BARCODE DETECTION AND DECODING IN ON-LINE FASHION IMAGE

A Thesis

Submitted to the Faculty

of

Purdue University

by

Qingyu Yang

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Jan P. Allebach, Chair

School of Electrical and Computer Engineering

Dr. Michael D. Zoltowski

School of Electrical and Computer Engineering

Dr. Joseph V. Rispoli

School of Biomedical Engineering and Electrical and Computer Engineering

Approved by:

Dr. Pedro Irazoqui

Head of the School Electrical and Computer Engineering Program

This is the dedicated to my parents Xingshan Yang and Huili Chu. Thanks for all your love and supports to my life.

ACKNOWLEDGMENTS

I would like to thank my major professor Jan P. Allebach for giving me the opportunity to work with him. Without his advice, help, encouragement, this research would not happen. I want to thank Prof. Michael D. Zoltowski helps and encourages me in my graduate program as well. I also would like to appreciate professor Joseph V. Rispoli gives me the sufficient skills and confidence from my starting of academic research.

I sincerely thank PoshMark research team, especially Zhi Li, Litao Hu, and Zhenxun Yuan, they give me helpful and valuable advice to me in this research work. Also, I would like to thank all my friends and the EISL group. The encouragement and help they gave me is an essential part of completing my thesis master program.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

A input image

I_g grayscale image

G_x gradient image in x axis

G_y gradient image in y axis

B binary image

T threshold

T_i i-th threshold

$feature_i$ i-th extracted feature

C_{max} the largest foreground cluster

C_i the region in input image corresponding to the largest foreground cluster

N Number of Images

TP True positives

TN True negatives

FP false positives

FN false negatives

ABBREVIATIONS

Acc	Accuracy
1-D	One-dimensional
2-D	Two-dimensional
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
RPN	Region Proposal Network
NMS	Non Maximum Suppression
IoU	Intersection over Union
GUI	Graphical User Interface
GPU	Graphics Processing Unit

ABSTRACT

Yang, Qingyu Master, Purdue University, May 2019. Barcode Detection and Decoding in On-line Fashion Image. Major Professor: Jan P. Allebach.

A barcode is the representation of data including some information related to goods, offered for sale which frequently appears in markets. Especially in the on-line fashion market such as the buy and sell market, barcodes on the tags of the sale items support identified information including producer, manufacturer, etc. The market need a system to automatically detect and decode barcode in real time. However, the existing method has a limitation in detecting 1-D barcode in some backgrounds such as tassels, stripes, and texture in fashion images. In this research, a focus is on identifying the barcode and distinguishing a barcode from its similarities. It is accomplished by adding a post-processing technique after morphological operations in the traditional method based on the hand-crafted features. Convolution Neural Network (CNN) is applied to solve this typical objective detection problem. The proposed algorithm has been validated using several examples. In addition, the performance and the results of the proposed algorithm have been compared with the other methods presented in the literature.

To decode a barcode, a Python-supported package including the existing common types of decoding schemes is widely used to decode the barcode. However, this commonly-used package has limitations in decoding the skewed barcodes. A pre-processing transformation step is added to process the strongly skewed barcode images in order to improve the probability of decoding success.

1. INTRODUCTION

Barcodes always carry essential information for their corresponding products in manufacture. Different types of barcode are used for various industries. In general, there are two types of barcode1-D barcode and 2-D barcode (Fig. 1.1). 2-D barcodes such as QR code and PDF 417 are developed based on the 1-D barcodes. The 2-D barcodes represent data by the shape and symbols. Also, each region in a 2-D barcode carries more information than the information in the 1-D barcode. In this project, we only focus on the 1-D barcode since it has a more extensive history and is more commonly used in industries. For the 1-D barcode, there are many types such as EAN-13, CODE 39, etc. The shapes of the typical 1-D barcodes are rectangles and consist of black and white stripes. Only minority barcodes include colored stripes. All 1-D barcodes record data by a variety of widths and spacings in parallel lines. The decoding scheme and detecting algorithms are based on these characteristics as well.

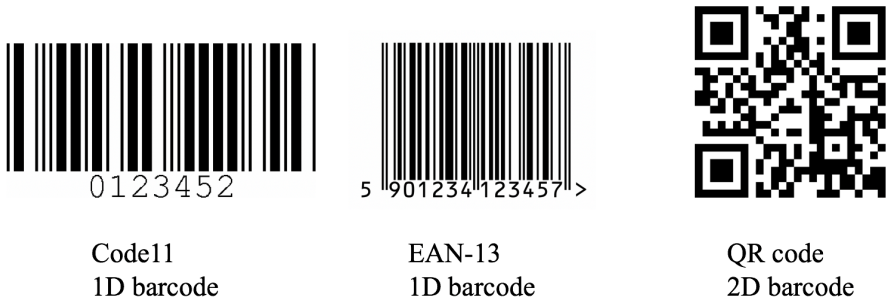

Fig. 1.1.: Examples of barcode types

1.1 Motivation

With the significance and a variety of applications, detecting and decoding barcode is much useful for online shopping platforms. As a result, they need to process a tremendous amount of images containing barcodes. Also, it would be labor-intensive to label and decode barcodes manually. The system of detecting and decoding barcodes has widely used in market and software development. However, the current system still has some limitations with skewed, small and rotated barcodes. For example, scanning always fails when the detecting device is far away from the barcode. Thus, getting more accurate on detecting and decoding barcodes becomes a useful question. In this project, we focus on the identification of barcode in online fashion images and decoding it. The goal is developing a robust automatic system to detect and decode barcodes. There are some common patterns used in fashion design such as tassels, brands logos, cells and stripes misclassified as barcodes in fashion images. For facing this challenge, we focus on distinguishing barcode and its similarities.

1.2 Related Work

1. Barcode detection in traditional method work

According to the characteristics of a 1-D barcode, Adrian Rosebrock realized the detection by implement the image processing method. The general idea of this work is based on edge detection to get the gradient image which represents the intensity of the image. The region that has concentrated parallel lines in an image will have the largest intensity. Then the morphological transformation will be used to detect the region that has the largest intensity. The detected region corresponds to the localization of the barcode in the original image. For obscure barcodes, many previous works add preprocessing in detection to get better results. Although the localization of barcode can be realized in this method, it has limitations when the image does not include a barcode. Alternatively, if the image has complex backgrounds, the results would be wrong.

2. Barcode detection in deep learning approach

CNN models which can extract more features from an image can largely improve the accuracy in object detection task. In 2015, Faster R-CNN [1] has been proposed in NeuralPS by S. Ren, etc. In 2017, J. Li, etc. implement Faster R-CNN to detect 1-D barcode and get higher accuracy [2]. However, their work based on the dataset which includes at least one barcode and clear backgrounds in each image. In other words, trivial scenes contain much more complicated backgrounds and bring detection task into a more challenging phase.

3. Barcode Decoding with Pyzbar

There are different barcode decoding packages in Matlab. For example, Aygun Baltaci developed the decoding solution for Code 39. Based on the scheme of different barcode types, the decoding solution is generated by transferring the number of pixels of parallel lines to a digital number. In addition, there is an online open source Pyzbar in Python package developed by Lawrence Hudson. This package can decode common types of barcodes. Also, lightly skewed and occluded images can also be decoded directly without any pre-processing.

1.2.1 The Proposed Solution

To detect and decode barcodes, the system will be constructed as two major parts as shown in Fig. 1.2. The first part is barcode detection. Then we use the detected result to crop the barcode region and apply our decoding part to generate the information in real time. Since stripes and barcodes share a lot of similar features, many stripe patterns will be mistakenly recognized as barcodes with the previous detecting system. Except for the influence of various background, small, obscure and occluded barcodes in the background also fail the system. We developed the traditional method based on hand-crafted features and implement a CNN model to reduce the limitations in previous detecting system. In the decoding part, we use a Pyzbar package to decode the common types of barcodes. Also, for the strongly

skewed barcode images, we propose the algorithm of perspective transformation that can warp barcode images to the unskewed position.

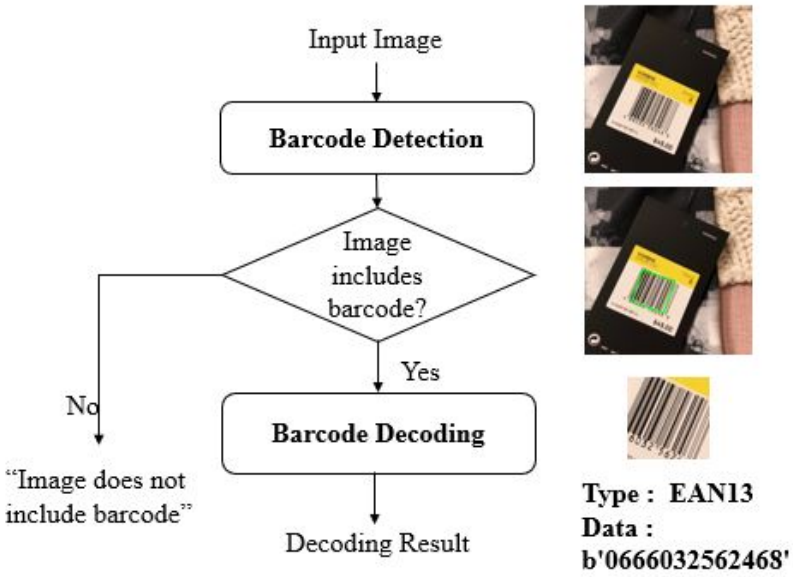

Fig. 1.2.: Overall structure of barcode detection and decoding system

2. BARCODE DETECTION SYSTEM

2.1 Barcode Detection with hand-crafted traditional method

The first development is the barcode detection system. Based on the previous work on detecting barcode, we implement both traditional method and deep learning approach to increase our detecting accuracy. According to the characteristics of general 1-D barcode, the classification of the barcode image and the non-barcode image has been realized. The first part will introduce the traditional method based on hand-crafted features. Then the second part will be the implement of using deep learning approach to extract features for identifying barcodes.

2.1.1 Construction of the detection system

The general idea of detecting barcode is using edge detection and morphological transformation [3] to figure out a cluster in an image most likely to be a barcode. Then we add post-processing to identify if the selected cluster corresponds to a barcode in the original image. The flowchart in Fig. 2.1 shows a general structure of the detecting system in the traditional method. After morphological transformation, the largest foreground cluster will be treated to have the largest possibility of corresponding to a barcode. In this system, the cluster is represented as C_{max}. We add a post-processing step to identify the pattern of matching C_{max} by setting multi-thresholds. The identifying system shows the results by bounding the detected barcode region. Fig. 2.1 displays an example of the detecting system. The original image is generated from Poshmark website.

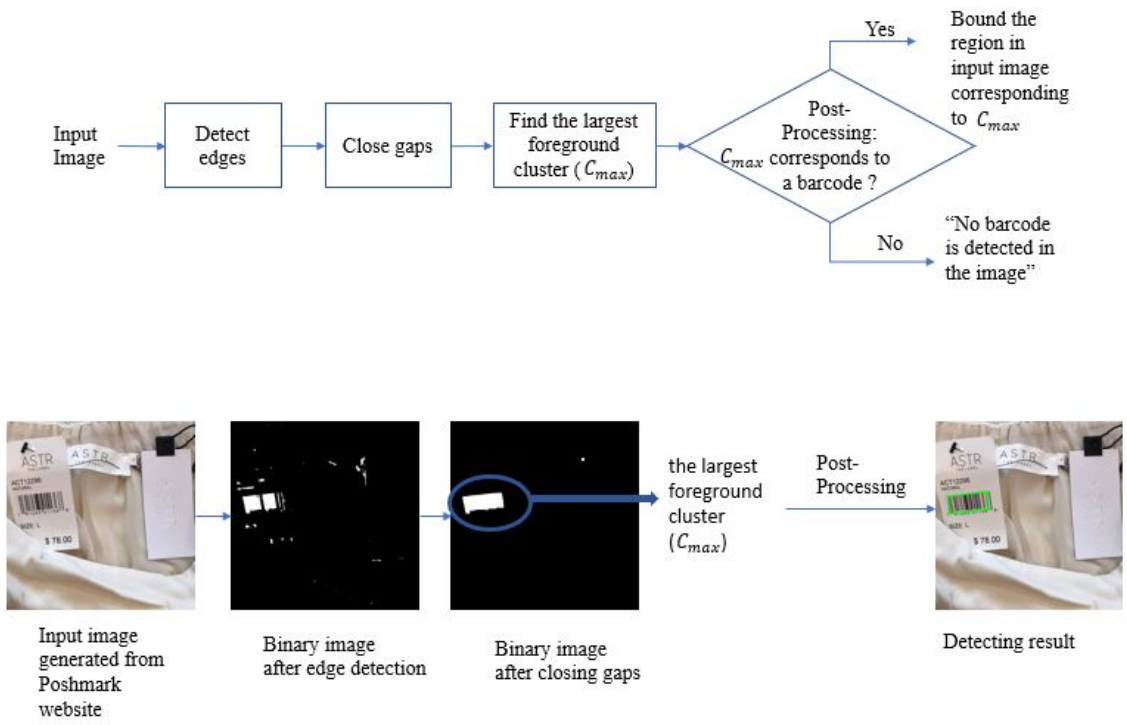

Fig. 2.1.: Structure and example of detecting barcode with hand-crafted traditional method

2.1.2 Edge Detection

A barcode consists of many parallel lines. The small edges between the lines can be detected by edge detection. For a 2-D color image, the first step to detect edge is changing color space from RGB space to grayscale (I_g) space by combining R, G, and B channels to a single channel using equation (2.1). In order to detect edges in the input image, we compute gradient image [4] to realize it. Because the gradient image represents the changing of intensities at each pixel. Sobel operator [5] is applied to the grayscale image to get gradient value in x and y direction at each pixel. The pixel in the gradient image is presented as G. Equation 2.2 shows the convolution of getting gradient image by 3×3 Sobel operator. According to a visible barcode has

smaller edges, a 9 x 9 Sobel operator is used in this method. But if we use a 3 x 3 Sobel operator, the gradient image will include more details which are not expected.

$$I_g = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2.1}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I_g \quad \text{and} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & 0 & -2 \end{bmatrix} * I_g \qquad G = \sqrt{G_x^2 + G_y^2} \tag{2.2}$$

Fig. 2.2 shows an example of edge detection. The gradient image is describing the intensity of our input image. It is obvious that the cluster corresponding to a barcode has been described based on the rapid change of intensities. Then the binary image splits the foregrounds and backgrounds. A tested threshold (T) which equals to 225 is used to get the binary image. If the pixel is larger than 225, we set the corresponding pixel to 1 in the binary image. Otherwise, we set it to 0. Before the threshold, an averaging filter [6] with 9×9 kernel is added to blur the gradient image for smoothing out the high-frequency noise. The results of the edge detection are binary images including some clusters. White pixels represent the foregrounds, and black pixels represent the background.

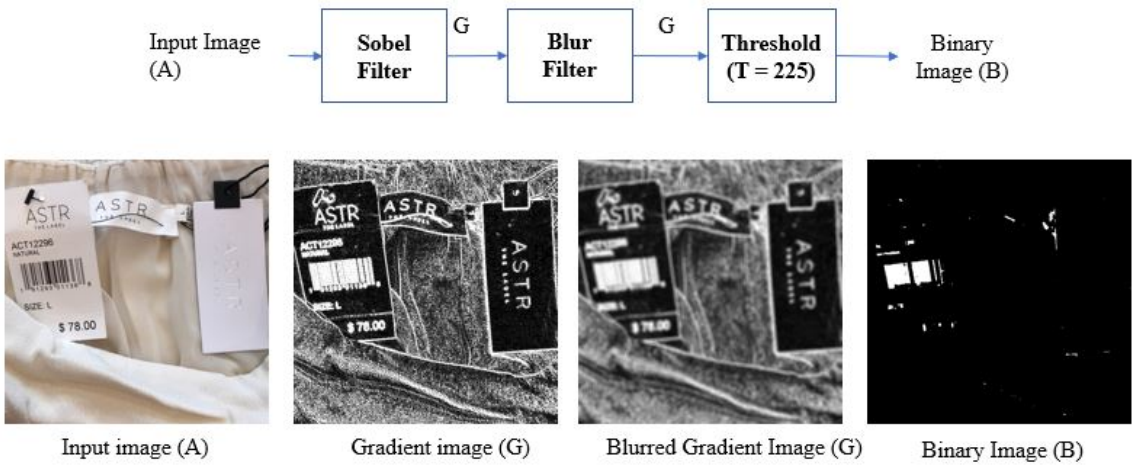

Fig. 2.2.: Structure and example of edge detection for barcode

2.1.3 Closing Gaps

After edge detection, we apply morphological transformation to fill out the narrow gaps between the white-pixel clusters in binary image. Morphological transformation includes several operations such as dilation, erosion, opening and closing to process the input images and get the output images with the same dimensions. Those operations are based on the neighboring pixels around the anchor pixels. The dilation and erosion operators [7] are applied for the binary image after detecting edges. erosion operates the object to make the size to be smaller. The algorithm in binary image is setting a pixel to 0 if any of the neighboring pixels have the value 0. In general, dilation makes the anchor clusters more visible and connect the neighboring clusters. So we use it to fill out the small spacings in the foreground clusters. In the binary image, white pixel is set to 1 and black pixel is 0. The dilation is setting the value of the anchor pixel to 1 if any neighboring pixels in the kernel have the value 1.

We construct a series of erosion and dilation to close the gaps of the foreground cluster in our binary image. The 21×7 kernel is applied to implement the operations. Because the kernel including larger width to better close the gaps between the parallel lines in vertical. With this specified kernel, erosion is used to remove the noises of white pixels. Then we use dilation to remove the small black pixel in the gaps and extend the larger white region. The final binary image is constructed by four erosion operations and four dilation operations.

The largest foreground cluster (C_{max}) after applying the morphological operations will be treated as the region that has the largest probability of corresponding to a barcode. For testing the algorithm without post-processing, we randomly download 200 fashion images from online shopping platforms. The region corresponding to the largest foreground cluster in image is labeled by green box. Without post-processing, Fig. 2.3 shows some examples for successful detection that correctly localize the barcodes. Fig. 2.4 displays some failure cases that the bounding regions are corresponding to stripes, grids which is similar to barcodes. But the labeled regions are not actually barcodes.

Fig. 2.3.: Successful cases from traditional method without post-processing

Fig. 2.4.: Failure cases from traditional method without post-processing

2.1.4 Post-processing

Fig. 2.4 displays the processing can figure out the region including the most stripes in the image. However, it does not achieve our goal that identify if any barcode is included in the image. Thus, we add a post-processing subsystem after finding the largest foreground cluster (C_{max}). The region with corresponding to C_{max} in input image is represented as C_i here. The post-processing is constructed by 4 steps to check if C_i matches the features in regular barcodes. Fig. 2.5 displays the construction. Through the comparison of 4 hand-crafted features ($feature_1,\ feature_2,\ feature_3,\ feature_4$) extracted from C_i and the defined thresholds ($T_1,\ T_2,\ T_3, and\ T_4$), C_i has been predicted as a barcode or not. All the conditions are based on the features from typical 1-D barcode. The following describes the four conditions for checking C_i.

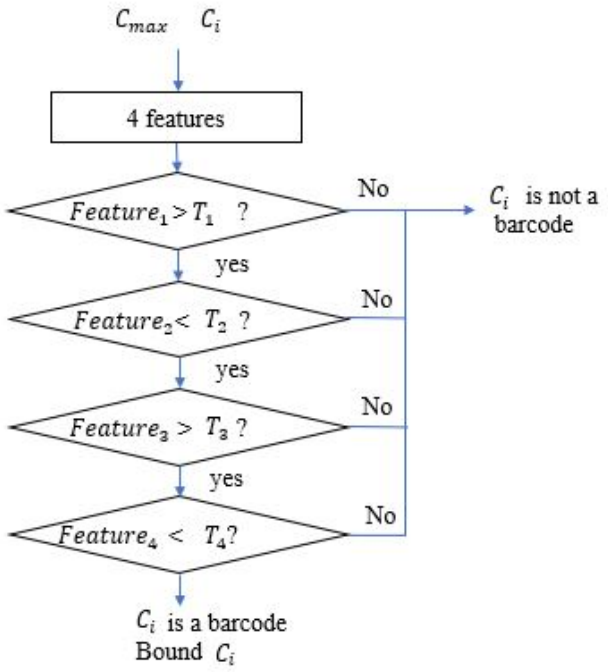

Fig. 2.5.: Structure of algorithm in post-processing

- Area of Barcode T_1

  From the 200 testing results, some failure cases in the images without a barcode have a small bounding box. Thus, the feature which represents the dimension of the largest foreground cluster is used to check if C_{max} corresponds to a visible barcode. As Equation 2.3, if the number of pixels in C_{max} is larger than T_1, the C_i will pass the first check.

$$feature_1 = \#\ of\ pixels\ in\ C_{max} \tag{2.3}$$

- Shape of Barcode T_2

  It is visible that the contours of typical barcodes are similar to a rectangle. With the general shape of barcode, we select the second threshold based on the comparison between the shape of C_{max} and a rectangle. We use implementation in OpenCV to calculate the area of the minimum bounding box of C_{max}. $feature_2$

is calculated by the radio of the number of pixels in bounding rectangular area and the number of pixels in C_{max} to be the second checking value (Equation 2.4). The number of T_2 is identified by testing 200 images.

$$feature_2 = \frac{\#\ of\ pixels\ in\ C_{max}}{\#\ of\ pixels\ in\ bounding\ rectangle} \tag{2.4}$$

- Corners in Barcode T_3

  A barcode is formed by black-and-white lines with different spacing and width. Because each line includes 4 corners, a barcode includes dozens of corners. Thus, detecting corners also helps us to identify if C_i is a barcode. We check this by applying Harris corner detection [8]. In addition, as the barcodes have different size in an image. The radio of the number of detected corners and the number of pixels in C_i is set to be checked (Equation 2.5). If C_i includes the enough corners that $feature_3$ larger than $T3$, it will pass third check.

$$feature_3 = \frac{\#\ of\ corners\ in\ C_{max}}{\#\ of\ pixels\ in\ C_{max}} \tag{2.5}$$

- Color Information in Barcode T_4

  Typical image of barcode is a binary image which only includes black and white pixels. Grayscale image uses pixel value which is a single number to represent the brightness of each pixel. The pixel values have a range from 0 to 255. 0 is taken to be black and 255 is taken to be white. In real world, barcodes shown in images always be 2-D color images which have three Channel R, G and B due to the effect of illumination, background when capturing picture. But it always looks similarly to the typical barcodes that we can identify the barcode by visual perception. Fig. 2.6 displays that real barcode region has smaller difference between R, G and B values than colorful stripes. Thus, we can use calculated averaging difference between R, G and B channels shown in Equation(2.6) to approximate the color information of C_i.

Fig. 2.6.: RGB values in colorful stripes and barcode region

$$feature_4 = \frac{\sum_{i=1}^{N}((R-G)+(R-B)+(G-B))}{N}, N = \# \tag{2.6}$$

To select the values of four thresholds, we test 200 images to obtain the specific numbers for higher accuracy. Table 2.1 displays the selected number.

Table 2.1.: Selected Values for thresholds in Post-processing

	Value	**Requirement**
Area of Cluster (T_1)	800	$> T_1$
Shape of Cluster (T_2)	0.6	$< T_2$
Corners in Cluster (T_3)	0.001	$> T_3$
Color Information in Cluster (T_4)	70	$< T_4$

2.1.5 Test Performance

To test the performance of our algorithm, 1000 images are randomly downloaded from Poshmark (an online shopping) website. Fig. 2.7 displays some examples of our testing images. The barcode in each image are expected to be bounded. Otherwise, there will be no bounding box in the image. In the results, the system will make the testing images to 2 classes. One is the image including at least one barcode, the other one is the image without any barcode. The detecting system is expected to bound barcode region. If the image does not have any barcodes, the system is expected to be no labels.

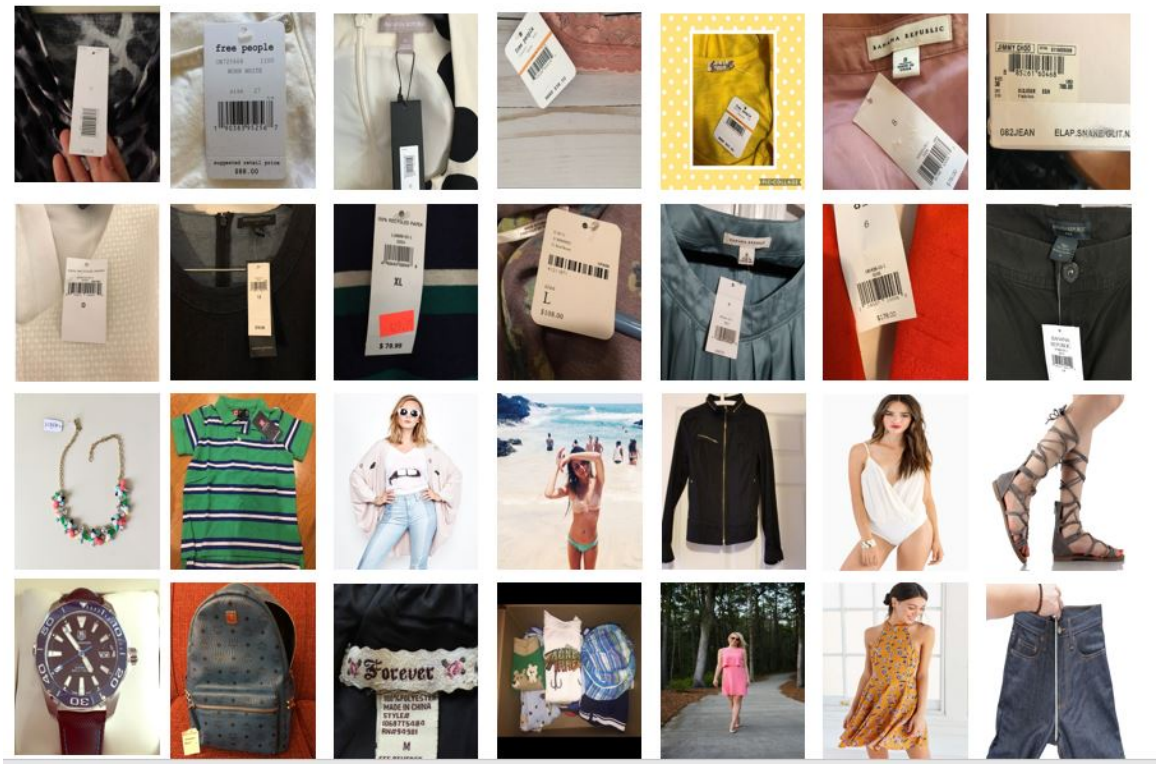

Fig. 2.7.: Examples of training dataset

In typical objective detection problem, true positives, true negatives, false positives and false negatives are used to show the performance of our classification system [9]. In this case, they are defined as the following:

- True Positives (TP): Image includes at least one barcode; The system detects the barcode in image.
- False Positives (FP) Image does not have any barcode; The system detects a wrong barcode in image.
- True Negatives (TN): Image does not have any barcode; The system does not label any region
- False Negatives (FN): Image includes at least one barcode; The system does not detect the barcode and it does not label any region

From analysis the results of 1000 testing images, the overall accuracy (Acc) is 82.69% and the testing process takes 44.52 seconds. Accuracy is calculated by equation 2.7. N_{TP} represents the number of true-positive cases. Also, N_{FP}, N_{TN} and N_{FN} are in similar. According to these values, the confusion matrix [10] which gives a visualization of algorithm performance is displayed in Fig. 2.8

$$Acc = \frac{TN + TP}{\#\ of\ total\ images} \tag{2.7}$$

	Predicted: No	Predicted: Yes	
Actual: No	TN = 755	FP = **83**	838
Actual: Yes	FN = **89**	TP = 73	162
	844	156	

Fig. 2.8.: Confusion matrix of testing 1000 images with traditional method

Fig. 2.9 and Fig. 2.10 display examples the true positives and the true negatives.

Fig. 2.9.: True positives with traditional method

Fig. 2.10.: True negatives with traditional method

Fig. 2.11 displays some false negatives. Some barcodes shows small and unclear in the images, the system cannot detect it. Also, if the backgrounds includes more obvious stripes than the barcode, the system cannot label the exact barcode region. Because the traditional method is based-on the intensities of images, unclear barcode region is hard to be detected. Fig. 2.12 shows examples of false positives. The method of threshold in post-processing still have limitations. Using single number to threshold cannot cover all the different conditions in fashion images.

Fig. 2.11.: False negatives with traditional method

Fig. 2.12.: True Positives with traditional method

2.2 Barcode Detection with deep learning method

2.2.1 Network: Faster R-CNN

In recent year, as the developing of deep learning, Convolutional Neural Networks (CNNs) [11] reach significant success in image classification task. Based on this success, many groups around the world try to use CNNs to solve image image recognition task and many of them already achieved impressive performances. In this project, we adopt Faster R-CNN [1] and modify it for fitting better in our task.

Before introducing the Faster R-CNN, we first illustrate the CNNs structure and functions. As shown in Fig. 2.13, the input will be did "convolution" operation with

the kernel. In practice, it is actually a cross-correlation operation. Therefore, it is a linear operation to the input. After "convolution", an non-linear active function will be applied to the output. The combination of "convolution" and active function, can non-linear transfer the input image to the feature spaces. And this non-linear transformation can be used to extract high-level features in images.

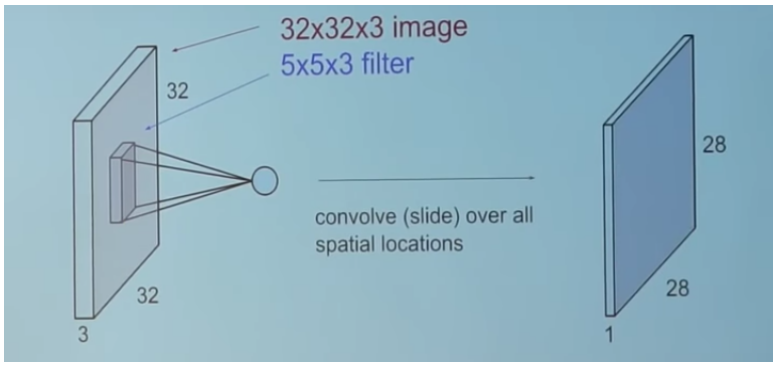

Fig. 2.13.: Convolution Operation

Barcode detection is a simplified image recognition task. In this case, we only need to localize and identify one kind of object: barcode. As shown in Fig. 2.14, Faster R-CNN consists of convolutional layers, Region Proposal Network(RPN), and classifier. The convolutional layer is used to extract many useful high-level feature in image. These features will be quantified to numbers and used by RPN and classifier. There are many different CNN models in image classification task. Since those CNN models have very good performances in image classification, the convolutional layers of them have strong ability to extract features in images. Therefore, we use the convolutional layers in inception-v2 [12] in our Faster R-CNN for barcode detection.

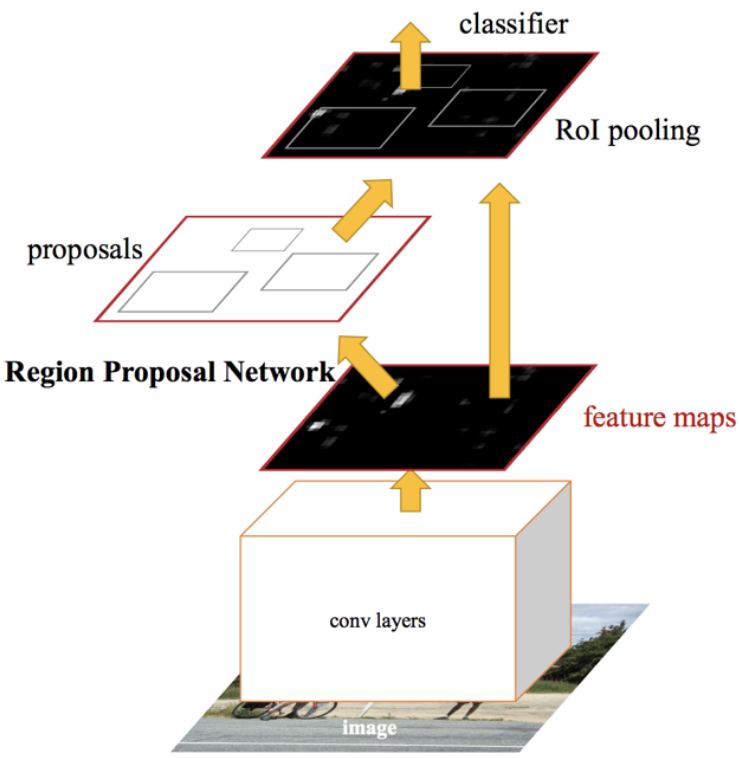

Fig. 2.14.: Faster R-CNN

In Faster R-CNN, the RPN is a one-layer network for region proposal. And the classify is a networks for identify the barcode. After convolutional layers, our system maps the input image to a high-dimension feature space, the feature map. The function of the RPN is selecting regions which contain barcode features. For each feature map, RPN will propose 9 different size sliding-windows, as shown in Fig. 2.15. Those selected region in feature map will remapped back to image. Therefore, thousands of bounding boxes will be generated in the input image. For each region, classifier will assign a score to them. The score is more like the probability of containing barcode in this region. There are thousands of bounding boxes generated, as shown in Fig. 2.16, which is a face detection example. Many regions around face will also be assign a very high score. In this case, we only need one bounding box. We used Non Maximum Suppression (NMS) to deal with this problem. As shown in Fig. 2.17, if the

Intersection over Union (IoU) is larger than the threshold, the regions with smaller score will be removed.

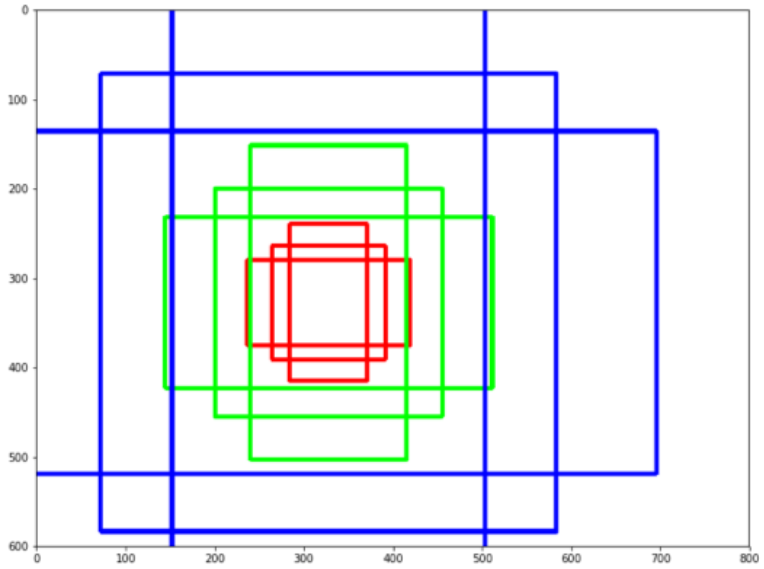

Fig. 2.15.: Different Anchor Size

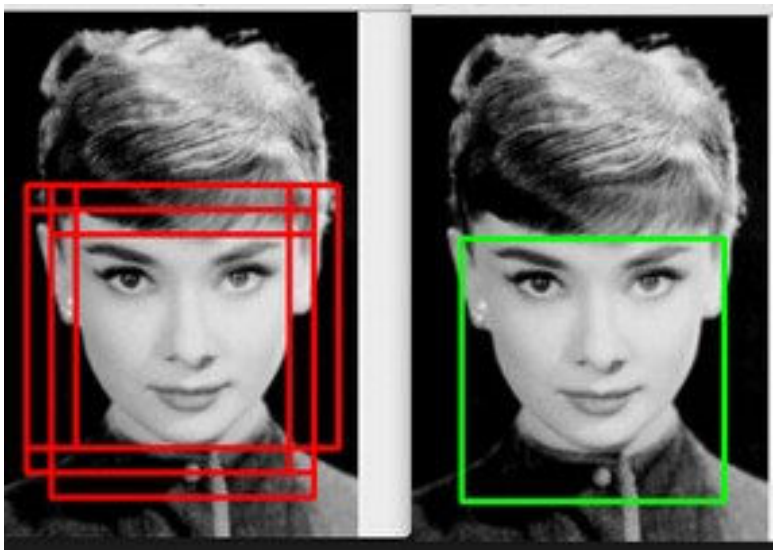

Fig. 2.16.: NMS Example in Face Detection

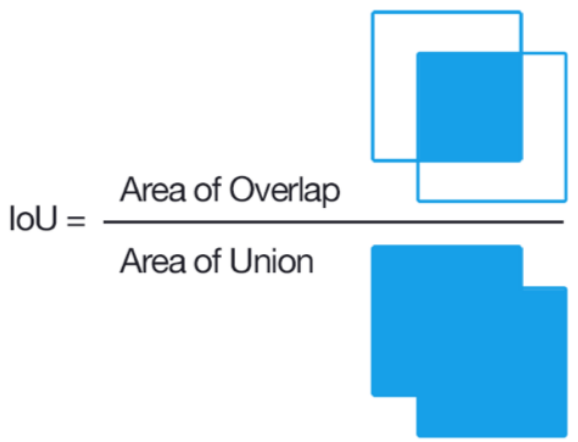

Fig. 2.17.: Intersection over Union

The classifier in Faster R-CNN is quite similar to the structure of other CNN model for image classification task. It is fully-connected layer. In our case, we only have one class, it is like a regression process that regress all those features in the region to one number. In our project, we also add an extra class to our network later, which will be illustrated in section 2.2.2

2.2.2 Adding New Class as Hard Negative Samples

In our experiment, we found that though the CNN model is powerful and get a good result. It still has large space for improving. After analyzing the false positive result, the selected regions do not have barcode, we found that our networks sometimes treat the clothes that containing stripe patterns as barcode. The reason is stripe pattern and barcode share many similar visual features. In other words, stripe patterns are hard negative samples. We should let our network pay more attention to this. To help our networks, Faster R-CNN, to focus more on the different features between barcode and stripe pattern. We modify the networks to recognize both barcode and stripe pattern and only draw bounding box for barcode regions. Based on this design, the classifier should focus on a classification task. The feature space should also contain more features to help classify barcode and stripe pattern.

2.2.3 Dataset Preparation

The total number of images in training dataset is 700. 500 images include barcodes and 200 images include stripes. For better matching our goal which is detecting barcode in fashion images. The data set is generated by taking pictures containing barcodes with varying backgrounds at fashion stores. The backgrounds include kinds of element in fashion stores such as clothes, floors, packages, etc. It also includes small, rotated and occluded barcode with different photographing angles and varying illumination.

We would like to obtain a model that can better distinguish the stripes and barcodes. The stripes will be treated as hard negative samples in our experiment. Thus, 200 online images with clear and colorful stripes are added to our training dataset.

The ground truth of the 700 training data is created by manual labeling. We use an open source toolbox to draw bounding boxes to label barcode regions and stripes with two different classes in each training image. Fig. 2.18 shows the GUI of our labeling tool. After using labeling tool, an extensible markup language (XML) data file recording the coordinates of labeling rectangular boxes will be generated for each training image. Fig. 2.19 shows some examples in our training images with barcodes. Fig. 2.20 shows some examples of images with stripes in the training dataset.

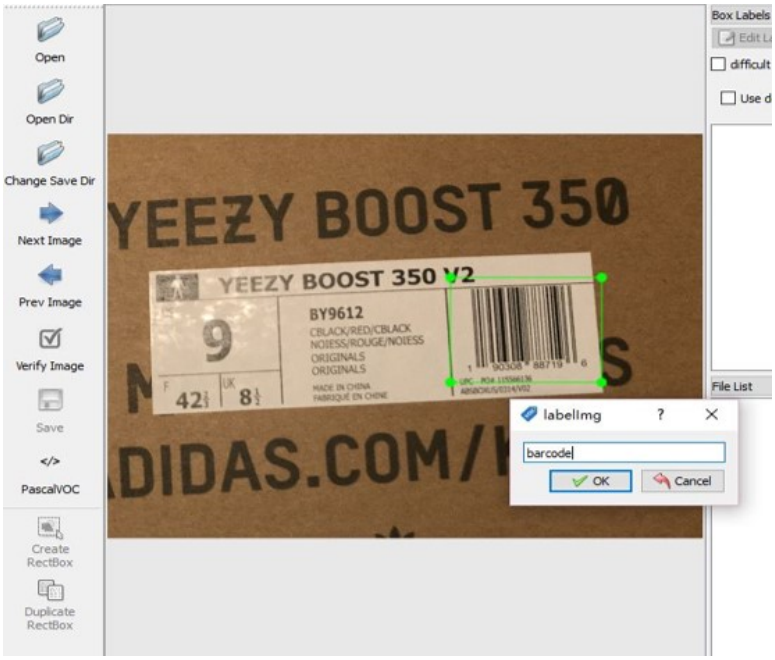

Fig. 2.18.: GUI of labeling tool

Fig. 2.19.: Examples of training dataset of "barcode" class with deep learning approach.

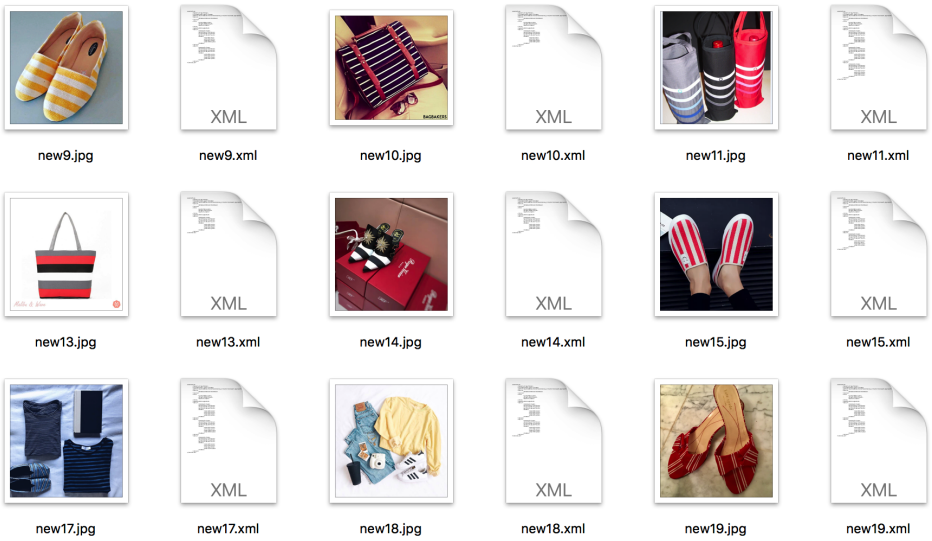

Fig. 2.20.: Examples of training dataset of "stripes" class with deep learning approach

2.2.4 Faster R-CNN Parameter Settings

Before we training our Faster R-CNN, we need to select hyperparameters. Choose good hyperparameters can help the model to get better performance.

- Learning rate: Since the convolutional layers are from the pre-trained model. The learning rate should be small. In our case, we select it as 0.0002.
- Batch size: As described in [13], large batch size may cause poor generalization. The result may too specify to our dataset. However, if batch size is small, it takes more time for computation. Since our dataset is small, computation time is not an issue. We set batch size as 1. The images are fed to our system one-by-one.
- Number of global steps: Each global step is a batch. The number should be large enough to let the training process converging. In our training process, there are 13000 steps for converging.

- The threshold of confidence value: Only the regions assigned with high confidence score can be selected. We set it as 93% to get better result.

2.2.5 Test Performance

As the above description said, the training set is generated with two classes: barcode and stripes. We use the 700 training images to create our detecting model with Faster R-CNN network. The performance is tested in 1000 fashion images which are downloaded from Poshmark website. For comparison, the testing dataset with the traditional method which is described in section 2.1.5. Because the network can extract more abstract features than hand-crafted traditional method, deep learning approach is able to get higher accuracy. Some obscure and occluded barcodes can also be detected in the fashion images in this method. Examples of true positives and true negatives are displayed in Fig. 2.21 and 2.22. Only green box will be focus to be our detected barcode region.

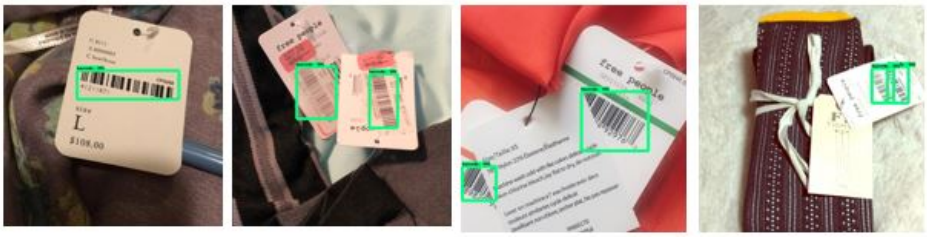

Fig. 2.21.: True positives with the developed deep learning method

Fig. 2.22.: True negatives with the developed deep learning method

For better comparison our development and previous work, we also repeat the training process without "stripes" ground truth images. Fig. 2.23 displays some failure case in previous work and successful results in our development with the same testing images.

Fig. 2.23.: Comparison of results with the previous and the developed deep learning method

In addition, the confusion matrix of previous works and our development has been shown in Fig. 2.24. It is obviously that adding new class "stripes" helps to decrease

false positives. The accuracy is calculated in the equation 2.7 in section 2.1.5. The accuracy with only one training class is 92.4%/ Our developed accuracy is 96.7% which is higher than the accuracy with previous one.

	Predicted: No	Predicted: Yes	
Actual: No	TN = 789	FP = **49**	838
Actual: Yes	FN = **27**	TP =135	162
	816	184	

	Predicted: No	Predicted: Yes	
Actual: No	TN = 826	FP = **12**	838
Actual: Yes	FN = **21**	TP = 141	162
	847	153	

Fig. 2.24.: Confusion matrices with the previous and the developed deep learning method.

Table 2.2 shows a clear comparison of these three method. The traditional method takes the shortest time, but it has lower accuracy. The deep learning method requires GPU. Also, the deep model takes longer time. But the deep learning approach gives us a higher accuracy to detect barcode.

Table 2.2.: Comparison of three detecting methods

Method	**Acc**	**Run time**
Traditional Method	82.8%	44.52 s
Previous Method with Deep Learning	92.4%	143.86 s
New Method with Deep Learning	96.7%	169.23 s

Even though our development based-on deep learning approach largely improves the accuracy, there are some limitations. Fig. 2.25 displays the wrong detection with barcode. Regular black-white stripes share much similar pattern with barcode, our system is confused with these cases. Much obscure texts collection on the tags also seems like the pattern of barcode, our model is confused to distinguish it. We also try to add the third class for obscure texts, but this class attack the previous model. The possible reason is obscure text share so many features with barcode that the network cannot detect it.

The most false negatives are due to the small barcode such as Fig. 2.26. The red circle in the figure is the manual label for failure detecting results. The dimensions of the barcodes in the figure are around 8×57 and 10×40. In the process of labeling ground truth, the dimension of bounding box need larger than 32×32 after resizing the image to 224×224. It results the limitation for detecting very small barcode. These limitations need to be solve in the future.

Fig. 2.25.: False positives in the developed deep learning method.

Fig. 2.26.: False negatives in the developed deep learning method

3. BARCODE DECODING SYSTEM

3.1 Barcode Decoding System

To obtain the information from the barcode, a decoding system is added after barcode detection. The bounded region will be cropped to be the input of the decoding system. Decoding barcode bases on its encoding. The general idea of decoding is generating the information from the spacings and widths of parallel lines in the barcode.

3.1.1 General Method for Decoding 1-D Barcode

A traditional method to decode a 1-D barcode is using edge detection to detect lines and correct the barcode image [14]. Then the specific information of spacings and widths are converted to a string of numbers according to the schemes. Fig. 3.1 displays a flowchart and an example to decode a typical EAN-13 barcode. Firstly, the 2-D color image is transferred to grayscale image. Then a threshold is used to get a binary image. Before threshold, an averaging filter is used to denoise the grayscale image to reduce high-frequency information. Then we apply Hough transformation to correct image with vertical lines. Based on the binary image after preprocessing, we use 0 and 1 to represent each width and spacing. According to different schemes, the binary array will be converted to a decimal code.

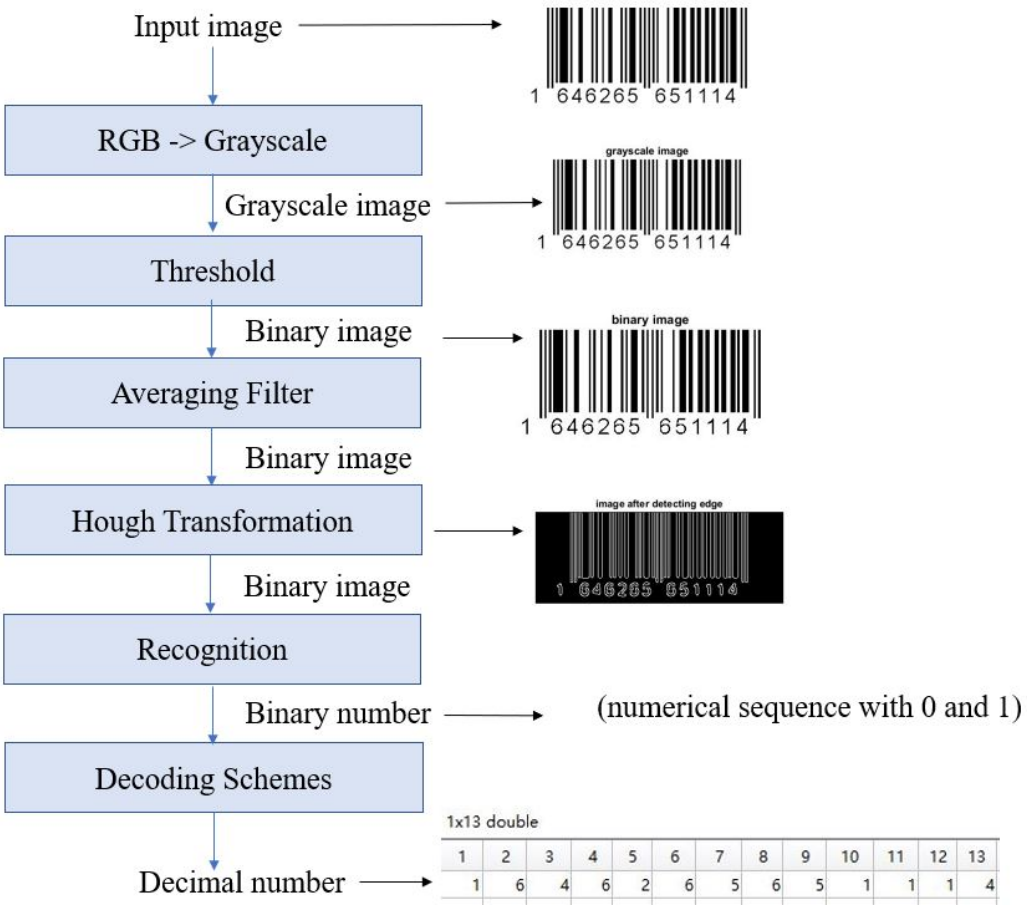

Fig. 3.1.: Structure of general method for decoding 1-D barcode.

3.1.2 Pyzbar Package to Decode Barcode

The Pyzbar package in Python is used to decode common types of barcodes. This package is set up with different decoding schemes, barcode correlation and pre-processing before decoding. Thus, even though the barcode is lightly skewed, it can also be straightly detected with the Pyzbar package. Also, it can extract the type of input barcode and decode the information. Fig. 3.2 displays two examples of results generated from this package.

Fig. 3.2.: Decoding results from Pyzbar package

3.1.3 Strongly Skewed Decoding System

For some strongly skewed images, Pzybar cannot decode it directly. The most current software on our mobile phone also cannot decode it. We use perspective transformation [15] by four corners points on the barcode to warp it. Fig. 3.3 shows the process.

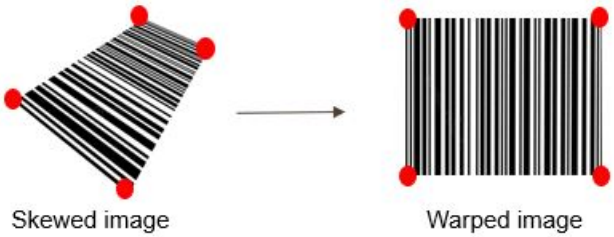

Fig. 3.3.: Process of warping strongly skewed image

The perspective transformation projects an original image to a new projective plane. It realized by matrix calculation (equation 3.1) of the coordinates. $[u, v]^T$ represents the coordinates in original image, and $[x, y]^T$ represents the coordinates

in projective mapping. Matrix (H) represents the general projective transform. Let h_{33} equal to 1. $\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$ is the matrix of linear transformation including scaling, shearing and rotation. Due to linear transformation, $[h_{31}\ h_{32}]$ is set to zero. $[h_{13}\ h_{23}]^T$ represents translations. The coordinates of labeled corners is applied to generate the H matrix.

$$\begin{bmatrix} x^{'} \\ y^{'} \\ w^{'} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \times \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad x = \frac{x^{'}}{w^{'}} \quad y = \frac{y^{'}}{w^{'}} \tag{3.1}$$

3.2 Barcode Decoding Results

The following Fig. 3.4 displays some successful examples from the detecting and decoding system.

Fig. 3.4.: Results of decoding lightly skewed images

A strongly skewed image can be detected with warping image is shown in Fig. 3.5

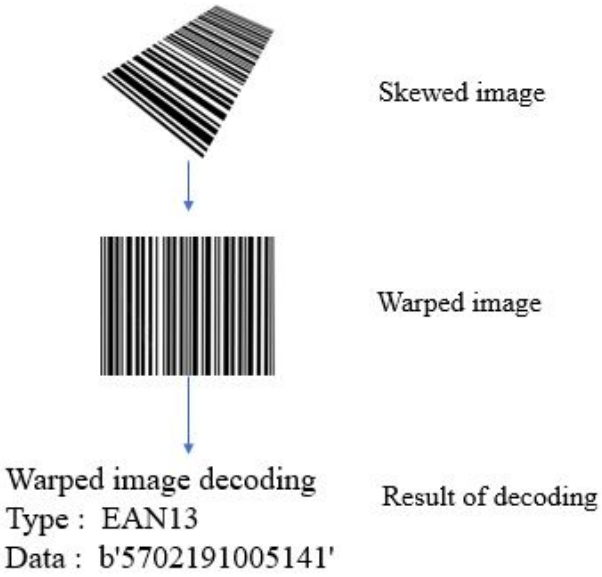

Fig. 3.5.: Results of decoding strongly skewed images

4. SUMMARY

The developed system can be desirable to identify images of fashion items that contain barcode, and to decode those barcodes. Also, the challenge that fashion items frequently contain strips that can be confused with barcodes has been solved in the development. We explicitly consider stripes as detection class, along with barcodes. In this work, the three approaches to barcode detection has been developed: the traditional approach uses hand-crafted features; the deep learning approaches with and without an explicit class for stripes, respectively. As a result, adding post-processing in hand-crafted traditional method solves some failure cases in false positives. In addition, the algorithms of deep learning approach are trained on a large corpus of photographs of fashion items acquired in retail outlets, and tested on images from the Poshmark website. From testing performance, the comparison is observed for those three methods in run time and accuracy. The proposed method with the explicit class for stripes with deep learning approach achieves the highest accuracy of 96.7%. To decode the significantly skewed barcodes, the Pyzbar package has been applied after warping the barcode image.

REFERENCES

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[2] J. Li, Q. Zhao, X. Tan, Z. Luo, and Z. Tang, "Using deep convnet for robust 1d barcode detection," in *Advances in Intelligent Systems and Interactive Applications*, F. Xhafa, S. Patnaik, and A. Y. Zomaya, Eds. Cham: Springer International Publishing, 2018, pp. 261–267.

[3] P. Bodnár and L. G. Nyúl, "Improving barcode detection with combination of simple detectors," in *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*. IEEE, 2012, pp. 300–306.

[4] D. Wang, "A multiscale gradient algorithm for image segmentation using watersheds," *Pattern recognition*, vol. 30, no. 12, pp. 2043–2052, 1997.

[5] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of solid-state circuits*, vol. 23, no. 2, pp. 358–367, 1988.

[6] H. G. Nash and J. R. Linford, "Low pass digital averaging filter," Mar. 11 1980, uS Patent 4,193,118.

[7] S. Simanovsky, I. M. Bechwati, M. Hiraoglu, and C. R. Crawford, "Apparatus and method for detecting objects in computed tomography data using erosion and dilation of objects," May 23 2000, uS Patent 6,067,366.

[8] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[9] A. S. Glas, J. G. Lijmer, M. H. Prins, G. J. Bonsel, and P. M. Bossuyt, "The diagnostic odds ratio: a single indicator of test performance," *Journal of clinical epidemiology*, vol. 56, no. 11, pp. 1129–1135, 2003.

[10] J. T. Townsend, "Theoretical analysis of an alphabetic confusion matrix," *Perception & Psychophysics*, vol. 9, no. 1, pp. 40–50, 1971.

[11] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[13] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[14] A. S. Dhua and M. Delgadillo, "Decoding barcodes," Apr. 9 2013, uS Patent 8,413,903.

[15] J. Mezirow, "Perspective transformation," *Adult education*, vol. 28, no. 2, pp. 100–110, 1978.