
# Building Management: Machine Vision

Submitted by

Xinchen YANG

Thesis Advisor

Prof. Dr.-Ing. Ralph HÄNSEL

INFORMATION TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
TECHNICAL UNIVERSITY OF APPLIED SCIENCES LÜBECK

A thesis submitted in fulfillment of the requirement to the degree
Bachelor of Science (B.Sc.)

2023

# Declaration of Authorship

I, Xinchen YANG, declare that this thesis titled, “Building Management: Machine Vision” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly conducted while in candidature for a degree at Technical University of Applied Sciences Lübeck.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

TECHNICAL UNIVERSITY OF APPLIED SCIENCES LÜBECK

# *Abstract*

Department of Electrical Engineering and Computer Science

Bachelor of Science (B.Sc.)

**Building Management: Machine Vision**

by Xinchen YANG

International students face the challenge to write a thesis at the end of their studies. As many rules and examples of theses at the Technische Hochschule Lübeck are available in German exclusively they are not suited for the international students. Although many good guidelines are available online they do not cover the specialties of the Technische Hochschule Lübeck. Therefore, a special guideline is necessary. Besides very specific rules for this university some common mistakes are covered in this guideline.

# *Acknowledgements*

Put your acknowledgements here! This is an example:

**Acknowledgments by Horst Hellbrück** I want to thank all staff members and students that have participated in this document by giving tips or by asking questions. When I have started this guide in 2009 with a vague idea in mind, I could not foresee that this document becomes such a great help for all the students. My special thanks go to Zhi Haolin and Ren Zhong from the international study program Information Technology at Technical University of Applied Sciences Lübeck together with ECUST in Shanghai. They have written most of the texts and provided references and figures. Without their valuable contribution this guide would have stayed at a conceptual level.

**Acknowledgements by Ren Zhong** I would like to express my gratitude to Prof. Dr.-Ing Horst Hellbrück, who wrote the first draft of this document, and offered me such a great opportunity to contribute to this meaningful work.

**Acknowledgements by Andreas Hanemann** I would like to thank the people who have worked on this document earlier (Horst Hellbrück, Zhi Haolin, Aida Bahta, Ren Zhong). Some advice from Hermann Hochhaus and Hans-Günter Kunze has also been included into the document.

**Acknowledgements by Denys Matthies** I would like to particularly thank Martin Ochoa, Vel, Johannes Böttcher, Steve Gunn, Sunil Patand, and Benjamin Petry for providing the basis for this new layout. Thank you!

The authors hope that students can benefit from the document and that more students may add contributions to this document to improve it.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

**CI** **C**ochlea **I**mplant
**DMI** **D**igital **M**usic **I**nstrument
**HCI** **H**uman **C**omputer **I**nteraction

*Dedicated to my someone,*
*who you really appreciate to have in your life.*

# 1 INTRODUCTION

Building management is one key aspect of facility management. It integrates various disciplines to ensure the functionality, comfort, safety, and efficiency of the building environment for people, processes, and technology. Energy management has become the major aspect of building management nowadays. The calculation of energy consumption of a building is heavily influenced by windows and their roller shutters: while windows emit most of the heat generated by sunlight to the environment, the roller shutters may have a huge impact on such transfer process. In order to optimize the energy consumption of the building, efficient automated building management and monitoring should incorporate available information of the building status. This includes the window status and the roller shutter positions.

This bachelor thesis aims to investigate and implement in machine vision algorithm for detection of the roller shutter status. Since the estimation of energy consumption is a continuous process, the implemented algorithm should be capable of accurate detection 24/7 and under various weather conditions. In this bachelor thesis, deep learning algorithm You only look once (YOLO) is selected to solve this object detection problem. The detection model is trained and tested via a limited data set obtained manually or by preset cameras on the THL campus.

This bachelor thesis is organized into three parts. The first part (Chapter 2-4) is a

literature review. In Chapter 2, several common solutions are systematically introduced, which includes computer vision, machine learning and Convolutional Neural Networks (CNNs). Chapter 3 explains the applications of CNNs on object detection and main features of YOLO algorithm. Chapter 4 analyzes the existing challenges in roller shutter detection under the requirements of this dissertation.

# 2 REVIEW OF METHODOLOGY

This chapter aims to conduct a preliminary task analysis and paper review, then a systematic introduction to possible methods for solving the given task is provided. A literature review on common methods will be presented to provide a comprehensive understanding of the topic.

## 2.1 Related Work

The detection of the roller shutters can surely be realized by human eyes and manual measurement. However, different lighting and weather conditions, for example, during the night or on rainy days, may have a huge impact on the preciseness of human eye recognition. In addition, it may cost a lot of time and labor to realize round-the-clock detection. To improve efficiency and reduce costs, previous studies have suggested the use of computers to replace humans in some aspects of these visual tasks, that is, the introduction of **machine vision**.

Few previous studies focused exactly on detecting the roller shutters of windows. However, as the principles and detection requirements are similar to many other parts of a building, candidate methods for this bachelor thesis can be referenced from studies that aimed to detect doors, windows, or other basic building components.

Digital signal processing is a common and traditional tool used for solving machine vision tasks. In a previous work, Jin et al. applied a content-based object detection approach to window detection, which utilized gradient transforms to detect edges [1]. Similarly, Recky et al. used an extended gradient projection method for detecting windows, and introduced a facade color descriptor based on k-means clustering in a CIE-Lab color space [2]. Sirmacek et al. utilized a set of steerable filters for L-shape features and perceptual organization rules to detect windows and doors on a rectified thermal image of the building [3]. These studies all showed promising experimental results, but the error rates were mainly attributed to different lighting conditions and window reflections. Therefore, characterizing the various appearances of real-world surfaces remains a challenging task for these legacy approaches.

As artificial intelligence advances, machine learning and deep learning algorithms have been applied to various machine vision tasks. Wang et al. designed a robust classifier for window detection in facades using the traditional machine learning algorithm Gentle AdaBoost. In recent studies, You Only Look Once (YOLO) has been used to train prediction models for tasks such as door detection in convenience stores. Researchers in a 2021 study utilized YOLOv4, which is considered state-of-the-art, and optimized the model to distinguish between convenience store glass doors and glass walls by incorporating surrounding objects in the scene. Biyomi et al. also used YOLOv5 for building envelope detection and compared its performance to other methods in a recent study. These studies suggest that applying machine learning and deep learning algorithms can lead to higher detection accuracy and can handle various scenarios. However, the training process may require large amounts of data and equipment, and may take longer than traditional digital signal processing methods.

As seen from the timeline of related studies, applying deep learning methods to machine vision tasks has become the main trend. However, digital signal processing and traditional machine learning methods can be more efficient in certain detection problems. Additionally, previous studies have primarily focused on determining the location of the target object, with the parameters of the target object itself being less crucial. In contrast, this bachelor thesis considers not only "where" but also "how much" as equally crucial. Since it is still unclear which method would be most appropriate for the current task, the following

sections will introduce these methods accordingly.

## 2.2 Machine Vision

Machine vision is a subcategory of computer vision, which is proposed by David Marr in his book vision[4]. It refers to the technology and methods used to provide imaging-based automatic inspection and analysis for such applications as automatic inspection, process control, and robot guidance, usually in industry [5].

The primary purpose of machine vision is to create models and data extracts and information from images [6]. Machine vision works by using an algorithm and optical sensors to stimulate human visualization to automatically extract valuable information from an object. Image processing and pattern recognition become two indispensable components for machines to output image understanding. Many techniques from artificial intelligence also play important roles in all aspects of computer vision [7].

As shown Figure 2.1, based on the review of previous studies, a machine vision task can generally be solved by using the traditional **Digital Signal Processing (DSP)** method, or **Machine Learning** related methods.

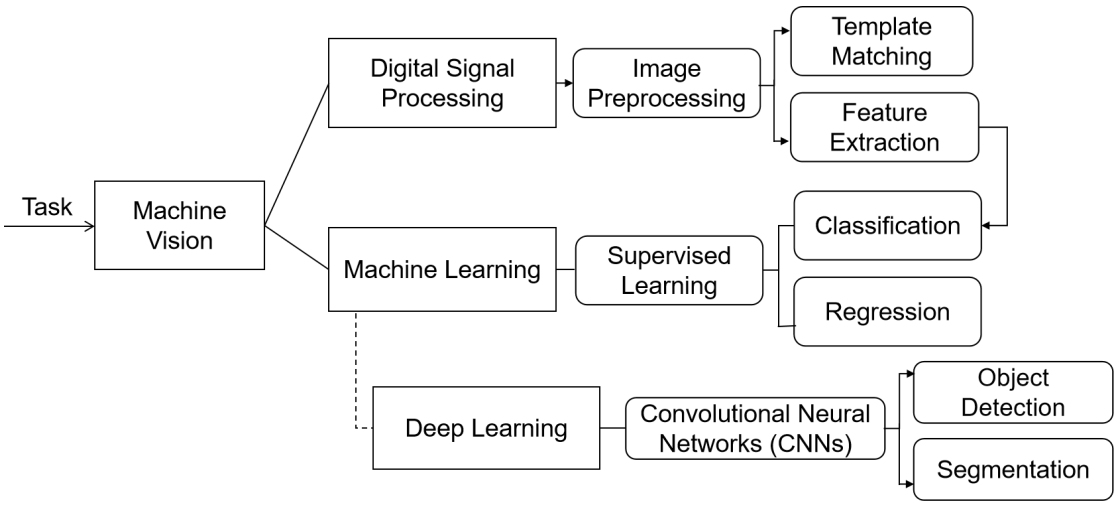


FIGURE 2.1: Methodology Overview

## 2.3 Digital Signal Processing

Digital Signal processing techniques can be applied to process digital images. A digital image is a two-dimensional matrix of pixel points, each with certain brightness and color information. By using DSP techniques, we can analyze these pixel points to extract useful information. Such process is called digital image processing. It mainly includes the following two aspects: Image preprocessing and feature extraction. Template matching is a commonly used DSP based technique for machine vision tasks. Introduction of template matching is also involved in Appendix .1.

### 2.3.1 Image Preprocessing

Due to the complexity of the acquisition environment and lighting, the quality of the image can be greatly disturbed, which in turn affects the accuracy of recognition. Therefore, before the image is recognized, the necessary preprocessing operations need to be performed. Basic preprocessing operations include graying, binarization, smoothing, enhancing, scaling, rotating and edge detection.

**Graying** is the process of changing the image to grayscale, while **binarization** changes the grayscale image into a grayscale image with values of 0 and 255. These two operations reduce the amount of data in the image and simplifies the image processing. **Smoothing** removes detailed information from images to reduce noise and image distortion, where median filter is a commonly used technique. **Enhancing** aims to improve the contrast, brightness, and detail of images, making them sharper and easier to see. Histogram Equalization is the state-of-art. **Scaling** and **rotating**, as their names indicate, change the size, direction, angle of the image for better follow-up processing and analysis.

**Edges** are very important features of an image in machine vision. This is because edges represent the demarcation line between different regions, and such division is crucial for segmentation and object recognition. **Edge detection** is the operation to detect such edges in the preprocessing stage. Canny algorithm [8] is a widely used gradient-based edge detection algorithm which can accurately

detect edges in the image with good noise immunity and efficient performance. Figure 2.2 shows the effect of aforementioned preprocessing operations.

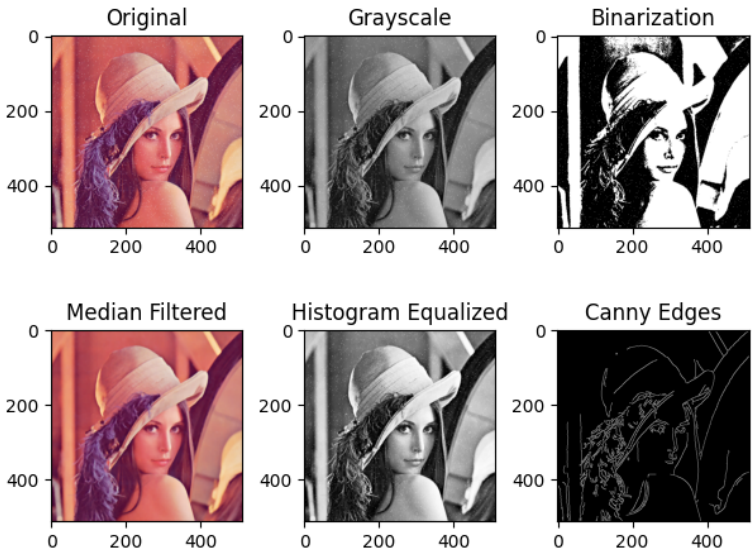


FIGURE 2.2: Effect of multiple preprocessing operations

### 2.3.2 Feature Extraction

Feature extraction is the basis of object recognition, which can extract important information from images like feature points. These feature points can be kept constant at different scales and angles, thus improving the accuracy of object recognition and classification. Several common feature extraction techniques are described below:

**Harris corner detector** [9], as the name infers, is used to extract corner feature points by calculating the difference in gray value between a pixel point in the image and its surrounding pixel points.

**scale-invariant feature transform (SIFT)** [10] transforms an image into a large collection of feature vectors, each of which is invariant to image translation, scaling, and rotation, partially invariant to illumination changes, and robust to local geometric distortion. Several improvements have been made in its improved algorithm **Speeded-up robust features (SURF)** [11].

**Oriented FAST and rotated BRIEF (ORB)** [12] is a faster, more robust local feature detector modified from SIFT and SURF to detect features points in images and generate descriptors. However, it is less robust to different lighting conditions and occlusion.

After feature descriptors are extracted, they are used as input to train the classifier which is then used for classification or object detection. Related machine learning terms will be explained in the next section.

## 2.4 Machine Learning

**Machine learning** is a sub-field of artificial intelligence that is concerned with building useful algorithms rely on a collection of examples, no matter naturally or artificially generated [13].

### 2.4.1 Supervised Learning

**Supervised learning** is one of the categories and the most common learning method of machine learning. The aim of supervised learning is to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a pre-labelled input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data. Such a predictive model can be based on two techniques: classification and regression, predicting discrete and continuous responses respectively. Most image-focused pattern-recognition tasks usually depend on classification using supervised learning.

### 2.4.2 Classification

**Classification** is the task which automatically assigning a discrete label to an unlabeled example, for example, assigning a given email to the "spam" or "non-spam" class. [13]

A classification problem in machine learning is typically addressed through the use of a classification learning algorithm. These algorithms take a set of labeled examples as input and generate a model that can be used to predict the label of a new, unlabeled example. The output of the model can be either a directly assigned label, or a numerical value that can be interpreted by an analyst to deduce the label. **Support Vector Machine (SVM)** is a typical classifier, and its feature is introduced in Appendix .2.

The evaluation of the classification result can be done by a **confusion matrix**, which can classify the test data into four categories: true positives (TP) , false positives (FP), true negatives (TN) and false negatives (FN). Correct Classification Rate (CCR) , a common assessment metric of classification models, can then be calculated as follow:

$$\text{CCR} = \frac{\text{TN} + \text{TP}}{\text{FN} + \text{FP} + \text{TP} + \text{TN}}$$

### 2.4.3 Regression

**Regression** is the task of predicting a real-valued label (often called a target) given an unlabeled example, for example, estimating stock price [13].

In supervised learning, a regression problem chooses a regression algorithm to take a collection of labeled examples as inputs and produces a model that can take an unlabeled example as input and output a target. The relation between predicted output and input can be expressed as $y' = f(x)$. Based on the function, a regression problem can be classified as linear regression, polynomial regression, etc. During the training of a regression model, **loss function** is used to evaluate the performance by calculating the error between predicted and true values. Mean Squared Error (MSE) is a common loss function, which can be calculated by $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. The smaller the MSE, the better the performance of the model will be. Figure 2.3 depicts an example of a linear regression model and the corresponding MSE.

In order to minimize the loss function of the model, **gradient descent algorithm** is used to obtain the best model parameters. Gradient descent algorithm is an

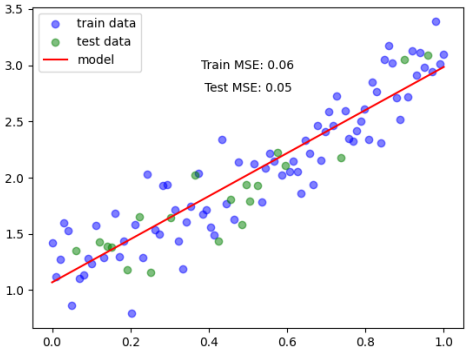


FIGURE 2.3: An example of a linear regression model

iterative optimization algorithm that reduces the value of the loss function by calculating the partial derivatives (gradient) of the loss function with respect to the model parameters and updating the model parameters along the negative gradient direction. The algorithm is used to continuously update the model parameters, and the optimal regression model can be gradually fitted to minimize the error between the prediction result and the true value.

Traditional machine learning classification and regression methods can largely improve the adaptability to different scenarios and tasks compared to the traditional digital signal processing. However, their generalization capabilities are still insufficient, while a large number of manual rules still need to be designed for feature extraction and dealing with occlusion.

## 2.5 Artificial Neural Networks (ANNs) and Deep Learning

**Artificial neural networks (ANNs)** is a very useful tool in supervised learning and is the foundation of deep learning. ANNs are computational processing systems of which are heavily inspired by way biological nervous systems (such as the human brain) operate. ANNs are mainly comprised of a high number of interconnected computational nodes, which we name as neurons. These neurons work distributively to collectively learn from the input in order to optimise its final output.

Figure 2.5 shows the architecture of a simple ANN and how a neuron processes its input. For every neuron, it calculates the sum of every input, which is the output of each neuron in the previous layer, with **weights** set to each input to measure its influence on the result. A **bias** is also set to adjust the intermediate result for **activation** afterward. The goal of activation is to add non-linear features into the output of a neuron, which helps neural networks discover and learn complex patterns, and to normalize such output to be an input of the next layer. Rectified Linear Unit (ReLU) is one of the commonly used activation functions, which simply turns all negative output values to 0.

An ANN is generally composed of three types of layers. The input layer loads inputs in the form of a multidimensional vector to the input layer, and distribute them to the hidden layer. The hidden layer will then make decisions from the previous layer and weigh up how a stochastic change within itself detriments or improves the final output, and this is referred to as the process of learning. Having multiple hidden layers stacked upon each-other is commonly called deep learning [14].

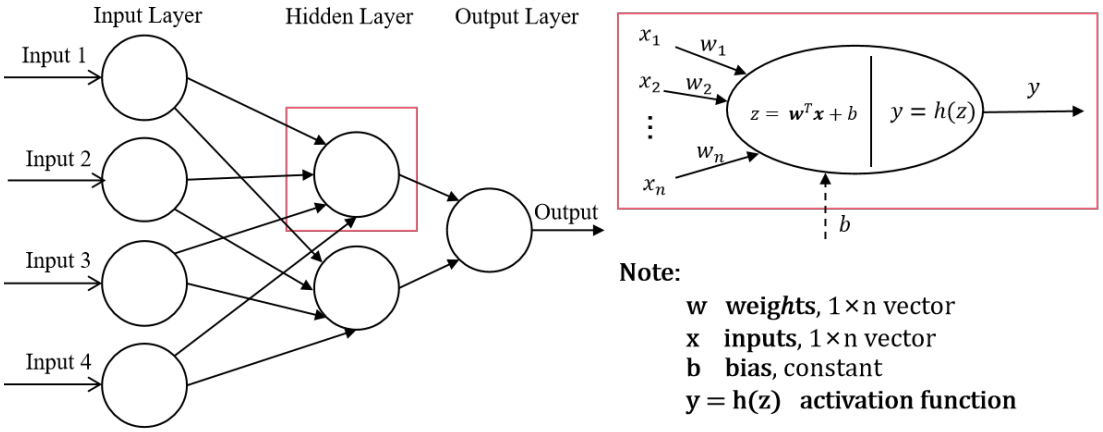


FIGURE 2.4: The architecture of a simple ANN network

One of the largest limitations of traditional forms of ANNs is that when computing image data, they tend to struggle with the computational complexity. This is because each neuron in fully-connected layers needs to be connected to all neurons in the previous layer, and this require a lot of computation to deal with the huge number of weights each neuron provides for every single pixel. Besides, even if the computing power is unlimited, the complexity will cause

another problem: overfitting, which occurs when a model is trained too well on a particular data set, to the extent that it starts to fit the noise or random fluctuations in the data rather than the underlying patterns or trends, resulting in poor performance on new, unseen data. In order to reduce the complexity of ANNs, **Convolutional Neural Networks (CNNs)** is introduced.

### 2.5.1 Convolutional Neural Networks (CNNs)

CNNs are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. The notable differences between two networks are their architectures, which makes CNNs more capable in the field of pattern recognition within images.

Figure 2.5 depicts the architecture of a general CNN. The basic functionality of this example CNN can be broken down into three key areas: convolutional layers, pooling layers and fully-connected layers.

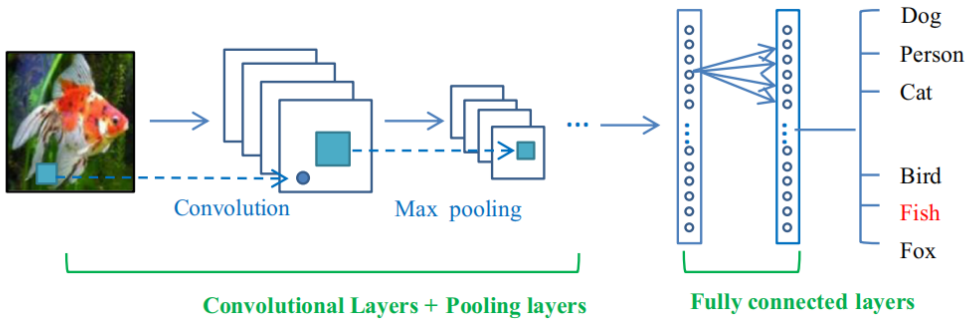


FIGURE 2.5: The pipeline of the general CNN architecture [15]

As can be implied from the name, **convolutional layers** play a vital role in how CNNs operate. Its main task is to extract features from the original input image by convolution operations. The output of the convolution layer can be used as the input of the next layer.

Feature extraction in the convolutional layers is realized by several learnable convolution **kernels**, which are used to convolve the input image to produce an output feature map. In a CNN, each neuron of the convolutional layer uses a kernel to process the input pixels adjacent to it. Every kernel can serve as a

filter to extract one specific feature of the input image like edges, corners, etc. Since such filters can be applied to any part of the input image and each type of filters is sharing the same parameters, the number of parameters we required is dramatically decreased. This important feature is called **parameter sharing**, which can reduce the risk of overfitting and improving the generalization ability of the model.

During the convolution operation, a kernel is used to slide over the input image, performing one convolution operation at a time, to produce the output feature map. As the input is glided through, the scalar product is calculated for each value in that kernel. Then activation function will be applied, and every kernel will obtain a corresponding activation map, of which will be stacked along the depth dimension to form the full output volume from the convolutional layer.

There are three hyperparameters that control the size of the output volume of the convolutional layer: depth, stride and zero-padding. **Depth** refers to the number of convolutional kernels in the convolutional layer. **Stride** is the distance that the filter moves along the input each time. **Zero-padding** is the simple process of padding the border of the input by setting it zero.

A **pooling layer** is usually attached after convolutional layers. It contains pooling operations used to reduce the size of the convolutional layer outputs, thereby reducing the model complexity and computing power required. Pooling operations can also improve the robustness and generalization of the model to reduce the risk of overfitting. The most commonly used pooling function is max pooling and average pooling, which take the maximum value in the pooling window as output.

Following several convolutional and pooling layers, **fully-connected layers** perform the high-level reasoning in CNNs. As the name implies, every neuron in fully connected layer has full connections to all activation in the previous layer. Their activation can then be calculated with a matrix multiplication with a bias offset. The fully connected layer eventually converts the 2D feature map into a 1D vector. The vectors derived can either be fed into a certain number of classifications or can be considered as feature vectors for further processing [16].

The input data is processed through convolutional layers, pooling layers, etc., and finally the prediction result of the model is obtained in the output layer. Such process is defined as forward propagation. However, the predicted value may have deviation from the real one to some extent. In order to diminish such deviation, we need use loss functions to calculate the deviation and updates each parameter by **back propagation** to minimize the loss. Similar to the process introduced in the regression model training, gradient descent algorithm is also used in back propagation for calculating new weights in that procedure.

### 2.5.2 Object detection

Object detection aims to automatically identify and localize a specific class of objects from an image or video. The target objects are marked given their locations via **bounding boxes**. In recent years, the development of deep learning has made significant progress in object detection. Deep learning-based object detection methods include two main types: single-stage detection and two-stage detection.

**Single-stage detection** requires only one neural network model, performing forward propagation only once to complete the object detection. **You Only Look Once (YOLO)** is a representative single-stage detection algorithm. Its main idea is to divide the feature map obtained from the output of CNN into multiple grids, with each grid predicting a fixed number of bounding boxes and their corresponding object class probabilities. Then, category probabilities and position information of each bounding box are adjusted to obtain the final prediction box. Finally, the Non-Maximum Suppression (NMS) algorithm is applied to remove redundant bounding boxes and obtain the detection results [17]. Besides the YOLO series, Single Shot MultiBox Detector (SSD), RetinaNet are another single-stage detection algorithms.

**Two-stage detection** can generally offer better accuracy. It involves two steps: proposal generation and detection. In the first stage, a region proposal network generates potential bounding boxes, and in the second stage, features of the proposed regions are extracted and used to classify and refine the boxes. **Faster**

**R-CNN** and **Mask R-CNN** are two of the most popular two-stage detection algorithms.

### 2.5.3 Segmentation

Unlike object detection, segmentation requires classification of each pixel in the image, rather than localizing and classifying only the boundaries of the object. Based on different segmentation results, it can be categorized into three types: semantic segmentation, instance segmentation and panoptic segmentation. An example is given in Figure 2.6.

**Semantic segmentation** is an approach detecting, for every pixel, belonging class of the object [18]. For example, all cars in an image are segmented as one object and background as another object. **U-Net**, named by its U-shape architecture [19], is one of the state-of-art based on CNN to process semantic segmentation.

**Instance segmentation** aims to identify the every pixel's belonging of object. It detects each distinct object of interest in the figure. For instance, every car in a figure is segmented as an individual object. **Mask R-CNN**, can also be used in instance segmentation task.

**Panoptic segmentation** combines the feature of semantic and instance segmentation. It can identify the belonging class of each pixels, meanwhile distinguishing them based on different instance. However, processing such a large number of object instances and backgrounds simultaneously would requires more resources. **Panoptic FPN** and **BlendMask** are two commonly used algorithms.

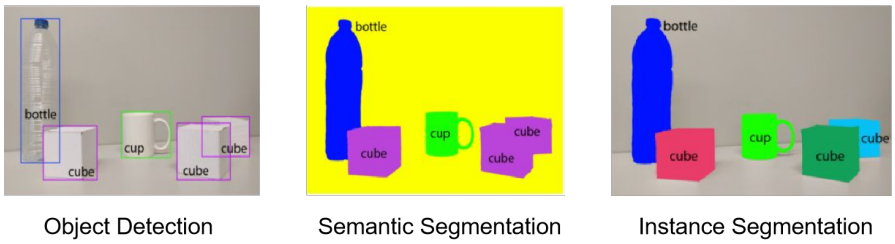


FIGURE 2.6: An example of object detection and segmentation [20]

# 3 CONCEPT

This chapter aims to identify a promising approach to address the task. A thorough analysis of the challenges associated with the task is conducted first. This analysis serves as a foundation for comparing and contrasting different methods and algorithms to determine the most effective approach.

## 3.1 Task Challenges

This study is confined to a particular building on the THL campus, and concentrates on a single facade. As a result, the roller shutters installed on this particular building are relatively homogeneous in terms of their features, such as texture, brands, and colors. Additionally, the images used for analysis are static, and do not require real-time processing. Despite these factors, the realization of the proposed detection method may face several challenges. Three main challenges are identified and discussed in the following sections.

### 3.1.1 Round-the-clock detection

Detecting objects reliably under different weather conditions and at different times of day poses a challenge. Changes in light intensity and color temperature

can significantly affect the appearance features of roller shutters, while camera's performance may also degrade due to factors such as lighting and noise, thereby affecting the image quality. Figure 3.1 depicts how the image of the same window would differ under various conditions.


FIGURE 3.1: Images of the same window taken at different time or under different weather conditions

### 3.1.2 Impact of occlusion

Occlusion is considered an existing and challenging issue in machine vision applications that can significantly hinder feature extraction and classification when the target is obscured. The problem is further exacerbated in the current study, where not only the roller shutter need to be detected, but its degree of closure must also be calculated. Figure 3.2 describes examples of occlusion scenarios in this task. Addressing this challenge requires a highly demanding solution to mitigate the impact of occlusion.


FIGURE 3.2: Images of various occlusion scenarios

### 3.1.3 Measurement of the Degree of Closure (DoC)

In order to accurately determine the percentage of how much a roller shutter is lowered (In this bachelor thesis, DoC is used to represent such percentage) , a reliable and consistent measurement method that can be applied to all situations is necessary. While the length and area of a window remain fixed in the real world, the shape of the window in the 2D world of digital photography can be affected by the camera angle, making it challenging to obtain clear and upright images for accurate measurements. Additionally, distinguishing the boundary between the window and the roller shutter may also pose difficulties.

## 3.2 Evaluation of Methods

In chapter 2, three promising methods were introduced, which are:

1. traditional digital signal processing techniques
2. traditional machine learning with classification and regression
3. deep learning with object detection and segmentation

Based on their performance for addressing the challenges mentioned above, one method will be selected for use in this study. This section presents an analysis of the strengths and limitations of each method. In order to ensure simplicity throughout this section, this bachelor thesis employs **DSP Method, TML Method** and **DL Method**, respectively, to represent these methods. Table 3.1 provides an overview of the performance of the three methods in addressing the challenges posed by the task, which would be explained in the following paragraphs.

The first challenge is round-the-clock detection, for which the DL method performs the best. DSP Method can be a fast and simple solution, but it requires several preprocessing steps to reduce the impact of lighting or noise. However, these steps should be designed carefully while adaptive processing steps may hinder the accuracy of detection to a great extent. The TML method has the

TABLE 3.1: Evaluation of Three Methods

| Methods | Round-the-clock Detection | Occlusion | Measurement |
|---|---|---|---|
| DSP Method | + | - | + |
| TML Method | ++ | + | + |
| DL Method | +++ | ++ | ++ |

**Note:** The more plus signs (+) indicate better performance, while the minus sign (-) indicates the method is not suitable for addressing the challenge.

advantage of being able to classify or regress new and unknown patterns or scenarios by learning the statistical patterns of the data, thus having stronger generalization ability. However, it still requires manual feature engineering, which demands specific knowledge and experience. On the other hand, DL method may require some time to train the model, but it can overcome the limitations of the other two methods. By learning large amounts of data, DL can automatically extract features and better generalize to new data. Since the speed of detection and training time are not the primary requirements of this study, the DL method is considered the most appropriate for addressing round-the-clock detection issues.

The second challenge is occlusion. DL method still performs best in tackling this issue. DSP Method like template matching relies on comparing the raw pixel values of the input image to a pre-defined template, which is not robust to changes in the appearance of the object due to occlusion. The TML method can improve the robustness of the model by selecting features that are insensitive to occlusion, and the classifier design can use a combination of multiple classifiers to mitigate the effects of occlusion. However, for different occlusion cases, the features need to be manually selected and designed again, which makes the implementation and adjustment of the algorithm more cumbersome. The DL method provides a better solution with the help of data augmentation in the training process, and can automatically learn feature representations in images, enabling them to recognize objects even when they are partially occluded. DL models can also incorporate multiple scales and viewpoints of the object to improve recognition accuracy.

In terms of measuring the DoC, the shape of the roller shutter is rectangular,

which matches the shape of a bounding box. Therefore, using bounding boxes as a representation is considered appropriate for this study. Although all three methods can provide bounding boxes to locate the target, the DL method can provide more detailed and accurate bounding boxes, as well as other information such as the class and confidence level of the target. This additional information enables more comprehensive and precise object detection and recognition, using techniques such as Non-Maximum Suppression (NMS), which outputs only the bounding boxes with the highest confidence level in a certain region. If a pixel-level measurement of DoC is required, a deep learning segmentation task can generate a mask of the roller shutters to improve the measurement accuracy.

Overall, the superior performance of the DL method, which surpasses the other two methods in every aspect, makes it the preferred approach for this bachelor thesis.

## 3.3 Evaluation of Algorithms

In order to select the most efficient algorithm for the current task, it is crucial to evaluate the candidate deep learning algorithms. While both object detection and segmentation algorithms can achieve recognition, this bachelor thesis will only focus on algorithms primarily designed for object detection. This is because although segmentation can provide precise classification of every single pixel, it requires far more training to achieve the same level of accuracy as object detection. On the other hand, as is mentioned above, the use of rectangular bounding box would be enough for measurement.

In the previous chapter, several one-stage and two-stage object detection algorithms were mentioned. This bachelor thesis includes R-CNN series, EfficientDet, YOLO series, RetinaNet, and SSD series as candidates for evaluation. To analyze the performance of these algorithms, two metrics were considered: Mean Average Precision (mAP) and Frames Per Second (FPS). While real-time detection was not a requirement for the task, higher FPS could indicate faster processing times when handling a large number of static images, and was therefore taken into consideration.

TABLE 3.2: Performance on COCO Dataset

| Algorithms | Proposal | Backbone | Size | FPS | mAP (%) |
|---|---|---|---|---|---|
| Faster R-CNN [21] | 01/2016 | ResNet-50 | - | 9.4 | 59.2 |
| R-FCN [22] | 03/2016 | ResNet-101 | - | 12 | 51.9 |
| FPN FRCN [22] | 10/2017 | ResNet-101 | - | 6 | 59.1 |
| RetinaNet [23] | 06/2018 | ResNet-101 | 800 | 5.1 | 57.5 |
| EfficientDet-D3 [24] | 03/2020 | Efficient-B3 | 896 | 23.8 | 65.0 |
| SSD [25] | 12/2016 | VGG-16 | 512 | 22 | 48.5 |
| SSD [22] | 12/2016 | ResNet-101 | 513 | 8 | 50.4 |
| DSSD [22] | 07/2017 | ResNet-101 | 513 | 6 | 53.3 |
| YOLOv3-SPP [26] | 04/2018 | Darknet-53 | 608 | 20 | 60.6 |
| YOLOv4 [27] | 04/2020 | CSPDarknet-53 | 608 | 23 | 65.7 |
| YOLOv5x [28] | 05/2020 | CSPDarknet-53 | 640 | - | 68.9 |
| YOLOv5x6 [28] | 05/2020 | CSPDarknet-53 | 1280 | - | 72.7 |

**Note:** (-) indicates that the corresponding metrics were not found.

This bachelor's thesis compiled the test results of each algorithm's most complex model (which maximize the mAP at the cost of FPS) on the COCO dataset (test-dev2017), as shown in Table 3.2. It can be seen that the later YOLO series achieved a generally higher mAP and faster detection speeds compared to the other algorithms. Among the YOLO series, YOLOv5 may be not the latest version of YOLO series, but it remains one of the state-of-art algorithms in the field of object detection and more researches are based on it than any other YOLO's later version. Therefore, this bachelor thesis chooses YOLOv5-v6.0 as the proposed algorithm to address the detection of roller shutter status.

## 3.4 Addressing Challenges Using Bounding Boxes

Now that object detection algorithm YOLOv5 is selected for this task, how it can solve the existing challenges need to be discussed before moving on to the implementation part. One of the main features of YOLOv5 and other object detection algorithm is using bounding boxes to localize the object. Although the performance of an algorithm can be decided by its architecture to a large extent,

the ability to tackle specific challenges can be improved if the bounding boxes are used wisely and certain rules are set.

As the round-the-clock detection can be well realized through existing YOLOv5 network architecture and feature extraction strategies, this section focuses mainly on the newly proposed solutions for addressing occlusion and measurement of DoC in roller shutter detection.

### 3.4.1 Solution for heavy occlusion

Unlike the common goal of object detection algorithms, which is to recognize objects despite all kinds of occlusion, the goal of this task is not only to recognize but also to provide precise measurements. Therefore, we do not want the model to sacrifice the accuracy of measurements by predicting the possible area of the entire roller shutter in the presence of heavy occlusion. To address this, a balance is defined to split occlusion scenarios into two categories: **Mild occlusion**, which the model should be robust enough to "see through", such as branches, leaves; **Heavy occlusion**, which should be exclude by the model, such as building components, sculptures, regardless of whether the part obscured is part of roller shutters. For heavy occlusion scenarios, this subsection will explain how a decision tree can be applied to tackle them.

To achieve this, the status of roller shutters are set as open, closed or blocked. This gives rise to another problem when we defining object classes. If we set shutter as the only object class, there will be only two different results based on whether roller shutters are detected or not. To be more specific, if roller shutter is not detected in the image, the status could either be open or blocked, which we cannot tell from each other based on the detection results. Therefore, this bachelor thesis add another object class to detect window glass, which can help determine the status.

As shown in Figure 3.3, ten scenarios are used to simulate the roller shutter status in reality. In some scenarios, although strong occlusion exists in the images, it has no influence on the determination of roller shutter status. This is because we only need to find out the demarcation between roller shutters and glass to

judge the DoC. Therefore, as long as such demarcation is clear, any occlusion would not be considered affecting the detection result. This also give rise to another method to calculate DoC, which will introduced later in this section.

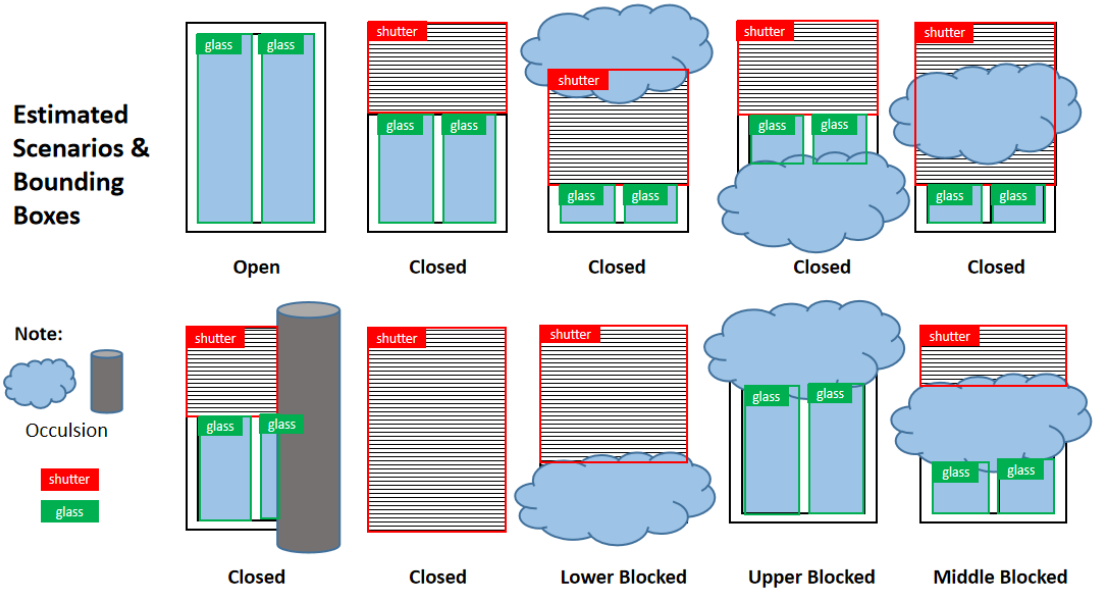


FIGURE 3.3: Estimated Scenarios and Bounding Boxes with Two Object Classes "Shutter" and "Glass"

To distinguish these scenarios by status, a decision tree is designed as figure 3.4 depicts, those scenarios can be split by simple logic comparing a few parameters of the bounding boxes. To be specific, **S3** and **G1**, which represent y coordinates of the lower left corner of shutter bounding boxes and the upper left corner of glass bounding boxes respectively. Those scenarios are split into four branches at first step based on the detection results:

**(1) Both the roller shutter and glass are detected.** This means the status is either closed or blocked. We need to examine whether the demarcation between the roller shutter and glass is clear by comparing G1 and S3. If G1 is approximately equal to S3, we can determine that the roller shutter status is closed. Otherwise, something is blocking the window in the middle, and the end of the roller shutter could be anywhere above G1 and below S3.

**(2) Only the roller shutter is detected.** In this case, we cannot tell whether the window glass is fully covered by the roller shutter or partly by occlusion. Therefore, we compare S3 with the height of the window, which is the height

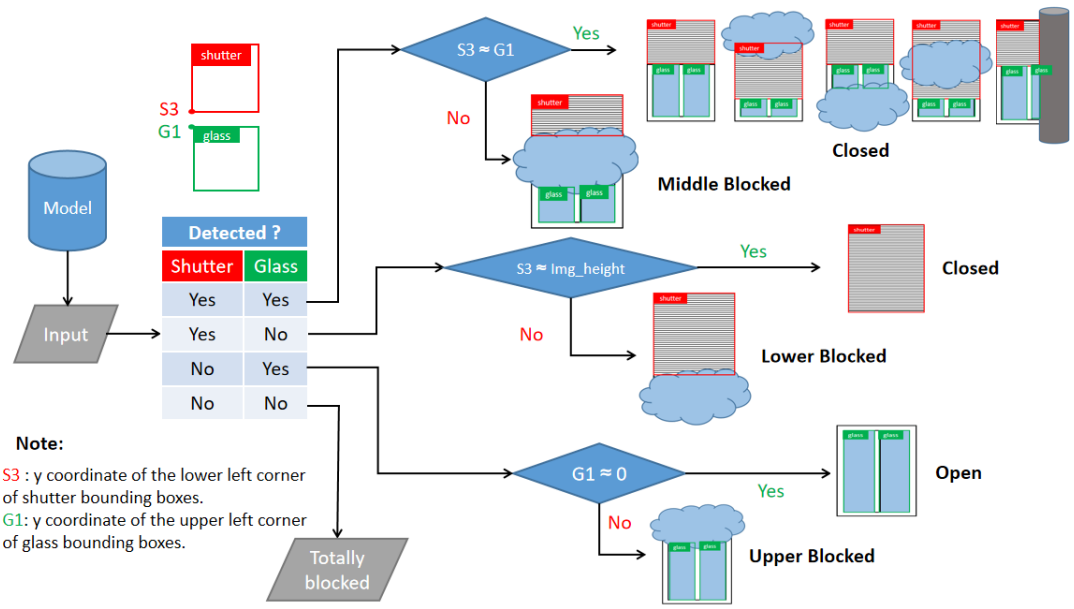


FIGURE 3.4: A Decision Tree to Determine The Roller Shutter Status

of the image in practice (images for detection contain only the window without any background). If S3 is approximately equal to the height of the image, the roller shutter is considered fully closed. If not, it is considered blocked, and the exact DoC cannot be determined.

**(3) Only the glass is detected.** With the absence of the roller shutter, we cannot simply say that the roller shutter is open. This is because the roller shutter may be covered by occlusion that covers the upper part of the image. To address this problem, we need to check the value of G1. If it approaches 0, which ensures that the upper parts of the image may not be the roller shutter, then we can determine the status as open. Otherwise, we cannot determine whether the area above G1 in the image is glass or roller shutter, which we mark as blocked.

**(4) Neither the roller shutter nor glass is detected.** This indicates a failure in detection, which could be due to the poor performance of the detection model or the window being completely blocked by occlusion. Assuming that the algorithm works well, we also mark this scenario as blocked.

By using this logic to determine the roller shutter status, the detection model should be robust enough to handle heavy occlusion. However, the logic is not

yet clear enough to put into practice because when comparing two parameters, "approximately equal to" cannot be understand by computer. Specific thresholds need to be determined and adjusted during implementation. Also, for mild occlusion, some optimizations have also been made, which are described in the next chapter.

### 3.4.2 Measurement of DoC

The most straightforward way to calculate DoC according to its definition is:

$$DoC = \frac{\text{Area of the Roller Shutter}}{\text{Area of Corresponding Window}} \times 100\%$$

As mentioned in previous evaluation of methods, the measurement of DoC can be achieved using bounding boxes due to their similar shape to roller shutters. However, The area of the bounding box may not express the true area of the Roller shutter. This is because bounding boxes are upright rectangular with their sides either vertical or horizontal. The roller shutter in the image, on the other hand, may not be upright due to shooting angles. Therefore, before detection, the cropped image of each window needed to be set upright to ensure the edge of the roller shutter in the picture is parallel to the edge of the bounding box. In addition, as the width of roller shutter is equal to the width of window and the height of window can be replaced by the height of cropped image in practice, the formula can simplified as

$$DoC = \frac{\text{Height of the Bounding Box}}{\text{Height of Image}} \times 100\%$$

However, the formula may be flawed in certain situations, as discussed in the previous subsection. Using the height of the bounding box may not be appropriate when dealing with occlusion problems, where the bounding box only represents the unobstructed part of the roller shutter. Therefore, as the y coordinate of the bounding box's upper boundary may be 0 in normal cases, the y coordinate of the lower boundary (S3) is used to represent the height of the roller shutter. The final optimized formula is described as:

$$DoC = \frac{\text{S3}}{\text{Height of Image}} \times 100\%$$

Since the height of the image is fixed and depends solely on the input size of the model, there is only one variable in the formula mentioned above. This means that this measurement method may be more resistant to errors. However, the validity of this formula depends on the assumption that each image for detection is only filled with a complete window, without any offset or redundant background in the vertical direction. While this may pose a challenge for localizing windows, the benefits of reducing error rates and minimizing the number of object classes for detection make it a worthwhile trade-off.

### 3.4.3 Fusion

To obtain accurate results, fusion is another effective method to mitigate the impact of heavy occlusion. In this bachelor thesis, three cameras are set up to capture images of the same building facade from different angles. This allows us to determine the optimal angle that provides the best representation of the roller shutter status and DoC.

As shown in Figure 3.5, we considered all 19 possible scenarios using three cameras and estimated the fusion results for both the roller shutter status and DoC. Our estimation was based on three pre-defined rules, as follows:

1. The status is considered "blocked" only when all camera angles produce "blocked" results.
2. If the final status is "blocked", the DoC range would describe the possible maximum and minimum values. If multiple cameras detect a "blocked" status, the range would be narrowed as much as possible.
3. If "open," "close," or both are detected, the final status is determined by the majority. If the final status is "closed," the average of the non-zero DoC values would be taken as the final DoC.

| Detected Status (Num) | | | Fusion Result | |
|---|---|---|---|---|
| Open | Closed | Blocked | Status | DoC |
| 1 | | | Open | 0% |
| | 1 | | Closed | $DoC_{closed}$ |
| | | 1 | Blocked | DoC_min to DoC_max |
| 2 | | | Open | 0% |
| | 2 | | Closed | $AVG(DoC_{closed})$ |
| | | 2 | Blocked | Max(DoC_min) to Min(DoC_max) |
| 1 | 1 | | ? | ? |
| 1 | | 1 | Open | 0% |
| | 1 | 1 | Closed | $DoC_{closed}$ |

| Detected Status (Num) | | | Fusion Result | |
|---|---|---|---|---|
| Open | Closed | Blocked | Status | DoC |
| 3 | | | Open | 0% |
| | 3 | | Closed | $AVG(DoC_{closed})$ |
| | | 3 | Blocked | Max(DoC_min) to Min(DoC_max) |
| 1 | 2 | | Closed | $AVG(DoC_{closed})$ |
| 1 | | 2 | Open | 0% |
| | 1 | 2 | Closed | $DoC_{closed}$ |
| 2 | 1 | | Open | 0% |
| 2 | | 1 | Open | 0% |
| | 2 | 1 | Closed | $AVG(DoC_{closed})$ |
| 1 | 1 | 1 | ? | ? |

FIGURE 3.5: All Detection Cases for up to three cameras and Predefined Fusion Results

The 19 possible scenarios discussed earlier can be summarized and generalized, as shown in Figure 3.6, based on the fusion results obtained through the predefined rules. However, in situations where the number of "open" and "closed" results are equal, it is difficult to determine the final status of the roller shutter accurately. Therefore, practice is needed before we make the decision.

| Case No. | Detected Status (Num) | | | Fusion Result | |
|---|---|---|---|---|---|
| | Open | Closed | Blocked | Status | DoC |
| 1 | A | | | Open | 0% |
| 2 | | A | | Closed | $AVG(DoC_{closed})$ |
| 3 | | | A | Blocked | Max(DoC_min) to Min(DoC_max) |
| 4 | a | A | X | Closed | $AVG(DoC_{closed})$ |
| 5 | A | a | X | Open | 0% |
| 6 | A | A | X | ? | ? |

**Note:** **X** : Any natural number.
**a, A:** Any positive integer, A > a

| | | | | | |
|---|---|---|---|---|---|
| Image 1 | 10% | 10% | 10% | 15% | 0%-20% |
| Image 2 | 0% | 0%-10% | 15% | 0% | 5%-30% |
| Image 3 | 20% | 0% | 20% | 0% | 0%-10% |
| Case No. | 4 | 6 | 2 | 5 | 3 |
| Status | Closed | ? | Closed | Open | Blocked |
| DoC | 15% | ? | 15% | 0% | 5%-10% |

FIGURE 3.6: Fusion Logic Summary (Left) and an Example (Right)

# 4 IMPLEMENTATION

This chapter provides a detailed explanation of how to detect roller shutters using the PyTorch framework and the YOLOv5-v6.0 algorithm.

## 4.1 Process Handover and Overview

This bachelor thesis is mainly focused on detecting the status of roller shutters, while the localization of window position has been addressed in another bachelor thesis titled "Building Management: AI Mapping Image to Construction Plan" by Tao Liu. For more information regarding window localization, please refer to the aforementioned thesis. This section will also give a brief introduction of the handover process.

As shown in Figure 4.1, the inputs for the entire process are building images captured by pre-set cameras on the THL campus. Tao's work involves processing these raw images into calibrated images and outputting window locations as coordinates in a text file. My work involves taking these calibrated images and window locations as inputs, and outputting the detection results, including the roller shutter status and the Degree of Closure (DoC) of each window, in an Excel file.

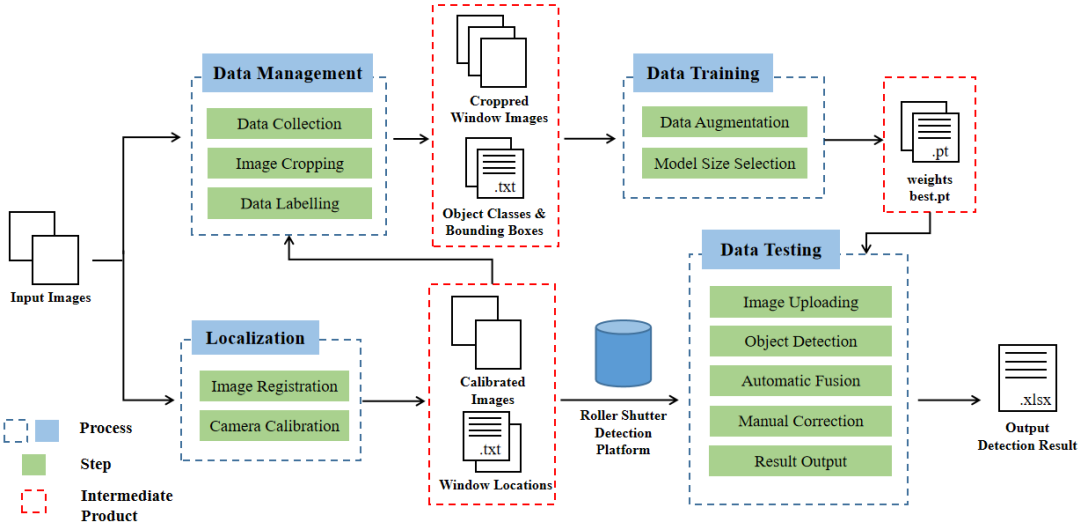


FIGURE 4.1: An Overview of the Work Flow

To enable an end-to-end process, a roller shutter detection platform is set up using the Qt Designer tool in Python, and the YOLO model is loaded for detection. Prior to loading the model, data management and data training are two critical modules for obtaining the necessary weights, which are essential for the detection model to function properly. Both raw input images and calibrated images are involved in the data collection process. These images are then manually cropped or cropped according to coordinates into separated window images for the labeling process followed. The following sections will provide a detailed explanation of each process module.

## 4.2 Data Management

Data Management involves three steps, which are data collection, image cropping and data labeling before adding to the training data set.

### 4.2.1 Image Acquisition and Preprocessing

### 4.2.2 Labeling Strategies

## 4.3 Data Training

After labeling, a text file recording the object classes and bounding boxes coordinates is generated for each cropped images. Before the training started, the model size of YOLO needs to be decided and customized data augmentation would also be an option.

### 4.3.1 Data augmentation

To improve the performance of detection model in different lighting conditions and robustness against occlusion. Two data augmentation methods are used, which are cutout and adjusting brightness. Cutout is the process to randomly dig out a specified number of small squares of a specified size from the image. This process can improve the robustness of the model against mild occlusions.

### 4.3.2 Model Selection

YOLOv5 offers five different models with varying sizes: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. Generally, larger models are more precise in object detection but require more time and higher performing equipment. Given the complexity of the current task and the available hardware (an NVIDIA RTX2060 GPU), YOLOv5m is selected as the preferred model. Additionally, the performance of YOLOv5n and YOLOv5s will also be compared with YOLOv5m in the evaluation chapter.

## 4.4 Detection Platform and Data Testing

After obtaining the weights from the training process, the model can be loaded to test its performance. In this bachelor thesis, a platform for detecting roller shutters is designed as the final product and evaluation tool. The GUI is shown in Figure 4.2. The following subsections will introduce its features and functions blocks.

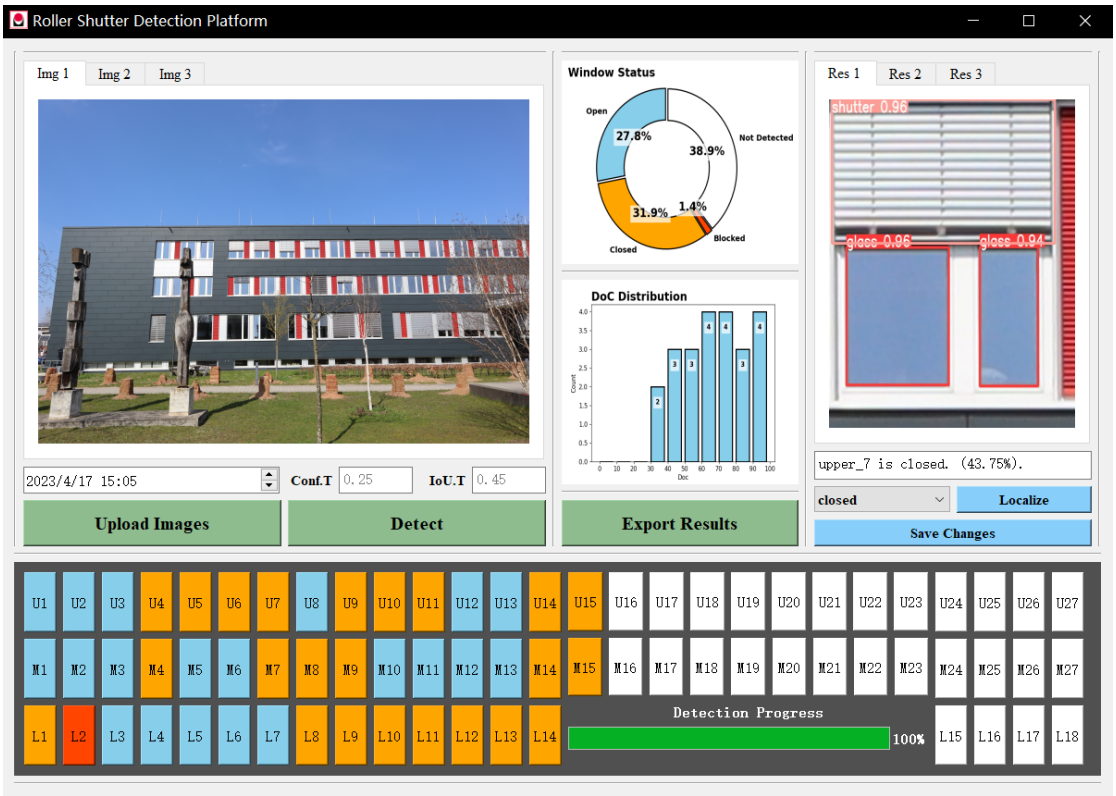


FIGURE 4.2: The GUI of roller shutter detection platform

Figure 4.3 offers an overview of the activity flow within the platform. The platform's functionality can be divided into four blocks: **Upload**, **Detect**, **Modify**, and **Export**. Three main functional blocks are accessed through three green buttons in the GUI, while the **Modify** block is not essential to the detection flow. Despite of this, the upper-left and lower parts of the GUI are designed to cater to the needs of checking and modifying the results.

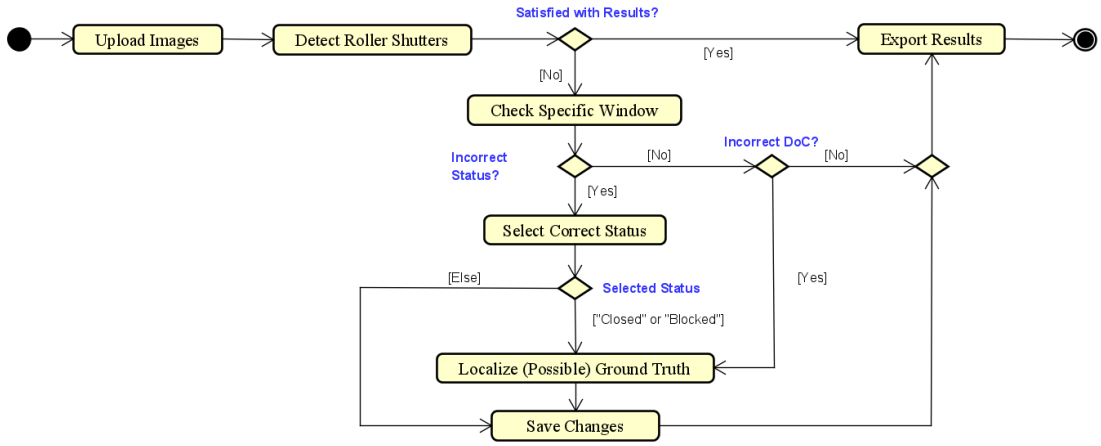


FIGURE 4.3: An Overview of the Work Flow (User Perspective)

### 4.4.1 Image Upload

The image upload process involves obtaining calibrated images and modifying them into cropped and upright images for further detection. It consists of two steps: **image selection** and **image preprocessing**. Figure 4.4 provides an illustration of how the uploaded images are processed.

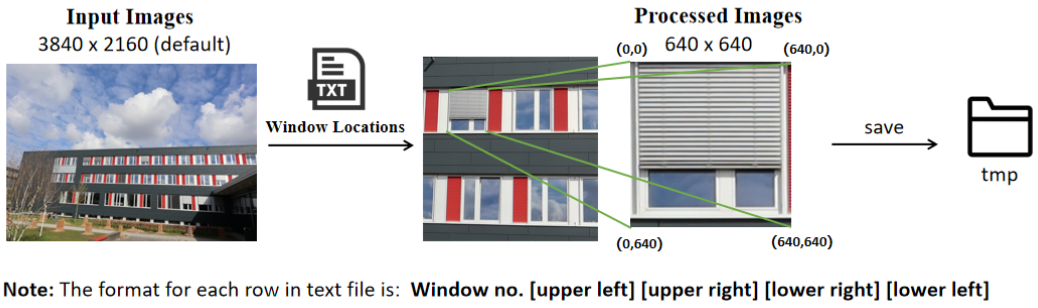


FIGURE 4.4: Preprocessing of Images Uploaded

**Image selection**: Upon clicking "Upload Images", users can select multiple calibrated images from their files. However, this platform only supports uploading up to three images at a time, based on the number of cameras set for the target building. Before the images can be uploaded, two validity checks are performed: First is to check if a text file with the same filename exists for each image because further cropping is based on the coordinates provided by such text file; Second is to check if the upload images are within the same period of time, which is

realized by comparing image taken time obtained from either the filename or file attributes.

**Image preprocessing**: Once the images pass two validity checks, they undergo a preprocessing step. This involves cropping the images based on the information provided in their corresponding text files. Each line in the text file contains the window number and the coordinates of its four corners. These coordinates are used to match and crop the corresponding window from the uploaded images. However, the resulting cropped images may not be upright, so a perspective warping technique is applied to map the coordinates onto the four corners of a $640 \times 640$ image. After these two processing steps are completed, the cropped and warped images are named according to their window number and saved in the temporary file, ready for the detection process.

### 4.4.2 Roller Shutter Detection

The detection process involves iterating through all the images in the temporary file, detecting the presence of glass and roller shutters, and drawing bounding boxes around them. In the GUI, two detection parameters can be set: the confidence threshold and the NMS (Non-Maximum Suppression) IoU (Intersection over Union) threshold. These parameters can be adjusted according to the specific requirements of the task. The default values are 0.25 for the confidence threshold and 0.45 for the NMS IoU threshold. The images with the bounding boxes are then saved in another file. During the detection process, two operations are performed simultaneously: **Status Display** and **Fusion**. Figure 4.5 provides an illustration of how these two operations are applied during the detection period.

**Status Display**: The duration of the detection process is determined by the number of processed images stored in the temporary file. For instance, processing one hundred images may take up to a minute. In addition to displaying a progress bar, this platform visualizes the iteration process using 72 buttons (located in the lower half of the GUI), which simulate the 72 windows of the target building. When the detection of an image is completed, the detected roller shutter status is displayed by changing the background color of the corresponding

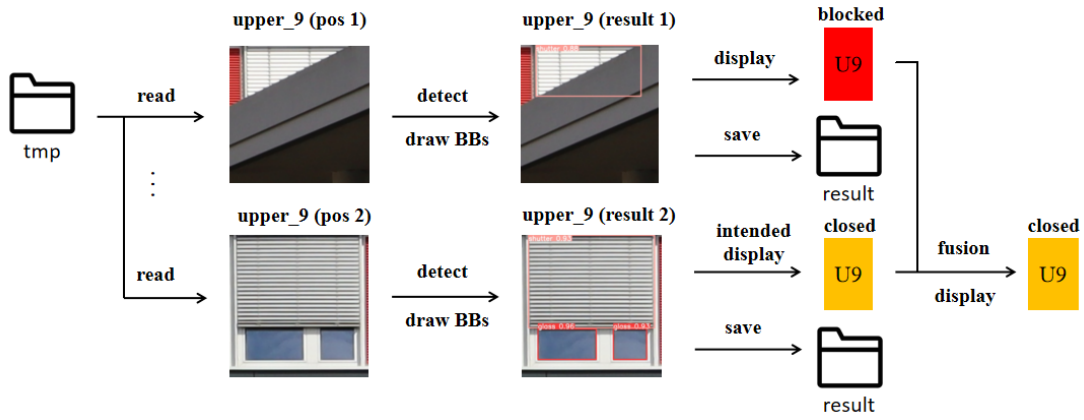


FIGURE 4.5: Display and Fusion during Detection Period

button: blue for open, orange for closed, red for blocked, and white for windows that have not been detected yet. Upon completion of the detection process, two graphs are displayed to illustrate the distribution of status, and DoC for the closed roller shutters, respectively.

**Fusion**: During a detection period, the color of the same button may change multiple times. These changes occur as a result of fusion when different results are obtained for a specific window. The fusion rules, which were introduced in the previous chapter, determine how these changes are handled. Instead of making a decision after the detection process, in practice, decisions are made each time a different detection result for the same window is obtained. However, the final fusion results remain consistent, ensuring that each window has a unique detection outcome.

### 4.4.3 Result Modify

After the detection is complete, users can view the detailed results of a specific window by clicking the corresponding button. If the results are incorrect, users can modify them by selecting the status and localizing the ground truth of the corresponding image. By clicking the "Save Changes" button, the detection results will be updated. Figure 4.6 demonstrates how users can check and modify the results. The "Localize" function allows users to set the modified DoC for the "closed" and "blocked" roller shutter statuses.

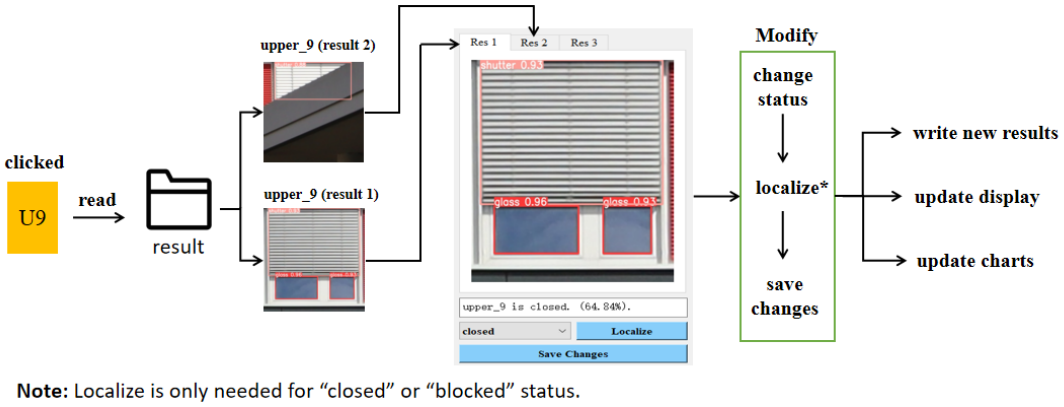


FIGURE 4.6: Check and Modify

For the "closed" status, users can precisely determine the location of the ground truth. In this case, localization involves setting the ground truth by selecting an image from the results. Users can localize the demarcation between the roller shutter and the window glass with a single mouse click in a newly opened window. The new DoC will then be set based on the mouse click position.

However, for "blocked" cases, localizing such demarcation is not possible. Users can only estimate a range to indicate the possible DoC. In this scenario, two mouse clicks are required to determine the possible locations where the DoC reaches its maximum and minimum values, respectively.

### 4.4.4 Report Export

This detection platform supports exporting the detection results in the form of Excel files. The default filename is set as the image capture time. Users have the option to change the filename and export it to any desired path according to their needs. The exported file includes the roller shutter status and DoC values for each window, along with two charts illustrating the distribution of status and DoC.

# 5

# EVALUATION AND OPTIMIZATION

# 6

# CONCLUSION

# REFERENCES

[1] H. Jin and M. Sakauchi, "Content-based objects detection for the recognition of building images," in *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, IEEE, vol. 2, 2001, pp. 705–708.

[2] M. Recky and F. Leberl, "Windows detection using k-means in cie-lab color space," in *2010 20th International Conference on Pattern Recognition*, IEEE, 2010, pp. 356–359.

[3] B. Sirmacek, L. Hoegner, and U. Stilla, "Detection of windows and doors from thermal images by grouping geometrical features," in *2011 Joint Urban Remote Sensing Event*, IEEE, 2011, pp. 133–136.

[4] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

[5] C. Steger, M. Ulrich, and C. Wiedemann, *Machine Vision Algorithms and Applications*. John Wiley & Sons, Mar. 12, 2018, 518 pp., Google-Books-ID: tppFDwAAQBAJ, ISBN: 978-3-527-41365-2.

[6] V. Wiley and T. Lucas, "Computer vision and image processing: A paper review," *International Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 29–36, 2018.

[7] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*. McGraw-hill New York, 1995, vol. 5.

[8] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986, Publisher: Ieee.

[9] C. Harris, M. Stephens, *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, Citeseer, vol. 15, 1988, pp. 10–5244.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.

[13] A. Burkov, *The hundred-page machine learning book*. Andriy Burkov Quebec City, QC, Canada, 2019, vol. 1.

[14] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[15] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016, Publisher: Elsevier.

[16] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018, Publisher: Hindawi.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[18] D. Guo, Y. Pei, K. Zheng, H. Yu, Y. Lu, and S. Wang, "Degraded image semantic segmentation with dense-gram networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 782–795, 2019.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234–241.

[20] Nitish, *Object detection and image segmentation*, 2017. [Online]. Available: https://nitishpuri.github.io/posts/machine-intelligence/object-detection-and-image-segmentation/.

[21] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2965–2974.

[22] J. Redmon. [Online]. Available: https://pjreddie.com/darknet/yolo/.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[24] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[25] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[26] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[27] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[28] Ultralytics, *Ultralytics/yolov5: Yolov5 in pytorch ; onnx ; coreml ; tflite*. [Online]. Available: https://github.com/ultralytics/yolov5.

[29] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[30] *Template matching*. [Online]. Available: https://docs.opencv.org/3.4/d4/dc6/tutorial_py_template_matching.html.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[32] https://aitechtogether.com/article/18854.html.

[33] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[35] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[36] T. Cheng, *Enhancing neural networks with mixup in pytorch*, 2021. [Online]. Available: https://towardsdatascience.com/enhancing-neural-networks-with-mixup-in-pytorch-5129d261bc4a.

APPENDIX

# A

# APPENDIX

## .1 Template Matching

**Template matching** refers to finding the part of the test image that is similar to the template image by comparing between the template image and the test image, which is achieved by calculating the similarity between the template image and the target in the test image, and can quickly locate the predefined target in the test image. A template can be an example, an instance; esp. a typical model or a representative instance [29].

Template matching is very similar to the principle of convolution, the template slides on the original image from the origin, calculates the degree of difference between the template and the place where the image is covered by the template, then the result of each calculation is put into a matrix and output as the result. After that maximum value in the similarity matrix is found, and this will give a position at which the input image matches the template image to the highest degree. That position is then be used to position the area which matches the template. Figure 1 shows an example of template matching.

Template matching is a simple and efficient way to detect the target object. Compared to machine learning techniques, template matching does not require a

large amount of training data and can therefore be used when the amount of data is limited. However, such method is sensitive to image transformations, and even slight environment changes would cause the failure, letting alone tackling the problem of occlusion. In order to improve the stability and accuracy of the algorithm, techniques like scale and rotation invariant feature detection are used to extract features from the image.


FIGURE 1: An example of template matching in OpenCV [30]

## .2 Support Vector Machine (SVM)

**Support Vector Machine (SVM)** is a type of classifier commonly used to tackle two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space [31]. In machine learning, the boundary separating the examples of different classes is called the decision boundary [13]. The algorithm aims to construct a maximally spaced hyperplane during the training so that the decision boundary has the maximum margin between the closest samples.

An example is shown in figure 2. As can be seen, the decision boundary L1 is set too close to the training samples, while L2 obtained by SVM algorithm maintains the largest margin. Two boundaries may make different predictions to the test sample marked in the figure, and these results are evaluated differently in the

confusion matrix (FN for L1 and TP for L2). This can show how L2 may obtain a better CCR as similar test samples increases.

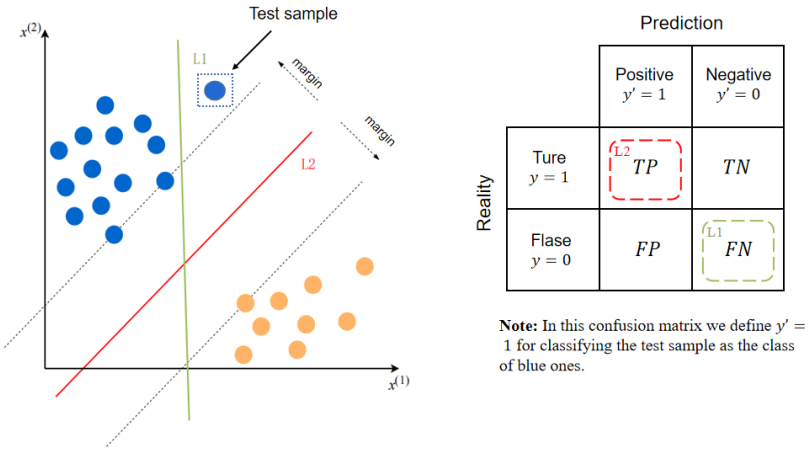


FIGURE 2: An example of SVM decision boundary (left) and the confusion matrix (right). L1 fails to classify the marked test sample correctly
[13]

## .3 YOLOv5-6.0

This section provides a comprehensive introduction to YOLOv5-6.0, which is utilized in this bachelor thesis. It covers various aspects, including the network architecture, loss calculation, and data augmentation method. Figure 3 illustrates the architecture of YOLOv5-6.0, which can be divided into three main components: the backbone, neck, and head.

### .3.1 Backbone

The Backbone of YOLOv5-6.0 version mainly consists of three modules: Conv Module, C3Net, and SPPF module.

**Conv Module (CBS)** . YOLOv5 encapsulates three functionalities within Conv module: convolution (Conv2d), batch normalization, and activation functions.

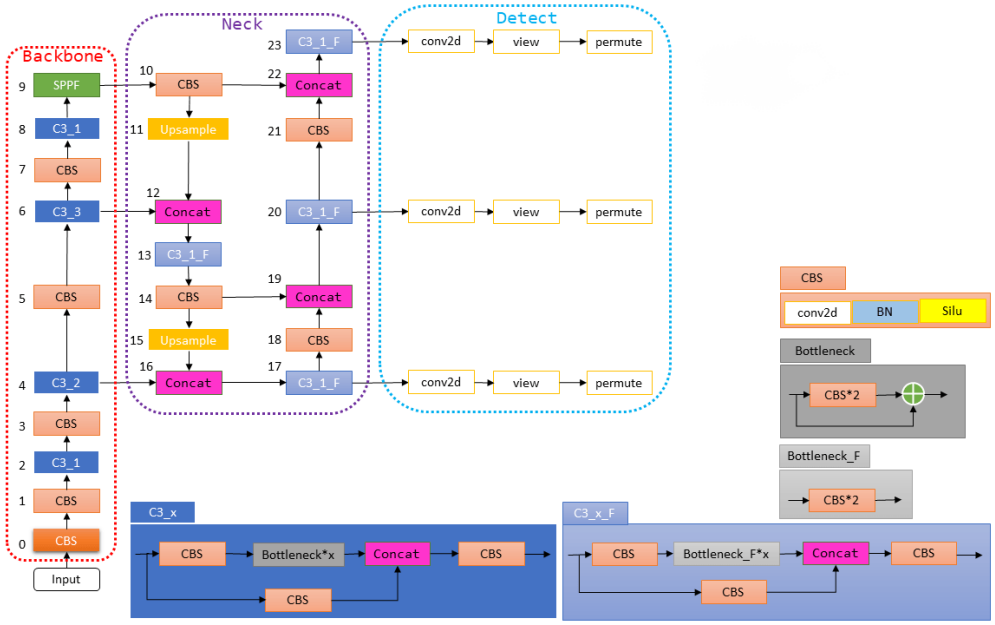


FIGURE 3: Network Architecture of YOLOv5-6.0 [32]

It also utilizes autopad(k, p) to achieve the effect of padding. In the YOLOv5-6.0 version, Swish (also known as SiLU) is employed as the activation function, replacing the previous versions' Leaky ReLU.

**C3Net**. C3Net stands for Cross-Cascade Network. It is based on CSPNet [33], which is designed to reduce computational complexity and improve inference speed while maintaining the detection and recognition accuracy of the model. Its main idea is to partition the gradient flow and allow it to propagate through different network paths. By using concatenation and transition operations, CSPNet achieves a richer combination of gradient information. Figure 4 gives an example of applying CSP to ResNet.

YOLOv5 draws inspiration from the CSPNet concept and applies it to the DarkNet53 backbone network. In the YOLOv5-6.0 version, the BottleneckCSP module from earlier versions is replaced with the C3 module. Both modules follow the CSP architecture, but they differ in the selection of correction units. The C3 module consists of three standard convolutional layers and multiple Bottleneck modules. The main difference between the C3 module and the BottleneckCSP module is that the Conv module following the output of the Bottleneck module is removed.

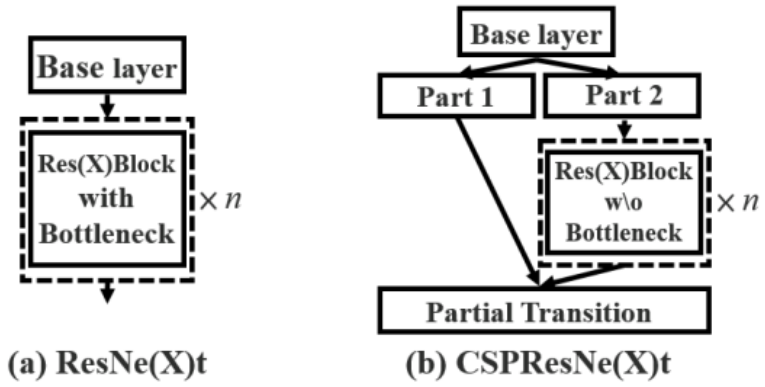


FIGURE 4: Applying CSPNet to ResNe(X)t [33]

**SPPF** [28]. In the YOLOv5-6.0 version, the SPPF module is used instead of the SPP module, where SPP stands for Spatial Pyramid Pooling. YOLOv5 draws inspiration from the concept of SPPNet. The SPPF module replaces the single large pooling kernel in the SPP module with multiple small-sized pooling kernels. This modification aims to further improve the runtime speed while preserving the original functionality of fusing feature maps with different receptive fields and enriching the expressive power of the feature maps.

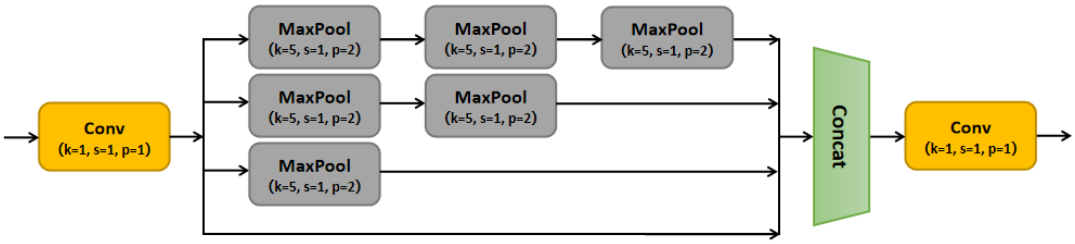


FIGURE 5: Structure of SPPF module

## .3.2 Neck

The Neck of YOLOv5 draws inspiration from the ideas of FPN (Feature Pyramid Network) and PANet (Path Aggregation Network).

**FPN**. FPN stands for Feature Pyramid Network, is a concept in object detection algorithms. In traditional object detection methods, predictions are often made solely based on features from the topmost layer of the network. However, it is

known that shallow features carry less semantic information but have stronger positional information, while deep features carry richer semantic information but weaker positional information.

The overall structure of FPN is illustrated in Figure 6. On the left side, there is a bottom-up propagation pathway, and on the right side, there is a top-down propagation pathway. The middle part represents the fusion of features through lateral connections.

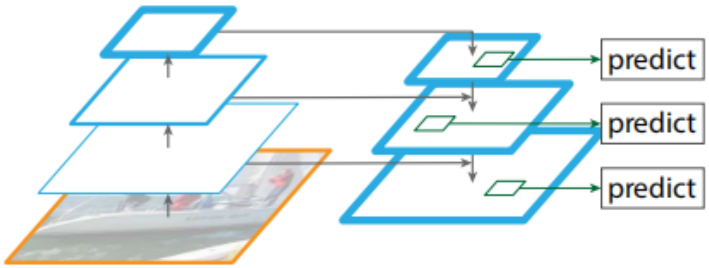


FIGURE 6: Structure of FPN
[34]

The bottom-up pathway corresponds to the forward propagation process of the network, which is associated with the backbone network mentioned earlier. During the forward process, the size of the feature map may change after passing through certain layers, while it remains unchanged after passing through other layers. The layers that do not change the size of the feature map are grouped into a stage. Therefore, the extracted features at each stage are the outputs of the last layer in that stage, forming a feature pyramid.

FPN addresses the issue of propagating deep semantic information to shallow layers through a top-down structure. However, the positional information from shallow layers does not influence the deep features. Additionally, in FPN, the top-down information flow needs to pass through the backbone network layer by layer, which leads to a higher computational burden due to the increased number of layers.

**PANet**. As shown in Figure 7, PANet introduces a bottom-up pathway in addition to FPN's top-down structure. After the top-down feature fusion, a bottom-up feature fusion is performed, allowing the positional information from the bottom

layers to be transmitted to the deep layers, thereby enhancing the localization capability across multiple scales.

Compared to FPN (as indicated by the red lines), PANet significantly reduces the number of feature maps that need to be traversed for the transmission of bottom-layer features (as indicated by the green lines). This makes it easier for the positional information from the bottom layers to propagate to the top layers.

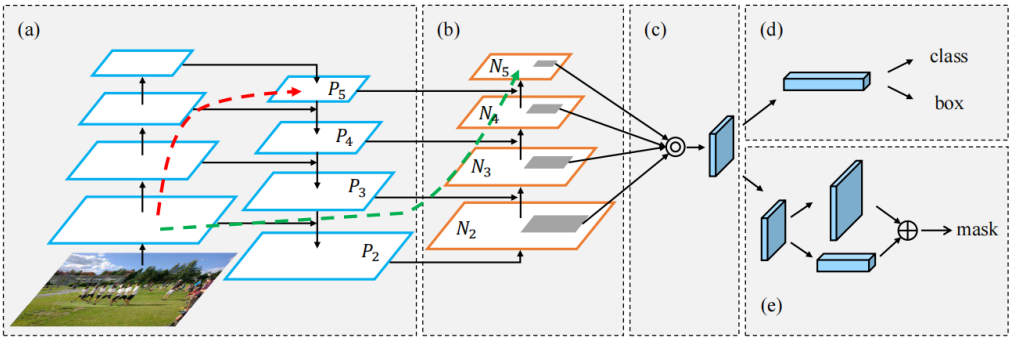


FIGURE 7: Illustration of framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion.
[35]

### .3.3 Head

In YOLOv5, the detection head takes the feature maps of different scales obtained from the neck and expands the channel dimension using 1x1 and 11x1 convolutions. The expanded feature channels have a size of (number of classes + 5) multiplied by the number of anchors per detection layer, where "5" corresponds to the predicted coordinates of the bounding boxes: center point (x, y), width, height, and confidence. The confidence represents the certainty or confidence level of the predicted bounding box and ranges from 0 to 1, where a higher value indicates a higher likelihood of the presence of an object.

The Head module consists of three detection layers, each corresponding to the feature maps of three different scales obtained from the Neck module. YOLOv5 divides the feature maps into grids based on their sizes and assigns three anchors with different aspect ratios to each grid on each feature map. These anchors are

used for predicting and regressing the targets, i.e., detecting objects within the image.

### .3.4 Loss Calculation

The loss function of YOLOv5 mainly consists of three components: bounding box loss (bbox_loss), classification loss (cls_loss), and confidence loss (obj_loss). The expression for the total loss is as follows:

$$Loss = box\_gain \times bbox\_loss + cls\_gain \times cls\_loss + obj\_gain \times obj\_loss$$

where box_gain, cls_gain, and obj_gain correspond to different loss weights. The default values for these weights are 0.05, 0.5, and 1.0, respectively.

**Bounding box loss**. YOLOv5 by default uses the CIoU (Complete Intersection over Union) metric to compute the bounding box loss. CIoU is an extension of DIoU (Distance-IoU), taking into account the aspect ratio of the bounding boxes, making the object box regression more stable. The calculation formula for the CIoU loss is as follows:

$$\text{CIoU} = \text{IoU} - \text{IoU}_{\text{CIoU}} + v(p, \text{gt})$$

where

$$\text{IoU} = \frac{\text{intersection area}}{\text{union area}}$$

$$\text{IoU}_{\text{CIoU}} = \frac{\text{center distance}^2}{\text{enclosing box diagonal distance}^2}$$

$$v(p, \text{gt}) = \frac{4}{\pi^2} \cdot \left( \arctan\left( \frac{\text{width}_{\text{gt}}}{\text{height}_{\text{gt}}} \right) - \arctan\left( \frac{\text{width}_p}{\text{height}_p} \right) \right)^2$$

**Classification loss**. YOLOv5 uses the binary cross-entropy function by default to calculate the classification loss. The binary cross-entropy function is defined as follows:

$$\text{Binary Cross-Entropy Loss} = -\left(y \log(p) + (1 - y) \log(1 - p)\right)$$

where **y** is the label corresponding to the input sample (1 for positive sample, 0 for negative sample), **p** is the predicted probability by the model that the input sample is a positive sample.

**Confidence Loss**. The confidence score of each predicted bounding box represents the reliability of that box. A higher score indicates a more reliable prediction and a closer approximation to the true bounding box. Regarding the confidence labels, previous versions of YOLO considered all grid cells containing objects (positive samples) to have a label value of 1, while the rest of the grid cells (negative samples) had a label value of 0. However, this approach led to the issue where some predicted boxes only surrounded the object without tightly enclosing it. Therefore, in YOLOv5, the confidence label for each predicted box is determined based on the CIoU between the predicted box and the ground truth box associated with the grid cell. It can be realized via the python code as follows:

$$tobj[b, a, gj, gi] = (1.0 - self.gr) + self.gr * score\_iou$$

where self.gr is the label smoothing coefficient. When self.gr is set to 1, the confidence label equals the CIoU.

## .3.5 Data Augmentation

Two data augmentation approaches are applied in YOLOv5-6.0, which are Mosaic and Mixup.

**Mosaic**. YOLOv5 continues to use the Mosaic data augmentation technique from YOLOv4 [27], which is an evolved version of the CutMix data augmentation method. As shown in Figure 8, the main idea behind Mosaic is to randomly select four images, perform random cropping on them, and then combine them into a single image for training data.

This technique helps improve the detection capability of the model, particularly in detecting occluded objects. By incorporating the Mosaic data augmentation, YOLOv5 enhances the model's ability to handle complex scenes and improve object detection performance.


FIGURE 8: Mosaic: a method of data augmentation [27]

**Mixup**. MixUp is a simple data augmentation method based on the idea of linearly combining the features and labels of two randomly selected samples to create a new training sample, as Figure 9 depicts. The main concept behind MixUp is to generate a new sample by taking a weighted sum of the features and labels of two randomly chosen samples. The formula can be described as follows:

$$x = \lambda x_1 + (1 - \lambda) x_2$$

$$y = \lambda y_1 + (1 - \lambda) y_2$$

where $x_1$ and $x_2$ represent two different input samples, $y_1$ and $y_2$ represent the corresponding labels for the two different input samples, and $\lambda$ represents the blending coefficient for the fusion of the two samples, which follows a Beta distribution.

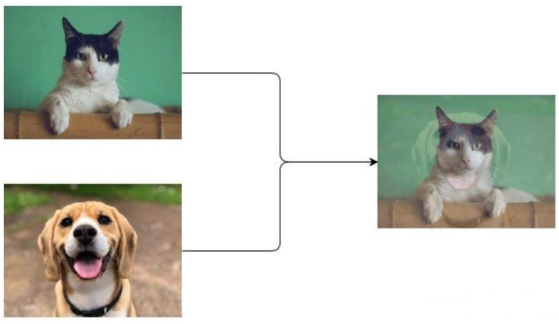

FIGURE 9: Simple Visualization of image mixup [36]