

9.3 Bunch of rolls

After the successful detection on individual roll images, we could extract the dimensions, but their was nothing to compare it to in order to calibrate the pixel dimensions. Also, to make it more realistic, it was necessary to implement the network on images with number of rolls in it. So, to do that, when at Vello, we took some pictures of the cabinets and measured a thickness in mm for the rolls using a digital Vernier Caliper.

Figure 41: Image to be tested

Initially, the approach was to train the network on previously acquired individual rolls and test it on the image displayed as Fig.41. Extract the coordinates of the detected bounding box and compare it with the dimensions in millimeter and try to calibrate it. However, the result was not very promising as expected. Surprisingly, the network was not able to detect any of the rolls in the image. So, the approach was improved and similar images with number of rolls were added into the training folder to achieve a better result. All the visible rolls in the images were labeled manually for both inner and outer ellipse as shown in Fig.42.

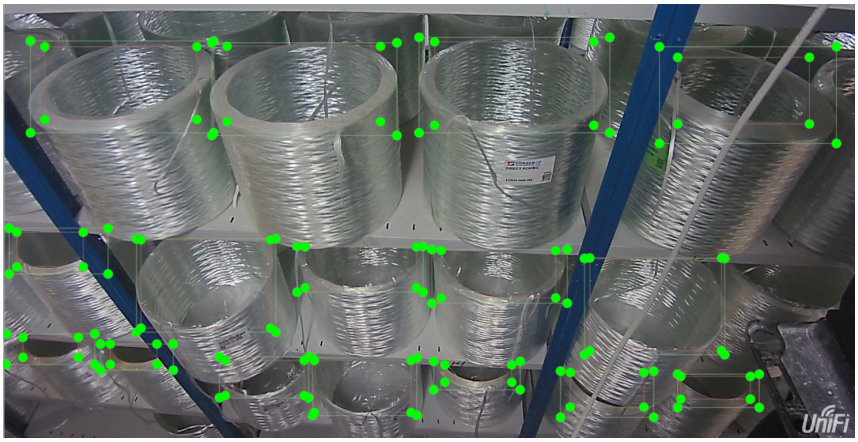
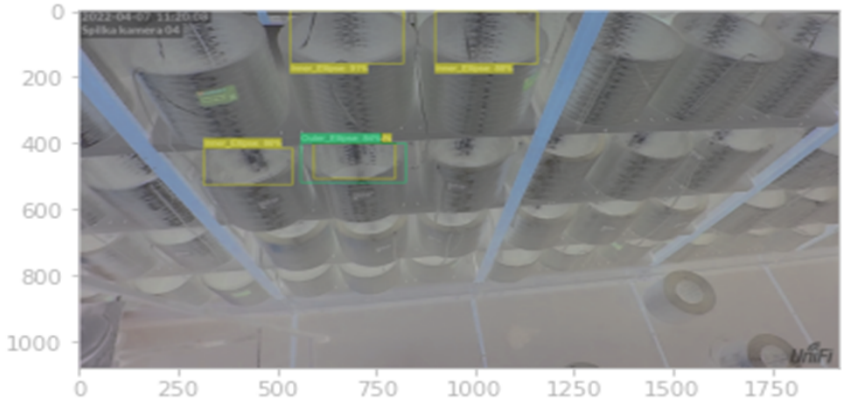


Figure 42: Labeling multiple rolls in an image

Only 10 images with multiple number of rolls were added to the training folder consisting of the labeled individual roll images to achieve the results shown in Fig.43. This means, we could train the on more of such images to enhance object detection for images with multiple number of rolls.




```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.475
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.784
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.616
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.326
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.493
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.548
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.641
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.696
```

Figure 43: Result obtained with more training images (top), Mean Average Precision (bottom)

So, we did that. More images like that were obtained at the manufacturing facility and all the rolls with distinctly visible edges was manually labeled. 20 such images were added and labeled. Then the image and xml pairs were moved into the training folder. Now, the training folder had 200 pairs and the testing folder had 40 pairs including the image with the rolls that was measured with a digital Vernier Caliper. The training was run and evaluation was done with the images in the test folder to obtain the result shown in Fig.52.

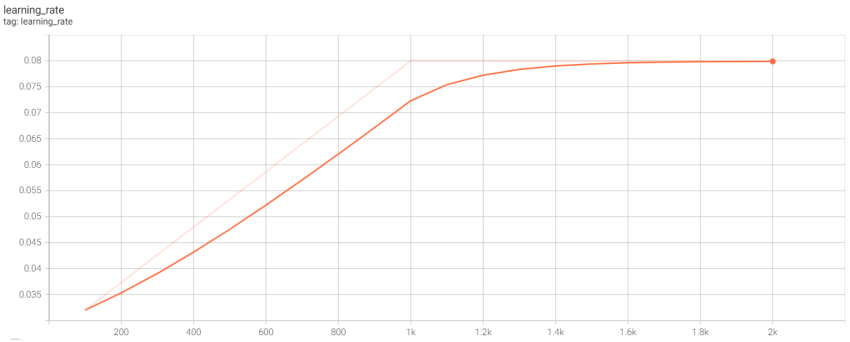


Figure 44: Learning rate for initial training

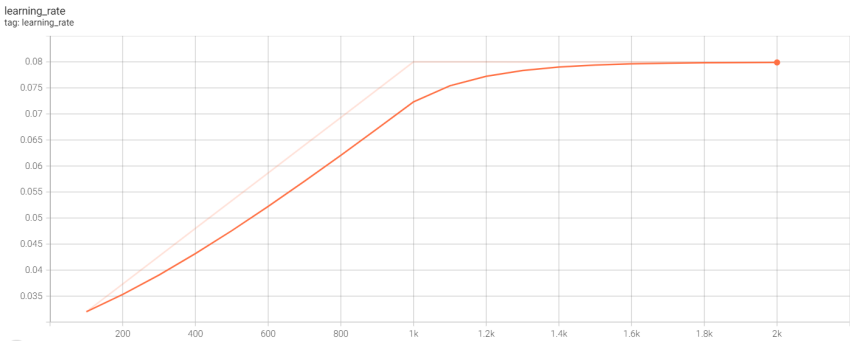


Figure 45: Learning rate after adding 20 images

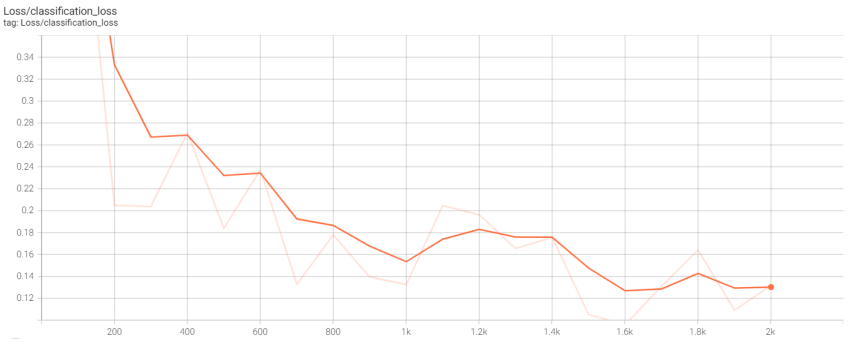


Figure 46: Classification loss for initial training

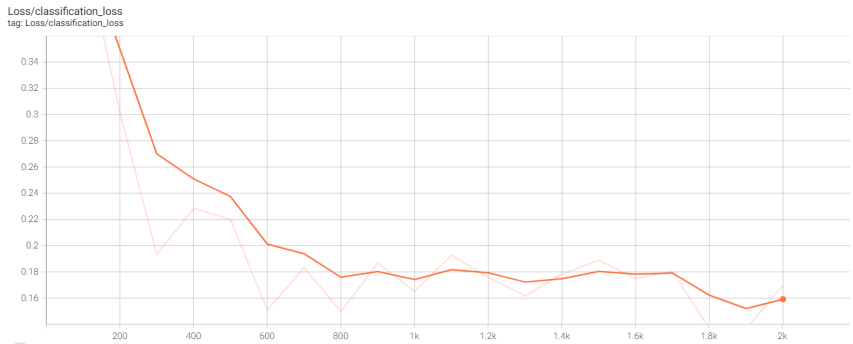


Figure 47: Classification loss after adding 20 images

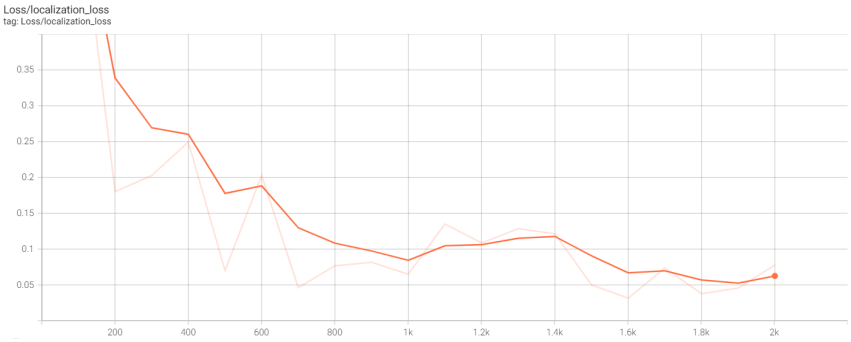


Figure 48: localisation loss for initial training

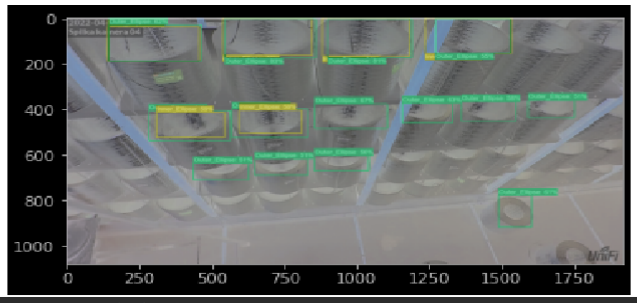
Figure 49: localisation loss after adding 20 images

```
INFO:tensorflow:Step 2000 per-step time 0.195s
I0607 12:17:09.410631 11180 model_lib_v2.py:705] Step 2000 per-step time 0.195s
INFO:tensorflow:{'Loss/classification_loss': 0.13156685,
 'Loss/localization_loss': 0.07769628,
 'Loss/regularization_loss': 0.14533125,
 'Loss/total_loss': 0.35459438,
 'learning_rate': 0.07991781}
I0607 12:17:09.410631 11180 model_lib_v2.py:708] {'Loss/classification_loss': 0.13156685,
 'Loss/localization_loss': 0.07769628,
 'Loss/regularization_loss': 0.14533125,
 'Loss/total_loss': 0.35459438,
 'learning_rate': 0.07991781}
```

Figure 50: losses for step 2000 for initial training

```
INFO:tensorflow:Step 2000 per-step time 0.195s
I0607 11:34:15.781232  1436 model_lib_v2.py:705] Step 2000 per-step time 0.195s
INFO:tensorflow:{'Loss/classification_loss': 0.1697647,
 'Loss/localization_loss': 0.13019311,
 'Loss/regularization_loss': 0.14424953,
 'Loss/total_loss': 0.44420734,
 'learning_rate': 0.07991781}
I0607 11:34:15.781232  1436 model_lib_v2.py:708] {'Loss/classification_loss': 0.1697647,
 'Loss/localization_loss': 0.13019311,
 'Loss/regularization_loss': 0.14424953,
 'Loss/total_loss': 0.44420734,
 'learning_rate': 0.07991781}
```

Figure 51: losses for step 2000 after adding 20 images




```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.383
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.655
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.381
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.120
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.503
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.246
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.374
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.557
```

Figure 52: Result obtained with hybrid images (top), Mean Average Precision (bottom)

Without tweaking any of the hyperparameters, we are still able to enhance the network's performance just by adding the training images. Although, the network is clearly not able to detect all the fibre rolls it is trained to detect and the precision values has decreased compared to the mAP displayed in Fig.43, the progress is still significant. Now that we have the network detected both inner and outer ellipse for 6 different rolls in two different panels, we can try calibrating by utilising the thickness measured in mm with a digital Vernier Caliper.

		y min	x min	y max	x max	Overlap Score
Inner	1,1	0.035734	0.074139	0.176207	0.239106	0.75
	1,2	0.384723	0.161655	0.484752	0.284769	0.58
	1,3	0	0.459811	0.153749	0.616791	0.85
	1,4	0	0.644308	0.144125	0.800398	0.62
	2,1	0	0.280557	0.151201	0.441639	0.86
	2,2	0.375474	0.310487	0.468203	0.42307	0.58
Outer	1,1	0.027908	0.076161	0.175651	0.242757	0.61
	1,2	0.377248	0.148142	0.498944	0.295241	0.64
	1,3	0.006329	0.469108	0.161819	0.621029	0.8
	1,4	0.005265	0.663012	0.142003	0.798291	0.55
	2,1	0.005386	0.284189	0.162569	0.448292	0.83
	2,2	0.369823	0.296937	0.485757	0.4342	0.76

	y max(outer - inner)	thickness (mm)	ratio
1,1	-0.00055645	8.62	-15491.1
1,2	0.01419186	16.38	1154.183
1,3	0.00806971	9.93	1230.527
1,4	-0.00212195	10.94	-5155.64
2,1	0.01136781	20.96	1843.803
2,2	0.0175534	14.06	800.9844

Figure 53: Bounding box coordinates for detected ellipse(top), calibration(bottom)

We know there is a vertical distortion. So, we cannot expect to have the same ratios for rolls in different panels. However, it would look good if we could obtain the same ratio for the rolls which are in the same panel at least. Negative value for the thickness in pixel dimension also can be discouraging here, but there are several things to consider here. Moreover, this is supposed to be more of a demonstration on how we can use the coordinates to calculate the thickness in pixel and relate it to the thickness in mm. The threshold for the overlap score was set to be 0.5. So, we can see the variations in accuracy of the detection. The fibre roll (1,3) and the fibre roll (2,1) are detected with satisfying accuracy for both inner and outer ellipses.

10 Discusssion and Conclusion

After meticulous research on different vision systems that can be used to automate the inspection of fibre roll status at Vello's manufacturing facility and keeping the objectives in mind, making use of deep learning was fairly chosen to be the most viable solution. Among different neural network architectures, Single Shot Detector was selected. To minimise the computational cost, one of the lightest SSD network, SSD MobileNet V2 FPNLite 320×320, was used. This particular network architecture was perfect to be implemented here because it could output coordinates of the bounding boxes which would be used later to obtain thickness of the fiber rolls in pixels.

In the beginning, 3D model was used to acquire images for training and testing the network. This was done to make sure that using the chosen network architecture can deliver. With encouraging results on individual roll images obtained from the 3D model, the network was now ready to be tested on real images. So, some of the images with multiple rolls in it was cropped and images with individual rolls was labelled to train the network. The quality of detection was poor compared to the detection on images of 3D model. Environmental factors were to blame as none of the parameters in the network architecture were improvised. Things like lighting, texture of the fibre rolls, camera positioning was completely under control with the 3D model images whereas this was not the case for the real images. The fibre rolls also have a plastic cover which can sometimes deceive even human eyes as an elliptical edge of the fibre roll. Hence, it did influence labeling, training and detection.

Also, even though horizontal distortion was eliminated, real images still had vertical distortion. And since we were cropping individual rolls out of the image, resolution for the images, going towards bottom of the cabinet, degraded. The network standardised the resolution for every image to be 320×320 which exacerbated the image quality. The amazing and really encouraging thing is, the network was still able to deliver qualitative object detection for most of the images in the test folder.

The bounding box coordinates output would only give us the thickness in pixels which is no good unless calibrated. So, some of the rolls' thickness was measured with a Vernier Caliper. The plan was to use the image with multiple number of these measured rolls for object detection and compare the pixel dimension with the dimension in millimeters. Hence, instead of using individual roll images, it was important to make use of images with multiple rolls. Moreover, this was going to be the case when the vision system would be actually implemented at the manufacturing facility. It was understood that the training also must be done with similar images labeled for multiple rolls in it. The result obtained was not very satisfying, but it was for sure encouraging, given that it was still delivering amidst all the limitations mentioned earlier. A simple way to relate the dimensions in pixel to the dimensions in millimeters was demonstrated. Detection accuracy for inner and outer ellipse varied for different detected fibre rolls. The rolls with highest accuracy for both inner and outer ellipse could be trusted.

The measurement precision solely depended on the quality of object detection achieved. The object detection quality is influenced by many factors though. So, it is clear that if we are able to eliminate the factors negatively impacting the detection quality, we can achieve precise measurements from the vision system. One of the factors was resolution of the images used for training and testing, specially the for the cropped out images. One of the improvisations to minimise the resolution degradation is to have higher resolution cameras acquire the images. The vertical distortion was also one of the reasons for getting low resolution images of the rolls on the bottom panels. So, a camera capable of capturing parallel image would be able to eliminate the vertical distortion. Also, if we look into an image, each of the fibre roll in an image seems to have a different positioning and this is due to the perspective nature of the camera we use. Moving camera that moves along the rail and has as fewer rolls as possible in a frame at a time could do the trick to avoid such variations and at the same time minimise the distortions. The change in thickness of the fibre roll is very slow, so the motion does not have to be very fast, which is an advantage here.

The network we are using is computationally cheap, but it resizes all the images to 320×320. However, if we make use of heavier SSD architecture like SSD ResNet50 V1 FPN 1024×1024, it could drastically improve the quality of object detection. For this to impact as desired, the training images must have high resolution. Not using the cropped out individual roll images could also help us achieve the quality of detection we desire. The only downside to using such heavy network is it can be computationally expensive. The other important environmental influence which can be improvised for better results is lighting. The rolls in different cabinets or even the rolls in the same cabinet but different panels, appeared to have different value because of inconsistent lighting. The fibre rolls seemed to have different materials even though they are actually the same. And this is one of the reasons for the network not being able to detect all the fibre rolls in an image. Implementing better lighting system would definitely improve our results significantly. The moving camera with a frame that can capture 4 or 6 rolls at a time can have a adequately powerful light source installed on it which can uniformly illuminates all the fibres present in the frame.

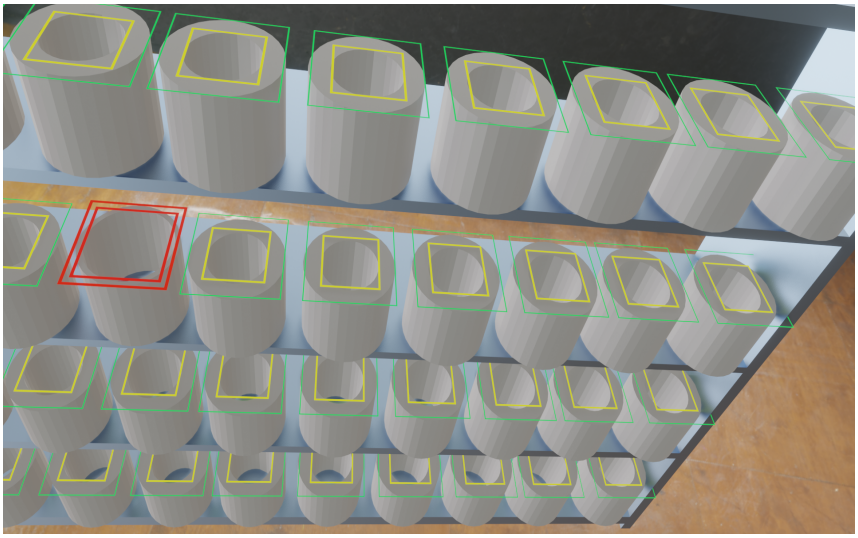


Figure 54: Basic concept of the interface

All these improvements could be made and the deep learning vision system could be implemented once again. More precise object detection can be achieved and calibration as demonstrated can be performed. Images for constant time interval can be fed to the network and detected changes can be studied to understand the pattern. The network will return the dimensions in millimeters once calibrated and the threshold for the minimum acceptable thickness can be set so that the interface can notify the operator with the fibre roll's thickness and location helping the operators to replace the rolls in time so that inconsistent profiles can be avoided. The time operators invested to inspect the fibre rolls manually can be invested on something more productive. These are some of the ways a vision system like this can positively impact the overall manufacturing. Therefore, deep learning vision system has a potential to be a robust vision system at Vello's manufacturing facility.

References

[1] *Application of the computer vision system for evaluation of pathomorphological images.* IEEE.

[2] *The quiet revolution in machine vision - a state-of-the-art survey paper, including historical review, perspectives, and future directions.* Melvyn L.Smith, Lyndon N.SmithMark, F.Hansen.

[3] *RGB to HSV.* https://scikit-image.org/docs/dev/.

[4] *Road Peculiarities Detection using Deep Learning for Vehicle Vision System.* IOP Conference Series: Materials Science and Engineering.

[5] *SSD: Single Shot MultiBox Detector.* Computer Vision – ECCV 2016.

[6] *Vision Systems in Manufacturing.* https://www.ptchronos.com/en-eur/blog/vision-systems-in-manufacturing.

[7] *What is a stereo vision camera?* https://www.e-consystems.com/blog/camera/camera-board/what-is-a-stereo-vision-camera/.

[8] Tânia M. Lima-Fernando Charrua-Santos Gerardo J. Osório Walid Barhoumi Amal Chouchene, Adriana Carvalho. *Artificial Intelligence for Product Quality Inspection toward Smart Industries: Quality Control of Vehicle Non-Conformities.* IEEE, 2020.

[9] Jorge Egea-Gregorio Raja-P. Pérez-Ruiz Manue Apolo O. Enrique, Martínez Guanter. *Deep learning techniques for estimation of the yield and size of citrus fruits using a UAV.* European Journal of Agronomy.

[10] Antonio Pietrosantob Consolatina Liguoria, Alfredo Paolillob. *An on-line stereo-vision system for dimensional measurements of rubber extrusions.* Measurement.

[11] darrenl. *LabelImg.* https://github.com/tzutalin/labelImg.

[12] Antonio Pietrosanto-Paolo Sommella Giuseppe Di Leo, Consolatina Liguori. *A vision system for the online quality monitoring of industrial manufacturing.* Optics and Lasers in Engineering.

[13] Daniele Grattarola. *Deep Feature Extraction for Sample-Efficient Reinforcement Learning.* POLITECNICO DI MILANO, 2017.

[14] Geoffrey Hinton. *Gradient Descent.* MACHINE LEARNING EXPLAINED, 2021.

[15] Yasser El-Sonbaty Ibrahim Abdelkader and Mohamed El-Habrouk. *Open MV: A Python Powered, Extensible Machine Vision Camera.* Dept. of Computer Science, Arab Academy for Science Technology, 2017.

[16] Jinghua Wei Jinrong Huang, Bijian Jian* and Sheng Liu. *Monocular vision system for distance measurement based on feature points.* Academic Journal of Engineering and Technology Science,, 2020.

[17] Heinrich Niemann Jochen Schmidt and Sebastian Vogt. *Dense Disparity Maps in Real-Time with an Application to Augmented Reality.* Proceedings Sixth IEEE Workshop on Applications of Computer Vision, 2002.

[18] Christophe Cudel Jean-Philippe Urban Karim Tout, Mohamed Bouabdellah. *Automated vision system for crankshaft inspection using deep learning approaches.* Fourteenth International Conference on Quality Control by Artificial Vision, 16.

[19] Socret Lee. *Implementing Single Shot Detector (SSD) in Keras: Part I — Network Structure.*

[20] Xuebing Li, Yan Yang, Yingxin Ye, Songhua Ma, and Tianliang Hu. *An online visual measurement method for workpiece dimension based on deep learning.* Measurement, 08.

[21] Erica Pensinia Konstantinos N.Plataniotis Lili Zhua, Petros Spachosa. *Deep learning and machine vision for food processing: A survey.* Current Research in Food Science, 15.

[22] Charles Rohlab Niels Maness Becky Cheary Lu Zhang Lucas Costaa, Yiannis Ampatzidisa. *Measuring pecan nut growth utilizing machine vision and deep learning for the better understanding of the fruit growth curve.* Computers and Electronics in Agriculture.

[23] Szymon Manduk. *Convolutional neural network 2: architecture.* AI Geek Programmer.

[24] Gabriele Canella Maria Cristina Valigi, Silvia Logozzo. *A Robotic 3D Vision System for Automatic Cranial Prostheses Inspection.* Advances in Service and Industrial Robotics, 2017.

[25] Anderson Carvalho Suman Harapanahalli Gustavo Velasco Hernandez Lenka Krpalkova Daniel Riordan Joseph Walsh Niall O'Mahony, Sean Campbell. *Deep Learning vs. Traditional Computer Vision.* Advances in Computer Vision, 2019.

[26] Josep Quintana Alexander Tempelaar Nuno Gracias Shale Rosen Håvard Vågstøl Kristoffer Løvall Rafael Garcia, Ricard Prados. *Automatic segmentation of fish using deep learning with application to fish size measurement.* ICES Journal of Marine Science.

[27] Nicholas Renotte. *TFODCourse.* https://github.com/nicknochnack/TFODCourse.

[28] Github Repository. *TensorFlow 2 Detection Model Zoo.*

[29] Adrian Rosebrock. *Measuring size of objects in an image with OpenCV.* https://www.pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/.

[30] TensorFlow. $ssd_mobilenet_v2/fpnlite_320x320$.

[31] Nunez-Iglesias J Boulogne F Warner JD Yager N Gouillart E Yu T the scikit-image contributors. van der Walt S, Schönberger JL. *scikit-image: image processing in Python.* The PeerJ Bioinformatics Software Tools Collection.

[32] Samuel J. Vidourek Zhaowei Chen, Michael R. Holtz and Hossein Alisafaee. *Deep-learning-based machine vision system for defect detection of fiber interlock cable.* Applications of Machine Learning 2021, 10.

[A.1] Applying different filters to the image

```
1
2  import numpy as np
3  from skimage import io
4  import matplotlib.pyplot as plt
5  from skimage.transform import rescale,resize, downscale_local_mean
6
7  camera = io.imread("st3.jpg",as_gray=True)
8
9  from skimage.filters import roberts,sobel, scharr,prewitt
10
11 edge_roberts = roberts(camera)
12 edge_sobel = sobel(camera)
13 edge_scharr = scharr(camera)
14 edge_prewitt = prewitt(camera)
15
16 fig, axes = plt.subplots(nrows=2, ncols=2, sharex=True, sharey =True,figsize=(8,8))
17 ax = axes.ravel()
18
19 ax[0].imshow(camera,cmap=plt.cm.gray)
20 ax[0].set_title('original␣image')
21
22 ax[1].imshow(edge_roberts,cmap=plt.cm.gray)
23 ax[1].set_title('Roberts')
24
25 ax[2].imshow(edge_sobel,cmap=plt.cm.gray)
26 ax[2].set_title('Sobel')
27
28 ax[3].imshow(edge_prewitt,cmap=plt.cm.gray)
29 ax[3].set_title('Prewitt')
30
31 for a in ax:
32     a.axis('off')
33
34 plt.tight_layout()
35 plt.show()
```

[A.2] Labeling images to create xml pairs [27]

```
!pip install --upgrade pyqt5 lxml
LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')
if not os.path.exists(LABELIMG_PATH):
    !mkdir {LABELIMG_PATH}
    !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}
if os.name =='nt':
    !cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc
!cd {LABELIMG_PATH} && python labelImg.py
```

[A.3] Setup Paths [27]

```
import os
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'

paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }

 files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}

for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

[A.4] Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD [27]

```
import wget
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_de
    tection\\packages\\tf2\\setup.py setup.py && python setup.py build && python setup.py install
    !cd Tensorflow/models/research/slim && pip install -e .
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model
```

```
_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
import object_detection
wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
```

[A.5] Create Label Map [27]

```
labels = [{'name':'Inner_Ellipse', 'id':1},{'name':'Outer_Ellipse', 'id':2}]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item␣{␣\n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

[A.5] Create TF records [27]

```
ARCHIVE_FILES = os.path.join(paths['IMAGE_PATH'], 'archive.tar.gz')
if os.path.exists(ARCHIVE_FILES):
  !tar -zxvf {ARCHIVE_FILES}

if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -
o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']} -o
{os.path.join(paths['ANNOTATION_PATH'], 'test.record')}
```

[A.6] Copy Model Config to Training Folder [27]

```
!copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_
MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
```

[A.7] Update Config For Transfer Learning [27]

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])

pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:        proto_str = f.read()
text_format.Merge(proto_str, pipeline_config)

pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_
MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PA
TH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_
PATH'], 'test.record')]

config_text = text_format.MessageToString(pipeline_config) with tf.io.gfile.GFile(files['PIPELINE_CONFIG'],
"wb") as f:                                                    f.write(config_text)
```

[A.8] Train the model [27]

```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'model_main_tf2.py')

command = "python␣{}␣--model_dir={}␣--pipeline_config_path={}␣--num_train_steps=2000".format(TRAINING_SCRIPT
, paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])

!{command}

print(command)
```

[A.9] Evaluate the model [27]

```

command = "python␣{}␣--model_dir={}␣--pipeline_config_path={}␣--checkpoint_dir={}".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'], paths['CHECKPOINT_PATH'])
print(command)
!{command}
```

[A.10] Load Train model from checkpoint [27]

```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util

configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-3')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

[A.11] Detect from an image [27]

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline

category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'ellipse_i_27.png')

img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
```

```
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
          image_np_with_detections,
          detections['detection_boxes'],
          detections['detection_classes']+label_id_offset,
          detections['detection_scores'],
          category_index,
          use_normalized_coordinates=True,
          max_boxes_to_draw=5,
          min_score_thresh=.8,
          agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

[A.12] Measurement of Circular Object with OpenCV

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

image = 'test_5.png'
img = cv.imread(image,1)
original = img.copy()
img = cv.cvtColor(img,cv.COLOR_BGR2GRAY)

img = cv.GaussianBlur(img,(21,21),cv.BORDER_DEFAULT)

circles = cv.HoughCircles(img,cv.HOUGH_GRADIENT,0.9,100,param1=80,param2=30,minRadius=30,maxRadius=70)
detected_circles = np.uint16(np.around(circles))
print('coordinates␣=␣',detected_circles)
print('shapes␣=␣',detected_circles.shape)

radius = np.array(detected_circles)
print('radius␣=',radius)


for i in detected_circles[0,:]:
   cv.circle(original,(i[0],i[1]),i[2],(50,200,200),3)
   cv.circle(original,(i[0],i[1]),2,(255,0,0),3)
   cv.putText(original,str(i[2]), (i[0]-70,i[1]-75),cv.FONT_HERSHEY_SIMPLEX,1.1,(255,0,0),2)

# print('radius are',radius)


cv.imshow('circles',original)
cv.waitKey()
```