



**AN AUTOMATIC WAREHOUSE INVENTORY  
MONITORING SYSTEM BASED ON TEXT EXTRACTION  
USING COMPUTER VISION**

**BY**

**NITIWAT ANANVAITHAYAKIJ**

**AN INDEPENDENT STUDY SUBMITTED IN PARTIAL  
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF ENGINEERING (LOGISTICS AND SUPPLY CHAIN  
SYSTEMS ENGINEERING)  
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY  
THAMMASAT UNIVERSITY  
ACADEMIC YEAR 2022**

THAMMASAT UNIVERSITY  
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

INDEPENDENT STUDY

BY

NITIWAT ANANVAITHAYAKIJ

ENTITLED

AN AUTOMATIC WAREHOUSE INVENTORY MONITORING SYSTEM BASED  
ON TEXT EXTRACTION USING COMPUTER VISION

was approved as partial fulfillment of the requirements for  
the degree of Master of Engineering (Logistics and Supply Chain Systems Engineering)

on June 24, 2023

Member and Advisor



---

(Associate Professor Warut Pannakkong, Ph.D.)

Member



---

(Associate Professor Jirachai Buddhakulsomsiri, Ph.D.)

Director



---

(Professor Pruettha Nanakorn, D.Eng.)

Independent Study Title	AN AUTOMATIC WAREHOUSE INVENTORY MONITORING SYSTEM BASED ON TEXT EXTRACTION USING COMPUTER VISION
Author	Nitiwat Ananvaithayakij
Degree	Master of Engineering (Logistics and Supply Chain Systems Engineering)
Faculty/University	Sirindhorn International Institute of Technology/ Thammasat University
Advisor	Associate Professor Warut Pannakkong, Ph.D.
Academic Years	2022

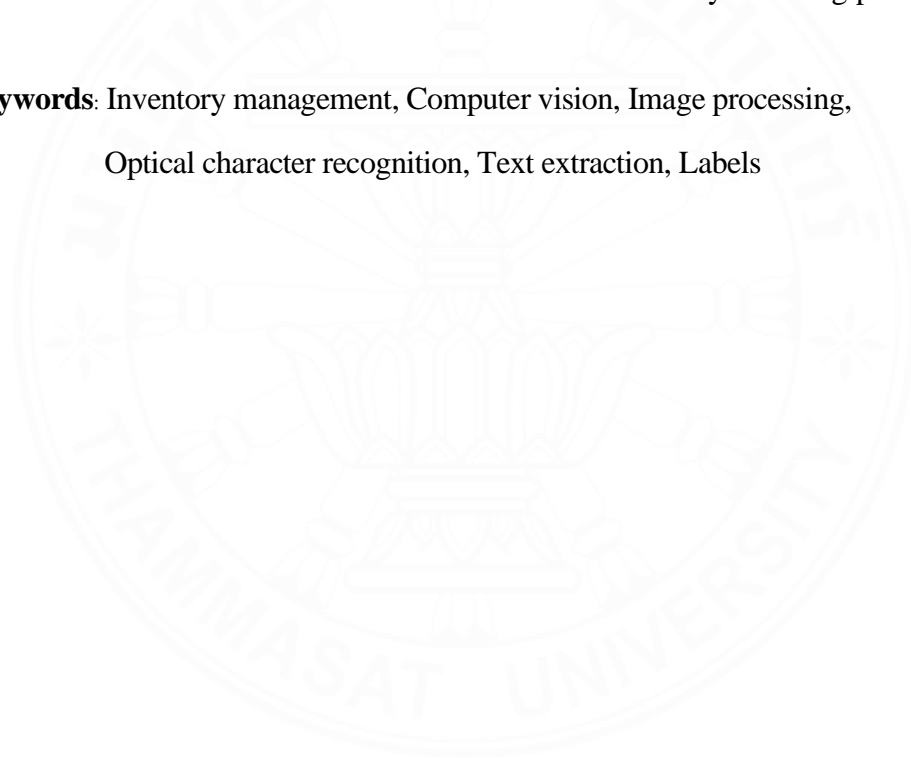
## ABSTRACT

Nowadays, most of product companies around the world do sales forecasting and inventory management to give themselves benefits to be able to take care of all customers' demand and avoid the shortage of supplies that might cause a lost in sales. Most companies then require at least a warehouse whether it is an in-house or outsourced warehouse to store products safely and be able to deliver them to assigned destinations. That is why warehouse management is an inevitable factor when we consider maximizing supply chain's profit.

In order to ensure that operating warehouses perform product storage correctly, companies usually request for an inventory checking report from warehouses, some for annually and some for semi-annually. This report normally presents meaningful keys that represent identity of each product such as product ID, pallet's tag ID, location of pallet on the rack, and quantity of products in that pallet. However, this request indirectly sends non-benefit impacts to the warehouse since inventory checking has been done manually which requires several workers to unload, check, and load pallets back on the rack and also needs to stop all transporting schedule to do the checking. With the following impacts, warehouse

needs to reschedule its normal activities and sometimes even postpone the delivery appointment. This study is going to find the solution for these drawbacks by replacing manual inventory counting with an automatic warehouse inventory monitoring system based on text extraction with the use of computer vision that can detect product label attached on every pallet on the rack, recognize and extract key information from label, and send digital text to store in document for comparing with the master data. Under the field of computer vision, technology of image processing will be applied in the part of detecting product label from captured images while optical character recognition will be applied to extract text from cropped image of label. With the help of technology, warehouse can reduce time used and decrease human error in the inventory checking process.

**Keywords:** Inventory management, Computer vision, Image processing,  
Optical character recognition, Text extraction, Labels



## ACKNOWLEDGEMENTS

First of all, I would like to express my deepest appreciation to Associate Professor Dr. Warut Pannakkong who has been a long time both advisor and supporter for this independent study. His support, guidance, and assistance have been invaluable throughout this journey. His expertise in the field of artificial intelligence and computer vision eased my study and work a lot. The insightful feedback and encouragement have been instrumental in shaping the direction and quality of this work. I am grateful for the opportunity to learn from his wisdom and experience.

I would also like to extend my heartfelt thanks to KNS Logistics Service Co., Ltd. and the warehouse staff, who provided valuable assistance, insights, and contributions to different aspects of this independent study. Their dedication, expertise, and collaboration have significantly enriched the outcomes and enhanced the overall quality of this work. Without the company, there would be no collection of training dataset that can improve the model up to this satisfied level.

Furthermore, I am grateful to Sirindhorn International Institute of Technology that provided the necessary resources and facilities to conduct this independent study. The support has been crucial in enabling the successful completion of this work.

My sincere appreciation also goes to my colleagues and friends who have offered their support and encouragement throughout this endeavor. Their insightful discussions, constructive feedback, and motivation have been truly inspiring and have helped me overcome challenges along the way.

In conclusion, I am humbled and honored to have had the opportunity to work on this study, and I am grateful to everyone who has contributed in any way. Your support and collaboration have made this journey fulfilling, and I am indebted to each and every one of you.

Nitiwat Ananvaithayakij

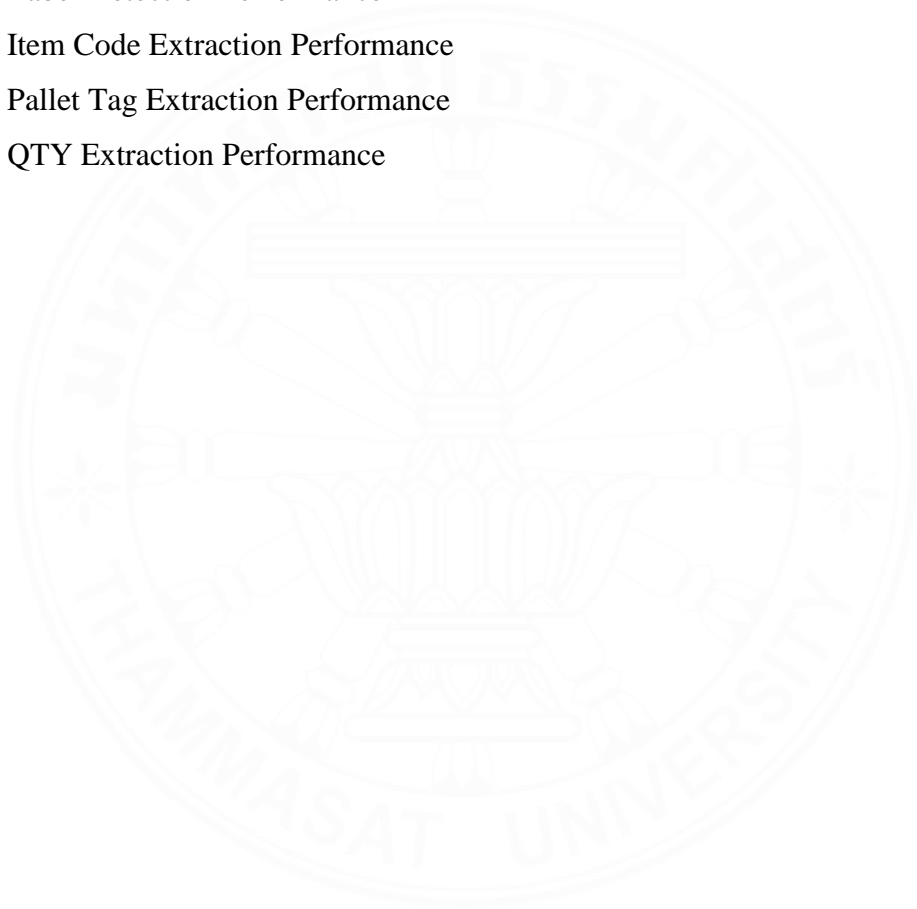
## TABLE OF CONTENTS

	Page
ABSTRACT	(1)
ACKNOWLEDGEMENTS	(3)
LIST OF TABLES	(6)
LIST OF FIGURES	(7)
CHAPTER 1 INTRODUCTION	
1.1 Background and Motivation	1
1.2 Problem Statement	1
1.3 Objectives	2
1.4 Study Limitations	2
CHAPTER 2 REVIEW OF LITERATURE	3
2.1 Related Research	3
2.2 Relevant Theories	5
2.2.1 Pre-processing Original Image	5
2.2.2 Extracting Text from Image	7
CHAPTER 3 DESCRIPTION OF STUDY	9
3.1 Methodology	9
3.2 Activities	11
3.2.1 Study of Image Processing and Text Extraction	12
3.2.2 Construction of Label Detection Model	17

3.2.2.1 Image Pre-processing	18
3.2.2.2 Shape Detection	19
3.2.2.3 Perspective Adjustment and Cropping	19
3.2.3 Construction of Text Extraction Model	20
3.2.3.1 Three Key Information Cropping	20
3.2.3.2 Three Key Information Extraction	21
3.2.4 Collecting Samples of Product Labels from Warehouse	21
3.2.5 Evaluation of Model Performance	22
CHAPTER 4 RESULTS AND DISCUSSION	25
4.1 Results	25
4.2 Discussion	26
CHAPTER 5 CONCLUSION AND RECOMMENDATIONS	29
5.1 Conclusion of the Project	29
5.2 Knowledge Gained	30
5.3 Recommendations for Future Work	31
REFERENCES	32
APPENDIX	
APPENDIX A	36
BIOGRAPHY	40

## LIST OF TABLES

Tables	Page
2.1 Comparison Between Past Researches and This Study	5
3.1 Gantt Chart of Scheduled Activities	12
3.2 Model Performance Evaluation Template	24
4.1 Label Detection Performance	25
4.2 Item Code Extraction Performance	25
4.3 Pallet Tag Extraction Performance	26
4.4 QTY Extraction Performance	26





## LIST OF FIGURES

Figures	Page
2.1 Output of Image Processing on Invoice	4
2.2 Output of Text Extraction from Taxi Receipt	4
2.3 Input Image After Performing Canny Edge Detection	6
2.4 Input Image After Performing Contour Detection	7
2.5 Process of Line Segmentation	8
2.6 Process of Word Segmentation	8
2.7 Process of Character Segmentation	8
3.1 Flowchart of Methodology	10
3.2 Original Input Image	13
3.3 Grayscale Image	13
3.4 Grayscale and Blurred Image in Both Axes	14
3.5 Image Adjusted by Thresholding Technique	15
3.6 Cropped Image of Apple	15
3.7 Axes Arrangement and Coordinates of Cropped Section	16
3.8 Rectangular Shape Detection through Finding and Drawing Contour	17
3.9 Original Input Image	18
3.10 Preprocessed Image	18
3.11 Shape Detection Performed on Original Image	19
3.12 Detection of Product Label	20
3.13 Cropped Pallet Tag Image	21
3.14 Cropped Item Code Image	21
3.15 Cropped QTY Image	21
3.16 Three Text Information Extracted from Image	21
3.17 A Training Dataset of Input Image	22
3.18 An Example Image of Normal Case Situation	23
3.19 An Example Image of Special Case Situation	23

4.1 Blurred Image	27
4.2 “RECEIVED” Mark Blocking Text Image	28



# CHAPTER 1

## INTRODUCTION

In this chapter, introduction to an automatic inventory monitoring system based on text extraction using computer vision is presented along with its background and motivation, current problems that the warehouse is encountering, objectives, and limitations of this study.

### 1.1 Background and Motivation

Since most of the product companies, retailers, or even manufacturers require at least one warehouse to store their products, goods, or raw materials respectively, it is rational that those companies would like to track their inventories stored in the contracted warehouses. They usually request their partnered warehouses to send inventory checking report for every year or more frequently like half a year. KNS Logistics Service Co., Ltd., a 3PL company which provides storage and warehousing service mainly to Toshiba Thailand, has been requested for this kind of report too. However, it takes too much manpower and time to execute which will be explained in detail in the next sub-section. This study is then not only my independent study but also a collaborated study between SIIT and KNS aiming to improve the process of inventory checking by applying computer vision technology.

### 1.2 Problem Statement

According to the inventory checking activity that most warehouses are requested to perform, common problems are that they have to terminate all other usual warehouse activities and spend too many hours to finish this checking activity which normally requires a few working days. This can lead to a loss of sales that warehouses possibly make. Moreover, the current inventory checking process is done manually by warehouse's employees. For KNS Logistics Service Co., Ltd., even it has an AS/RS system in the warehouse, employees still need to unload, check, and load pallets back on rack when

performing the inventory checking process. This can create more chance of human error and also decrease the trustworthiness of the warehouse to its supply chain.

In response to these two critical problems; long inventory checking process time and high possibility of human error, this study is focusing on developing an automatic inventory monitoring system that will reduce checking time and human error by applying the current popular technology, computer vision, which will be conceptually explained and demonstrated later on.

In addition, each pallet stored on rack in KNS's AS/RS system has a label attached on which contains text information such as product's code, pallet's tag number, and number of quantity. Traditional inventory checking process at KNS is to check whether these information on pallet correctly matches with information in the master data that is stored as .xlsx format or not.

## **1.2 Objectives**

This study aims to achieve three objectives which can be divided into three phases.

1. To develop the system that is capable of detecting label from the collected input images, recognizing 3 text information on product label image; Tag Number, Item Code, and Quantity Number, and transforming them into digital text..
2. To evaluate the performance of the invented inventory monitoring system through testing images collected from warehouse of KNS Logistics Service Co., Ltd. with the required overall accuracy of 85%.

## **1.3 Study Limitations**

All text information represented on product label in this study is printed in number or English language only which makes it easier to be recognized than the handwritten one or the other language one. Moreover, dataset used in this study is collected from a single warehouse, KNS Logistics Service Co., Ltd. which means other warehouses might have different product label's configuration.

## **CHAPTER 2**

### **REVIEW OF LITERATURE**

#### **2.1 Related Research**

For the past few years, artificial intelligence has been in the trend of research and development which allows researchers to apply computer vision to be used in various industries. Several new technologies that have just been introduced to the world are also based on computer vision. Szelisk (2011) defined computer vision as a field aiming to describe images by interpreting, reconstructing, and extracting properties from them such as shapes, colors, and distances. There are different other names that can be defined as computer vision such as machine vision system, visual image system, and image system (Fernandes, Dórea, & Rosa, 2020). Since computer vision can be applied to any kind of problems that relate to images or videos, some articles found out that this technology can be helpful in defect detection problems. Raveendran and Chandrasekhar (2022) integrated computer vision techniques and deep learning model to inspect and classify defects in semiconductor manufacturing. He et al. (2021) applied image processing which is a part of computer vision to develop machine positioning. Apart from manufacturing industry, Fernandes, Dórea, & Rosa (2020) used computer vision in the field of animal science. The researchers monitored animal's behavior, measured phenotypes of interest such as body weight and condition score, identified live animal, and assessed beef cuts composition through computer vision and sensor technologies.

There are several articles in the recent years that integrated two technologies of computer vision which are image processing and text recognition and applied them to solve their interested problems like extracting text information from images taken from cameras or phones. Although the previous researchers applied this knowledge to many fields such as bill extraction and taxi receipt extraction which are illustrated in Figure 2.1 and Figure 2.2 respectively, none of them use this capability to do text extraction from product label attached to every pallet on warehouse's racks which is illustrated in Table 2.1. Moreover,

actual data from KNS warehouse will be used in this in study which will enhance this study to have more potential in the industrial world.

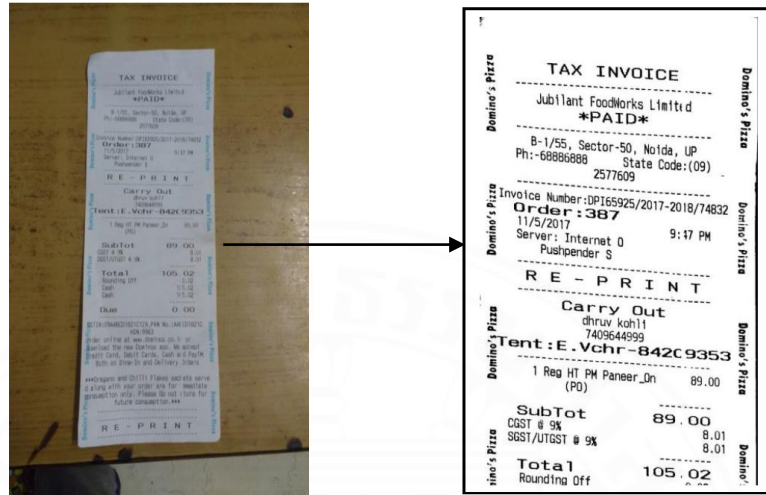


Figure 2.1 Output of Image Processing on Invoice



Figure 2.2 Output of Text Extraction from Taxi Receipt

**Table 2.1** Comparison Between Past Researches and This Study

Publication	Year	Image Processing	Text Extraction	Data Mining	Machine Learning	Neural Network	Text Extraction from Bill/Receipt	Text Extraction from Label	Actual Data from Industry
Sidhwa <i>et al.</i>	2018	✓	✓				✓		
Manoharan	2019	✓	✓						
Karthikeyan and Vanitha	2019	✓		✓					
Yindumathi <i>et al.</i>	2020	✓	✓		✓		✓		
Liu <i>et al.</i>	2020					✓	✓		
Kumar <i>et al.</i>	2020	✓	✓				✓		
Thorat <i>et al.</i>	2022	✓	✓						
This Study	2022	✓	✓					✓	✓

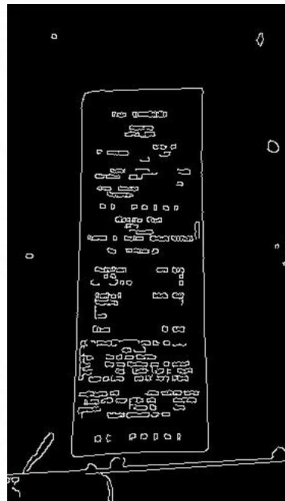
## 2.2 Relevant Theories

In order to apply knowledge of computer vision technology to solve inventory checking problem in the warehouse, OpenCV and Tesseract OCR are two crucial engines to build an automatic inventory monitoring system. The first engine is for doing image processing which aims to exclude noise and unnecessary parts from original image and make an interested text part much easier to be later on extracted by applying some filters. Then the second engine does the job to recognize and extract text information from image and send it to specified destination.

### 2.2.1 Pre-processing Original Image

Image processing is a technique used to pre-process an input image which all collected images must go through. This process needs to be done deliberately because Tesseract OCR in the next process may provide a poor-quality text extraction if its input images are not perfectly pre-processed. OpenCV has been a current well-known tool to perform an image processing. It is an open source computer vision library that contains a collection of image processing algorithms developed by Intel (Sidhwa et al., 2018) It provides various activities that users can do with their images such as resizing, color detection, and shape detection through Python language. Sidhwa et al. (2018) used OpenCV to detect bill or invoice from image and filter out unnecessary noise before passing it to next process. In their case, Canny edge detection function was initially used to convert original image into black and white where white lines separate two high contrast areas which representing edge of the invoice as shown in Figure 2.3. Then the next process

was executed by contour detection function which draws contour in color along the edge that was detected earlier as shown in Figure 2.4. Article of Yindumathi, Chaudhari, & Aparna (2020) also mentioned that OpenCV has been widely used to do image's edge cutting. Changing image's color from RGB to grayscale or binary is also available in OpenCV. This approach allows computer to detect contours or shapes easier than original color. This type of image improvement was one process of OCR process flow in an article of Thorat et al. (2022). However, there are some articles selected other techniques to pre-process original images. For example, improved SSD network was developed and used by Liu et al. (2020) to locate taxi's receipt information region in original images instead of using image processing.



**Figure 2.3** Input Image After Performing Canny Edge Detection



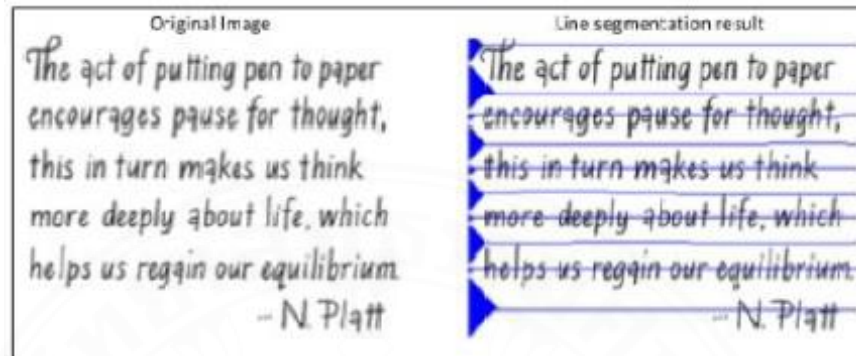


**Figure 2.4** Input Image After Performing Contour Detection

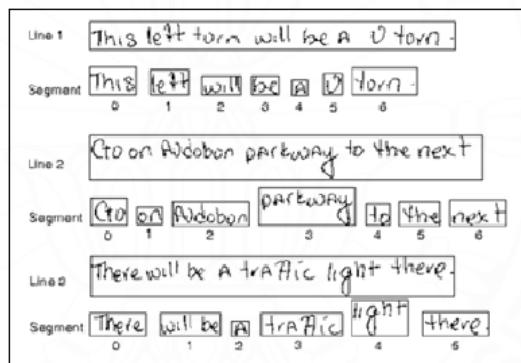
### 2.2.2 Extracting Text from Image

After noise and unnecessary parts of image have been cropped out and interested parts have been pre-processed, text information now should be easily extracted. One popular text extraction technique that has been widely used is optical character recognition or OCR. Tesseract is one of the open-source OCR engines that was developed at Hewlett Packard (Sidhwa et al., 2018). It applies text segmentation to extract printed or handwritten text from images. There are three levels of segmentation process which are line segmentation, word segmentation, and character segmentation as shown in Figure 2.5, Figure 2.6, and Figure 2.7 respectively. The first level aims to classify text information by lines and then moves to the next level that finds space between words in each line that was previously detected to perform word segmentation. The last level is to dilute words into the smallest level, characters. Sidhwa et al. (2018), Yindumathi, Chaudhari, & Aparna (2020), Kumar et al. (2020), and Thorat et al. (2022) all used Tesseract OCR to convert text information in any images into digital text format although some had used different pre-processing techniques. However, Liu et al. (2020) used different approach to do character recognition. The researchers selected to use CNN-GRU combined neural network to recognize text instead of three levels of segmentation. They proposed their text recognition methodology for taxi receipt based on neural network without applying

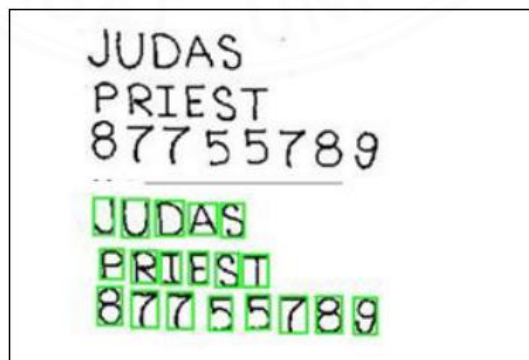
traditional image processing and ordinary optical character recognition because their problem was quite different from normal situation. Their dataset, taxi receipt, contains bad printing and irregular arrangement of text which ordinary text segmentation cannot solve.



**Figure 2.5** Process of Line Segmentation



**Figure 2.6** Process of Word Segmentation



**Figure 2.7** Process of Character Segmentation

## **CHAPTER 3**

### **DESCRIPTION OF STUDY**

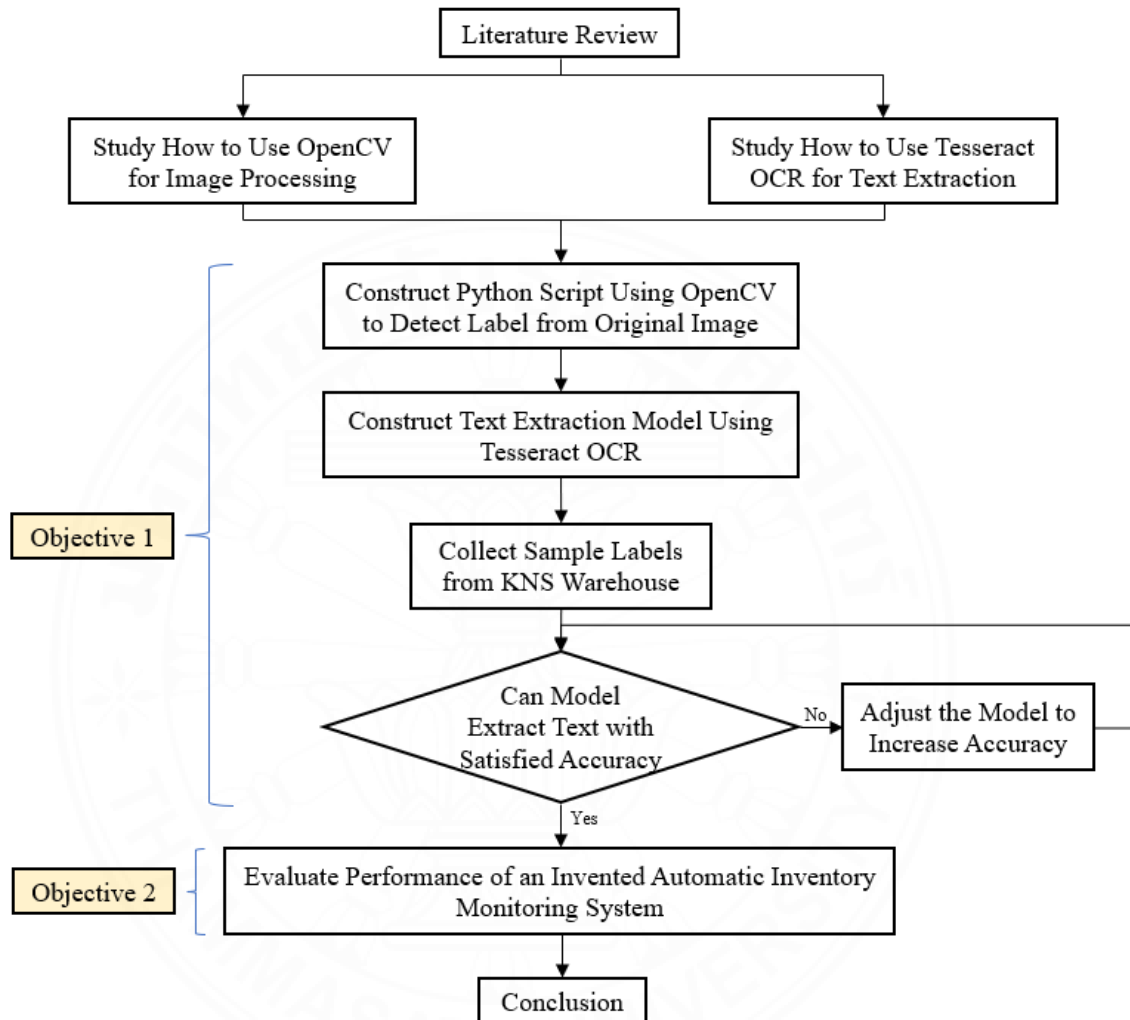
According to those literatures recited in Chapter 2, this study applies two techniques, image processing and text extraction, to be a core knowledge to perform several step operations defined as methodology which will be represented as flow chart. This chapter also provides an activity plan which is called Gantt Chart. Details of each activity will be presented too in this chapter.

#### **3.1 Methodology**

There are two big model construction to be performed following knowledge gained from Chapter 2. The first model is to construct label detection model that is able to perform image processing on input images. Then the second model is a text extraction model that is able to extract three key text information from the preprocessed image of product label. OpenCV library from computer vision is initially used to perform image processing. The first model should be developed to be capable of pre-processing an image, detecting product label from captured image, and cropping out unnecessary parts. Then OCR is applied through Tesseract OCR library to develop a text extraction model to be capable of extracting text information from label image and converting it from image format to text format. The extracted text is expected to be sent out of the system, automatically compared with the master data, and updated the master file. However, there are more details in between than these two big processes to achieve this study's goals.

Figure 3.1 represents flowchart of methodology which each element of it will be explained in detail later on this sub-section. In this figure, each activity needs to be executed to respond three objectives identified in Chapter 1. An initial process of this study's methodology is to do literature review related to image processing, text extraction, and computer vision. Then KNS Logistics Service Co., Ltd. comes to take a vital role as it provides the dataset for this study in form of a set of samples of product labels with an amount around a hundred labels. During this initial phase, OpenCV and Tesseract OCR

library need to be studied in order to be able to use as main tools to construct a model that can both perform image processing and text extraction in sequence.



**Figure 3.1** Flowchart of Methodology

After knowledge and dataset are prepared, a model is ready to be constructed. Python is a language used to construct a script for both models. OpenCV library is used first. In this phase, input images are required to be preprocessed; for example, cropping out unnecessary sections and leaving only label part. Then Tesseract OCR library is used next to construct a Python script that can convert text information in image format to text format

and also be able to send this text to check similarity with the master data and update the master file.

When the model can fully operate, performance evaluation will be done to assure that inventory monitoring system can predict text with satisfied accuracy. The testing images that individually contain product label directly came from KNS Logistics Service Co., Ltd. which have the same pattern of information fulfillment. So, it is important to make sure that all testing images were captured under the same environment and aligned on the same perspective so that this study could simulate as close as possible to what would actually happen when this system is launched into the warehouse.

At this point of the entire methodology, this study is able to produce an automatic inventory monitoring system based on text extraction using computer vision that can be applied to help KNS Logistics Service Co., Ltd. to reduce process time and human error for doing annual or semi-annual inventory checking. There are other several advantages of this study not for only one warehouse company but also for the whole supply chain to recognize how computer vision technology is worth to invest and how it can save cost for every element of the supply chain.

### **3.2 Activities**

In order to follow the mentioned flowchart and be able to complete every task in time, all activities should be addressed in the schedule manner. This following Gantt Chart in Table 3.1 represents each activity that needs to be executed and its duration in the manner of period of time. Moreover, details of each activity in Gantt Chart are thoroughly explained so that readers can follow and apply knowledge gained from this study to their tasks or projects.

**Table 3.1** Gantt Chart of Scheduled Activities

Activities		Jan-23				Feb-23				Mar-23				Apr-23				May-23				Jun-23			
		Week																							
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Semester 2/2022	Review literatures related to image processing and text extraction												PP												FD
	Study how to write Python script using OpenCV library for image processing												PP												FD
	Construct image processing model that can detect product's label from input image												PP												FD
	Study how to write Python script using TesseractOCR library for text extraction												PP												FD
	Construct text extraction model that can convert text in label image to text format												PP												FD
	Add script to transfer received text to database Excel file for inventory monitoring purpose												PP												FD
	Collect training data from KNS												PP												FD
	Create a storyboard of the system from inputting images to getting updated Excel file												PP												FD
	Train and test automatic warehouse inventory monitoring system												PP												FD
	Adjust the model to increase accuracy in text extraction and finalize the model												PP												FD
	Write final report and create final presentation preparing for final defense												PP												FD

The abbreviations in the Gantt Chart are explained below:

PP = Proposal and Progress Presentation (1 Apr)

FD = Final Defense (24 Jun)

### 3.2.1 Study of Image Processing and Text Extraction

This is the first process to be performed after all literatures were reviewed. I started learning how to construct the label detection model first and then following by how to construct the text extraction model. Both of them were constructed and executed through Python script using Visual Studio Code as a code editor software program. There are two main libraries that need to be installed in order to construct the two models. OpenCV is the first one. Open Computer Vision is an open source of computer vision that contains a collection of image processing functions. PyTesseract is another library that needs to be installed. It provides an interface for using Tesseract OCR engine which is a technology that is capable of extracting text from images.

After all libraries are set, image processing is always the first process to perform to any input image. There are five functions that I found useful in this study which will be briefly explained and illustrated through the change of Figure 3.2.



**Figure 3.2** Original Input Image

#### cv2.cvtColor

Changing color of the image is the first function to be used. Normally, images have three channels of color which are red, green, and blue which is difficult for computer to process. This is why it needs to be changed to one channel only which is grayscale. Figure 3.3 illustrates how image is changed according to this function.



**Figure 3.3** Grayscale Image

### cv2.GaussianBlur

Next function is to blur image which is important because it helps reduce unwanted noise and remove high-frequency details of the image but retain main structure and edges of image which make it easier to detect interested components from blurred image than clear image. Users are allowed to configure whether they want to blur the image in X-axis, Y-axis, or both axes. They are also allowed to set the level of blur through number. Figure 3.4 illustrates an example of blurred image.



**Figure 3.4** Grayscale and Blurred Image in Both Axes

### cv2.threshold

The third function is a key factor of label detection model. This function converts color or grayscale images into binary images. It allows users to specify threshold value that when any pixel has value over the threshold, it will be converted to whether black (0) or white (255) as the users defined. For example, if the threshold is set as 200 and the pixel that is over the threshold is set to be white (255), any pixels that holds a value higher than 200 will be converted to white while any pixels that holds value below 200 will be converted to black which is shown in Figure 3.5. Since the labels of this warehouse are printed in white color and stamped onto the packaging of product which has different color, this thresholding function then is very useful to help detect product label.





**Figure 3.5** Image Adjusted by Thresholding Technique

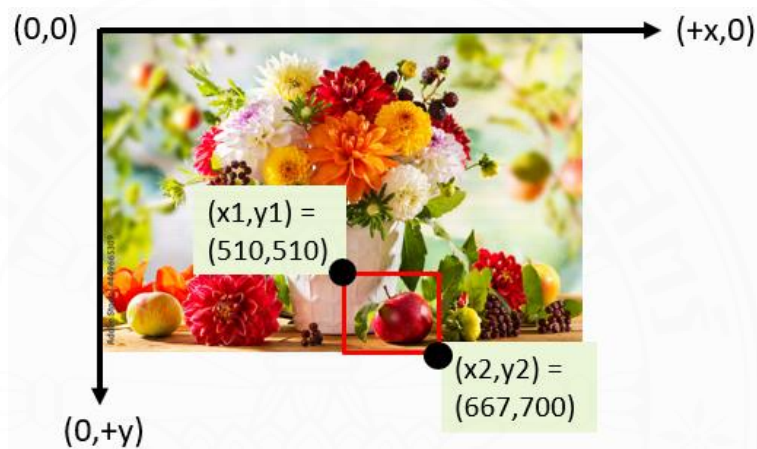
image[y1:y2, x1:x2]

This function has a purpose to keep only interested segment from an original image. It can be considered to be a cropping function. In order to specify the coordinates of interested area, users need to identify a dimension of the whole image first. “print(image.shape)” is a prerequisite function that can be used to identify a dimension of the image. This function will identify the height and length of the image respectively. After the dimension is known, the users now can determine which part of the image they want to crop through (x1,y1) and (x2,y2) where point 1 represents top left corner while point 2 represents bottom right corner of the cropped image. From this example, an original image has the height and length of 667 and 1,000 pixels respectively. If an apple in the middle is an interest, trial and error is done to determine two coordinates that will crop out only apple from original image. The result is shown in Figure 3.6.



**Figure 3.6** Cropped Image of Apple

However, the users need to be noted that the arrangement of axes in computer vision is not usual as conventional axes. The origin is always at the top left corner of an image which states the coordinate of  $(0,0)$ . X-axis starts counting with positive value when the axis goes on to the right which seems to be the same with conventional one. Y-axis however starts counting with positive value when the axis goes down. In this case of cropping apple out of an original image, it has been found that point 1 of the cropped section should be  $(510,510)$  and point 2 should be  $(667,700)$  which are indicated in Figure 3.7.



**Figure 3.7** Axes Arrangement and Coordinates of Cropped Section

#### cv2.findContours and cv2.drawContours

Since the image now is in black and white, an interested section in the image is ready to be detected through its edge and structure. These functions play an important role to find contours and draw a curve or a line on the edge of the contours. Actually, there are several sub-functions that the users can utilize to find contours. For this case, I used “cv2.CHAIN\_APPROX\_NONE” so that there will be no approximation on edges of contours which means the result will be more precise. Moreover, the users can add more condition to the process of finding contour. For example, it can be stated to find only contour that contains four curves so that only rectangular shape contour will be detected and drawn with curve or line. So, this function can apply to perform shape detection too. Figure 3.8 represents a contour which is in rectangular shape.



**Figure 3.8** Rectangular Shape Detection through Finding and Drawing Contour

Another part of the study moves to how to perform text extraction through the library called PyTesseract. Fortunately, there is only one function that needs to be studied which is “pytesseract.image\_to\_string”. This function will utilize Tesseract OCR engine for text extraction on the preprocessed image. The engine technically perform a series of processes to extract text which were already explained in Chapter 2, including text localization, character segmentation, feature extraction, and recognition. During the study, one important technique that I learned to increase an accuracy of text extraction is to input an image containing only interesting text instead of the whole image. This technique then requires a few more times of image cropping.

### 3.2.2 Construction of Label Detection Model

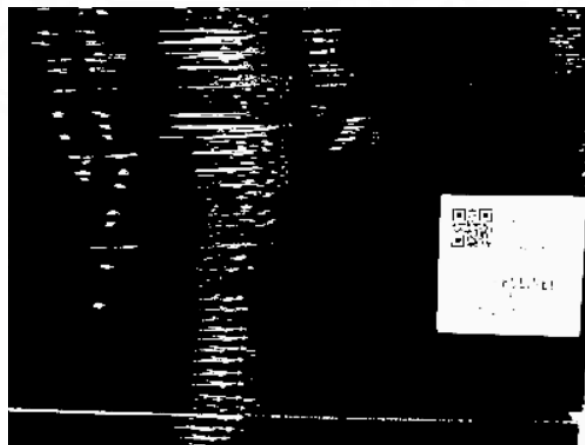
When all libraries and functions were installed and studied, the first model of label detection could be now constructed. The structure of label detection was designed to have three main parts. Pre-processing an image is the first part. This is a part where an original input image meets the system. When an image is perfectly preprocessed, the next part of the model will attempt to detect the rectangular shape that represents the product label. If the label is detected, the last part will adjust its perspective and crop out unnecessary section so that there will be only product label comes out as an output. Following three sub-sections represent each part of the label detection mentioned above.

### 3.2.2.1 Image Pre-processing

From the study in section 3.2.1, I selected to perform changing the color of an image from RGB to grayscale, blurring an image both X and Y axis, and applying thresholding technique to an image respectively defined as an image pre-processing. Figure 3.10 illustrates the preprocessed image of Figure 3.9 that already went through these processes. At this point, a rectangular shape section obviously appears upon an input image which represents the interesting section, product label. However, since each input image has its own color of product's package and different brightness, the threshold value in the thresholding function needs to be adjusted manually every time.



**Figure 3.9** Original Input Image



**Figure 3.10** Preprocessed Image

### 3.2.2.2 Shape Detection

From the preprocessed image received from 3.2.2.1, it is quite a simple task to detect contour from black and white image because the structure and edge of the label are so obvious. This part of the model requires to use function finding and drawing contours with the condition that drawing only occurs when the contour have four curves so that it is surely a rectangular contour. Moreover, there is an additional condition I put into the model which relates to an area that the contour covers. There is a minimum area set as a threshold so that this model still operates smoothly when there is another rectangular shape in an input image. Figure 3.11 illustrates a shape detection process executed on a preprocessed image from previous process but displaying the highlighted contour on an original image.



**Figure 3.11** Shape Detection Performed on Original Image

### 3.2.2.3 Perspective Adjustment and Cropping

The last part of the label detection model is to adjust the perspective of the label from whatever perspective it lies on to conventional perspective. This part will also crop out unnecessary section and remain only a label image. This process was done through two main functions which are “cv2.getPerspectiveTransform” and “cv2.warpPerspective”. Figure 3.12 presents the product label after passing through all processes of the label detection model.



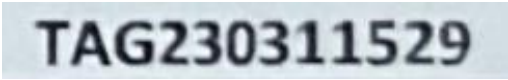
**Figure 3.12** Detection of Product label

### 3.2.3 Construction of Text Extraction Model

Now that the label of product is in the pocket, next step is to extract text information from it. In this case, there are three key text information needed to be considered and extracted which are pallet tag, item code, and quantity number. For a traditional inventory checking process, warehouse staff would manually compare these three information on the product label to the one recorded in the master file. Extraction of these three information then is the goal of this model. It can be seen in Figure 3.12 that each text information locates in different locations of the label. From the study in 3.2.1, the less text information in an image increases the accuracy of text extraction process. That is why there are two parts in text extraction model. First part focuses on cropping these three key text information into three different images. Another part focuses on using a function in PyTesseract library to extract text from each image.

#### 3.2.3.1 Three Key Information Cropping

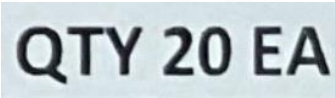
Since there are three locations of text, there then are three separate images of each text information. Fortunately, location of each text approximately locates at the same coordinates for all label. That allows us to be able to use the same setting of cropping function or at most adjust for a little. Figure 3.13, 3.14, and 3.15 illustrate cropped images of three different text information.



**Figure 3.13** Cropped Pallet Tag Image



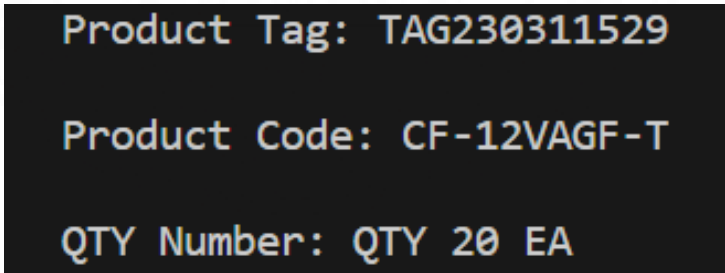
**Figure 3.14** Cropped Item code Image



**Figure 3.15** Cropped QTY Image

### 3.2.3.2 Three Key Information Extraction

Each text image then needs to be extracted using a “pytesseract.image\_to\_string” function through PyTesseract library which was explained in 3.2.1. This process directly relates to the resolution of an input image. If the input image is captured with high resolution, it is easier for the Tesseract OCR engine to extract text in it. Figure 3.16 illustrates three outputs of three text extractions on pallet tag, item code, and quantity number.

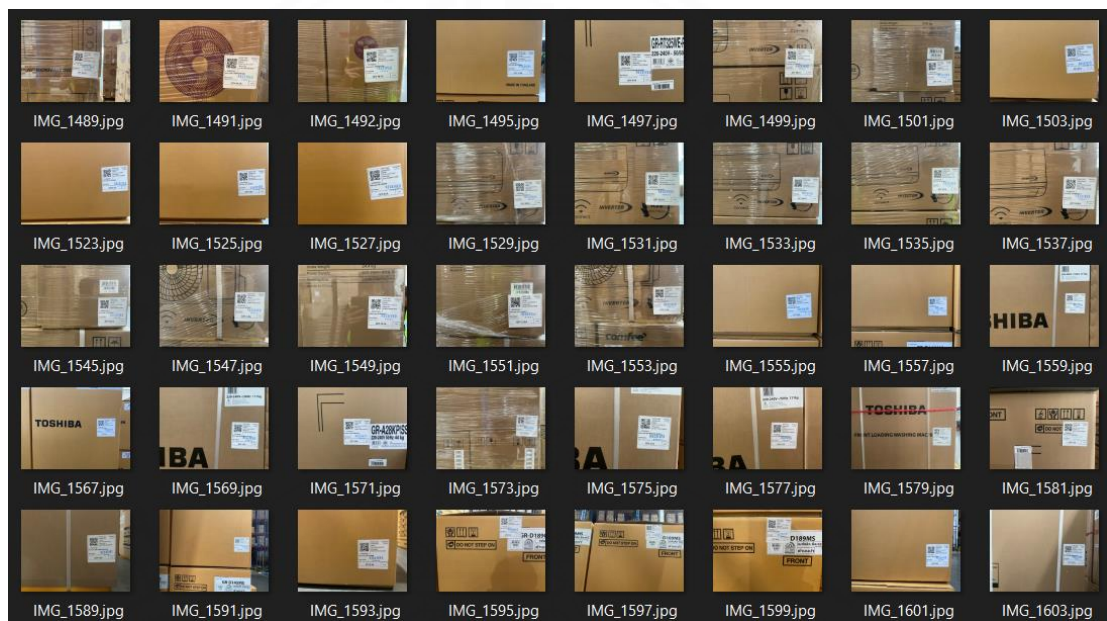


**Figure 3.16** Three Text Information Extracted from Image

### 3.2.4 Collecting Samples of Product Labels from Warehouse

After both models were constructed, a dataset of input images should be collected so that evaluation of the models can be conducted. For this process, I went to the warehouse

of KNS Logistics Service Co., Ltd. to collect as much as possible of images. There are over a thousand pallets stored in their AS/RS system which all of them have a product label on but only images of pallets that are already unloaded from the system aiming for being transported can be taken. So, there are totally 105 images of product label that I collected from the warehouse. Figure 3.17 presents one part of the collection of images that will be used as a training dataset for evaluating performance of the models while Figure 3.9 is an example of this dataset.



**Figure 3.17** A Training Dataset of Input Images

### 3.2.5 Evaluation of Model Performance

Now both models and training dataset are prepared for performing performance evaluation. First, I decided to join label detection and text extraction models together so that it would be easier in model testing process because both label detection and text extraction would be executed after one click of running the model. The dataset was also classified into two categories; normal case and special case. Normal case of input image is that the product label is located on the product package clearly without any blockers that can be unexpected noise to the label detection process and that all three parts of text



information which are printed on the label can be visible clearly without any noise such as blurred image. Figure 3.18 is a good example representing image that is classified into normal case situation. On the other hand, special cases are the images that were blurry taken, had unusual markings blocking interesting text, had other labels overlapping the interesting product label, had plastic film sealing over the product label, had the product label stamped at the edge of the product package which make it difficult to detect label's edge, or even had a strap lying upon the product label. Figure 3.19 is a good example of image that is classified into a special case situation because there are two more labels on the product package which overlaps the interesting label.



**Figure 3.18** An Example Image of Normal Case Situation



**Figure 3.19** An Example Image of Special Case Situation

After the data preparation was done, there are then total 61 images that are classified into the normal situation for this automatic warehouse inventory monitoring system. The procedure of evaluating performance of this model can be divided into two main parts according to a number of processes that were explained earlier. The first part is to evaluate the accuracy of this model in detecting the product label from an input image. Then the second part aims to evaluate the accuracy of this model in extracting text from the preprocessed image where there are three texts to be extracted for each image; pallet tag, item code, and quantity number. So, it can be said that there would be four tables of accuracy. The model can be considered to meet to the standard only when all accuracy levels are over 85%.

Table 3.2 represents a template of two vital metrics that was used to evaluate the performance of the model using class precision and class recall. Class precision is a measurement that indicates the proportion of correctly predicted positive instances which is known as “True Positives” out of all instances predicted as positive which are the combination of “True Positives and False Positives”. This metric reflect the ability of the model to avoid false positives. Class recall, on the other hand, measures the proportion of “True Positives” out of all actual positive instances which are the combination of “True Positives and False Negatives”. This metric reflects the ability of the model to capture all positive instances. It can be stated that “Higher precision indicates fewer false positives and better performance while higher recall indicates fewer false negatives and better performance”.

**Table 3.2** Model Performance Evaluation Template

Accuracy			
	true Positive	true Negative	class precision
pred. Positive			
pred. Negative			
class recall			

## CHAPTER 4

### RESULTS AND DISCUSSION

This chapter will present the outcome of the performance of the constructed model that was evaluated after all processes in Chapter 3 have been accomplished. Moreover, there will be a section of discussion of the results too.

#### 4.1 Results

Since the model performance was evaluated separately according to its purpose, there are four parts of the result which are reported as Table 4.1, Table 4.2, Table 4.3, and Table 4.4 using the template of class precision and class recall metrics explained previously in Chapter 3. There are totally 61 images classified as normal cases and used to find the model performance. The first table relates to label detection model while the rest relates to text extraction model. The performance of the overall model can be considered to be good since the standard level of accuracy (at least 85%) that was set up earlier is satisfied.

**Table 4.1** Label Detection Performance

<b>Label Detection Accuracy</b>		<b>100.0%</b>	
	true Positive	true Negative	class precision
pred. Positive	61	0	100.0%
pred. Negative	0	0	#DIV/0!
<b>class recall</b>	100.0%	#DIV/0!	

**Table 4.2** Item code Extraction Performance

<b>Product Code Extraction Accuracy</b>		<b>88.5%</b>	
	true Positive	true Negative	class precision
pred. Positive	54	N/A	100.0%
pred. Negative	7	0	0.0%
<b>class recall</b>	88.5%	N/A	

**Table 4.3** Pallet Tag Extraction Performance

<b>Pallet Tag Extraction Accuracy</b>		<b>100.0%</b>	
	true Positive	true Negative	<b>class precision</b>
pred. Positive	61	N/A	100.0%
pred. Negative	0	N/A	N/A
<b>class recall</b>	100.0%	N/A	

**Table 4.4** QTY Extraction Performance

<b>QTY Extraction Accuracy</b>		<b>100.0%</b>	
	true Positive	true Negative	<b>class precision</b>
pred. Positive	61	N/A	100.0%
pred. Negative	0	N/A	N/A
<b>class recall</b>	100.0%	N/A	

## 4.2 Discussion

From the result, three out of four tables show that the performance of the model in term of accuracy is up to 100% while the remaining one table can reach up to 88.5% which already passes the criteria that was set in Chapter 3. The table that has an issue is a table representing a performance of the item code extraction. There are 7 images that the model incorrectly predicted. After looking into those images and analyzing the situation to determine the root cause of this issue, it was found that the issue was all about the unclear font. There are two characters that were misunderstood and incorrectly extracted. The first character is “5”. Since the font that the warehouse’s label designer use does not provide number 5 a sharp edge, two out of seven incorrect predictions then extract this number as alphabet “S”. Another character is “0”. This is universal issue if the users do not select the font carefully because zero can be seen as alphabet “O” sometimes. In my case, the remaining five incorrect predictions occurred by predicting “0” to be “O”. This situation rarely happens with pallet tag or quantity number because their texts are not the combination of alphabets and numbers like item code. However, this issue can be easily solved and bring the model to the higher standard by training the Tesseract OCR engine

more about the difference between these two characters so that the model can classify them by itself without changing the design of the label. However, changing fonts of the label seems to be an easier work to perform because it is an improvement at the root of this issue.

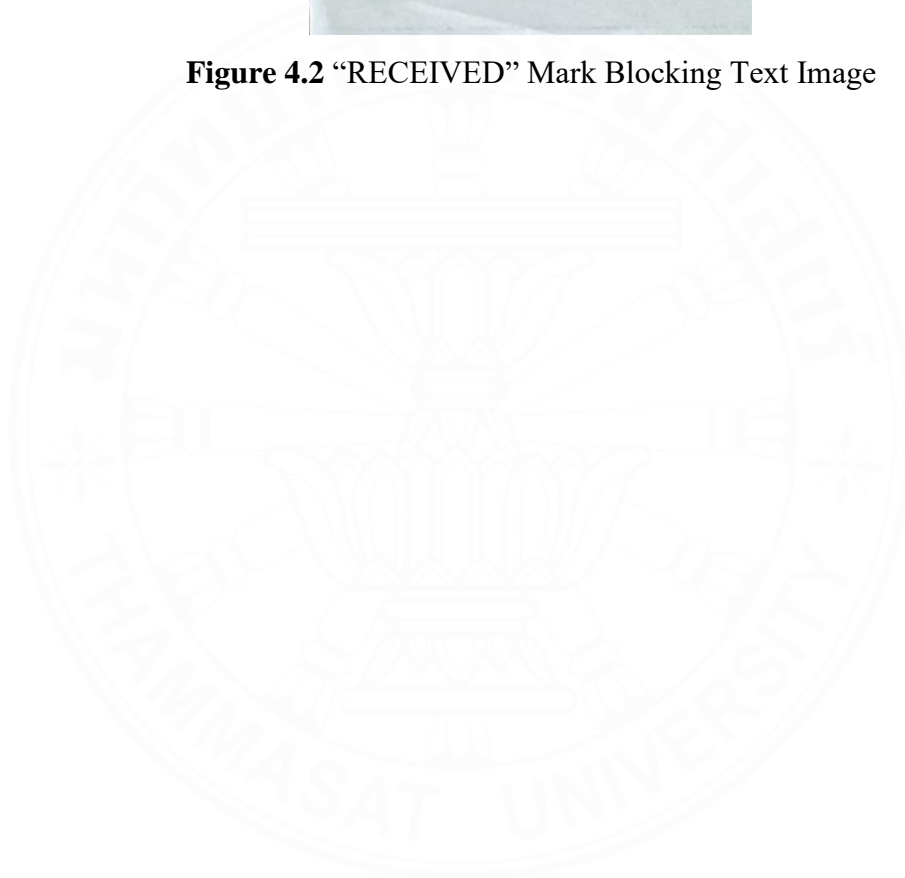
For the special case situation, the remaining 44 images are separated into 6 categories based on its specialty. There are 2 images that are blurred, 3 images that have “RECEIVED” marks blocking an interesting text, 11 images that their product labels are sealed by plastic film, 23 images that their product labels are overlapped by other unwanted labels, 6 images that product labels are placed too close to an edge of the product package which creates difficulty in detecting edge and structure of the product label, and 3 images that their product labels have a strap lying on. I tried to run all special case images through the model and it turned out that product label of 39 out of 44 images cannot be detected which means they technically cannot reach to the text extraction process. Only 5 images from two categories, including blurred image and “RECEIVED” mark blocking interesting text image, that pass through the label detection part. However, text information in these 5 images cannot be extracted correctly. For the first two images, text information on the image is too blurred to be correctly extracted. Figure 4.1 is an example of this case. For other three images, text information on the image has “RECEIVED” mark lying on the text which makes the model perform incorrect extraction. Figure 4.2 is an example of this case.



**Figure 4.1** Blurred Image



**Figure 4.2** “RECEIVED” Mark Blocking Text Image



## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

#### **5.1 Conclusion of the Project**

This study aims to develop an automatic warehouse inventory monitoring system that is able to first detect the product label that is stamped on the product package and then extract three interesting text information located in different locations on the label. There are then two models to be constructed which are label detection model and text extraction model. After the required Python libraries which are OpenCV and PyTesseract were installed onto a software program called Visual Studio code, the necessary functions of both libraries were studied so that they can be wisely selected to use. The construction of label detection and text extraction models was executed with a careful selection of functions in order to create models that correctly respond to the stated problem of the warehouse. These two models are also made to combine together to become only one model that can both detect label and extract three interesting text information with one click. At this point, objective number one has been fulfilled with the ability of this model that satisfies the requirement. However, this model needs to be evaluated to see whether its performance in term of accuracy can meet the standard level that was determined to be 85% or not.

In order to perform a performance evaluation, training dataset is another key factor apart from the automatic warehouse inventory monitoring system that has been constructed. Thanks to KNS Logistics Service Co., Ltd., a partner of this study, the collection of product label were collected from its warehouse to become the training dataset for the model. There are 105 images collected from the warehouse which can be separated into two categories. 61 images are classified as normal cases and 44 images are classified as special cases. Only 61 normal case images are in the focus of model performance evaluation while the rest plays as an extra role of this study. With the completed model in one hand and a training dataset in another, the performance evaluation can be executed. The class precision and class recall metrics were utilized to measure the performance.

There should be two parts of performance which are for label detection performance and text extraction performance. However, there are three sections of text to be extracted which are pallet tag, item code, and quantity number. So, there are totally four parts of performance to be reported. As a result, three out of four gather a 100% accuracy as a performance while the last one which is item code extraction performance can reach up to 88.5%. From these reported number, it is clear to claim that my automatic warehouse inventory monitoring system outperforms the standard accuracy level that was set at 85%.

## **5.2 Knowledge Gained**

Since this study is related to the current technology, computer vision, opportunity is what can be realized during the study of this topic. There are still many gaps in both academic and industrial aspect. Text extraction has been introduced to the world many years ago but people have applied it just to convert text in an image format to digital format. There are many more problems that can apply this technique to solve but they do not have awareness of this technology. Moreover, process of pre-processing before sending images to undergo the next station is super vital. This is a lesson teaching that preparation is much more important than an execution state.

For the academic perspective, image processing is seemed to be fundamental step to do before image is used. OpenCV library has covered almost everything that image should be preprocessed such as cropping, color changing, and resizing. It was not hard to understand how to write Python language according to OpenCV library; however, it is quite difficult to analyze which commands should be selected when encountering one problem. Tesseract OCR is another interesting library. Although there are less commands than OpenCV, its ability to perform its duty is powerful. From now on, text that can be seen everywhere as a picture or video can be converted to digital text format. This can be applied to companies that are trying to move themselves forward by converting traditional data storage to digital data storage too.



## 5.2 Recommendations and Future Works

There are two main aspects of the improvement that can be accomplished in the future. The first one is about how to improve the current model to be able to have a higher performance level. The second aspect is how to reduce the special case situations so that the model can be applied to more images. For the first aspect, discussing the change of font on product label with the warehouse should be done because it is a much easier solution than training the Tesseract engine to get used to and understand the difference between “5” and “S” and “0” and “O”. For the second aspect, there are six cases to deal with in order to convert special cases into normal cases. Overlapping of other labels on the interesting labels seems to be the most frequent problem since 23 out of 44 special case images have this problem. There might be more training on the warehouse staff to not stamp the product label over other labels or at least leave a space between the labels so that the label detection could be done freely. The next vital problem which creates 11 cases out of 44 is that there is a plastic film sealing over the product label which creates the difficulty in detecting label since there will be reflecting light on the image that reflects on the plastic film which can be considered as noise to an image. To solve this problem, the warehouse should rearrange the process whether to not have the plastic film sealing at all or seal the plastic film first and then attach the product label later. The next two issues are that the product label is attached at the edge of the product package and that sometimes there are a strap lying on the product label. These two issues can be solved easily through the policy of the warehouse for its staff to be more careful when attaching the product label and when wrapping a strap around the package not to interrupt the label. “RECEIVED” mark that blocks the interesting text is same kind of issue as well which requires more carefulness from the staff when stamping the mark onto the label not to interrupt the three interesting texts. Apart from these, the model has been working well in the normal situation with high performance level. So, the automatic warehouse inventory monitoring system is now useful and helpful for the readers to apply knowledge and techniques included in this study to their projects or tasks in order to develop new technological innovations that can help the industry sector work with higher efficiency in their warehouse management fields.

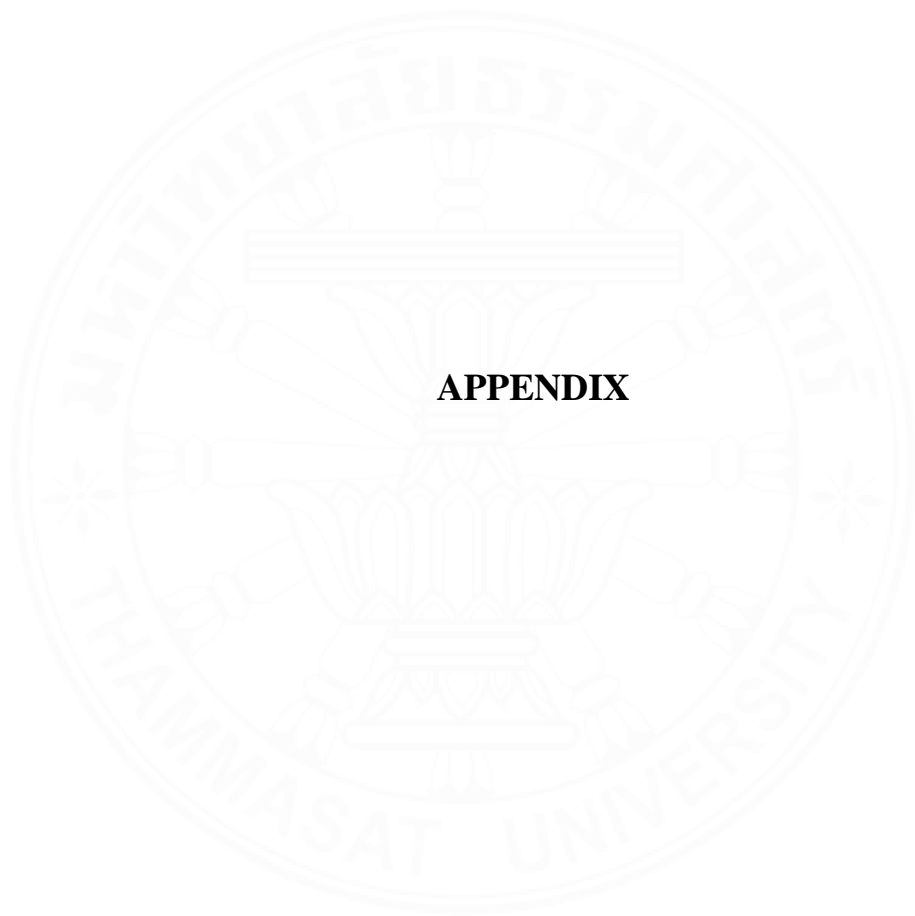
## REFERENCES

- Ben Sassi, N., Averós, X., & Estevez, I. (2016). Technology and poultry welfare. *Animals*, 6(10), 62.
- Fernandes, A. F. A., Dórea, J. R. R., & Rosa, G. J. D. M. (2020). Image analysis and computer vision applications in animal sciences: an overview. *Frontiers in Veterinary Science*, 7, 551269.
- He, W., Jiang, Z., Ming, W., Zhang, G., Yuan, J., & Yin, L. (2021). A critical review for machining positioning based on computer vision. *Measurement*, 184, 109973.
- Javaid, M., Haleem, A., Singh, R. P., Rab, S., & Suman, R. (2022). Exploring impact and features of machine vision for progressive industry 4.0 culture. *Sensors International*, 3, 100132.
- Karthikeyan, U., & Vanitha, M. (2019). A Study on Text Recognition using Image Processing with Datamining Techniques. *International Journal of Computer Sciences and Engineering*, 7(2), 1-6.
- Kumar, V., Kaware, P., Singh, P., Sonkusare, R., & Kumar, S. (2020, September). Extraction of information from bill receipts using optical character recognition. In 2020 International Conference on Smart Electronics and Communication (ICOSEC) (pp. 72-77). IEEE.
- Li, C., Li, J., Li, Y., He, L., Fu, X., & Chen, J. (2021). Fabric defect detection in textile manufacturing: a survey of the state of the art. *Security and Communication Networks*, 2021.
- Li, N., Ren, Z., Li, D., & Zeng, L. (2020). Automated techniques for monitoring the behaviour and welfare of broilers and laying hens: towards the goal of precision livestock farming. *animal*, 14(3), 617-625.
- Liu, W., Yuan, X., Zhang, Y., Liu, M., Xiao, Z., & Wu, J. (2020, June). An end to end method for taxi receipt automatic recognition based on neural network. In 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (Vol. 1, pp. 314-318). IEEE.

- Long, S., He, X., & Yao, C. (2021). Scene text detection and recognition: The deep learning era. *International Journal of Computer Vision*, 129(1), 161-184.
- Manoharan, S. (2019). A smart image processing algorithm for text recognition, information extraction and vocalization for the visually challenged. *Journal of Innovative Image Processing (JIIP)*, 1(01), 31-38.
- Mizan, C. M., Chakraborty, T., & Karmakar, S. (2017). Text Recognition using Image Processing. *International Journal of Advanced Research in Computer Science*, 8(5).
- Nasirahmadi, A., Edwards, S. A., & Sturm, B. (2017). Implementation of machine vision for detecting behaviour of cattle and pigs. *Livestock Science*, 202, 25-38.
- Natei, K. N., Viradiya, J., & Sasikumar, S. (2018). Extracting text from image document and displaying its related information. *J. Eng. Res. Appl*, 8(5), 27-33.
- Raveendran, S., & Chandrasekhar, A. (2022). Inspecting and classifying physical failures in MEMS substrates during fabrication using computer vision. *Microelectronic Engineering*, 254, 111696.
- Sidhwa, H., Kulshrestha, S., Malhotra, S., & Virmani, S. (2018, October). Text extraction from bills and invoices. In 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN) (pp. 564-568). IEEE.
- Skurowski, P., Nurzyńska, K., Pawlyta, M., & Cyran, K. A. (2022). Performance of QR Code Detectors near Nyquist Limits. *Sensors*, 22(19), 7230.
- Szeliski, R. (2011). Computer vision. Texts in computer science. *Texts in Computer Science*.
- Thorat, C., Bhat, A., Sawant, P., Bartakke, I., & Shirsath, S. (2022). A Detailed Review on Text Extraction Using Optical Character Recognition. *ICT Analysis and Applications*, 719-728.
- Wang, K., Babenko, B., & Belongie, S. (2011, November). End-to-end scene text recognition. In 2011 International conference on computer vision (pp. 1457-1464). IEEE.

Yindumathi, K. M., Chaudhari, S. S., & Aparna, R. (2020, July). Analysis of image classification for text extraction from bills and invoices. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.





**APPENDIX**

## APPENDIX A

### PYTHON SCRIPT OF AN AUTOMATIC WAREHOUSE INVENTORY MONITORING SYSTEM

```

import cv2, numpy as np, pytesseract, openpyxl, xlswriter, pandas as pd
pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files\\Tesseract-
OCR\\tesseract.exe'

widthImg = 640
heightImg = 640
#####
inputPic = "Training_Dataset/IMG_1533.jpg"

def preProcessing(img):
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (5,5), 1)
    _,imgThreshold = cv2.threshold(imgBlur,200,250,cv2.THRESH_BINARY)
    kernel = np.ones((5,5))
    imgThres = cv2.dilate(imgThreshold, kernel, iterations=2)
    return imgThres

def getContours(img):
    biggest = np.array([])
    maxArea = 0
    countours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
    for cnt in countours:
        area = cv2.contourArea(cnt)
        if area > 100000:
            print(f"This is selected area: {area}")
            perimeter = cv2.arcLength(cnt, True)
            print(f"This is perimeter: {perimeter}")
            approx = cv2.approxPolyDP(cnt, 0.02*perimeter, True)
            print(approx)
            if area > maxArea and len(approx) == 4:
                biggest = approx
                maxArea = area
                cv2.drawContours(imgContour, cnt, -1, (255,0,0), 30)
    cv2.drawContours(imgContour, biggest, -1, (255,0,0), 100)
    return biggest

def reorder(myPoints):
    myPoints = myPoints.reshape((4,2))
    myPointsNew = np.zeros((4,1,2), np.int32)
    add = myPoints.sum(1)
    myPointsNew[0] = myPoints[np.argmin(add)]
    myPointsNew[3] = myPoints[np.argmax(add)]

```

```

diff = np.diff(myPoints, axis=1)
myPointsNew[1] = myPoints[np.argmin(diff)]
myPointsNew[2] = myPoints[np.argmax(diff)]
return myPointsNew

def getWarp(img, biggest):
    biggest = reorder(biggest)
    pts1 = np.float32(biggest)
    pts2 = np.float32([[0,0], [widthImg,0], [0,heightImg], [widthImg,heightImg]])
    matrix = cv2.getPerspectiveTransform(pts1,pts2)
    imgOutput = cv2.warpPerspective(img, matrix, (widthImg,heightImg))

    imgCropped = imgOutput[20:imgOutput.shape[0]-20, 20:imgOutput.shape[1]-20]
    imgCropped = cv2.resize(imgOutput,(widthImg,heightImg))

    return imgCropped

def stackImages(scale,imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range(0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape[:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale,
scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y],
(imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)
                    if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor(
imgArray[x][y], cv2.COLOR_GRAY2BGR)
                    imageBlank = np.zeros((height, width, 3), np.uint8)
                    hor = [imageBlank]*rows
                    hor_con = [imageBlank]*rows
                    for x in range(0, rows):
                        hor[x] = np.hstack(imgArray[x])
                    ver = np.vstack(hor)
            else:
                for x in range(0, rows):
                    if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                        imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
                    else:
                        imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1],
imgArray[0].shape[0]), None,scale, scale)
                        if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x],
cv2.COLOR_GRAY2BGR)
                    hor = np.hstack(imgArray)
                    ver = hor

```

```

    return ver

img = cv2.imread(inputPic)
imgContour = img.copy()

imgThres = preProcessing(img)

biggest = getContours(imgThres)

if biggest.size != 0:
    imgWarped = getWarp(img, biggest)
    imgArray = ([img, imgThres],
                [imgContour, imgWarped])
    cv2.imshow("Result", imgWarped)
else:
    imgArray = ([img, imgThres],
                [img, img])
stackedImages = stackImages(0.1, imgArray)

cv2.imshow("Work Flow", stackedImages)
cv2.imwrite(f"Output_1/{inputPic[21:25]}_label.jpg", imgWarped)

labelPic = f"Output_1/{inputPic[21:25]}_label.jpg"
image = cv2.imread(labelPic)

# Crop TAG (having new image as [y1:y2 , x1:x2])
tag = image[270:310, 10:260]

# Crop ITEM CODE
code = image[320:360, 160:380]

# Crop QTY NUMBER
qty = image[500:580, 130:360]

cv2.imshow("tag", tag)
cv2.imshow("item code", code)
cv2.imshow("qty", qty)
cv2.imwrite(f"Output_1/{inputPic[21:25]}_tag.jpg", tag)
cv2.imwrite(f"Output_1/{inputPic[21:25]}_code.jpg", code)
cv2.imwrite(f"Output_1/{inputPic[21:25]}_qty.jpg", qty)

cv2.waitKey(0)

# Open image file
tag = f"Output_1/{inputPic[21:25]}_tag.jpg"
code = f"Output_1/{inputPic[21:25]}_code.jpg"
qty = f"Output_1/{inputPic[21:25]}_qty.jpg"

# Recognize text using Tesseract
tagText = pytesseract.image_to_string(tag)
codeText = pytesseract.image_to_string(code)

```



```
qtyText = pytesseract.image_to_string(qty)

# Print recognized text
print(f"Product Tag: {tagText} \nItem code: {codeText} \nQTY Number: {qtyText}")

# Load the Excel file
workbook = openpyxl.load_workbook('DataBase_Checking.xlsx')

# Select worksheet
worksheet = workbook['Sheet1']

# Write a value to specific cells
worksheet['B3'] = tagText
worksheet['B4'] = codeText
worksheet['B5'] = qtyText

# Save the changes
workbook.save('DataBase_Checking.xlsx')
```



## BIOGRAPHY

Name Nitiwat Ananvaithayakij  
Education 2022: Bachelor of Engineering (Industrial Engineering)  
Sirindhorn International Institute of Technology  
Thammasat University

