TECHNISCHE HOCHSCHULE LÜBECK

# Bachelor thesis

# Building Management – Determination of Roller Shutter Status


Submitted by: Xinchen Yang

Department: Electrical Engineering and Computer Science

Degree program: Information Technology

First examiner: Prof. Dr. Hänsel


Date handing out: 15th March 2023

Date handing in: 15th June 2023

A. Hanemann

(Prof. Dr. Andreas Hanemann)
Head of Examination Board

**Task description:**

Building management is one key aspect of facility management. It encompasses multiple disciplines to ensure functionality, comfort, safety and efficiency of the built environment by integrating people, place, process and technology. The major aspect of building management nowadays is energy management. Energy consumption of a building is hardly influenced by the windows and its roller shutters. This also has an effect on the energy harvesting of a building due to the sun light.

Efficient automated building management and monitoring should incorporate available information of the building status. This includes the window status and the roller shutter positions. A camera setup to monitor the roll shutter is available on the THL campus. The key idea is to monitor the shutter status w.r.t. open or closed. The primary sensor is the camera system. Machine vision algorithms should be used to provide the roller shutter status.

The task for this thesis is to investigate and implement an machine vision algorithm for determination of the roller shutter status. The implemented algorithm should be able to run 24/7. It should provide reasonable results at day and night and under different weather conditions.

In a first stage a literature review should be performed and a promising algorithm should be selected. A classic algorithm (digital signal processing) or machine learning and deep learning approaches may be used. Furthermore, the algorithm should be implemented and tested on a limited data set. The implementation should be done in python and OpenCV framework.

As prerequisite the registration of the buildings construction plan to each camera image is available or can be done manually. The estimation of the window positions should not be solved in this thesis.

TECHNISCHE HOCHSCHULE LÜBECK

**Department of**
**Electrical Engineering and Computer Science**
**Dean's Office - Board of Examiners**

**Statement on the bachelor thesis**

I assure you that I wrote the work independently, without outside help.

Only the indicated sources has been used in the preparation of the work.
The text or the sense of the text are marked as such.

I agree that my work may be published, in particular that the work may be submitted to third parties for inspection or that copies of the work may be made for forwarding to third parties.

13.06.2023
Date

Yang.Xinchen
Signature


# Building Management - Determination of Roller Shutter Status

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

TECHNICAL UNIVERSITY OF APPLIED SCIENCES LÜBECK

# *Abstract*

Department of Electrical Engineering and Computer Science

Bachelor of Science (B.Sc.)

**Building Management - Determination of Roller Shutter Status**

by Xinchen YANG

The detection of building components is crucial for accurately estimating energy usage in building management. While previous studies have utilized machine vision algorithms to detect doors, windows, and other components, there has been a lack of focus on roller shutters. Moreover, traditional machine vision algorithms face challenges in terms of running 24/7 and effectively handling occlusion. This bachelor thesis presents a novel approach for round-the-clock detection of roller shutter status and position using the object detection algorithm YOLOv5. In this research, we leverage the surrounding window glass to support status determination and employ fusion logic to achieve enhanced performance when multiple camera positions are deployed. Our detection platform is developed using Python, and the experimental results showcase an impressive overall detection accuracy of 97.13% and an Error in degree of closure (DoC) of 0.973 on our customized test dataset. The source code of this project is available at `https://git.mylab.th-luebeck.de/xinchen.yang/building-management-machine-vision`.

# *Acknowledgements*

As I'm approaching the end of my college journey, this bachelor thesis signifies a new beginning in my academic life. Over the past three months, I have dedicated all my efforts to this work, incorporating everything I have learned throughout my previous years, and finally I come to another crossroad of my life.

I would like to express my sincere gratitude to Prof. Dr.-Ing. Ralph Hänsel, my first instructor. Your invaluable ideas and guidance have been instrumental in helping me complete this task. I have also gained valuable insights into the research process from you. Your instructions on the appropriate structure, length, and content of an academic thesis have equipped me to become a novice scholar. I also appreciate the time you dedicated to our meetings, reading through our theses, and the effort you put into gathering the necessary dataset. Besides, you gave me a feeling that we are a team. You taught us the use of GitLab, which could be really useful tools in teamwork.

I would also like to thank my classmate and colleague, Tao Liu. You dedicatedly carried out the majority of the data acquisition work. I sincerely wish you obtain further success in your studies in the USA. I want to thank Bo Ya, who gave me the very first idea of deep learning and taught me some basics. Furthermore, I would like to extend my thanks to Prof. Chahabadi Djahanyar for serving as the second instructor for my thesis. Thank you for dedicating your time and effort to read my thesis and for attending the final oral defense.

Finally, I want to express my deepest love and gratitude to my parents. Your support and sacrifices gave me the opportunity to explore a new chapter of my life on the other side of the continent. During countless moments of stress and pressure, you provided me with unlimited patience and comfort, for which I am truly grateful. I also want to express my heartfelt regrets to my late grandpa. You expected me to become a top engineer, and I'm right on the path. It saddens me that you couldn't witness my graduation. Your absence is deeply felt, and I will always cherish the memories we shared.

This bachelor thesis is dedicated to all the extraordinary individuals I have encountered throughout my 23 years of life. May your future be illuminated with boundless success and joy.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 INTRODUCTION

## 1.1 Motivation

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 a 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.

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 roller shutter status and positions.

## 1.2 Goal and Task Requirements

The task can be clearly understood by referring to Figure 1.1. This bachelor thesis aims to investigate and implement in machine vision algorithm for the detection of the roller shutter.


FIGURE 1.1: A sketch of the task description [1]

The desired algorithm should be able to determine the roller shutter status and quantify the degree of closure. Since the estimation of energy consumption is a continuous process, the implemented algorithm should be capable of detection 24/7 and under various weather conditions. The implementation should be done in Python and OpenCV framework and is tested via a limited dataset.

This bachelor's thesis serves as the second phase in the image processing pipeline and focuses mainly on detection. The initial localization of windows in the building images is realized by another correlated bachelor thesis titled "Building Management: AI Mapping Image to Construction Plan" by Tao Liu.

## 1.3 Organization

This bachelor thesis consists of six chapters: In **chapter 2**, a comprehensive review of relevant literature and potential methods are provided. In **chapter 3**, we first analyze task challenges, then we evaluate different methods and algorithms to determine the most effective approach. In **chapter 4**, we describe how the selected algorithm is implemented in a Python environment. In **chapter 5**, the algorithm is tested on a limited dataset, and a detailed evaluation of the results is presented. In **chapter 6**, we provide a comprehensive summary of the research findings, draws meaningful conclusions, and outlines recommendations for future research.

# 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 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 previous work, Jin et al. applied a content-based object detection approach to window detection, which utilized gradient transforms to detect edges [2]. 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 [3]. 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 [4]. 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 [5]. 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 [6]. Biyomi et al. also used YOLOv5 for building envelope detection and compared its performance to other methods in a recent study [7]. 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 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 [8]. 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 [9].

The primary purpose of machine vision is to create models, data extracts and information from images [10]. 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 [11].

As shown in 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.

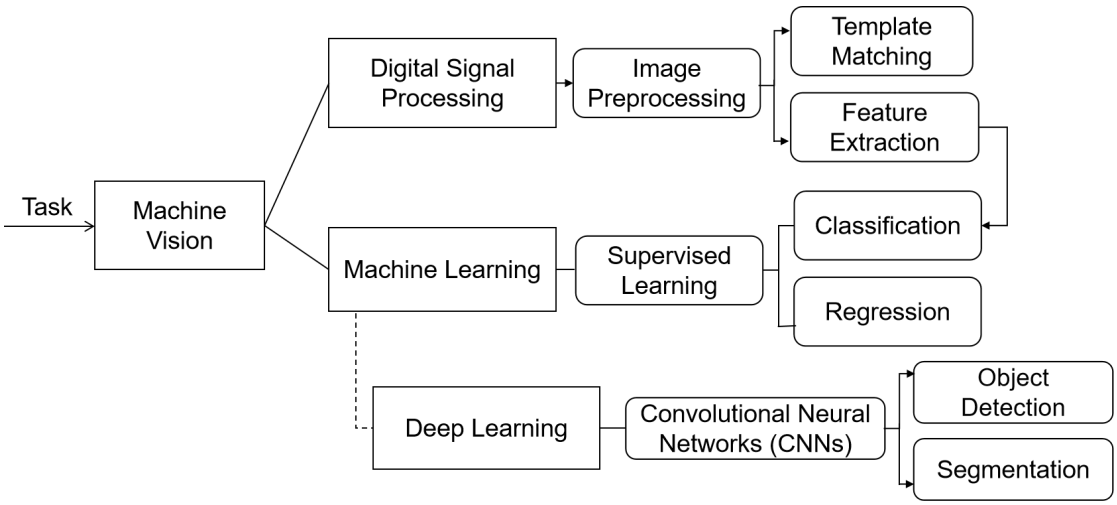


FIGURE 2.1: Methodology overview

## 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 a 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. The introduction of template matching is also involved in Appendix A.

### 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.

**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 [12] is a widely used gradient-based edge detection algorithm that can accurately detect edges in the image with good noise immunity and efficient performance. Figure 2.2 shows the effect of the aforementioned 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. Common feature extraction techniques include: Harris corner detector [13], scale-invariant feature transform

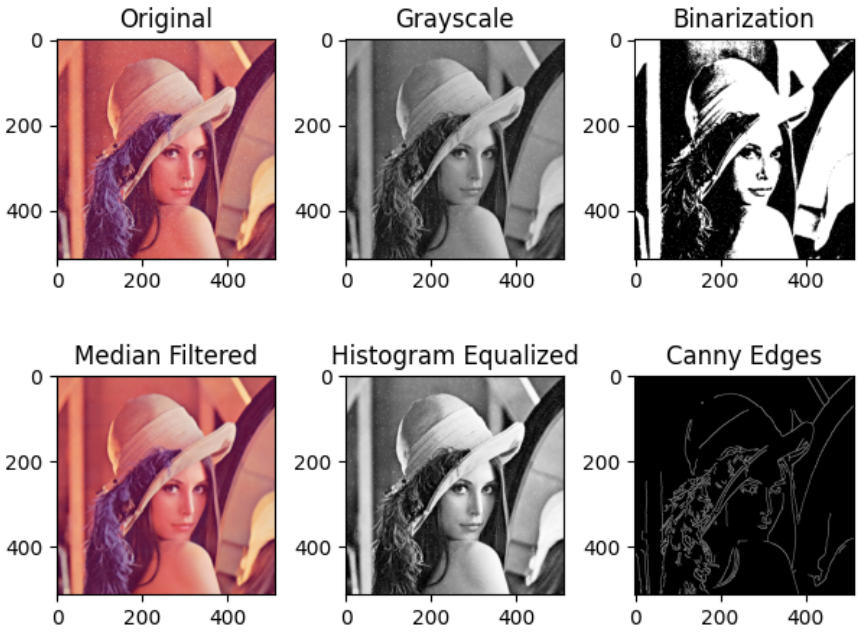


FIGURE 2.2: Effect of multiple preprocessing operations

(SIFT), Speeded-up robust features (SURF) [14] and Oriented FAST and rotated BRIEF (ORB) [15].

These feature descriptors have the shared objective of extracting meaningful and invariant features from images, enabling accurate object recognition and classification. They demonstrate robustness to scale, rotation, and other image variations while providing efficient and reliable feature extraction capabilities.

After feature descriptors are extracted, they can be 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 that rely on a collection of examples, no matter whether naturally or artificially generated [16].

### 2.4.1 Supervised learning

**Supervised learning** is one of the categories and the most common learning method of machine learning. Supervised learning aims to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes pre-labeled 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 that automatically assigns a discrete label to an unlabeled example, for example, assigning a given email to the "spam" or "non-spam" class [16].

A classification problem in machine learning is typically addressed through the use of classification learning algorithms. 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 B.

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:

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

### 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 [16].

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.

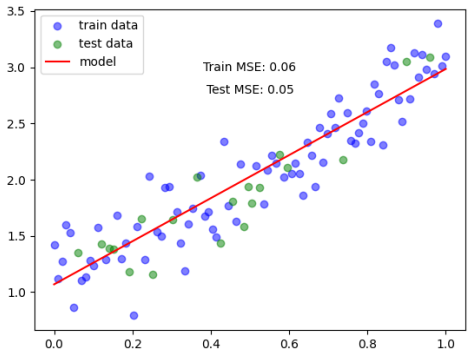


FIGURE 2.3: An example of a linear regression model

To minimize the loss of the model, **gradient descent algorithm** is used to obtain the best model parameters. It is an iterative optimization algorithm that calculates the partial derivatives (gradient) of the loss function concerning the model parameters and updates 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 traditional digital signal processing. However, rules still need to be designed manually for feature extraction.

## 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 that are heavily inspired by the 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 to optimize its final output.

Figure 2.4 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 and distributes 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 [17].

One of the largest limitations of traditional forms of ANNs is that when computing image data, they tend to struggle with computational complexity. This is

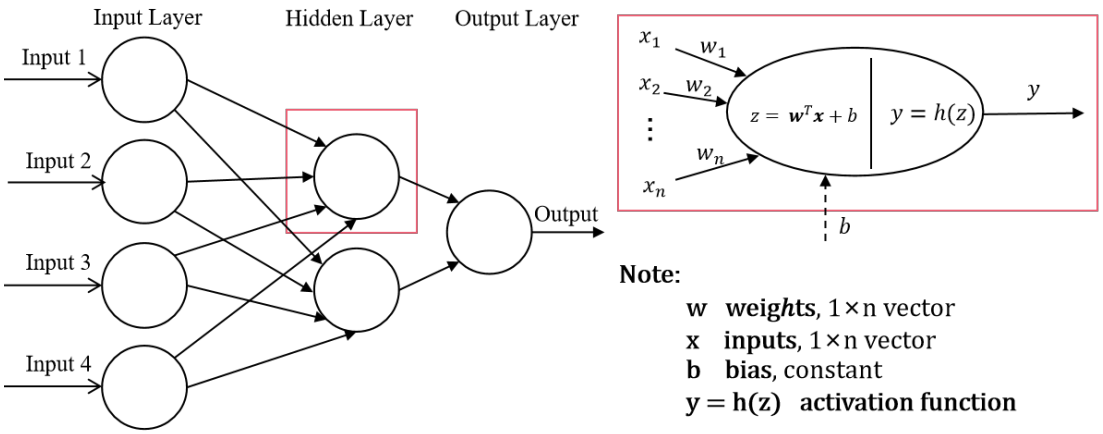


FIGURE 2.4: The architecture of a simple ANN network

because each neuron in fully-connected layers needs to be connected to all neurons in the previous layer, and this requires a lot of computation to deal with the huge number of weights each neuron provides for every single pixel. Besides, overfitting is another problem, 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. 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-optimize through learning. The notable differences between the 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.

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

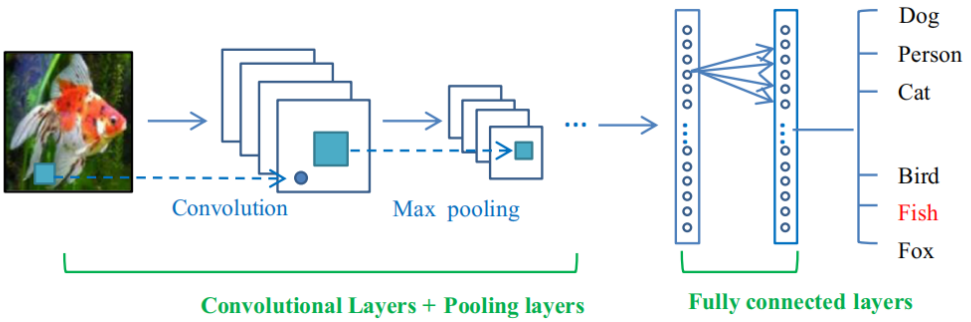


FIGURE 2.5: The pipeline of the general CNN architecture [18]

by convolution operations. The output of the convolution layer can be used as the input of the next layer.

In convolutional layers, learnable convolution **kernels** are used to extract features from the input image, producing an output feature map. Each neuron in the layer applies a kernel as a filter to extract specific features like edges or corners from adjacent input pixels. **Parameter sharing** allows the same filters to be applied across the entire image, significantly reducing the number of parameters needed. This feature helps prevent overfitting and improves the model's generalization ability.

During convolution, the kernel slides over the input image, calculating scalar products for each value and applying an activation function. Each kernel generates an activation map, and these maps are stacked to form the output volume of the convolutional layer along the depth dimension.

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 or average pooling, which take the maximum or average value in the pooling window as output, respectively.

Following several convolutional and pooling layers, **fully-connected layers** perform the high-level reasoning in CNNs. As the name implies, every neuron in a

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 [19].

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 a process is defined as forward propagation. However, the predicted value may have a deviation from the real one to some extent. To diminish such deviation, we need to use loss functions to calculate the deviation and update each parameter by **back propagation** to minimize the loss. Similar to the process introduced in the regression model training, the gradient descent algorithm is also used in backpropagation 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. It divides the feature map generated by a CNN into grids, where each grid predicts a fixed number of bounding boxes and their associated object class probabilities. The algorithm adjusts the category probabilities and position information of each bounding box to obtain the final predicted boxes. Non-Maximum Suppression (NMS) is then applied to remove redundant bounding boxes and obtain the final detection results [20]. Besides the YOLO series, Single Shot MultiBox Detector (SSD), and RetinaNet are other 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 common two-stage detection algorithms.

To evaluate the performance of the object detection model, **mean Average Precision (mAP)** is a commonly used metric that combines the precision and recall of the algorithm across different classes to provide a comprehensive assessment. To be more specific, mAP is the mean value of Average Precision (AP), which is obtained by plotting the precision-recall curve and calculating the area under the curve. mAP ranges from 0 to 1 and greater value may indicate better performance.

The value of mAP also varies under different IoU thresholds. IoU stands for Intersection over Union, its calculation is shown as follows:

$$\text{IoU} = \frac{\text{Intersection Area}}{\text{Union Area}}$$

where the Intersection Area refers to the area of overlap between the predicted bounding box and the ground truth bounding box, while the Union Area refers to the combined area of both bounding boxes.

As can be inferred, a larger IoU value may indicate a more precise prediction of bounding boxes, which is usually harder to achieve. A higher IoU threshold simply sets a higher demand that only the prediction whose IoU value is over the threshold could be considered correct.

Based on this, **mAP@0.5** and **mAP@0.95** (mAP at IOU threshold 0.5 and 0.95 respectively) are two commonly used metrics in the evaluation of object detection algorithms. While mAP@0.5 captures overall detection performance with a more relaxed threshold, mAP@0.95 focuses on the algorithm's ability to precisely localize objects and achieve high accuracy in challenging scenarios. The choice of which metric to use depends on the specific requirements.

If a pixel-level classification is required, **segmentation** is another technique in deep learning. Its introduction is involved in Appendix C.

# 3 CONCEPT

This chapter aims to identify promising approaches and solutions. A thorough analysis of the task challenges 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 Overview and Challenges

In this bachelor thesis, we deploy two cameras to detect the roller shutter status of a particular building facade (as shown in Figure 3.1). Also, the degree of closure (DoC) for the roller shutters should be quantified. The detection method needs to be able to run 24/7 under various weather conditions.


FIGURE 3.1: Example images from different camera positions

As the task requirements indicate, we should focus on both precise recognition and measuring. Based on this, three main challenges are identified and discussed in the following subsections.

### 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 the camera's performance may also degrade due to factors such as lighting and noise, thereby affecting the image quality. Figure 3.2 depicts how the image of the same window would differ under various conditions.


FIGURE 3.2: Images of the same window taken under different 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 does the roller shutter need to be recognized, but its degree of closure must also be calculated. Figure 3.3 describes examples of occlusion scenarios in this task. Addressing this challenge requires a highly demanding solution to mitigate the impact of occlusion.


FIGURE 3.3: Images with various occlusion scenarios

### 3.1.3 Measurement of the degree of closure (DoC)

To accurately determine the percentage of how much a roller shutter is lowered (In this bachelor thesis, **DoC, ranging from 0 to 100**, 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 Selection 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

Based on the previous literature review and extensive information search, it is evident that all three methods have their respective strengths and limitations. Table 3.1 provides an overview of how these three methods perform in addressing the task challenges and detailed analyses are followed.

TABLE 3.1: Performance of candidate methods in addressing task challenges

| Methods (→)<br>Challenges (↓) | Digital Signal Processing | Traditional Machine Learning | Deep Learning |
|---|---|---|---|
| Round-the-clock Detection | (+) high processing speed<br>(+) high interpretability<br>(-) manual feature extraction<br>(-) sensitive to noise | (+) high interpretability<br>(+) high generalization ability<br>(-) manual feature engineering<br>(-) training required | (+) automatic feature extraction<br>(+) high generalization ability<br>(-) training required<br>(-) low interpretability |
| Occlusion | (+) no training requirements<br>(-) complex modeling required<br>(-) sensitive to changes | (+) high interpretability<br>(+) high generalization ability<br>(-) manual feature engineering<br>(-) training required | (+) high adaptivity<br>(+) hierarchical feature learning<br>(-) training required |
| Measurement | Edge detection & template matching<br>(-) sensitive to environments<br>(-) complex template design | Regression algorithms<br>(-) training required | Neural networks & bounding box<br>(+) complex modeling capability<br>(-) training required |

**Note:** (+) indicates the strength and (-) indicates the weakness. All these evaluations are based simply on general rules and experience, and the evaluation may not be reasonable in other scenarios.

### 3.2.1 Selection criteria

Considering the task requirements and personal circumstances, the following criteria are used to select the most promising method:

**Feasibility**: Considering my limited understanding of machine vision and the limited time available for my bachelor thesis, I prefer a method easy to implement and doesn't require in-depth expertise in a specific field.

**Reliability**: This involves accuracy and consistency. We want the selected method to provide accurate status predictions and precise DoC values. Also, the detection performance should remain consistent and robust under different lighting conditions and occluded scenarios.

**Optimizability**: The chosen method should be easily tested and evaluated for future improvements. This implies that the results are presented in an understandable form, and its algorithms can be readily accessed and modified.

### 3.2.2 Decision making

**The Digital Signal Processing (DSP) method** requires designing several preprocessing steps to reduce the impact of lighting or noise. Feature extraction and modeling also remain a manual process, which can be challenging to accomplish without sufficient knowledge in the field. As for reliability, the DSP method provides precise outcomes for anticipated scenarios, but such rules can be hardly generalized to other unknown features due to environmental changes. In occlusion handling, we expect a rather poor performance in edge detection and template matching. The advantage of the DSP method is the high interpretability of its procedure and results, making it easy to maintain and optimize.

**The Traditional Machine Learning (TML) method** requires manual feature engineering, which demands specific knowledge and experience. Also, training is a time-consuming process required to obtain an effective classifier. However, it's able to classify or regress new and unknown patterns or scenarios by automatically learning the statistical patterns of the data, thus having a stronger

generalization ability to provide accurate and consistent results. Regression algorithms can be used to predict DoC values and can maintain a high level of interpretability when the relationship between inputs and outputs is linear.

**The Deep Learning (DL) method** enables automatic feature extraction in object detection. Its learning process is highly encapsulated, requiring no in-depth understanding of the network architecture. Despite the need for labeling and training, this method is highly feasible and easy to implement. During training, the DL method automatically extracts features and demonstrates better generalization to new data. It employs hierarchical feature learning, enabling it to accurately recognize objects even when partially occluded. The DoC measurement is achieved using neural networks and bounding boxes, which offer better performance in learning complex features. They also provide an intuitive and clear visualization of the target object's location and area. However, a challenge with deep learning processes is their black-box nature, which hinders the explanation of their decision-making process and makes further optimization difficult.

After thorough consideration, **the Deep Learning method** is selected for this bachelor thesis due to its remarkable feasibility and reliability. While in optimization, the deep learning method may pose more challenges in understanding its working principle compared to the DSP method, we believe that the potential benefits outweigh this trade-off.

## 3.3 Selection of Algorithms

In subsection 2.5.2, several one-stage and two-stage object detection algorithms were mentioned. To analyze the performance of these algorithms, two metrics are considered: Mean Average Precision at IOU threshold 0.5 (mAP@0.5) and Frames Per Second (FPS). Although real-time detection is not a requirement for the task, higher FPS could indicate faster processing times when handling a large number of static images and is therefore taken as a secondary metric.

This bachelor thesis aggregates the test results of each algorithm's best 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

TABLE 3.2: Performance of algorithms on COCO Dataset

| Algorithms | Proposal | Backbone | Size | FPS | mAP@0.5 |
|---|---|---|---|---|---|
| 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. All FPS measurements are conducted using a GTX 1080Ti or RTX 2080Ti GPU.

achieved a generally higher mAP and faster detection speeds compared to the other algorithms. Among the YOLO series, YOLOv5 maybe not be the latest version of the YOLO series, but it remains one of the state-of-art algorithms in the field of object detection, and more research is 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

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

YOLOv5 uses bounding boxes to localize the object. Although the performance is largely decided by its architecture, the ability to tackle specific challenges can be improved if the bounding boxes are used wisely and certain rules are created.

### 3.4.1 Solution for heavy occlusion

The goal of the current task is not only to recognize objects but also to provide precise measurements. Therefore, we do not want the model to sacrifice the accuracy by predicting the entire roller shutter in the presence of heavy occlusion. To address this, we split occlusion scenarios into two categories: **Mild occlusion**, which the model should be robust enough to "see through", such as sparse branches; **Heavy occlusion**, which should be excluded by the model, such as building components, sculptures, and heavy leaves, 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 is set as "open", "closed" or "blocked". This gives rise to another problem when we define object classes. If we set shutter as the only object class, the misdetection of roller shutter may indicate two indistinguishable possibilities: "open" and "blocked". Therefore, this bachelor thesis also detects window glass, which can help determine the status.

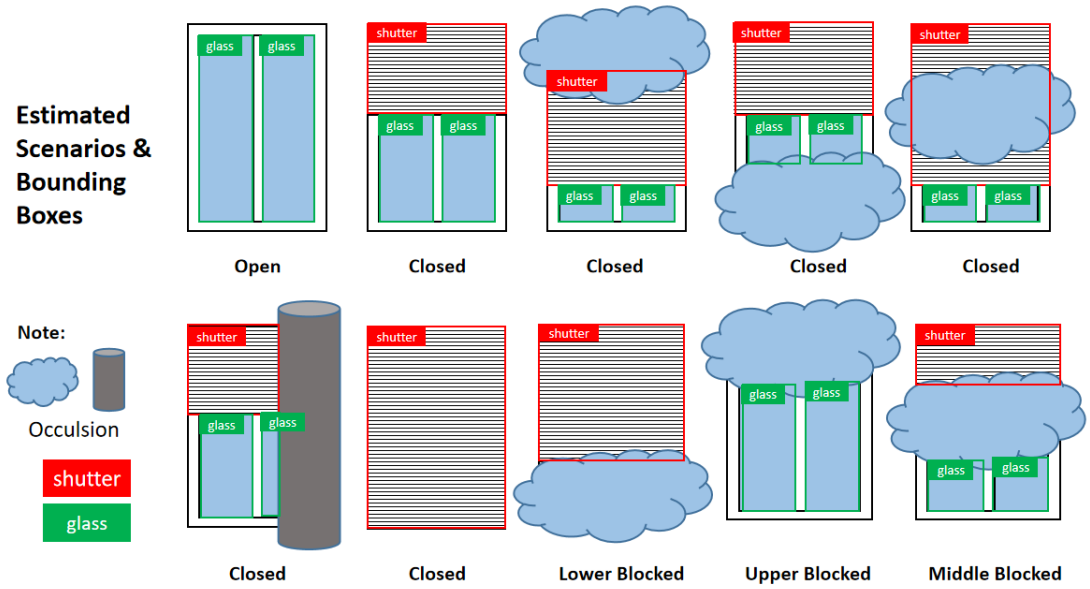


FIGURE 3.4: Estimated scenarios and bounding boxes with two object classes

As shown in Figure 3.4, ten scenarios are used to simulate the roller shutter status in reality. In some scenarios, although strong occlusion exists in the images,

it does not influence judging roller shutter status and DoC. This is because we only need to find out the demarcation between roller shutters and glass.

To distinguish these scenarios, a decision tree is designed as Figure 3.5 depicts. Two parameters of the bounding boxes: **S3** and **G1** are compared after those scenarios are first split into four branches at based on the detection results:

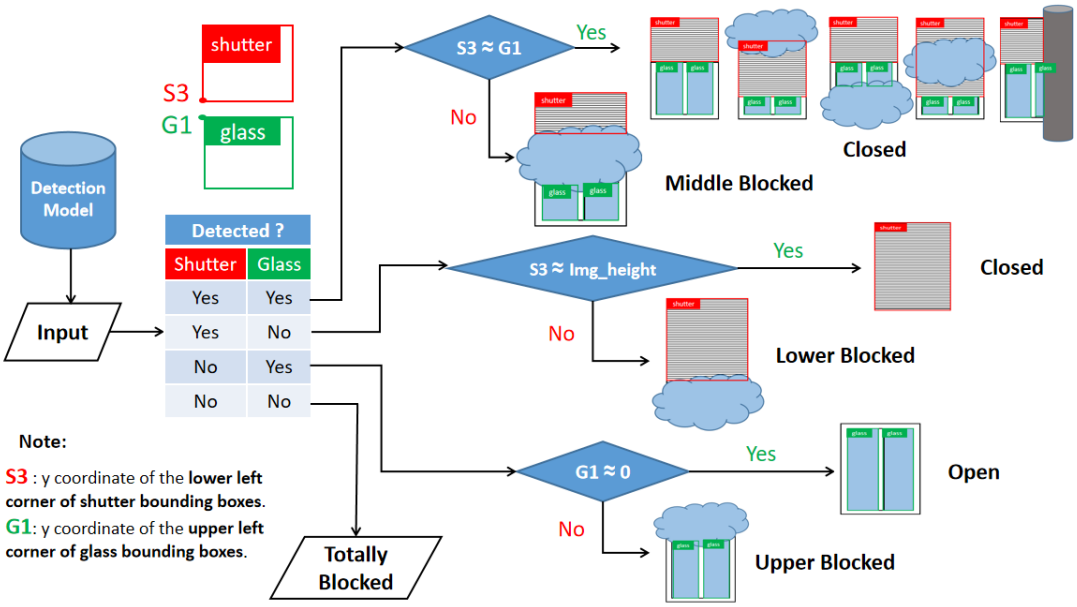


FIGURE 3.5: A decision tree to determine the roller shutter status

**(1) Both the roller shutter and glass are detected:** We need to determine if the roller shutter is closed or blocked. By comparing G1 and S3, if they are approximately equal, it indicates that the demarcation between the roller shutter and glass is clear, and the status is closed. Otherwise, if there is an obstruction in the middle, the DoC would remain unclear and we consider it blocked.

**(2) Only the roller shutter is detected:** We cannot determine if the window glass is fully covered by the roller shutter or partially occluded. Therefore, we compare S3 with the height of the image, which ideally should be equal to the height of the image. If two values are approximately equal, the roller shutter is considered fully closed, otherwise, it is considered blocked in the lower part, and the exact degree of closure cannot be determined.

**(3) Only the glass is detected:** Without the roller shutter present, we cannot assume it is open. There might be an occlusion covering the upper part of a

roller shutter. To address this, we examine the value of G1. If G1 approaches 0, indicating that the upper parts of the image are unlikely to be the roller shutter, we classify the status as open. If G1 is far from 0, it is uncertain whether the area above G1 corresponds to glass or the roller shutter, and we mark the status 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.

In practice, "approximately equal to" cannot be understood by the computer. Therefore, three thresholds are defined to be applied in practice, as is shown in table 3.3. The preset values of these thresholds need to be evaluated in practice.

TABLE 3.3: Definition of thresholds set to determine status

| Threshold | New expression | Replaced expression | Preset value |
|---|---|---|---|
| $TH_{middle}$ | S3-G1 $<TH_{middle}$ | S3 $\approx$G1 | 50 |
| $TH_{lower}$ | S3 $>TH_{lower}$ | S3 $\approx$img_height | 500 |
| $TH_{upper}$ | G1 $<TH_{upper}$ | G1 $\approx$0 | 100 |

**Note:** The preset image height is 640.

### 3.4.2 Measurement of DoC

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

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

The measurement of DoC can be achieved using bounding boxes due to their similar shape to roller shutters. However, bounding boxes are always upright rectangular, while the roller shutter in the image may be distorted due to shooting angles. Therefore, before detection, the cropped image of each window needed to be set upright. As we assume the cropped images include only target windows, the formula can be simplified as:

$$\text{DoC (\%)} = \frac{\text{Area of the Bounding Box}}{\text{Image Size}} \times 100(\%)$$

As the width of roller shutters and windows is equal, we can take only height into measurement. However, using the height of the bounding box may be inappropriate when dealing with occlusion problems, where the bounding box only represents the unobstructed part of the roller shutter. Therefore, we apply the y coordinate of the lower left corner of shutter bounding boxes (**S3**) to represent the height of the roller shutter. The final optimized formula is described as:

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

As the image height is fixed, the formula has only one variable S3. This makes the measurement method more resistant to errors. However, its validity relies on the assumption that each detection image contains only a complete window without any vertical offset or background. Although this trade-off may have strict requirements in localizing windows, it can greatly reduce error rates.

### 3.4.3 Fusion

In this bachelor thesis, multiple cameras are set up to capture images of the same building facade from different angles. This allows us to mitigate the impact of heavy occlusion and obtain more precise results of roller shutter status and DoC.

As shown in Figure 3.6, we considered all 19 possible scenarios using at most 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, and average of non-zero DoC 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 | ? | ? |

**Note:** $\mathbf{DoC_{closed}}$ is the DoC value of the roller shutter with status "closed".

FIGURE 3.6: All scenarios for up to three cameras and pre-defined fusion results

The 19 possible scenarios discussed earlier can be summarized and generalized, as shown in Figure 3.7, based on the fusion results obtained through the pre-defined 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

| | | | | | |
|---|---|---|---|---|---|
| Camera 1 | 10% | 10% | 10% | 15% | 0%-20% |
| Camera 2 | 0% | 0%-10% | 15% | 0% | 5%-30% |
| Camera 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 3.7: 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. The Python project and dataset have been uploaded on GitLab[1].

## 4.1 Process Handover and Overview

This bachelor thesis mainly focuses on detecting the status and DoC of roller shutters, while the localization of window position has been assigned to Tao Liu. For more information regarding window localization, please refer to his thesis. This section will give a brief introduction to 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 DoC of each window, in an Excel file.

To enable an end-to-end process, a roller shutter detection platform is set up using the Pyside2 package and the Qt Designer tool in Python, and the YOLO

[1] https://git.mylab.th-luebeck.de/xinchen.yang/building-management-machine-vision

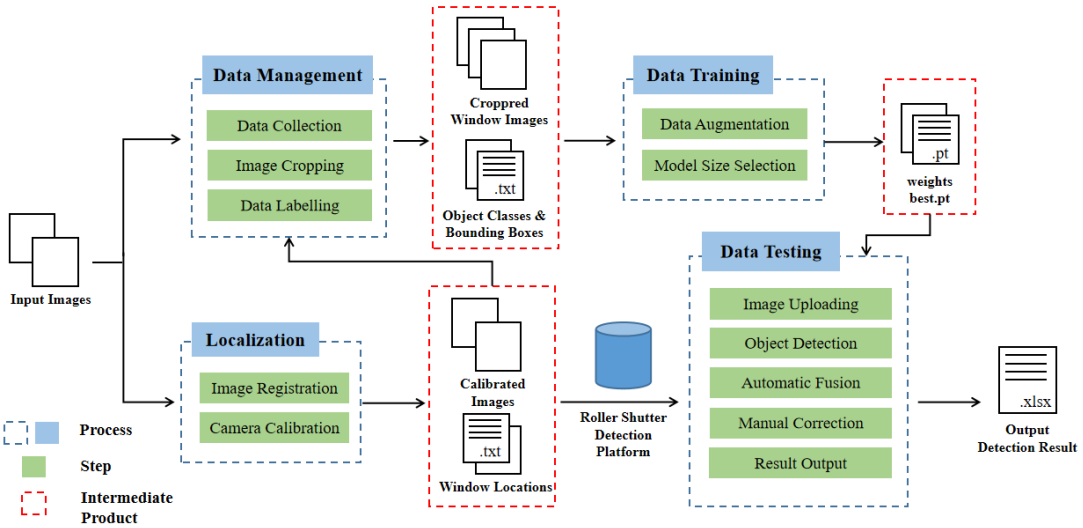


FIGURE 4.1: An overview of the workflow

model is loaded for detection. Before loading the model, data management and data training are two indispensable steps to obtain the necessary model weights. The following sections will provide a detailed explanation of each process.

## 4.2 Data Management

Data Management involves the acquisition of raw images and several preprocessing steps to build the final dataset for training and testing.

In this bachelor thesis, 228 building images taken by two video cameras set on the THL campus are selected. These images are taken from 4:00 to 22:40 on the 4th and 16th of May, 2023. Each image selected has a time interval of at least 20 minutes and contains information on the same 44 windows located at the left part of the building. The detailed separation of the data is shown in Figure 4.2. As can be seen, to ensure the differences between the test and training sets, a cross-selection is applied. After the training set and test set are initially divided, these images are calibrated into undistorted images and then cropped according to the window location data from Tao's work. Those cropped window images will then be labeled for training, validation, and testing.

During the labeling process, two object classes are defined, which are "glass" and "shutter". As some window images can be hardly recognized due to the poor image qualities, to ensure the preciseness of the detection model, these images are excluded from the data set. Therefore, the total number of original window images involved in the training process is deducted to 818 for validation and 3826 for training (7652 after data augmentation).

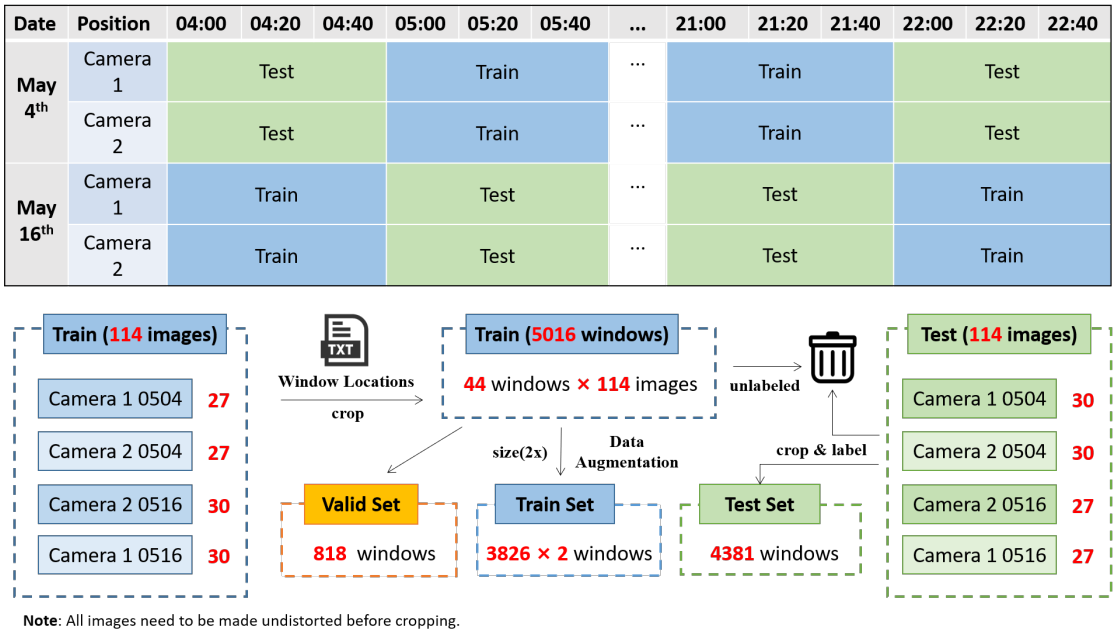


FIGURE 4.2: Selection and separation of the dataset

## 4.3 Data Training

After labeling, a text file recording the object classes and bounding box coordinates is generated for each cropped image. Before the training started, the model size of YOLO needs to be decided, and customized data augmentation would also be an option. The services of labeling and data augmentation are provided by **roboflow**[2], a computer vision platform for data preprocessing, annotation, and model training.

[2]https://universe.roboflow.com

### 4.3.1 Data augmentation

To improve the performance of the detection model in different lighting conditions and make it robust against occlusion. Two data augmentation methods are applied to expand the training set, which are adjusting brightness and cutout. The augmentation process is shown in Figure 4.3.

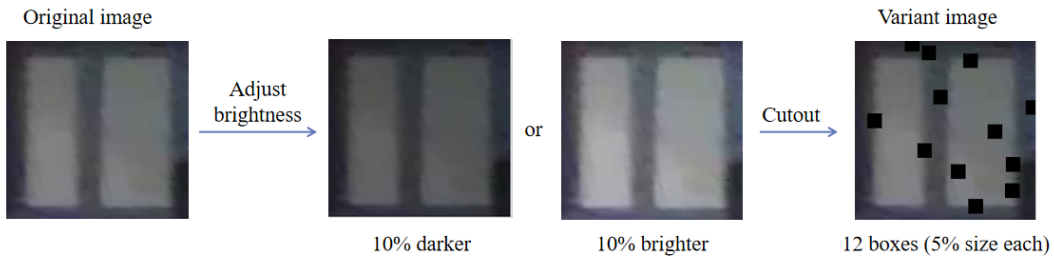


FIGURE 4.3: Data augmentation process applied on each original image

As can be observed, a variant is produced based on each original image in the training set. Firstly, the original image is randomly adjusted to be 10% brighter or 10% darker. This helps simulate the lighting condition at different times, thereby improving the performance of round-the-clock detection. After that, the cutout is applied on the adjusted images, to randomly dig out 12 small boxes with 5% the size of image each. This process can improve the robustness of the model against mild occlusions. Finally, the variant images are added to the final training set. As each original image would produce a variance, the size of the training set after data augmentation will be doubled to 7652 images.

### 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.

### 4.3.3 Training outcome

The training process will start by running the **train.py** in the official YOLOv5 package. After training for 200 epochs, the weights that achieved the best performance on the validation set were saved as "best.pt", which can be loaded during the detection process. The hardware information and training outcomes can be accessed on GitLab[3]. Here we only investigate the training performance, as shown in Figure 4.4.

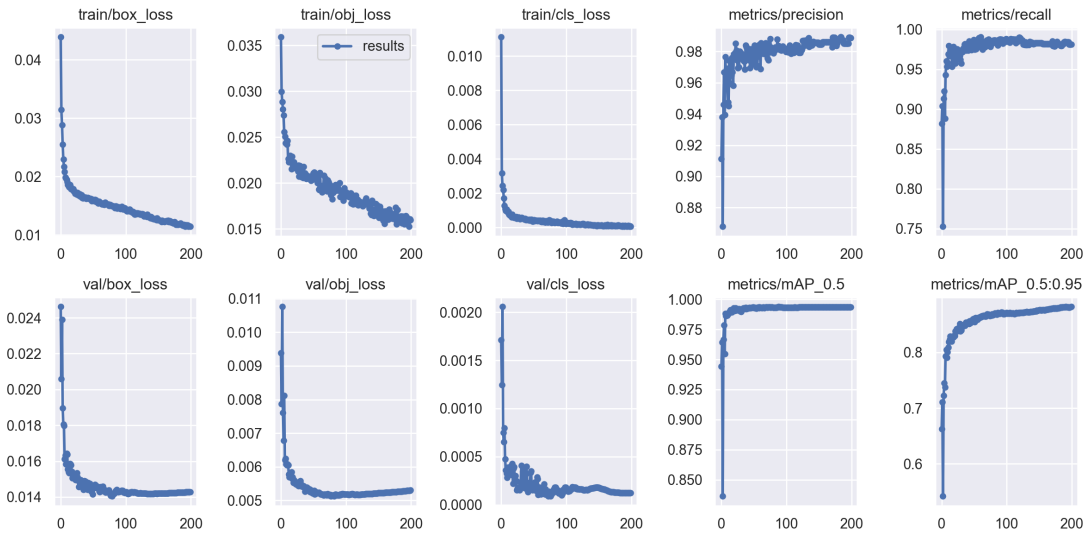


FIGURE 4.4: Training results of 200 epochs

As observed, the training losses significantly decrease to a very low level after 200 epochs. However, there is a slight increase in the validation loss during the later epochs, suggesting the possibility of overfitting if training continues. Therefore, based on this observation, it can be considered that 200 epochs are sufficient for training the model.

Notably, the model achieved an mAP@0.5 of 0.994 and an mAP@0.95 of 0.885 on the validation set. These high mAP scores indicate that the model's training is successful and the weights obtained are suitable for further detection.

[3] `https://git.mylab.th-luebeck.de/xinchen.yang/building-management-machine-vision/-/tree/main/Training%20outcome`

## 4.4 Data Testing

In this bachelor thesis, a platform for detecting roller shutters is designed as the main contribution. The project is implemented using PySide2 and Qt Designer. The traditional Model-View-Controller (MVC) structure is also applied to ensure the separation between the logic module and GUI, as well as the extensibility and maintainability provided by reusable methods. The MVC structure of this project can be seen in Figure 4.5.

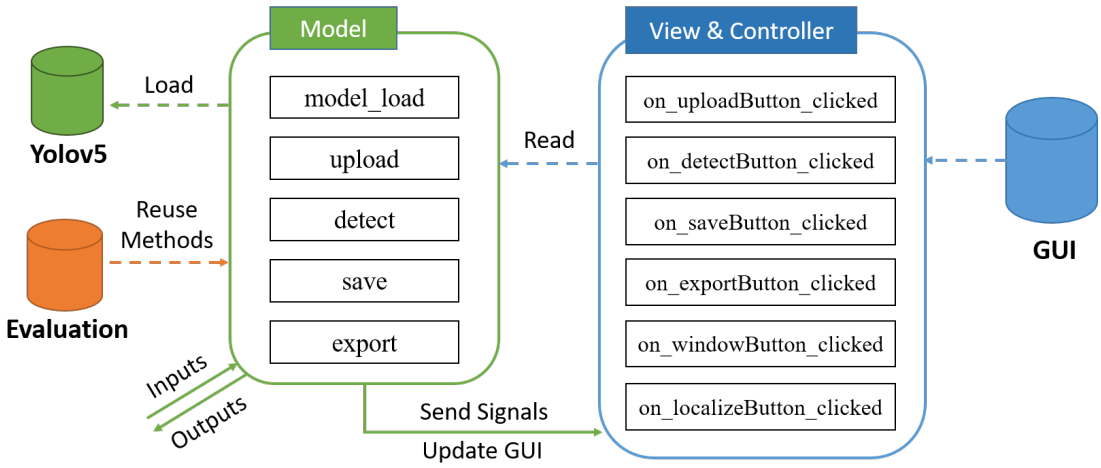


FIGURE 4.5: The MVC structure applied in the project

As shown, the model block contains all the detection-related methods, which can also be reused in the next evaluation section. The controller block and view block are combined as a single entity in PySide2 to update the GUI, which represents the roller shutter detection platform shown in Figure 4.6. The following subsections will introduce its features and functional blocks.

Figure 4.7 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.

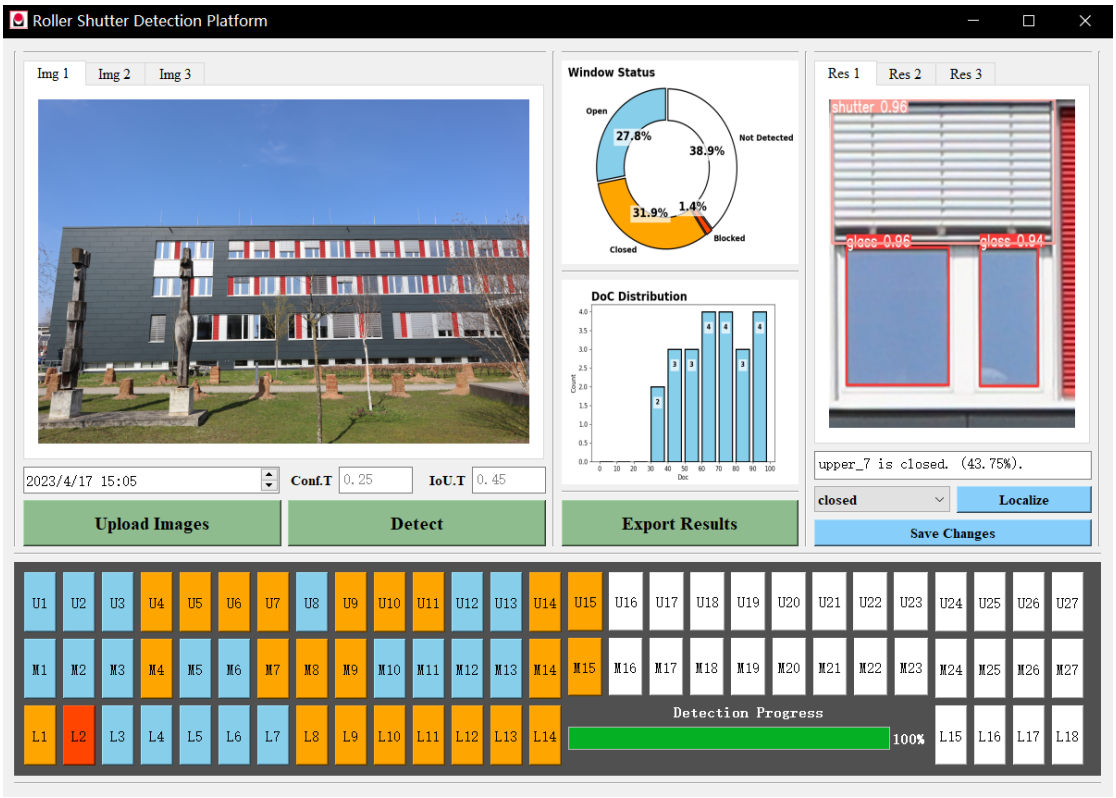


FIGURE 4.6: The GUI of roller shutter detection platform

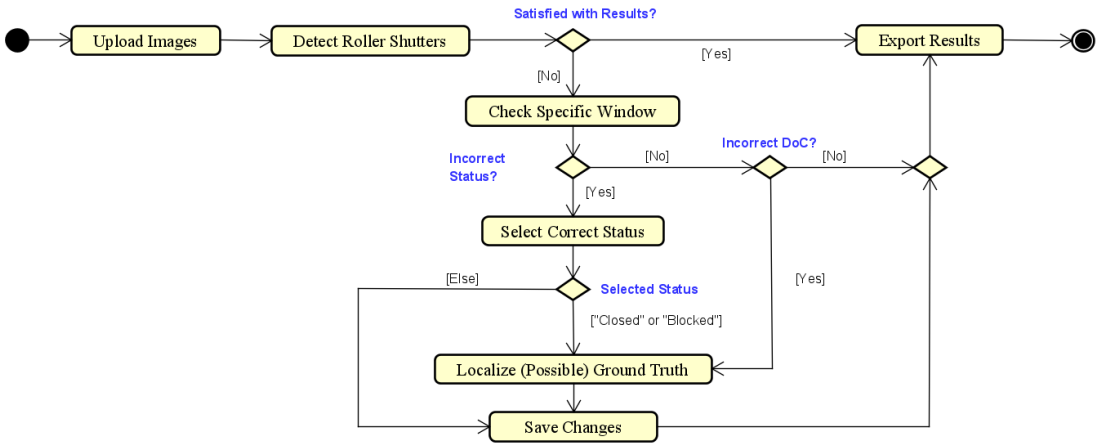


FIGURE 4.7: An overview of the workflow

### 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 4.8 illustrates how the uploaded images are processed.

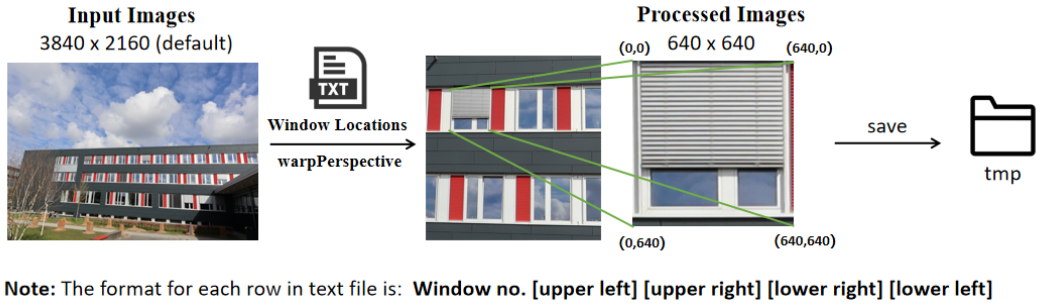


FIGURE 4.8: Preprocessing of images uploaded

**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 possible 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 standard text file 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.

**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 we apply a **warpPerspective** method in OpenCV to map the coordinates onto the four corners of a $640 \times 640$ image. After that, images are saved in the temporary file, ready for the detection process.

### 4.4.2 Roller shutter detection

The detection process involves iterating through images in the temporary file, detecting glass and shutters, and drawing bounding boxes around them. In the GUI, two detection parameters can be set accordingly: the confidence threshold and the NMS IoU threshold. The images with the bounding boxes are then saved in another file named "result". During the detection process, two operations are performed simultaneously: **Status Display** and **Fusion**. Figure 4.9 illustrates how these two operations are applied during the detection period.

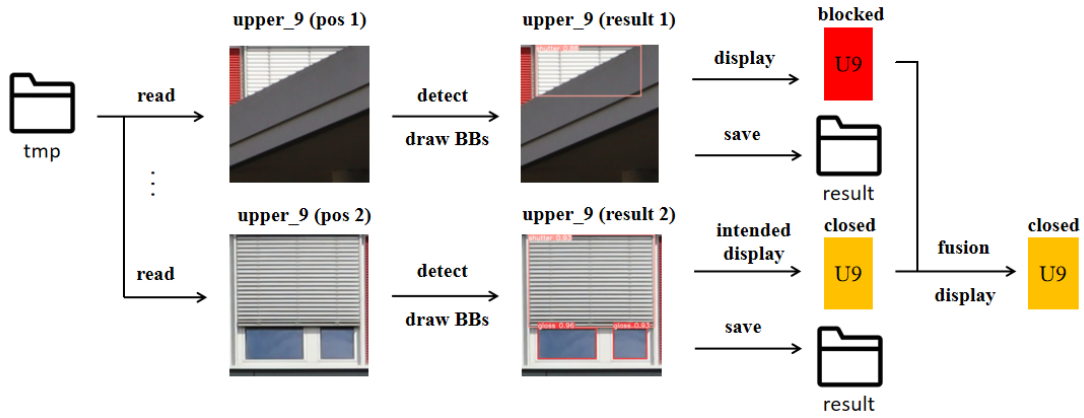


FIGURE 4.9: Display and fusion during detection

**Status display**: This platform visualizes the detection process using a progress bar and 72 buttons (located in the lower half of the GUI), representing the 72 windows. The determination of roller shutter status follows the decision tree explained in subsection 3.4.1. After the detection of each image completes, the detected roller shutter status is displayed by changing the background color of the corresponding button: blue for open, orange for closed, red for blocked.

**Fusion**: During the detection period, the status of the same roller shutter may change multiple times. This occurs as a result of fusion when different results are obtained from different camera positions. These changes are handled using simple if-else statements and based on the fusion rules introduced in subsection 3.4.3. In practice, Fusion operations are applied each time when a different detection result for the same window is obtained. Upon completion of the whole detection process, two graphs are displayed to illustrate the distribution of status, and DoC for the closed roller shutters, respectively.

### 4.4.3 Result modify

Users can view the detailed results of a specific window by clicking the corresponding button. If any modification is needed, users can select the modified status and localizing the ground truth on current window image. By clicking the "Save Changes" button, the detection results will be updated. Figure 4.10 demonstrates how users can check and modify the results.

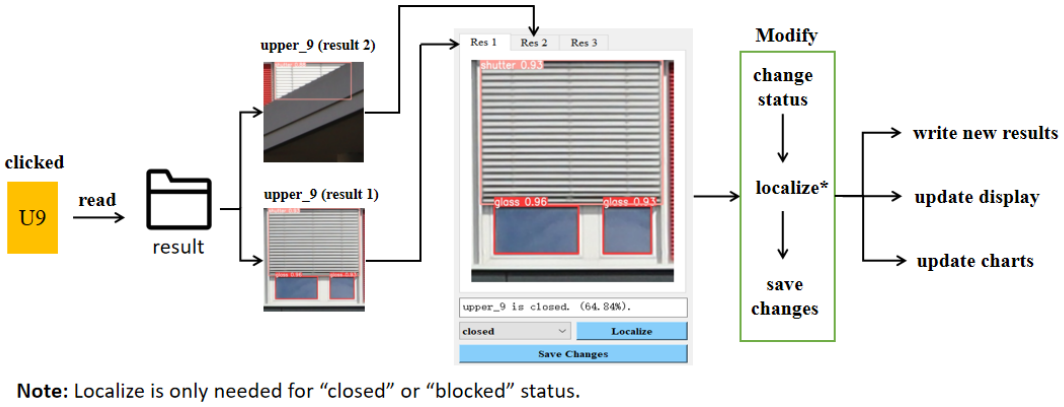


FIGURE 4.10: Checking and modifying detection results

The "Localize" function allows users to set the modified DoC for the "closed" and "blocked" roller shutter statuses. To be more specific, Users can precisely determine the location of the ground truth for the "closed" status by selecting an image and clicking on the demarcation between the roller shutter and the window glass. However, for "blocked" cases, users can only estimate a range by clicking twice to determine the maximum and minimum of possible DoC.

### 4.4.4 Report export

This detection platform supports exporting the detection results in the form of Excel files. It is realized by using the **openpyxl** package in Python. 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

In this chapter, we evaluate the performance of the detection model on the test set and in practical implementation. After that, we examine the suitability of the task premises and identify existing limitations.

## 5.1 Performance of the Detection Model

The evaluation of the detection model is conducted using the cropped test dataset, which consists of 4381 labeled window images. The evaluation is performed by running the val.py script provided in the official YOLOv5 package. In this section, a comparison will be made between the model with data augmentation and the one without.

### 5.1.1 Evaluation metrics

As introduced in section 2.5.2, the mean Average Precision (mAP) is a commonly used metric for evaluating object detection algorithms. In this bachelor thesis, we not only expect the model to detect all objects without missing them but also to provide precise localization. Therefore, we have chosen two metrics, namely

**mAP@0.5** and **mAP@0.95**, to assess the basic detection requirements and the advanced precision requirements, respectively.

### 5.1.2 Results analysis

The evaluation results of the model are shown in the following Table 5.1 and 5.2. To examine the influence of the data augmentation as well, the model trained without data augmentation is also included as a contrast.

TABLE 5.1: Evaluation of model with data augmentation

| Class | Images | Labels | Precision | Recall | mAP@0.5 | mAP@0.95 |
|---|---|---|---|---|---|---|
| All | 4381 | 9738 | 0.887 | 0.839 | 0.907 | 0.709 |
| Glass | 4381 | 7536 | 0.936 | 0.892 | 0.945 | 0.73 |
| Shutter | 4381 | 2202 | 0.837 | 0.786 | 0.87 | 0.689 |

TABLE 5.2: Evaluation of model without data augmentation

| Class | Images | Labels | Precision | Recall | mAP@0.5 | mAP@0.95 |
|---|---|---|---|---|---|---|
| All | 4381 | 9738 | 0.767 | 0.812 | 0.822 | 0.608 |
| Glass | 4381 | 7536 | 0.846 | 0.925 | 0.95 | 0.731 |
| Shutter | 4381 | 2202 | 0.688 | 0.699 | 0.694 | 0.484 |

As observed, the model with data augmentation achieved a higher mAP@0.5 and mAP@0.95 for all classes, reaching 0.907 and 0.709. These high values indicate that the model can accurately detect objects as well as provide precise locations for the majority of cases. Although metrics for the shutter class are relatively lower, it is still acceptable. This suggests that the detection model performs better in predicting the glass class compared to the shutter class.

On the other hand, the model without data augmentation shows a slight increase in both two metrics for the glass class, but it performs far poorly in detecting the shutter class. This deficiency would hinder the detection of shutters and further impact the fusion process. Additionally, the average number of labels for the glass class per image is nearly three times that of the shutter class. However, only one correctly detected glass label per image would be sufficient to support

the status determination. Therefore, the metrics for the shutter class are more critical. Considering this trade-off, it is not deemed worthwhile.

Overall, the model exhibits significant improvement after applying data augmentation, making it suitable for practical implementation.

## 5.2 Performance of Practical Implementation

The performance of detection in practice is enhanced through the use of a decision tree for analyzing the roller shutter status and a fusion logic introduced in the Concept chapter. To evaluate the practical performance, the original test set of 114 undistorted building images is used. Also, images with the same taken time and different camera positions are grouped in pairs to evaluate the fusion performance. This evaluation process involves running the evaluation.py[1] script, which is a part of the project and designed to output results that assess the effectiveness of the detection model on the dataset in practice.

### 5.2.1 Setting of the ground truth

According to the task requirements, we mainly focus on obtaining two parameters of each roller shutter at a given time, which are:

1. the status of the roller shutter
2. the DoC (degree of closure) of the roller shutter

Although the labeling process on the cropped test data set can provide the ground truth of each roller shutter position, these bounding boxes may not make sense unless we examine each bounding box to determine the status and DoC manually based on human cognition. Therefore, the status and DoC ground truth is set manually for all 44 windows in every tested building image and it is based on two premises:

[1] `https://git.mylab.th-luebeck.de/xinchen.yang/building-management-machine-vision/-/blob/main/detection_platform/evaluation.py`

1. The localization of window positions is precise, which means the DoC calculation formula itself is error-free.
2. The limitation of human cognition is considered, which means "blocked" status would be involved in ground truth when the status cannot be inferred from the window images.

The ground truth information is then saved as text files. The format of each line is like "**window_no status DoC**" (e.g., "upper_1 closed 89.54"). Each line is compared with the corresponding ground truth during the detection process of the evaluation module and then output the results to calculate the evaluation metrics mentioned next subsection.

### 5.2.2 Evaluation metrics

Based on the two desired parameters, two evaluation metrics are set accordingly to evaluate the performance of the detection platform in obtaining these two parameters, which are: **Detection Accuracy** and **Error of DoC**.

**Error of DoC** is used to examine the preciseness of localizing roller shutters. It calculates the absolute difference between the predicted DoC and the ground truth DoC of "closed" roller shutters. The calculation formula can be shown as follow:

$$DoC_{\text{error}} = |DoC_{\text{pred}} - DoC_{\text{gt}}|, \text{ if status = "closed"}$$

**Detection Accuracy** is used to examine the accuracy of classification. It is the proposition of **correct predictions** which fulfills both the two conditions below:

1. The predicted status is the same as the recorded ground truth status.
2. The Error of DoC (if exists) is no greater than 10.

The calculation formula can be expressed as follows:

$$NP_{\text{correct}} = NP_{\text{correct status}} - NP_{DoC_{\text{error}}>10}$$

$$\text{Detection Accuracy} = \frac{NP_{\text{correct}}}{NP_{\text{total}}} \times 100\%$$

where $NP$ represents the Number of Predictions.

### 5.2.3 Results overview and factors analysis

Here we present the performance of each camera position and the overall performance after fusion, which are shown in the following Table 5.3.

As can be observed, the overall performance after fusion surpasses the individual performance of any single camera position. The fusion approach achieves a detection accuracy of 97.13% with an average error of DoC of only 0.937. These results indicate that the fusion process has significantly enhanced the performance. Although a lower rate of correct "blocked" status detected is also observed, it has little impact on real-world scenarios since the "blocked" ground truth will be minimized with the optimization of camera position.

Furthermore, a notable difference is observed between the two camera locations, with camera 1 exhibiting much worse performance compared to camera 2. This disparity can be attributed to the higher number of incorrect "blocked" predictions for "closed" windows in camera 1. Such misdetections may result from the occlusion caused by dense leaves and branches, which are prevalent in real-world scenarios.

Based on the analysis presented in the Concept chapter, round-the-clock detection is another challenge for the given task. While the significant impact of occlusion on the performance across different camera positions has already been observed, the influence of varying lighting conditions remains unknown. Thus, an analysis of each time slot to examine the performance changes throughout the day is also conducted.

Figure 5.1 depicts how the detection accuracy and error of DoC would differ throughout the day. The fusion results (green line) show a more stable and better performance over the other two camera positions. To examine the impact of lighting conditions on the final detection results. In Figure 5.2 we only picked out the fusion result as a reference.

TABLE 5.3: Performance of the detection platform before and after fusion

| Camera Position[2] | Camera 1 | | | Camera 2 | | | Camera 1 & 2 (Fusion) | | |
|---|---|---|---|---|---|---|---|---|---|
| Ground Turth (→) Prediction (↓) | Open | Closed | Blocked | Open | Closed | Blocked | Open | Closed | Blocked |
| Open | 1021 | 20 | 0 | 1122 | 17 | 13 | 1140 | 15 | 12 |
| Closed | 3 | 989 | 7 | 2 | 1239 | 0 | 4 | 1272 | 7 |
| Blocked | 120 | 305 | 43 | 20 | 58 | 37 | 0 | 27 | 31 |
| Correct Status (%) | 89.25% | 75.27% | 86.00% | 98.08% | 94.29% | 74.00% | 99.65% | 96.80% | 62.00% |
| Correct Status (Sum) | | 2053 | | | 2398 | | | 2443 | |
| Error of DoC >10 | | 7 | | | 2 | | | 7 | |
| Correct Prediction | | 2046 | | | 2396 | | | 2436 | |
| Detection Accuracy | | 81.58% | | | 95.53% | | | 97.13% | |
| Error of DoC (Average) | | 1.645 | | | 1.098 | | | 0.937 | |

**Note:** All detection results are obtained under a default setting of confidence threshold 0.25 and NMS IoU threshold 0.45.

The detailed evaluation results can be found at https://git.mylab.th-luebeck.de/xinchen.yang/building-management-machine-vision

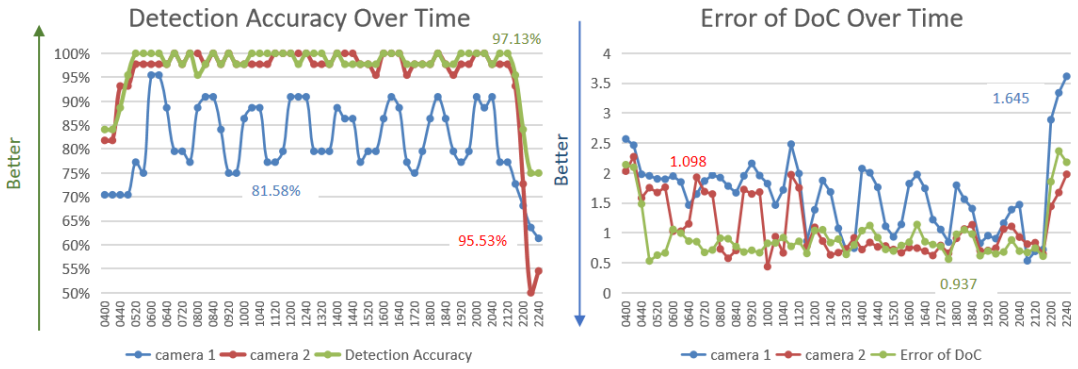


FIGURE 5.1: Detection accuracy (left) and error of doC (right) over time

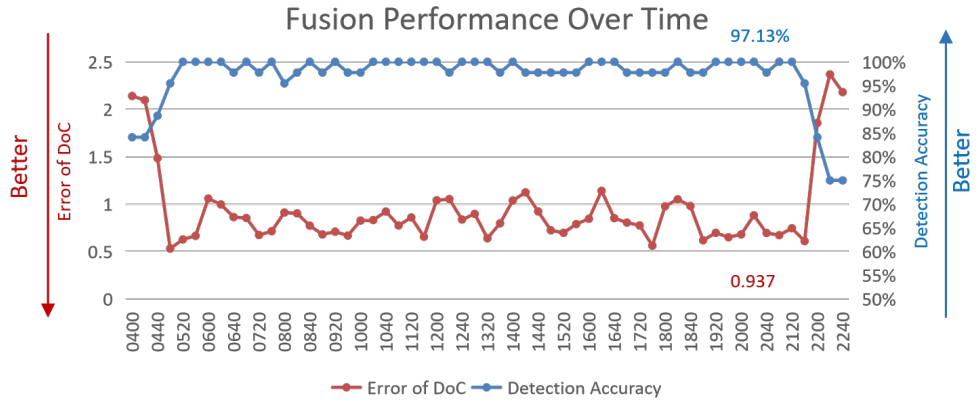


FIGURE 5.2: The overall performance of fusion over time

As can be observed, the detection accuracy remains high and stable from 6:00 to 22:00 but is decreased dramatically before and after that period. The error of DoC shows a reverse trend, but it also indicates the truth that the performance is getting worse. This means after getting completely dark during the night, the detection platform would perform badly with decreasing detection accuracy and increasing error of DoC. Although the detection model turns out to be precise during the day, there is still much room to develop for scenes with poor lighting conditions.

Overall, the evaluation results show improved performance in both metrics after fusion. Although there are noticeable effects from different camera positions and lighting conditions, which indicates the task challenges still maintain a considerable influence on detection performance, the current model has significantly

mitigated their impact and achieved promising results.

## 5.3 Discussion

This section aims to evaluate the assumptions and preset values used during the implementation, based on the results of the evaluation. It will also address the existing limitations of the current detection model.

### 5.3.1 Assumptions and preset values assessment

To obtain the current evaluation results, assumptions and preset values have been proposed, which are not necessarily reasonable. This subsection gives a preliminary assessment of them.

**Confidence Threshold:** The confidence threshold is set as the default 0.25 in the detection process, and it is a rather low threshold aiming to detect as many objects as possible. Despite the concern that more false positives would be detected, the current threshold turns out to obtain better performance in both detection accuracy and error rate, as can be observed from the comparison of different thresholds in Table 5.4.

TABLE 5.4: Overall performance at different confidence threshold

| Confidence Threshold | 0.15 | 0.25 | 0.35 | 0.45 |
|---|---|---|---|---|
| Detection Accuracy | 95.97% | 97.13% | 95.73% | 95.14% |
| Error of DoC | 0.984 | 0.937 | 0.990 | 1.041 |

**Thresholds to determine window status:** In section 3.4.1, we introduced three thresholds: $TH_{lower}$, $TH_{middle}$ and $TH_{upper}$ to distinguish the partially blocked scenarios. Having applied a rather loose restriction (500, 50, 100 for the three thresholds respectively), the detection model gets a satisfying improvement in performance after fusion and the preset values turn out to be appropriate (detailed evaluation is in Appendix E). However, such thresholds are just considered suitable for current scenarios. More experiments are needed to obtain more general thresholds.

**Open-Closed Fusion:** In subsection 3.4.3, a fusion decision remains unsolved with an equivalent number of predicted status "open" and "closed" (assumed "closed" during implementation). The evaluation results discovered 22 cases on the test dataset. The number of ground truth statuses is 4 for "open", 17 for "closed" and 1 for "blocked". Therefore, we can infer that determining such cases as "closed" would have a higher detection accuracy on the given dataset, but this conclusion may be not persuasive enough due to the limited dataset.

### 5.3.2 Limitations

The evaluation results have indicated several limitations of the study conducted in this bachelor thesis, which are mainly the model training and detection logic.

**Model Training**. The poor performance of the detection platform during nighttime can be attributed to the low accuracy in labeling night images due to recognition uncertainty. The dataset was originally set up with images captured from 4:00 to 22:40, assuming that the excluded time would have similar lighting conditions and the roller shutter would remain unchanged during the night. However, this exclusion has significantly reduced the number of night images available. To make matters worse, many night images have poor quality and are difficult to recognize, leading to their abandonment during the labeling process. This can result in inadequate learning of features for night scenarios during training, ultimately causing detection failures.

**Detection Logic**. The detection logic involves the decision tree to determine the status and the fusion logic to improve the overall performance. These methods have still much room to be improved. For the decision tree, the optimal thresholds still need to be determined. For the fusion logic, this bachelor thesis only applied a rough majority rule, while in reality, the decision-making process could be more complex. For example, if one camera position is considered far more suitable for detecting one certain window, then the weights of its prediction should not be simply considered equivalent to those of other camera positions. In the current experiment, camera 2 has a better detection accuracy than camera 1. If a more thoughtful weighting can be applied in the fusion, the overall performance may be further improved.

# 6 CONCLUSION

## 6.1 Key Findings and Contributions

In this bachelor thesis, the task of detecting the roller shutter status and degree of closure is accomplished by applying the YOLOv5 object detection algorithm. A detection platform is developed using Python and Qt Designer. On the test dataset, an overall detection accuracy of 97.13% and an average Error of Degree of Closure (DoC) of 0.937 are achieved between 4:00 and 22:40.

This thesis introduced a decision tree to determine the roller shutter status under heavy occlusion. The introduction of the "blocked" status reduces the risk of assigning hardly recognized roller shutters to ambiguous categories of "open" or "closed". Additionally, the newly developed fusion logic significantly improves detection performance when multiple camera positions are involved.

The model training utilizes a labeled dataset consisting of 7652 images for training, 818 images for validation, and 4381 images for testing. YOLOv5m is selected as the desired model size, and the training epoch is set to 200. The best weights obtained from the training process achieve an mAP@0.5 of 0.994 on the validation set and 0.907 on the test set.

The roller shutter detection platform, serving as the final contribution, provides an end-to-end GUI with a wide range of functions, including visualization of

the detection process, modification of detection results, report exporting, and model evaluation. The code structure is developed using the MVC (Model-View-Controller) architecture, ensuring the independence between business logic and GUI, thereby facilitating the maintenance and reusability of methods.

However, several limitations are identified during the evaluation. The detection platform exhibits decreased performance during nighttime, and different camera positions may significantly affect the detection accuracy. Furthermore, the preset values used in the implementation still require optimization through further investigations.

## 6.2 Future Work

To address the existing limitations, future work should focus on four main areas of improvement:

**Optimizing the Dataset**. The training dataset should be expanded for the detection model to learn more features of roller shutters and window glass. In the current experiment, due to the limited dataset, we did not pay special attention to the related weather condition of each image. The future study should consider weather to involve winter images, for example. Moreover, it is crucial to improve the accuracy of labeling to ensure correct ground truth, enabling the model to learn more precise features under challenging environments.

**Investigating other algorithms**. Exploring other object detection algorithms, such as DETR, which has shown promising results compared to the YOLO series, could enhance the performances.

**Optimization of Detection Logic**. The detection logic should be optimized through additional experiments and comprehensive factor analysis. The desired thresholds of decision trees should be obtained via sufficient experiments.

**Extending Functions of the Detection Platform**. The current detection platform is just a demo to give an insight into how the detection algorithms and logic work. The current MVC structure still needs to be optimized for further connection with other modules, such as the database.

# REFERENCES

[1] R. Hänsel, 2023. [Online]. Available: https://cloud.th-luebeck.de/index.php/apps/files/?dir=/Bachelor_ECUST/00_Kickoff&openfile=3806643/BA_Building_Management_20230307.pdf.

[2] 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.

[3] 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.

[4] 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.

[5] X. Wang, K.-L. Wen, X.-H. Ying, and H.-C. Chen, "Appling semantic method to windows detection in facade," in *2011 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2011, pp. 837–841.

[6] K. Prompol, C.-Y. Lin, S. Ruengittinun, H.-F. Ng, and T. Shih, "Automatic door detection of convenient stores based on association relations," in *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, IEEE, 2021, pp. 1–2.

[7] N. Bayomi, M. El Kholy, J. E. Fernandez, S. Velipasalar, and T. Rakha, "Building envelope object detection using yolo models," in *2022 Annual Modeling and Simulation Conference (ANNSIM)*, IEEE, 2022, pp. 617–630.

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

[9] 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.

[10] 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.

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

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

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[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] 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.

[33] 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.

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

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

[36] 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.

[37] 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.

[38] 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.

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

# A
# APPENDIX

## A Template Matching

**Template matching** refers to finding the part of the test image that is similar to the template image, which is achieved by calculating the similarity between them. Also, it can quickly locate the predefined target according to the template 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 used to position the area which matches the template. Figure A.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 a method is sensitive to image transformations, and even slight environmental changes would cause the failure, let alone tackle the problem of occlusion. 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 A.1: An example of template matching in OpenCV [30]

# B 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 [16]. 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 B.1. As can be seen, the decision boundary L1 is set too close to the training samples, while L2 obtained by the 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 increase.

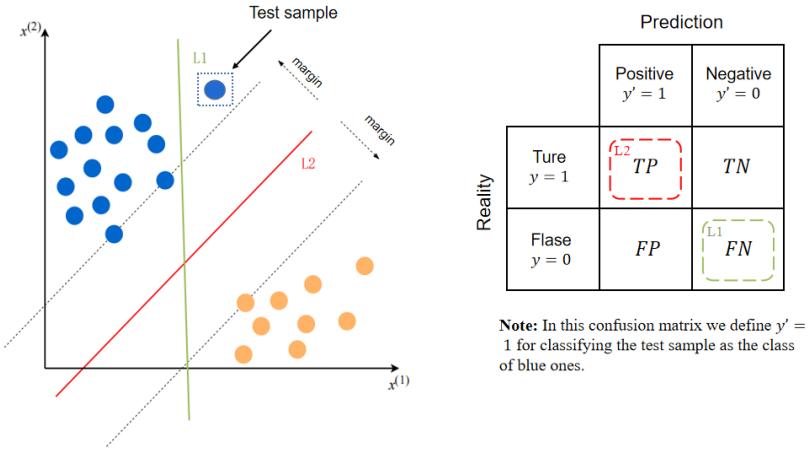


FIGURE B.1: An example of SVM decision boundary (left) and the confusion matrix (right). L1 fails to classify the marked test sample correctly
[16]

# C Segmentation

Unlike object detection, segmentation requires the 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 C.1.

**Semantic segmentation** is an approach detecting, for every pixel, belonging class of the object [32]. For example, all cars in an image are segmented as one object and the background as another object. **U-Net**, named by its U-shape architecture [33], is one of the state-of-art based on CNN to process semantic segmentation.

**Instance segmentation** aims to identify every pixel belonging to an 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 tasks.

**Panoptic segmentation** combines the feature of semantic and instance segmentation. It can identify the belonging class of each pixel, meanwhile distinguishing

them based on different instances. However, processing such a large number of object instances and backgrounds simultaneously would require more resources. **Panoptic FPN** and **BlendMask** are two commonly used algorithms.

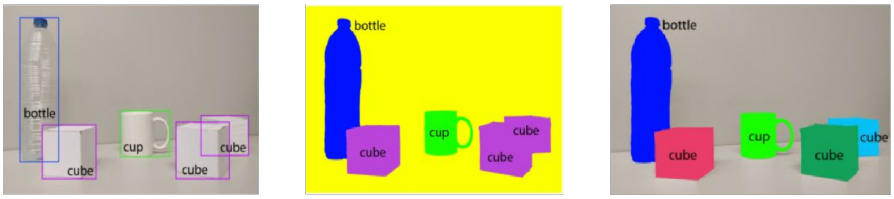


FIGURE C.1: An example of object detection and segmentation [34]

# D 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 D.1 illustrates the architecture of YOLOv5-6.0, which can be divided into three main components: the backbone, neck, and head.

## D.1 Backbone

The Backbone of the 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. 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 [36], which is designed to reduce computational complexity and improve inference speed while maintaining the detection and recognition accuracy of the model.

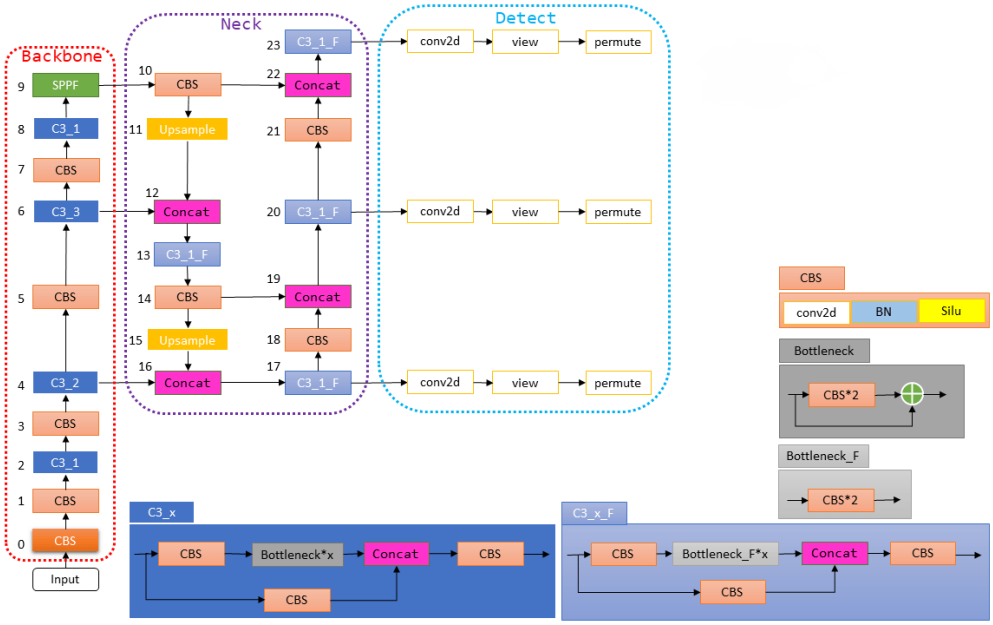


FIGURE D.1: Network Architecture of YOLOv5-6.0 [35]

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 D.2 gives an example of applying CSP to ResNet.

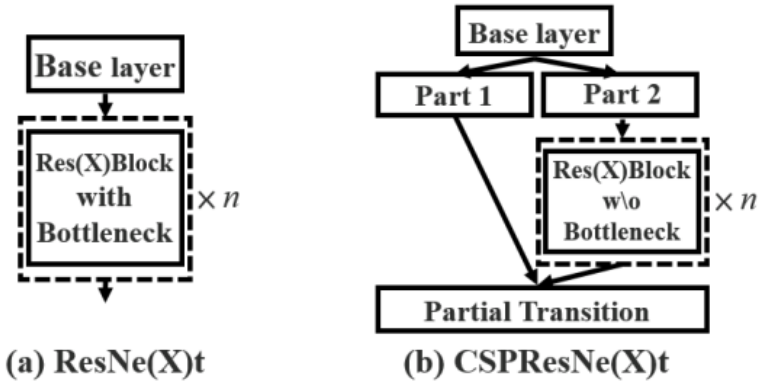


FIGURE D.2: Applying CSPNet to ResNe(X)t [36]

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. Another difference between the C3 module and the BottleneckCSP module is that the Conv module following the output of the Bottleneck module is removed.

**SPPF:** In YOLOv5-6.0, the Spatial Pyramid Pooling Fusion (SPPF)[28] module is used instead of the SPP (Spatial Pyramid Pooling) module. 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 D.3 depicts the structure of SPPF.

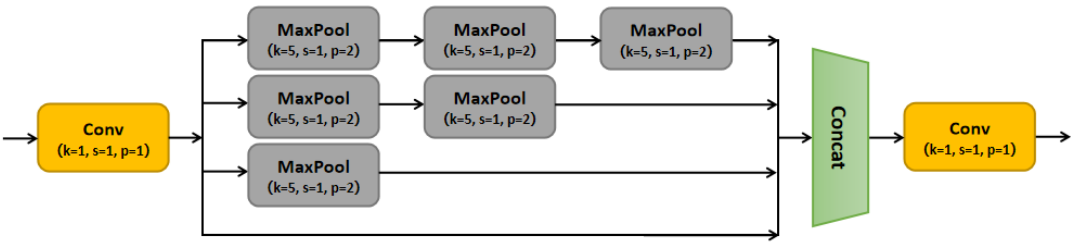


FIGURE D.3: Structure of SPPF module

## D.2 Neck

The Neck of YOLOv5 is based on FPN (Feature Pyramid Network) and PANet (Path Aggregation Network).

**FPN:** 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 D.4. 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.

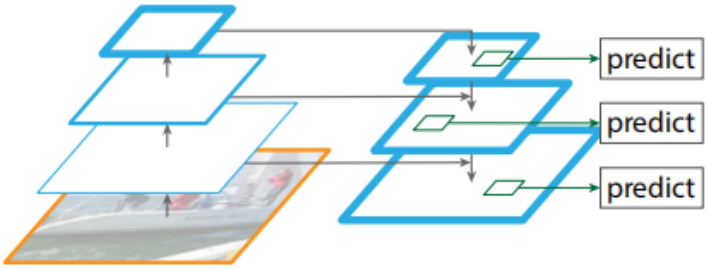


FIGURE D.4: Structure of FPN
[37]

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 D.5, 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.

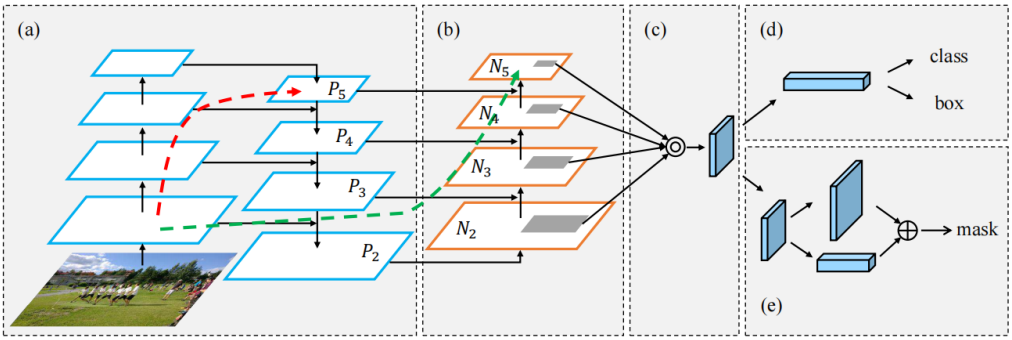


FIGURE D.5: Illustration of the framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion.
[38]

## D.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 (the 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.

## D.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:

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

where

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

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

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

**Classification loss:** YOLOv5 uses the binary cross-entropy function by default to calculate the classification loss. It 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 the positive sample, 0 for the negative sample), **p** is the predicted probability by the model that the input sample is positive.

**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.

## D.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 D.6, 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.

**Mixup:** It 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 D.7 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 D.6: Mosaic: a method of data augmentation [27]

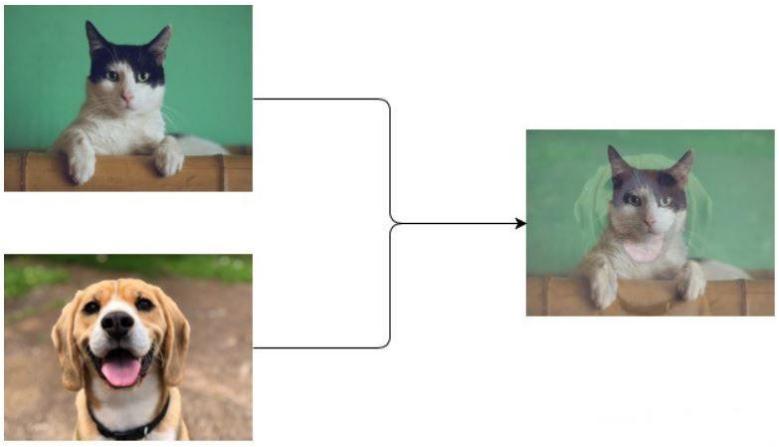

FIGURE D.7: Simple visualization of image mixup [39]

# E Evaluation of Thresholds

This section aims to evaluate the preset values of three thresholds used to determine the status of a roller shutter. The definitions and preset values can be referred to in Section 3.4.1.

## E.1 Improvement of performance

Before we investigate the optimal thresholds, the primary question is whether the setting of these thresholds has improved the overall performance. A comparison between the performance of the model that applied these thresholds and the other without (the model with no partially blocked status detected because of these thresholds) is shown in Table E.1.

TABLE E.1: Performance with and without thresholds applied

| | Detection Accuracy | Error of DoC | $NP_{DoC_{\text{error}}>10}$ |
|---|---|---|---|
| With thresholds | 97.13% | 0.937 | 7 |
| Without thresholds | 96.13% | 0.945 | 23 |

**Note:** $NP_{DoC_{\text{error}}>10}$ refers to number of predictions with Error of DoC $> 10$.

As can be observed, these thresholds do increase the performance of the model in every aspect. Notably, the number of predictions with an Error of DoC $> 10$ is dramatically decreased, which indicates that the fusion of the DoC tends to be more precise.

## E.2 Appropriateness of thresholds

As the performance can be attributed to various factors, it's hard to obtain the optimal thresholds. We can still investigate whether the preset thresholds are appropriate or not.

We suppose that each threshold influences the performance independently. By keeping two thresholds unchanged, we can test adjacent values of the rest

threshold to see if they have a similar performance. If the performance of the preset threshold is decreased dramatically compared with either of the adjacent values, it may indicate that the threshold is set too strictly and needed to be adjusted to a rather loose one. The table below gives an overview of the performance with the preset threshold values compared with that of adjacent values. As can be observed, these thresholds turn out to be appropriate for detection.

TABLE E.2: Performance of adjacent threshold values

| | $TH_{lower}$ | | $TH_{upper}$ | | $TH_{middle}$ | |
|---|---|---|---|---|---|---|
| | Value | Accuracy | Value | Accuracy | Value | Accuracy |
| (-20) | 480 | 96.89% | 80 | 97.13% | 30 | 97.09% |
| (-10) | 490 | 96.97% | 90 | 97.13% | 40 | 97.13% |
| Current | **500** | **97.13%** | **100** | **97.13%** | **50** | **97.13%** |
| (+10) | 510 | 97.05% | 110 | 97.13% | 60 | 97.13% |
| (+20) | 520 | 97.05% | 120 | 96.73% | 70 | 97.13% |

**Note:** Other two thresholds are set as "Current" values when the value of one changes.