POLITECNICO DI TORINO

Master's degree in ICT for Smart Societies

Master's Thesis

Shot-blasted DataMatrix code recovery for Industry 4.0 product traceability systems

Supervisors
Prof. Elisa Ficarra
Prof. Luca Barbierato
Prof. Raffaele Ferri

Candidate
shaohuan Wei

September 2020

Abstract

In the Industry 4.0 era, companies improved manufacturing efficiency through the "digital world". The purpose of the thesis is to recover the shot-blasted DataMatrix code, using the product unique identity which encoded in the DataMatrix code to build a multi-agents traceability platform, in order to obtain a complete and reliable database which contains product information and attributes, to achieve innovative Industry 4.0 product management and predictive maintenance. The main objectives are: i) product surface DataMatrix code position localization, ii) shot-blasted DataMatrix code recovery and decoding, iii) develop a smart way to collect data from the agents, and implement Manufacturing Execution System (MES) tracking.

Keywords: Shot-blasted DataMatrix code; Manufacturing Execution System; Industry 4.0 product management.

Acknowledgements

First and foremost, I would like to show my deepest gratitude to my supervisor Prof. Elisa Ficarra a respectable, responsible, and resourceful scholar, give me the opportunity to study my thesis topic. I also extend my thanks to Co-tutor: Prof.Luca Barbierato and Dr.Raffaele Ferri. Without their impressive kindness and patience, I could not complete my thesis. They provide me the data, describe some important concepts and definitions, give me constant guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With I4.0 development, Company proposes to build a multi-agent system (MES) to detect the product quality. In order to obtain a traceability database and reliable dataset to analyze the relationship between the production process and product quality, achieve product management and predictive maintenance. The first chapter introduces the conception about I4.0, MES system and DataMatrix code which record product identity, help us understand the thesis background.

1.1 Background

I4.0

Modern industrial development has lasted several years, nowadays industry enters into the 4th Generation Industrial revolution (I4.0). For the operator viewpoint, the goals of Industry 4.0 are to achieve a higher level of operational efficiency and productivity (such as reduce the setup and processing times, maintenance cost, implement product quantity, delivery, and flexibility). As well as the higher level of automatization[1], I4.0 connects the physical to the virtual world and will bring computerization and inter-connect into the traditional industry, for instance, machine to machine communication[2]. For the market viewpoint, this new industrial stage is also affecting the competition rules, the structure of the industry, and customer's demands[3].

As many authors mentioned, I4.0 is defined as Cyber-Physical System based on heterogeneous data and summed up as an interoperable manufacturing process. I4.0 characteristics are integrated, adapted, optimized, service-oriented, which correlate with algorithms, Big Data (BD), Internet of thing

(IoT), cloud computation (CC), services (Platform as a Service; Infrastructure as a Service). In a word, the principles of I4.0 are interoperability, virtualization, decentralization, real-time capability, service orientation and modularity[1].

During the I4.0 development move forward, with the base technologies above, some front-end technologies developed: smart supply chain, smart working, smart manufacturing, smart product[4]. Smart chain and working aim at providing efficiency to operational activities, for instance, short-term and long-term Human-Centered manufacturing, adaptive supply chain[5]. Regarding the smart manufacturing and smart product, I4.0 intelligent manufacturing system (IMS) framework consists of smart design, smart machines, smart monitoring, smart control, smart scheduling[6].

For each industry process included in I4.0, product quality is one of the main features. The principal aim of quality management is adopting high-productive modern technologies, which are reliability and maintainability. In general, Six Sigma (definition, measurement, analysis, improvement, control) methodology is used for industrial management to improve the quality[7]. In a word, Product quality management aims to define, measure, develop and control the quality.

I4.0 intelligent manufacturing system (IMS) development has significantly enhanced the products' quality. In IMS, some smart processes such as smart sensors, smart monitoring, and smart control also help to achieve predictive maintenance, predictive maintenance aims to predict when equipment failures might occur, prevent the occurrence of the failures by performing maintenance, moreover reducing maintenance costs and activities.

Combine IMS with the quality management and the predictive maintenance. Thesis monitoring, collecting the product state with product unique MES-ID which is recorded in DataMatrix code, aims to manage the product and obtain the relationship between process and quality, to achieve predictive maintenance in the future.

MES

In I4,0 documents should shared in different enterprises and systems; the agent analyzes and detects the product in parallel in the Multi-Agent System (MAS), also called Manufacturing execution system (MES). According to agents' individual information and decision, they cooperate and service

each other to accomplish the whole work. MES includes several functions such as document management, maintenance management, product tracking, performance analysis, it can improve efficiency and reactivity to adversities, increase flexibility; robustness and adaptability[8]. The implementation of document management depends on the database, which records dynamic data and provides data to the control center. In the thesis, to promote product quality, multi-agents are all detection agents, and each product should be detected with the agent's sequence, to obtain the final state.

A multitude of detection agents in the manufacturing execution system (MES) cooperate with each other to detect the product quality state. Agent scans the DataMatrix code which prints on the product to recognize the product's identity, uploads documents with the corresponding identity to the platform, the document contains the detection result and the reason once the detection result belongs to negative. In order to improve the working efficiency, the defective product will not follow the sequence anymore, meaning that in the platform the control center receives all messages generated by every agent and decides whether the process continues, the detection agent does not detect the product when getting the unqualified state after scanning the DataMatrix code.

In the MES system, the company decided to record product MES-ID in the DataMatrix code, let us introduce what is the DataMatrix code, and which kind of DataMatrix code we adopt.

DataMatrix code

DataMatrix code consists of black and white cells, the basic structures of DataMatrix code are the finder pattern and data region[9]. Finder pattern contains two L-shapes, solid L-pattern and alternating L-pattern. Solid L-pattern is used to determine the size, orientation and location, alternating L-pattern is used to define the number of rows and columns in the symbol. Between with the finder pattern, data is encoded in the data region (Figure 1.1). Let's name DataMatrix code as DataMatrix in simple. To avoid DataMatrix cells damaged with external factors, the redundancy error correction codewords also recorded in the data region. The Reed-Solomon algorithm is a standard algorithm for error correction generation, with error correction codewords DataMatrix can be decoded, even if some blocks are broken. Meaning that the decoded DataMatrix always gives us the correct information, the information is trustworthy, otherwise, it cannot be decoded. However according to the DataMatrix guideline[10], the Reed-Solomon al-

gorithm has a limitation about correction capability, the capability depends on the DataMatrix modules number. If errors are less than the correction capacity, we have the actual result, otherwise, output an empty result.

How "error correction codewords" recorded in DataMatrix? ASCII encodes data to decimal and hexadecimal, according to the hexadecimal, Reed-Solomon algorithm uses a series of hex numbers to represent error correction codewords. Finally, all hex data and hex error correction codewords are transformed to binary serial, then partition binary serial with 8 bits per byte, and filled in the modules (Figure 1.2).

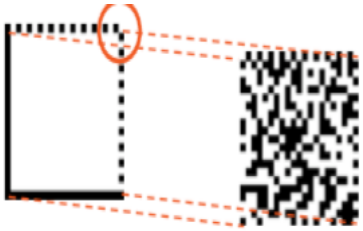


Figure 1.1: DataMatrix structure

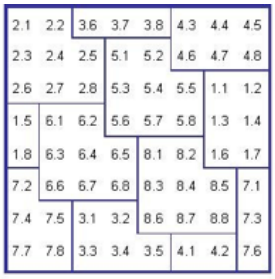


Figure 1.2: DataMatrix 8*8 modules

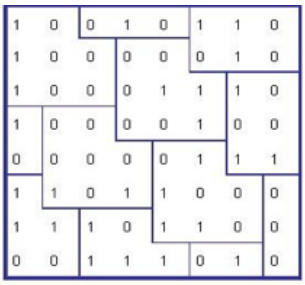


Figure 1.3: DataMatrix 8*8

For example, we encoded a number string '123456' into DataMatrix, ASCII encoding the string to "142;164;186" and transform to hex number:'8E','A4','BA', then Reed-Solomon algorithm use five hex error correction codewords to represents the data: '72','19','05','58','66'.Finally,the hex code would be:'8E,A4 BA,72,19,05,58,66' and the binary code would be:10001110 1010100 10111010 01110010 00011001 00000101 01011000 01100110 (Figure 1.3)(table 1).

DataMatrix has different attributes (table 2), in the thesis from the image which the company gave us, the starting point is 14*14 DataMatrix. Moreover, hypotheses DataMatrix always be 14*14, since it has capacity 16 numbers and ten characters, the capacity is enough to record ID serial numbers.

However, the DataMatrix we received is "shot-blasted DataMatrix". Shot-blasted DataMatrix means the DataMatrix which uses a laser irradiation erosion method to print on the product, processed by shot blasting together with the product. Shot blasting is used for surface production before further

processing, such as welding and coloring, it is a technological process for removing various impurities by abrasion. After shot blasting, the DataMatrix called shot-blasted DataMatrix (Figure 1.4). Because the shot-blasted DataMatrix only has rough contour moreover, contours are not smooth and not continuous, we also call it "rough contour DataMatrix", this kind of DataMatrix cannot be decoded by the scanner.

Thus, the principle objective of the thesis is to recover the rough contour shot-blasted DataMatrix code, moreover develop the platform demo to simulate the detection agents cooperation process, in order to obtain the product quality state and a reliable product information dataset.

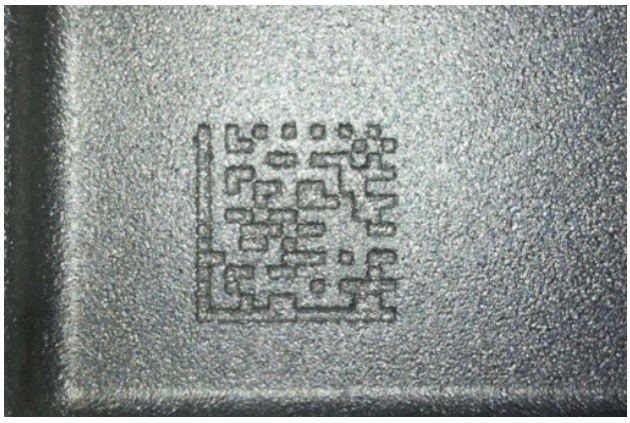


Figure 1.4: Shot-blasted DataMatrix code

1.2 State of the art

MES Platform
Because this MES detection agents cooperation system is under construction, some specific features like service function, client request, and the database schema are gradually improving, which means we need to design and prototype a new platform system to satisfy the different requests.

DataMatrix code recovery
Shot-blasted DataMatrix is a newly raised problem because it cannot be decoded by the usual decoder. After searching a lot, it is hard to find the literature related to this kind of DataMatrix. We can say this rough contour DataMatrix recovery is a new project. We do not have the reference,

and we do not have the baseline to measure recovery performance. This background requests us to try different methods to recover the DataMatrix, compare methods to show which method has the worthy value to improve in the future.

Therefore, this MES system development is a long-time schedule, what the thesis did is to initiate the system creation, raise some ideas and opinions, in order to construct a complete system that could be adopted in the company.

1.3 Objective and purpose

Database creation is a way to manage product information in an intelligent and effective approach. Aggregating the information from different agents is useful to analyze the relationship between the production information and product quality. In the end, the analysis report helps the company optimize the production process and improve product quality, the company will have a good performance in product management and predictive maintenance, achieving I4.0 improvement.

The thesis objective is to design a platform to obtain a reliable database that contains the product information and product quality state. The platform is used for quality detection purposes agents to upload information to a database through the product's unique identity, which is encoded in DataMatrix and printed on the product. However, the DataMatrix is a rough DataMatrix which cannot be decoded. Therefore, the thesis introduces two steps for database creation: the **DataMatrix recovery** and **MES platform building**.

DataMatrix code recovery
The objective of recovery is to transform the shot-blasted DataMatrix to standard DataMatrix with one pixel per cell, to help the scanner decode the DataMatrix and obtain the products identity. The product ID is a unique identity in the detection agent MES system, using unique identity, the system can track and check the product's state. Chapter 2 introduces the DataMatrix recovery flow, which contains **"positioning stage"** and **"recovering stage"**, and introduces different methods adopted in each stage.

MES Platform development
Chapter 3 introduces protocols adopted in the platform, moreover the platform and database structure. A smart platform development, aims to collect

the data from the machine with the technical sensors, update the product state after the multi detection agents process, and record the product analysis report. The platform can significantly improve the work efficiency rather than manually recording.

In conclusion, before the thesis we have the following questions:

1) How can we create a traceability system? What's the structure of the platform and Database?

2) What functions should the platform achieve?

3) How can we decode shot-blasted DataMatrix code with computer vision?

4) Which recovery techniques perform better and are better suitable for a real case scenario?

Chapter 2

Shot-blasted DataMatrix code recovery

Section 2.1 explains the tool and method adopted in DataMatrix code recovery flow, includes some image processing algorithms, image pre-processing, position localization process, and recovering process. 1). Pre-processing focuses on image binarization and denoising. 2). The position localization process is adopted in two different scenarios; auto positioning is adopted in the factory and manually positioning is used for the mobile phone application. 3). Recovering process using two succession methods to transform the contour DataMatrix code to the traditional type.

Section 2.2 2.3 2.4 are the recovery results in the study case, to testify the analysis flow. Thesis also proposes to use Neural Network in the recovering process, it is introduced in the additional section 2.5. In the following content DataMatrix code also called DataMatrix.

2.1 Tools and method

Despite the fact that the literature does not mention the rough contour DataMatrix recovering method, some of them recover the rough DataMatrix which has filled cells. Ladislav.K uses aspect ratio condition to define a closed region which contains DataMatrix solid L-shape, then using two edges of closed region to define solid L-shape, obtaining DataMatrix location[11]. Feng.L applies an algorithm to searching solid L-shape closed edges in two opposite directions from one point[12]. Qiang.H not only detects solid L-shape but also detects dash border by robustness RANSAC line detection method[13].

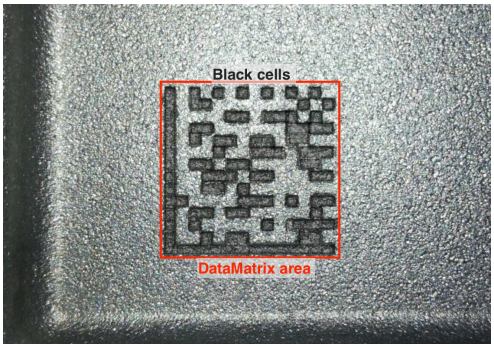
The typical analysis flow they adopted is positioning DataMatrix firstly and then based on the precision location, to recover the DataMatrix. I also adopt this flow and call it **"positioning and recovering analysis flow"**.

2.1.1 Stage1 Pre-processing

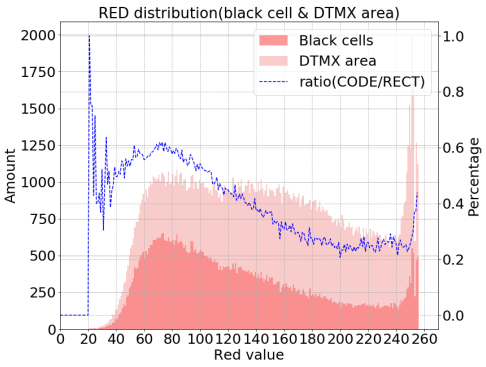
Before we process the image, firstly, we should know the image attributes. The sample image (Figure 1.4) has resolution ratio 72*72, and it is an RGB image. RGB image has three channels, values in each channel are from 0 to 255, the higher value represents the darker color. However, the three-dimension matrix increases the algorithm complexity, we usually create a new channel from these three channels, but before that, I check the RGB distribution, see whether the single color has more representativeness.

RGB channel selection
High representative color means we can easily recognize the DataMatrix from that color, all the black cells in DataMatrix should have similar color value and have large differences with white cells and surrounding circumstance. Therefore, I mark all positions which belong to black cell and then select a DataMatrix square, to understand black cells color distribution in DataMatrix square (Figure 2.1a).

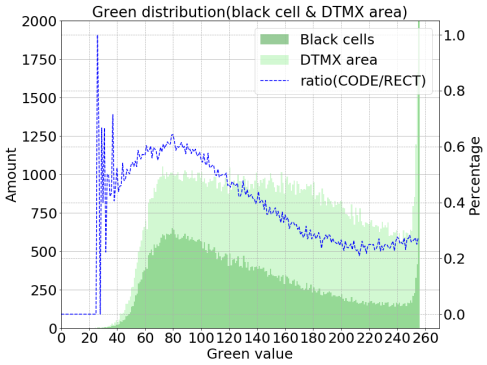
In the image (Figure 2.1b;2.1c;2.1d), the x-axis is the color value from 0-255, the y-axis is the number of appearances. Moreover the dark color distribution represents all positions which should be the black cell, the light color distribution represents all pixels in the DataMatrix area. Blue dash line is the ratio between the number of appearances in black cells and the DataMatrix area for the same color value. From the distribution we see red; green; and blue color have similar color distribution and the black cells' color is not in evidence, which means we cannot extract the DataMatrix from a single color.



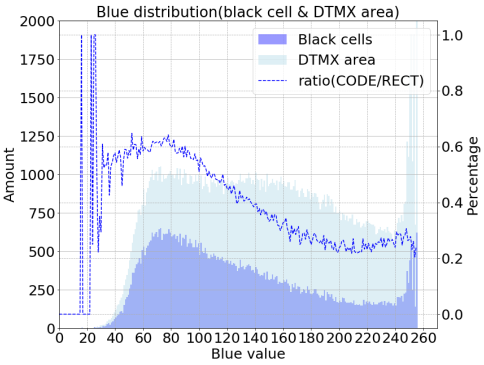
(a) Black cells



(b) Red distribution



(c) Green distribution



(d) Blue distribution

Figure 2.1: RGB distribution

Image binarization

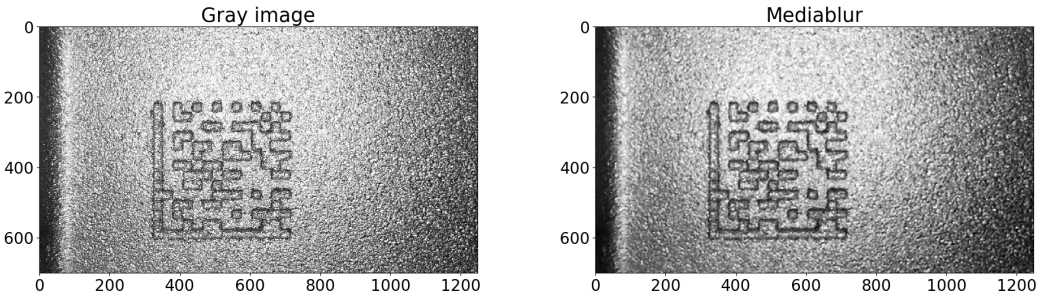
"A binary image is one that consists of pixels that can have one of exactly two colors, usually black and white, used for the image segmentation, the morphological operations, the distance transform, and gathering orientation-free metrics" [Wikipedia[1]]. Before image binarization, the image should be a 2D matrix, which means the image should be a single-channel image.

The red channel, blue channel and green channel constitute a single-single, people called this single-channel as 'grey channel'(Eq.2.1), the grey channel image called **grey image** (Figure 2.2a). Many different methods used to create the grey channel, using the maximum value of RGB or using the

[1]https://en.wikipedia.org/wiki/Binary_image

mean value of RGB, I adopt the typical method which is named 'equal grey' method.

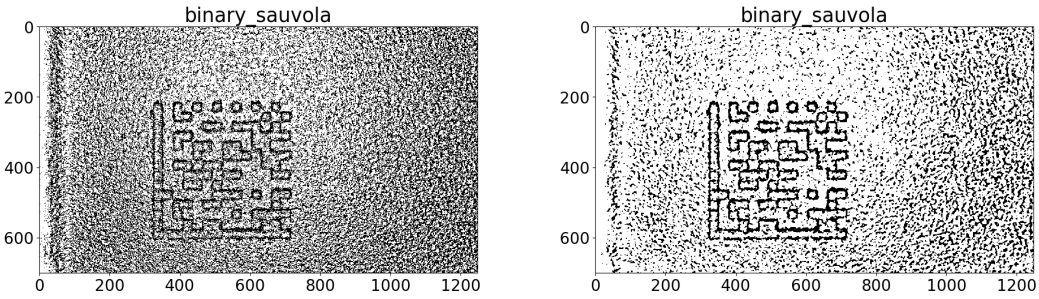
$$Grey\ value = B * 0.114 + G * 0.587 + R * 0.299 \qquad \text{(Eq.2.1)}$$



(a) Gray image (b) Blurred image

Figure 2.2: Gray and Blurred image

After transforming three-channels to one single grey channel, the grey image can be binarized. However, before binary the image, the typical way is to use a filter to smooth the grey image and omit the image noise. The filter is a sliding window, the median value of all numbers replaces the pixel in the center of the window. The image after smoothing is called a **blurred image** (Figure 2.2b). Binary the blurred image, rather than the grey image is a way to omit the noise, otherwise is it not easy to distinguish between foreground and background (Figure 2.3).



(a) Grey image binarization (b) Blurred image binarization

Figure 2.3: Grey image and blurred image binarization

In a word, the binarization process is a filter process, the grey value which higher than the threshold should be assigned as 1, on the contrary, assigned as

0. Different binarization algorithms obtain different thresholds and different binarization performance.

- Algorithm 1: OTUS algorithm
  OTUS algorithm is used to perform automatic image thresholding from 0-256, separate pixels into two classes, "the threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance" [Wikipedia]. The equation is introduced very clearly in Wikipedia. In a word after iterating each gray value from 0-255, the OTUS algorithm gives the binarization result which has an optimized threshold.

- Algorithm 2: Adaptive threshold algorithm
  Adaptive threshold algorithm separates the image into small windows, different windows have a different threshold, the threshold can be the average value of the window or the summation with gaussian weights. This algorithm has less suitability because mostly the window threshold is not sufficient to obtain a clear binary image.

- Algorithm 3: Niblack's algorithm
  Same as the adaptive threshold algorithm, separating the image with several windows. However, Niblick's algorithm (Eq.2.2) uses mean (m) and standard (s) deviation to determine the threshold for each window. Typically the k default as 0.2. Let us use the default value to obtain a Niblack's binary image.

$$T = m(x, y) - k * s(x, y) \qquad \text{(Eq.2.2)}$$

- Algorithm 4: Sauvola's algorithm
  Sauvola's algorithm (Eq.2.3) uses a more complex equation than Niblack's method, to obtain different binarization thresholds in each separated window. Not only use mean (m) and standard (s) but also compute the maximum standard deviation (R) of grayscale to calculate the threshold. With the default value of k, we obtain a clearer binary image.

$$T = m(x, y) * (1 + k * ((s(x, y)/R) - 1)) \qquad \text{(Eq.2.3)}$$

After grey image processing, blurred image processing, and binarization processing, we obtain a binary image with less noise, and base on the binary image to locate the DataMatrix.

2.1.2 Image processing algorithm

1) Binary image processing

Here we introduce some algorithms that process the binary image, which we use in the following sections.

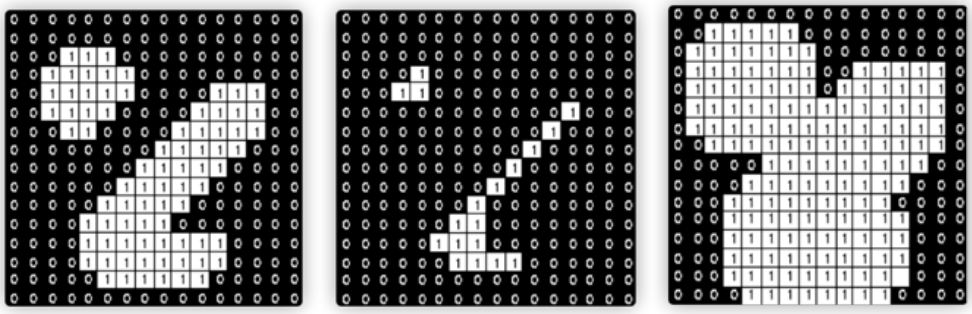
- **Morphology algorithm**

Morphology is a collection of non-linear operations related to the shape or morphology of features in an image, removing the imperfection of noise and texture by accounting for the image's form and structure.

The basic operations of morphology algorithms are erosion and dilation (Figure 2.4).

(a) Erosion: While the convolution kernel slides on the binary image, if **all** elements in the kernel are white, the value in the center of the kernel should be marked as white, else it should be black. Erosion is used for eliminating the white noise and cutting off the connection part.

(b) Dilation: While the convolution kernel slides on the binary image, if **any** elements in the kernel are white, the value in the center of the kernel should be marked as white, on the contrary, it should be black. Dilation is used for expanding the contour which shanked after erosion.



(a) Binary Image (b) Erosion (c) Dilation

Figure 2.4: Morphology algorithm

With the basic algorithm, different combinations have different effects:

* Close: The close method adopts erosion firstly before dilation is used to omit the image noise and remain the principal feature.

* Open: The Open method adopts dilation firstly before erosion is used to connect breakage blocks.

* Top hat: Make the difference between the original binary image and "open image", remain the image lightness part.

* Black hat: Make the difference between the "close image" and the original binary image, remain the image darkness part.

* We can also calculate the image gradient. The original binary image subtracts the erosion image is the image inner gradient and dilation image subtract the original binary image is the outer gradient.

- **Contour detection algorithm**

The contour detection algorithm started from Suzuki in 1985 (Figure 2.5 [2]) and improved in the 'OpenCV' library. In short, when the algorithm meets a new black pixel, it will scan the pixels around that black pixel and select the first one in clockwise order, until back to the original black point, and then using the same method to detect whether an inner contour exists, finally mark every contour. In the meantime, the algorithm can reconstruct a full hierarchy of nested contours.

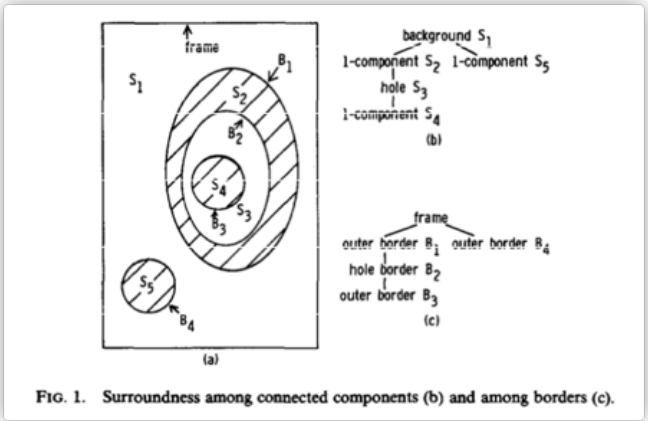


FIG. 1. Surroundness among connected components (b) and among borders (c).

Figure 2.5: Contour detection algorithm

[2]S. Suzuki, "Topological structural analysis of digitized binary images by border following", Computer Vision, Graphics, and Image Processing vol,30, pp.32-46,1985

- **Hough Line detection**

Hough line detection algorithm is an algorithm to detect the black line in binary image and improved in the "OpenCV" library. In a word, it transforms the orthogonal coordinate system to the polar coordinate system. In the polar coordinate system, each point represents a line in the orthogonal coordinate system.

Algorithm 1 Hough Line detection

Input:
Binary image;
Output:
Start and end coordinate of the detected line

while unmarked black pixel **do**
  for θ in 2π **do**
    $\rho = x\cos\theta + y\sin\theta$ record
    record ρ and θ
    mark the point
  end for
end while
record the start and end coordinate of the line which ρ and θ always appearance

The most important feature that should be noticed is, the line detection algorithm not only detects the continued lines but also detects the line which has a gap in the middle, it is suitable for the breakage DataMatrix contour, and this is the reason to select the Hough line algorithm. However, the disadvantage of the Hough line algorithm is the line's representativeness, sometimes the detected line cannot represent the real edge, in other words, many similar lines will be detected in a wide black edge.

2) Image rotation

Affine transformation rotation algorithm

Affine transformation is the process of performing a **linear transformation** in a vector space and adding a **shift translation**, aims to form another vector space.

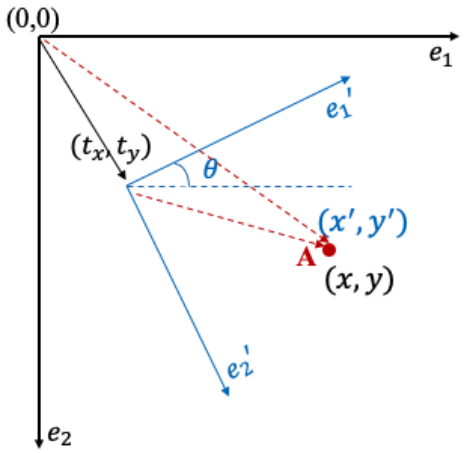


Figure 2.6: Affine transform

According to the affine transform definition, after the coordinate axis rotate in anticlockwise degree around point (t_x, t_y), the equation between original "coordinate system A" (e_1, e_2) and new "coordinate system B" (e_1', e_2') in the same point follows the red vector triangle Eq.2.4 shown in the image (Figure 2.6). **Linear transformation** is vector multiplication and **shift translation** is the origin movement distance. Because (e_1, e_2) are orthogonal vectors, the inverse matrix is equal to the transpose matrix Eq.2.5. The point in "coordinate system A" is presented by "coordinate system B" with equation Eq.2.6. Obviously, the modulus of vectors e_1', e_2', e_1, e_2 are all equal to 1, under the rule of vector multiplication using equation Eq.2.7 to show the relationship of two vectors in two coordinate systems.

$$(e_1\ e_2)\begin{pmatrix} x \\ y \end{pmatrix} + (0\ 0) = (e_1'\ e_2')\begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{Eq.2.4}$$

$$(e_1\ e_2)^{-1} = (e_1\ e_2)^T \tag{Eq.2.5}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = (e_1\ e_2)^T (e_1'\ e_2')\begin{pmatrix} x' \\ y' \end{pmatrix} + (e_1\ e_2)^T \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{Eq.2.6}$$

$$
\begin{aligned}
\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} e_1^T e_1' & e_1^T e_2' \\ e_2^T e_1' & e_2^T e_2' \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + (e_1\ e_2)^T \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
&= \begin{pmatrix} |e_1|\,|e_1'|\cos(\theta) & |e_1|\,|e_2'|\cos(\frac{\pi}{2}-\theta) \\ |e_2|\,|e_1'|\cos(\frac{\pi}{2}+\theta) & |e_2|\,|e_2'|\cos(\theta) \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + (e_1\ e_2)^T \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
&= \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + (e_1\ e_2)^T \begin{pmatrix} t_x \\ t_y \end{pmatrix}
\end{aligned}
\tag{Eq.2.7}
$$

Imagine that the "image 1" has a point "A" which is ready to rotate with (t_x, t_y) in anticlockwise θ degree (Figure 2.7), and the aim is to obtain the coordinate of "A-r" after "image 1" rotation.

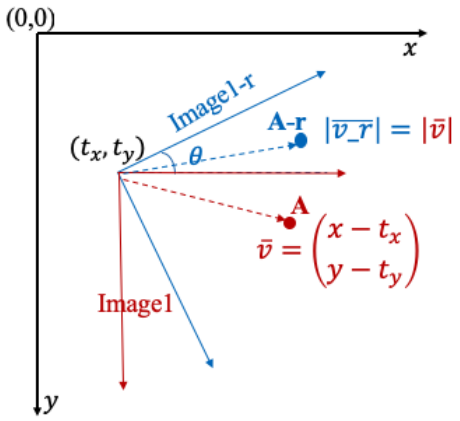


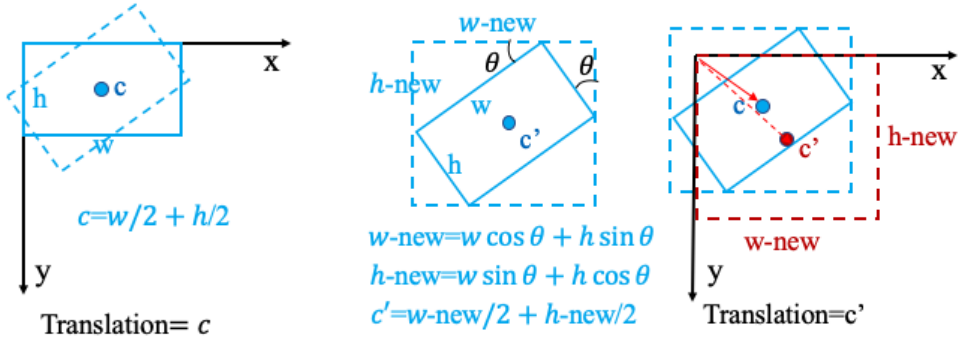
Figure 2.7: Affine transform rotation matrix

The vector of point 'A' in the 'image 1' coordinate system is $\begin{pmatrix} x - t_x \\ y - t_y \end{pmatrix}$ and this vector won't change even if image rotation. Therefore, the vector of point 'A-r' is also $\begin{pmatrix} x - t_x \\ y - t_y \end{pmatrix}$in the image1-r coordinate system. According to Eq.2.7 the position of 'A-r' in the original coordinate system is shown in equation Eq.2.8. The matrix which is able to transform the point coordinate is called the **"affine transform rotation matrix"**.

$$
\begin{aligned}
\begin{pmatrix} x_r \\ y_r \end{pmatrix} &= \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x - t_x \\ y - t_y \end{pmatrix} + (e_1 \; e_2)^T \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
&= \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x - t_x \\ y - t_y \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
&= \begin{pmatrix} \cos(\theta) & \sin(\theta) & t_x - t_x\cos(\theta) - t_y\sin(\theta) \\ -\sin(\theta) & \cos(\theta) & t_y - t_y\cos(\theta) + t_x\sin(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\end{aligned}
\tag{Eq.2.8}
$$

In conclusion, the rotation matrix has two parameters, rotation angle and rotating origin point. From the original image's "transform rotation matrix" and "point coordinate", it is possible to obtain the new coordinate after rotating the image with the "affine transform rotation algorithm". The manually positioning process rotates the image which has oblique DataMatrix and keeps DataMatrix in "horizontal-vertical" direction.

The typical method to rotate the image is using the central point of the image as the rotation origin point, however, the image perhaps exceeds the coordinate system after rotating with the center point (Figure 2.8a), which means that the translation distance which in "transform rotation matrix", is not enough to obtain the image completely. To avoid this happening, we change the translation distance in the image aspect way.



(a) Image center rotation (b) Bounding rectangular center rotation

Figure 2.8: Rotation origin point

To ensure that the coordinate system contains the complete rotated image, the circumscribed rectangular should be accepted at least. The circumscribed

rectangular acceptance condition is to increase the translation distance from the original image center (c) to the circumscribed rectangular center (c'). The center of circumscribed rectangular (c') is the average of width and length which relate to the image rotation angle and image dimension (Figure 2.8b).

2.1.3 Stage 2 Position localization process

According to the literature related to the DataMatrix recovery, their DataMatrix recovering process is based on the closed region. The closed region could be part of DataMatrix, but it must include solid L-shape, then find solid L-shape from the region to locate the DataMatrix.However, the problem I faced in the image is the DataMatrix contour integrality and image noise.

1) Shot-blasted DataMatrix code situation

Because of the manufactory technology problems, after the shot blasting process, a multitude of noises appear on the image, moreover the DataMatrix only reserves the contour. The DataMatrix contours are rough contours, which means it does not have straight edges, and it has some cracks on the edges. Breakage contours cannot constitute the closed region.

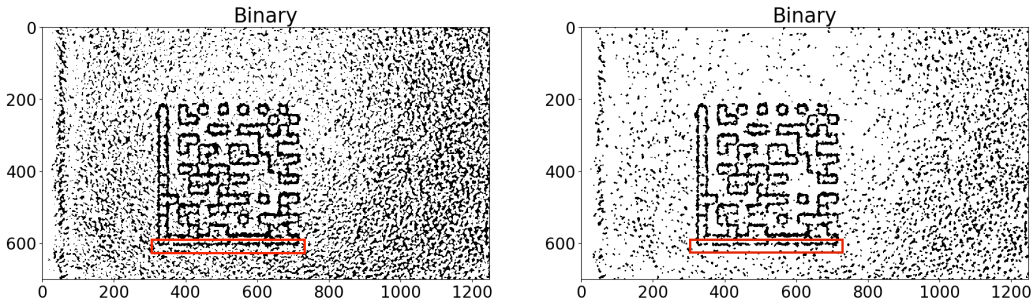
Thus, we adopt two different methods to complete the contour or omit the noise, because problems contradict each other:

a) The first idea to solve the problems is to repair the breakages. Using the morphology algorithm to connect the edges, aims to obtain a completely DataMatrix solid L-shape.

we adopt the **"morphology close method"** to process the image, which the order of process is dilation before erosion. The reason to select the close method because of the prime objective is to repair the crack on the edges, and the dilation method can connect separate edges, however in the dilation binary image, there are many noises, this phenomenon exceeds my expectation. Lots of noise will influence the analysis process in the following steps.

Therefore, using erosion after dilation to eliminate the noise, in order to obtain a clear binary image.

Image (Figure 2.9a) shows the close algorithm adopted result, because we use closed method, few parts of the noise are omitted, however, the breakage still exists on the edges, marked by the red square especially. Since the close algorithm cannot repair the breakage contour and only remove a few noises, we say this dilation and erosion method is not a suitable algorithm.



(a) Morphology close algorithm

(b) Morphology open algorithm

Figure 2.9: Morphology algorithm adopted on binary

b) The second idea to solve the image problem is to remove the noise and repair the crack as much as possible in the less noisy image.

Opposite with the first close method, the process which erosion firstly and then dilation is called **"open method"**. After the open algorithm, the original binary image transforms into a new binary image (Figure 2.9b). The new image has less noise around the DataMatrix, it seems better than the morphology close method, however, more breakages exist on the edges, marked by the red square. Because some individual part on the edge is mistaken for the noise and removed during the erosion process.

After adopting the close algorithm or open algorithm, the noise problem can be solved more or less (open method seems better), but the breakage on the edge cannot be repaired in the final. Breakage contour problem means the traditional methods that find solid L-shape from closed regions to locate DataMatrix is not suitable for our DataMatrix. Because our DataMatrix is a rough contour DataMatrix, the L-shape range is not a closed region since it is broken. Therefore, it is hard to obtain the solid L-shape or alternate L-shape, in other words, even if the L-shape is obtained in the image, we

cannot guarantee that this pseudo L-shape can represent the precision DataMatrix location, because breakage appearance in every possible segment on the edge, and the DataMatrix connected noise also has the probability to affect the edges' length and width.

Since the traditional positioning method does not work, I looked for other methods to obtain the DataMatrix position.

2) Sliding searching algorithm

sliding algorithm is based on the binary image and binary kernel, slide kernel on the binary image, for each sliding step, calculate the matching accuracy between kernel and the kernel part in the image.

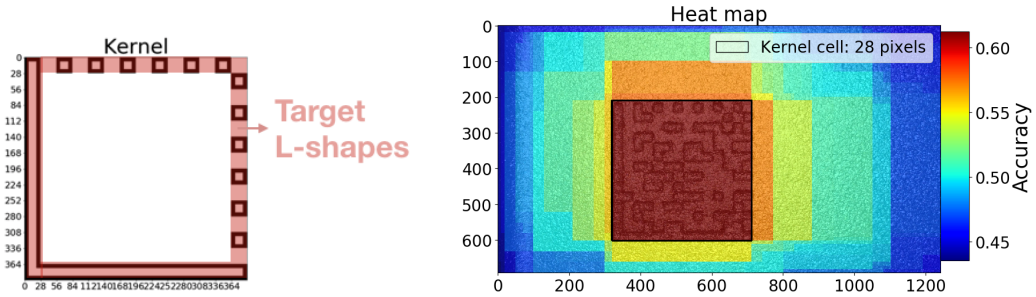
- **Binary kernel**

The binary kernel (Figure 2.10a) has the following features:

- 14 cells on each column and row.
- L-shapes outlines reservation.
- Target L-shapes.
- Black edges are quarter wide of the cell.

In kernel 14 cells on each line because at the beginning we hypothesis all DataMatrix codes belong to the 14*14. The reason for reserve L-shapes outlines because the shot-blasted DataMatrix binary image only has L-shapes contours. Moreover, In the kernel, we do not care about the data region, because the data region does not have constant shape, instead L-shape has.

Therefore, the matching content between kernel and image kernel part, is the L-shape black and white pixels. In the kernel the black occupied quarter width for each cell, I set this default value after observing the image of shot-blasted DataMatrix. However, kernel cell dimension is unknown, therefore it is necessary to try different dimensions and select the most suitable one.



(a) Binary kernel (b) Kernel sliding searching algorithm result

Figure 2.10: Kernel and kernel sliding result

- **Sliding searching algorithm**

While the kernel slides on the binary image with the vertical and horizontal direction, for each sliding step, calculate the accuracy of black and white color with correspondingly proportion, the black proportion is the white color percentage in the target L-shape, vice versa. The reason to compute accuracy for two colors is to avoid some widely black lines, especially the black wide L-shape. If black color is considered only, some widely black lines cannot be ignored.

$$accuracy_{avg} = accuracy_{BK} * weight_{BK} + accuracy_{WH} * (1 - weight_{BK}) \quad \text{(Eq.2.9)}$$

$$accuracy_{BK} = \frac{correctly\ matched\ black}{kernel\ L-shape\ black\ pixels} \quad \text{(Eq.2.10)}$$

$$accuracy_{WH} = \frac{correctly\ matched\ white}{kernel\ L-shape\ white\ pixels} \quad \text{(Eq.2.11)}$$

$$weight_{BK} = \frac{kernel\ L-shape\ white\ pixels}{total\ pixels\ in\ kernel\ L-shape} \quad \text{(Eq.2.12)}$$

Using the heat map to represent the sliding searching result, in the heat map the kernel is filled with the same matching accuracy value and the accuracy value depends on the kernel position, moreover, the lower accuracy is replaced by higher accuracy.

In the study case, I select a suitable kernel in which the cell has 28 pixels, and the kernel slide step is one pixel. Move the kernel from top left to bottom right on the 'open algorithm' image which we obtained before, calculate

the matching accuracy to show on the heat map (Figure 2.10b). From the heat map, we see when the kernel position is close to the real DataMatrix, the matching accuracy becomes higher and higher, until to the maximum which is around 0.7. The minimum match accuracy is around 0.48 because many white pixels matched but black pixels are not matched. Thus, in the heat map, the kernel position which has the maximum matching accuracy, marked as the black square is almost near to the real DataMatrix position, this result testifies that our sliding searching algorithm can be used to obtain the DataMatrix precision location.

3) Position localization process

The positioning localization process uses the sliding searching algorithm which was introduced before. Two main factors should be considered while the sliding algorithm is used, one is DataMatrix orientation and another one is the sliding area.

- Orientation means the DataMatrix L-shape keep around 90 degrees but not parallel with the image edges, in this case, it is necessary to rotate DataMatrix to the "horizontal-vertical" direction, because we default the kernel in sliding algorithm moves horizontally and vertically, it is easy to operate and calculate with these directions.

- The sliding area is a factor to influence the calculation time, if we can control the sliding area, the calculation time can be decreased. Calculation time is a fundamental standard to measure the algorithm performance.

We do not consider the problem with the shear angle, because we have a rough contour DataMatrix. If the image has a wide shear angle, the recovered contour's condition becomes worse, it is another complex project.

Under these two problems, the thesis design two positioning localization processes for two kinds of users, one group of users are **manufacturing machines**, another group of users are **workers**. For manufacturing machines, the camera is fixed on the machine, therefore the orientation problem does not need to be considered, but the scanning area should be the whole image, the DataMatrix could be printed in every possible place on the product. For the workers, the thesis introduces a method used for the mobile phone application, workers use a mobile phone to scan the DataMatrix on the product, aims to check the product identity. We can give users a small scanning region in advance before scanning, to decrease the waiting time, however, a friendly

application should accept user scanning with different orientations.

Program 1: Auto position localization
Auto position localization is adopted in the factory, the process is requested to locate the DataMatrix on the original image automatically. When an RGB image input, the first step is the image binarization as we introduced in "step 1"(chapter 2.1.1), to obtain a binary image. However, many noises are still existing in the binary image, the denoise processing could be "open morphology algorithm". After the open method, the number of noises decreased obviously, however we can make it better.

- **Step 1: Binary image denoising (area filtering)**

Since the erosion method in the open algorithm cannot remove the noise which has a relatively larger area, I set an area threshold to filter the noise.

Obviously, every noise is a closed region and it has a closed contour. Obtaining the noise contour means the noise's area and perimeter are easy to know. Using the "contour detection algorithm", which introduced in chapter 2.1.2 , I obtain every noise's border and area in the binary image and filter the noise which has the area less than 100. In the end receive a new binary image, which less noise remains than the open method (Figure 2.11). This new binary image can be used in the next process "sliding searching algorithm".

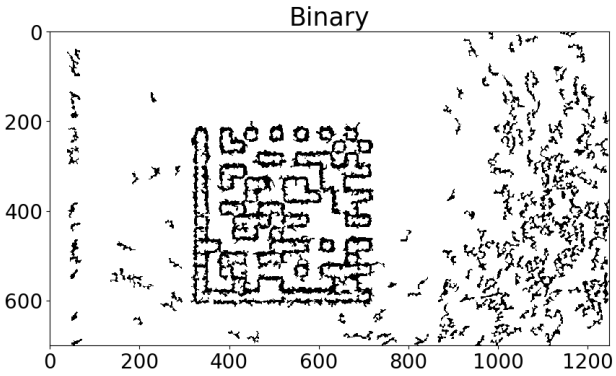


Figure 2.11: Area filtering binary image

For the "sliding searching algorithm", we already know the kernel's structure. However, we do not know the kernel dimension, meaning that the suitable kernel dimension should be selected by iterating different cell dimensions, for

instance the Figure 2.12 presents the maximum matching accuracy changing with different kernel cell dimensions sliding. The maximum value is the maximum accuracy in the heatmap (Figure 2.10b), heatmap of "sliding searching algorithm" depends on different kernels and sliding pixel steps. However, it spends lots of time during iteration. We know the calculation time is the primary standard to measure the process, to reduce the calculation time, it is better to obtain a few kernel dimension options and select the best one from them.

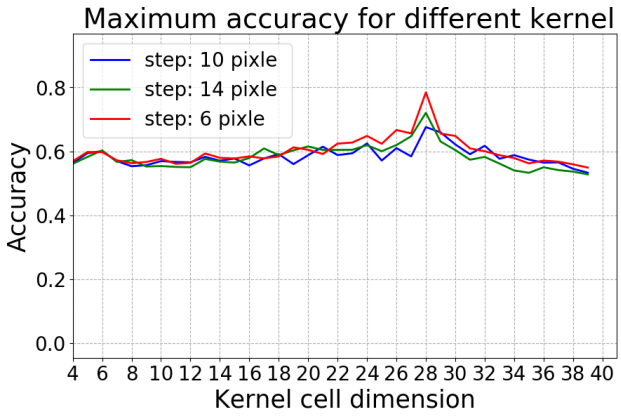


Figure 2.12: Different kernel sliding result

- **Step 2: Line detection**

Obtaining kernel dimension options means we should know the possible dimension of the DataMatrix. Detecting the L-shape of DataMatrix is a direct method to obtain the dimension, L-shape consists of two lines, therefore line detection is the first step to detect the L-shape. The algorithm of line detection named "Hough Line detection" is introduced in chapter 2.1.2.

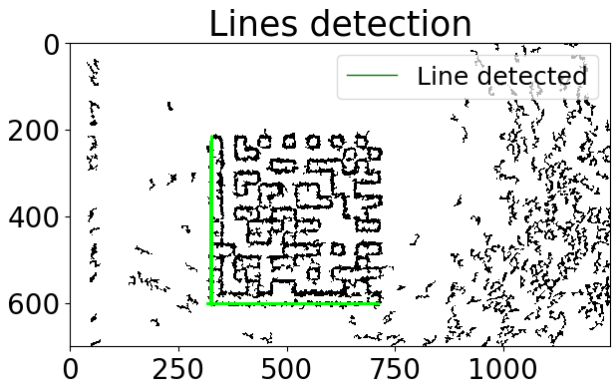
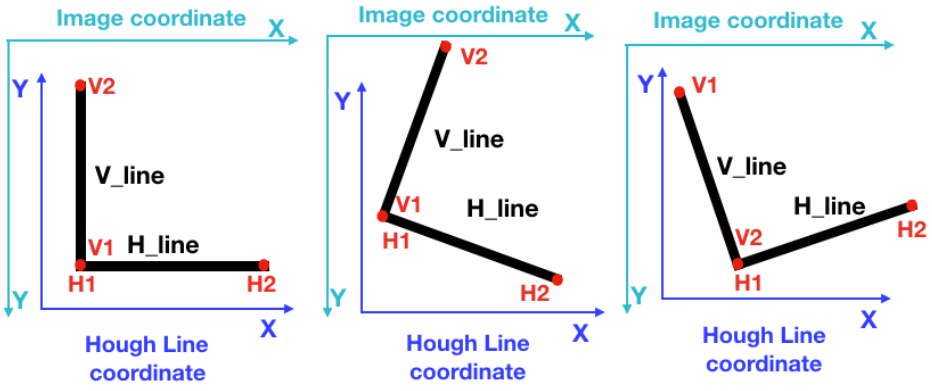


Figure 2.13: Hough line detection

In the experiment, I set a length threshold to filter shorter lines, and accept the line which has a 10 pixels gap. From "Hough line detection" result (Figure 2.13), the good phenomenon is the length of the detected line is near to real L-shape, however, the vertical edge of L-shape has more than two green lines inside. Therefore, in case of other lines out of DataMatrix, and similar lines overlapped, the next step is obtaining the L-shape from all detected lines, and from the L-shapes length to select kernel dimension options.

- **Step 3: L-shape detection**

Lines from the line-detection algorithm are represented by two points, the 'beginning point' and 'ending point'. From two points, other features like the slope, distance and rotation angle between Y-axis are easy to be computed and utilized in L-shape detection. Pay attention, the coordinate origin utilized in "Line detection algorithm" is the left bottom of the image, thus the order of two points in one single line is from the left to right and from the bottom to above, However, the coordinate of two-point is under the image coordinate system which the origin is the left above of the image. These two different coordinate systems should be noticed very carefully (Figure 2.14).



(a) Ideal condition (b) Clockwise condition (c) Anticlockwise condition

Figure 2.14: L-shape in different coordinates

The L-shape detection aims at obtaining the L-shapes form many lines, in the meantime, receives the L-shape length and vertical deviation angle. In the auto position localization process, it is not necessary to adopt the DataMatrix rotation. The fixed camera ensures the DataMatrix is in ideal condition (Figure 2.14a), but in the manually position localization process in the next part, it is required to consider the orientation problem. Therefore, I introduce the algorithm in the following (Algorithm 2), the different processes will adopt different detection results.

Algorithm 2 L-shape detection

Input:
Detected Lines;
Output:
L-shapes(length and orientation);

while unmarked Line **do**
Obtain the coordinate of $start_1$ & end_1 & $length_1$ & $slant\ angle_1$
for other unmarked Line **do**
Obtain the coordinate of $start_2$ & end_2 & $length_2$ & $slant\ angle_2$

if $line_1$ is horizontal & $point_1 \cong point_2$ & $angle_1 + angle_2 \cong 90°$ & $length_1 \cong length_2$ **then**
recover angle = 0°(Ideal orientation)
else if $start_1(y) > end_1(y)$ & $start_1 \cong start_2$ & $angle_1 + angle_2 \cong 90°$ & $length_1 \cong length_2$ **then**

recover angle = $angle_2$ (Clockwise condition)
else if $start_1(y) < end_1(y)$ & $end_1 \cong end_2$ & $angle_1 + angle_2 \cong 90°$ & $length_1 \cong length_2$ **then**
recover angle = $angle_1$ (Anticlockwise condition)
end if
end for
record L-shape
mark the lines in L-shape
end while
Calculate the length and average recover angle of L-shape

In short, L-shape constitutes two lines, which have adjacent vertex; similar length, and right-angle. Moreover, the L-detection algorithm restricts that the DataMatrix accepts the rotation between -90° and 90°.

After L-detection, the length of L-shape is readily known, however, the L-shape we obtained is pseudo-L-shape, we cannot guarantee this is the real DataMatrix shape, what L-shape gives us, is the pseudo length. Therefore, according to the pseudo length, we can create the kernel cell options, which are $[\frac{length}{14} - 1, \frac{length}{14}, \frac{length}{14} + 1]$. Divide 14 because at the beginning we hypothesis the DataMatrix belongs to type 14*14.

Thanks to L-shape, we have the kernel cell options, instead of searching every possible cell dimension. Optional cells save a lot of calculation time as we expect.

- **Step 4: DataMatrix rough range selection**

The image (Figure 2.12) shows the maximum matching accuracy result tendency. Iterate with the kernels which have different cell dimensions and different steps, for each kernel select the maximum accuracy in the accuracy heat map as we introduced in "sliding searching algorithm". However, because of the kernel cell dimension and step, the kernel is hard to match the DataMatrix within the precision location. Perhaps it is smaller than the DataMatrix area or skips the DataMatrix area because of the larger sliding step. Therefore, using the sliding searching algorithm to obtain the precision DataMatrix location has two requests: 1) the suitable kernel cell, 2) the sliding step must equal one pixel.

However,one pixel sliding in the whole image is not a wise choice, it wastes time because many steps far away from DataMatrix are useless. Thus, I create another range called "rough range", the rough range is an area that

is smaller than the original image and contains DataMatrix, meaning in the rough range all kernels matching accuracy at least equal to the threshold, which default as 0.7. The sliding step to obtain rough range should be larger than 1, we test 6 pixels 10 pixels 14 pixels, they are working well, we hypothesis the step is default as 10 pixels. Then Base on the rough range to search the precision location which will be introduced later.

Therefore, searching a rough range before obtaining the precision location is a way to reduce the calculation time. The binary kernel uses large sliding steps instead of one pixel, slide on the original binary image.

The Rough range searching based on the heat map result, the searching process is the following:

Algorithm 3 Rough range selection

Input:
1.Detected L-shapes
2.Binary image
Output:
DataMatrix Rough range;

for lines in L-shape **do**
cell options list append([length/14-1,length/14,length/14+1])
end for

Obtain the unique cell options list

for cell in cell options list **do**
Using default step and kernel slide on the image
Record the maximum matching accuracy
end for

Select the cell which has maximum matching accuracy, and it's whole matching results
Normalize the results
Compute the accuracy threshold according the Normalize threshold(default 0.7)
Select the kernel position which accuracy $\geq$ accuracy threshold
Use bounding rectangular to represent the total selected positions

- **Step 5: Precision location selection**

Based on the rough range, we reset the sliding step equal to one pixel, with one pixel sliding, the location achieves the request that the DataMatrix can be presented by the minimum square area, it is the precision location. The searching method is similar to rough range searching, redoing the "sliding searching process" and selecting the most representative cell from the options. In the selected heat map, the DataMatrix position is the kernel position which has the maximum matching accuracy.

- **Conclusion**

In conclusion, the auto position localization contains 6 steps: binarization process; Image denoising; Line detection; L-shape detection and kernel selection; rough range selection, and precision location selection. After these processes, we get the DataMatrix location automatically. We cannot guarantee it is the real DataMatrix location, but the square should be near to the actual square

Program 2: Manually position localization

Auto position localization is designed for the factory which we do not care about the DataMatrix orientation problem, because the fixed camera is settled in the production line. However, the product should be scanned and recognized in different scenarios, it satisfied I4.0 conception. Thus, we create another process that could be considered to use in the mobile phone application.

Manually position localization is improved from the auto position localization. In the manually positioning process, before scanning, the application should give users a defined scanning region, and request users to put DataMatrix inside the initial scanning area, as other applications did. The scanning area can be understood as the DataMatrix rough range in the auto-positioning process, which is introduced above.

In manually positioning, the advantage is to ignore the rough range selection process, but the application should accept different orientation scenarios. Since the "sliding searching algorithm" we introduced before only accepts the kernel sliding with vertical and horizontal direction, the method to solve the orientation problem is to rotate DataMatrix, let DataMatrix L-shape in "horizontal-vertical" orientation.

- **Manually positioning steps**

Apply "Affine transform rotation algorithm" which was introduced in chapter 2.1.2 in the manually positioning process. The analysis flow is similar to the auto positioning process, moreover adopting the recovery angle from detected L-shape in the affine transformation process. The analysis flow is:

  * Scanning area selection; Binarization process; Image denoising;
  * Line detection; L-shape detection and kernel optional range selection;
  * Affine transformation;
  * Sliding searching algorithm
  * Precision location selection.

Briefly speaking, based on the initial scanning area, the binarization and denoising process aims to obtain a clearly binary image, the process is introduced in the auto positioning process before. In the binary image select the relative longer lines which could constitute the L-shape. Iterating all the lines to select all possible L-shapes, record the length and vertical deviation angle which obtained from the L-shape detection algorithm. Vertical deviation angle is used for rotating the DataMatrix in "vertical-horizontal orientation" and length is used for obtaining the kernel options. Therefore, the DataMatrix position is the kernel position which has the maximum matching accuracy, after the "sliding searching algorithm" search with some optional kernels.

Notice that the sliding algorithm defaults the kernel slides in the vertical and horizontal direction, the rotated scanning region has some redundancy area that could not be scanned, because the length in the redundancy area is shorter than kernel dimension (Figure 2.15). In other words, the initial square scanning does not accept the larger DataMatrix rotation in a wide-angle, because the DataMatrix should always be settled inside the scanning region.

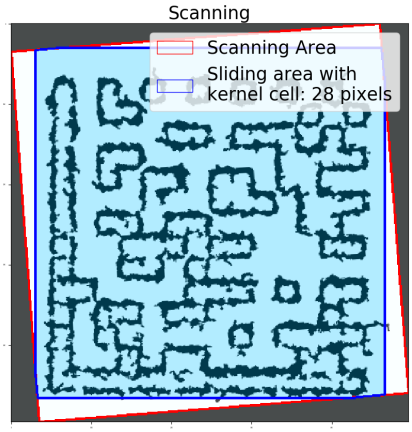
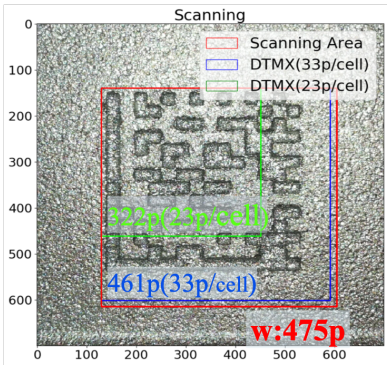


Figure 2.15: Scanning area rotation

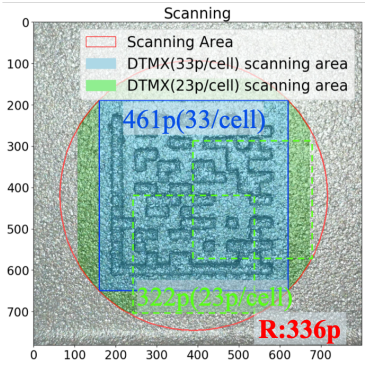
- **Initial scanning area comparison**

To solve the rotation degree issues and to decrease the calculation time as much as possible, I test three different initial scanning areas. Compare areas and analyze the advantages and disadvantages of each option.

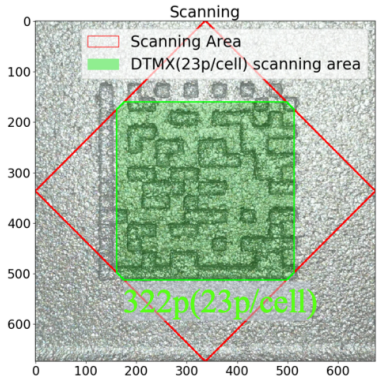
Three initial scanning areas are square area (Figure 2.16a), circle area (Figure 2.16b), and rhombus area (Figure 2.16c). These areas have a common feature that accepts the DataMatrix at most 33 pixels per cell. Assume 33 pixels because in this study case we have DataMatrix with 28 pixels per cell, the scanning area should be larger than DataMatrix. Thesis uses the maximum capacity constraint to compare the performance of different methods, however, the capacity parameter should be optimized in the real mobile phone application, which is another project in the future.



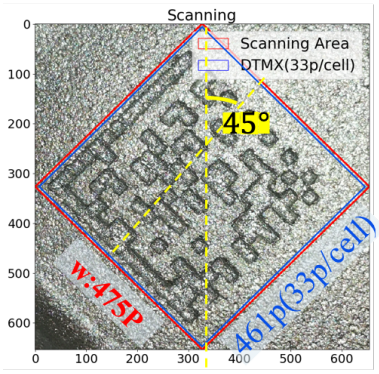
(a) Square scanning area



(b) Circle scanning area



(c) Rhombus scanning area



(d) Rhombus scanning area 45°

Figure 2.16: Initial scanning area comparison

Different scanning regions have different performance, thesis focus on the **calculation time performance** and **usability**. In the table 2.1, list the comparison results to see which is more suitable for the application.

	Accumulate Calculation time(s)(27-29 pixel/cell)	**Performance**
Square area	2.4s	Pro: Near real-time. Con: Few angle rotations.
Circle area	$\geq 10s$	Pro: Full angle rotations. Con: More time costing.
Rhombus area	$\leq 2.4s$	Pro: Less time costing. Con: Higher usage requests.

Table 2.1: Initial scanning region comparison

For the square area, since the initial scanning area is little larger than the DataMatrix area, the sliding algorithm needs less calculation time. When the kernel dimension decreases, the number of iterations increases and calculation time correspondingly increases. In the previous experiment, we use some kernel options to decrease the time, therefore we also focus on some optional kernel instead of all possible dimensions. In the study case, our kernel options are [27,28,29] pixel per cell, the accumulated calculation time for optional kernels is around 2.4 seconds, it is near-real-time, more or less acceptable for the mobile phone application (Figure 2.17). However, the disadvantage for this kind of initial area is the larger DataMatrix cannot rotate in a wide-angle.

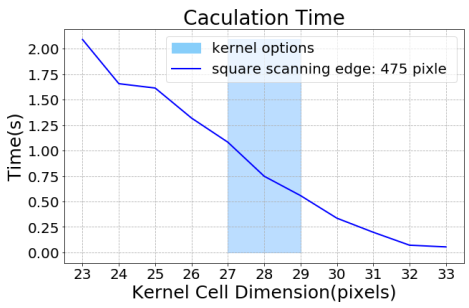


Figure 2.17: Square sliding time

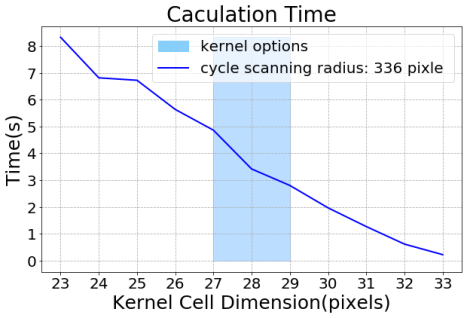
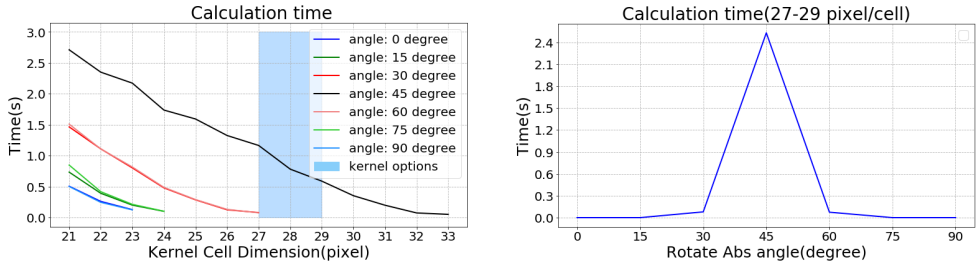


Figure 2.18: Circle sliding time

For the circle initial area, if the largest inner cut rectangular accepts kernel cell 33 pixels, meaning that the DataMatrix which is smaller and equal than 33 pixels per cell can be rotated with every possible angle, it is the advantage for this circle scanning region. However, when the kernel becomes smaller and smaller for example from 33 reduce to 23 pixels, for the same scanning area the number of iterations increases, moreover, the scanning area becomes larger and larger (Figure 2.16b), in the meantime the kernel has more and more possible positions. Therefore, the calculation time increases faster than the square area, because in the square region, the scanning area is constant. In the study case, the accumulated calculation time for all optional kernels is more than 10 seconds, under the circle initial area condition which contains a maximum of 33 pixels per cell DataMatrix (Figure 2.18).



(a) Rhombus sliding time (b) Rhombus accumulate time(27-29)

Figure 2.19: Rhombus area sliding time

In some scenarios the mobile phone is requested to keep the same distance with DataMatrix, in this condition we choose rhombus as the initial area, the rhombus is the square which rotates 45 degrees. Choose rhombus because it is a way to connect the calculation time and rotation angle. Specifically, when the input DataMatrix is a smaller one, for example under 23 pixels per cell (Figure 2.16c), the default rhombus performs well, because the scanning area is small. However, if a larger DataMatrix input, the way to accept large DataMatrix is to rotate the mobile phone, keep DataMatrix inside the rotated rhombus (Figure 2.16d). Different rotation angles accept different maximum kernel dimensions. In our condition, since we know the kernel options, according to the curve of calculation time (Figure 2.19a), obviously the rhombus which rotated under 15 degrees cannot accept our optional kernels. Therefore, the accumulated searching time of these three kernels depends on the rhombus absolute rotation angle (between 15°-75°) and less than 2.4 seconds (Figure 2.19b), absolute angle means the phone can rotate in clockwise also in anticlockwise.

Thus, the advantage of rhombus scanning is the calculation time decreasing. Suppose the user rotates the phone at a suitable angle which accepts DataMatrix inside the scanning region and close to the edges. In this case, the calculation time is less than the square area (obviously, 45° rhombus rotation has the same performance as the square). However, if a smaller DataMatrix rotated in a larger angle, it cost time. The worst condition is rotating rhombus with 45°, the performance is the same with the square area in the first case. The disadvantage of the rhombus is the usability, it requires a higher requirement to users, larger DataMatrix needs a larger rotation angle.

In conclusion: according to the performance, the scanning region adoption

depends on the scenario and scenario request. In the thesis, we adopt the **square initial area**, because it is the typical shape we use in daily life, and has a balanced performance between calculation time and usability.

4) Conclusion

In the position localization stage, since our shot-blasted DataMatrix has breakage contours, we design a "sliding searching algorithm" to search the DataMatrix location in the image, detect lines and L-shape in order to obtain some optional kernel dimension to reduce the calculation time. We make two different positioning algorithms (Figure 2.20), in order to adapt in different scenarios. "Auto position localization" is design for the factory. Since the industrial camera requests to scan the whole image, we use "rough range" searching process and "precision location" searching process to reduce the searching time."Manually position localization" is used for the mobile phone application, we have a defined scanning region in advance, however, the scanning orientation is variable. Using the L-shape oblique angle, we rotate the DataMatrix to "vertical-horizontal" orientation and search the location. Moreover, we test different initial scanning regions, study the performance variation.

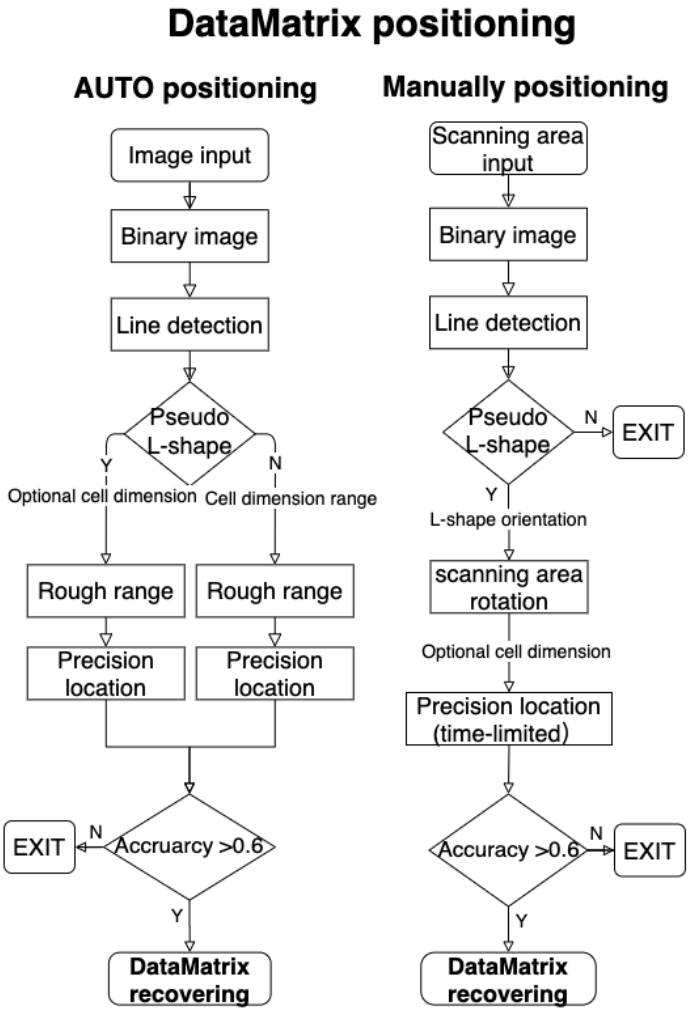


Figure 2.20: Position localization flow

2.1.4 Stage 3 DataMatrix code Recovering process

The position localization process gives us the DataMatrix position, because of the step in the sliding searching algorithm is one pixel, the received location has a higher confidence level. The next step is to recover the DataMatrix cell by cell, transforming the DataMatrix to the traditional standard format, using one-pixel blocks to represent each cell.

DataMatrix recovering methods focus on cell performance. Gridding the DataMatrix, for each cell using the percentage of the black pixels to deter-

mine the cell belongs to the black or white, this is the shared method in the literature[11][14]. However, the problem we faced is: our DataMatrix belongs to rough contour DataMatrix, the cell only reserves the contour, therefore the black percentage method is not suitable for the contour DataMatrix condition (Figure 2.21).

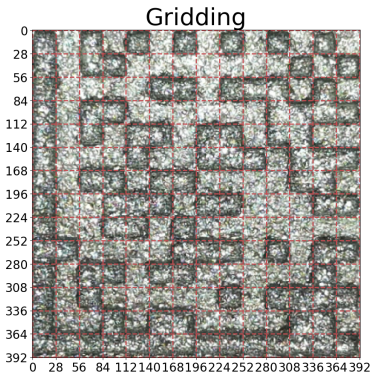


Figure 2.21: Gridding contour DataMatrix

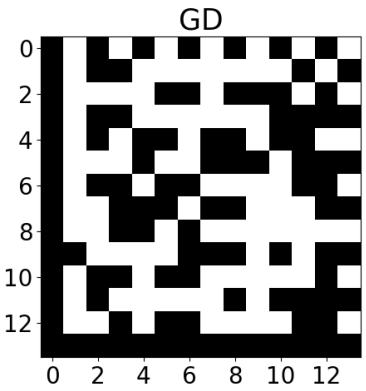


Figure 2.22: Ground truth

1) Conception

Here we introduce some basic conceptions which we use in this subsection.

I) Ground truth
Since we already know this DataMatrix sample's content, we create the standard DataMatrix show in the image (Figure 2.22) called **"ground truth"**. Compare recovered DataMatrix with the ground truth to measure the DataMatrix recovering performance and using the confusion matrix to represent the recovery result.

II) Confusion matrix
Confusion matrix is a specific table layout that allows visualizing the algorithm error distribution. Confusion matrix has two dimensions ("actual" and "predicted"). In the confusion matrix each row represents the instances in an actual class while each column represents the instances in a predicted class. Moreover, each block has the algorithm performance meaning: Positive(T); Negative(N); True Positive (TP); False Positive (FP); True Negative (TN); False Negative (FN) (Figure 2.23[3]).

[3]https://en.wikipedia.org/wiki/Confusion_matrix

Thesis focuses on the FN and FP, FN means the cell which should be black but predicted white (named **'Erasure'**) while FP means the cell which should be white but predicted as black (named **'Error'**). According to the ratio between erasure and error and the summation of erasure and error, to measure the recovering performance (Figure 2.24).

		Predicted class	
		P	N
Actual class	P	TP	FN
	N	FP	TN

Figure 2.23: Confusion Matrix

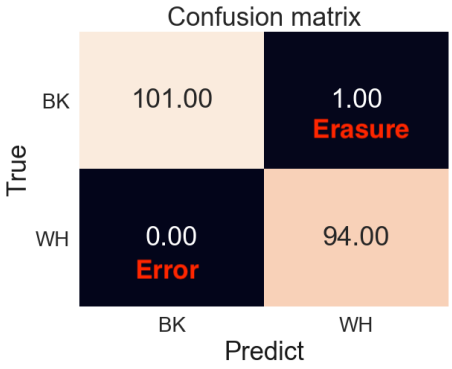


Figure 2.24: Error and Erasure

III) Recovering Baseline

Since the rough contour DataMatrix is not mentioned in any other literature, we put forward this new object and create the baseline for this kind of object.

The purpose of DataMatrix recovering is to decode the content. According to the DataMatrix attributes table (table 2) in the annex, different DataMatrix has different threshold ratios between error and erasure. Thanks to the error correction codewords, the DataMatrix can be decoded, if the ratio and the total instance number of error and erasure are less than the threshold, otherwise, this DataMatrix will lose its function.

Thesis adopts 14*14 type of DataMatrix, therefore the baseline in the thesis is the following:

$$\{\frac{eo}{ea} \leq 5/7, eo + ea \leq 12\} \bigcap \{ea = 0, eo + ea \leq 12\} \quad \text{(Baseline)}$$

eo: Error ea: Erasure

IV) Normalizing algorithm

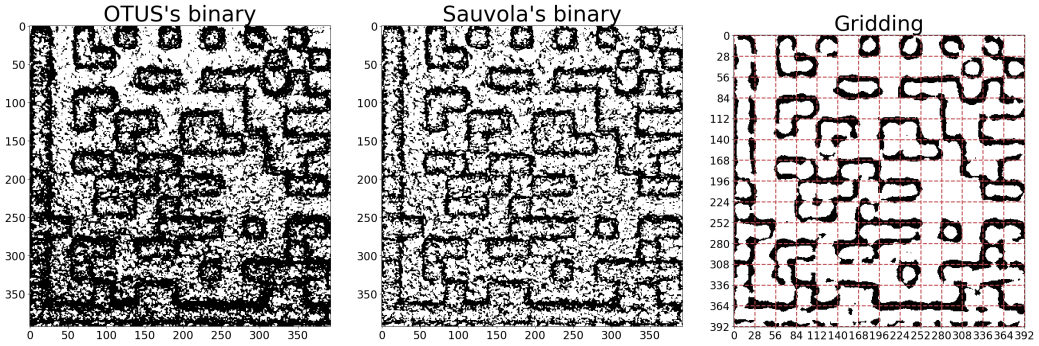
Normalize is the way to eliminate the effects of the dataset unit, by

scaling the numbers. The thesis adopts the "Min-Max Feature scaling" scale results from 0 to 1, to process the result matrix Eq.2.13.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{Eq.2.13}$$

2) Pre-processing

To solve the problem of "rough contour DataMatrix", I improve the method which the literature mentioned. From the beginning, since we already obtain the DataMatrix location, to avoid the other parts which do not belongs to the DataMatrix position effect the DataMatrix binarization result, it is better to re-do the binarization process. Same as the positioning process, I select the Sauvola's binary image as default binary algorithm because it has less noise than other methods like Otsu's image (Figure 2.25a Figure 2.25b).



(a) Otus's binary image (b) Sauvola's binary image (c) Sauvola denoising image

Figure 2.25: DataMatrix binary image

After binarization, many noises still exist, adopting the smoothing and morphology algorithm which was introduced in "positioning processing", to obtain the denoising image shown in the image (Figure 2.25c). Since in the beginning, we hypotheses the DataMatrix belongs to 14*14, we can divide the image into 196 small parts. From the gridding result, each small part can represent the cell in DataMatrix, it verifies the positioning process works well. With the gridding image, I create two methods to recover each cell to obtain a complete DataMatrix.

The two methods of contour DataMatrix recovering are **"contour cell recovering"** which focus on the cell's contour and **"Filled cell recovering"**

which focus on the whole cell.

3) Method 1: Contour DataMatrix Recovering

"Contour cell recovering" using the cell's contour to recover the cell to the one-pixel block. In the beginning, I created a "standard contour cell", which black edge occupied a quarter of the cell width, it seems like the cell in the binary kernel which is introduced in "sliding search algorithm". Compare each cell in the gridding map with the standard cell to calculate the black edge accuracy and using the accuracy threshold to classify the gridded cells. Different with the sliding searching algorithm, here we only consider the black color, because in the DataMatrix binary image, black pixels almost belong to the cell edge, moreover single-color accuracy computation costs less time than double colors.

$$accuracy_{BK} = \frac{correctly\ matched\ black\ pixels}{standard\ cell\ black\ pixels} \qquad \text{(Eq.2.14)}$$

In the contour cell recovering algorithm, we extract and recover the data region of DataMatrix, data region has 12*12 cells and it records information. Other DataMatrix parts like the solid L-shape and alternate L-shape have constant shapes, are not necessary to be recovered from the binary image. Therefore, the recovering process is the following (Algorithm 4):

Algorithm 4 Contour cell recovering

Input:
1.Gridding DataMatrix
2.Standard cell
Output:
Regular DataMatrix;

Extract the data region (12*12)

for cells in gridding data region **do**
compute and record the black color matching accuracy with standard cell
end for

```
Normalize the total result
Normalize threshold=1

while Normalize threshold≥0.5 do
  compute the accuracy threshold according the Normalize threshold
  for cell in gridding data region do
    if cell accuracy≥accuracy threshold then
      return black cell
    else
      return white cell
    end if
  end for

  Add L-shape and decode recovering result

  if Successful decoded then
    return BREAK
  else
    Normalize threshold =Normalize threshold-0.1
  end if
end while
```

According to the recovering result, sometimes it cannot be decoded because some cell connection parts have fewer or do not have black edges. Less black edges cause these parts are mis-predicted. Therefore the second method "filled cell recovering" is to recover the cell connection part.

4) Method 2: Filled cell recovering

To avoid the cell connection part cannot be recovered, after the contour cell recovering process, using the filled cell recovering process to improve the DataMatrix recovering result.

"Filled cell recovering" method comes from the reference literature. Calculate the black pixels percentage in the cell to determine the cell. Before the black pixel counting, cells should be filled, especially filling the cell connection part. The filling algorithm is developed from the "Contour Detection Algorithm" (Figure 2.5) which is introduced in chapter 2.1.2. Each cell contour could be obtained using "Contour Detection Algorithm". Theoretically,

the cell contour should be complete, therefore the contour should be classified as outer outlines and inner outlines, we select the inner outlines and fill black color inside. However, the contour is roughly reserved, some contours are broken, therefore, the breakage contour does not have inner and outer outlines and it cannot be filled. In the end, the filled cell, which has more than 90% of black pixels, belongs to black cell.

In the "filled cell recovering process", I do not apply the morphology erosion algorithm to connect the contour's crack. Because the distance between contours is very close, once using erosion algorithm, the crack is successfully repaired, in the meantime, different cell's contours are also connected. In the end the recovering result has lots of black area that should not exist.

5) Recovering result addition

Now we have two methods to recover the DataMatrix, 'contour cell recovering' and 'filled cell recovering'. The final step is to compensate black pixels with each other, obtaining the final recovering result, which not only has single cells but also has cell connection parts.

6) Conclusion

In conclusion, the contour recovering algorithm using two methods to recover the cell and cell connection part, transform the rough contour DataMatrix to a traditional style with one pixel for each block. In the end, decoded the recovery result by Python open source library ("pylibdmtx").

The confusion matrix is used to measure the recovering performance since we have the ground truth in advance, however in the real usage scenario, users do not have the ground truth, therefore this algorithm should give them the decoding result quickly and stability. With the recovering algorithm, I design a DataMatrix recovering flow (Figure 2.26). After the DataMatrix image binarization and the data region extraction, firstly using the contour cell recovering method to recover the DataMatrix, in the meantime the normalize threshold (BET) which is used to determine the cell default as 1.0. If the first method cannot be decoded, try the second method. Furthermore, if the second method decoding result is still empty, update the threshold until the threshold is less than 0.5. In my experiment, when threshold equal to 0.5, many white cells are mis-predicted as black, or maybe 0.5 is suitable for DaraMatrix with thin edges, but we do not want to analyze the DataMatrix

with weak edges, it is an unclear DataMatrix actually.

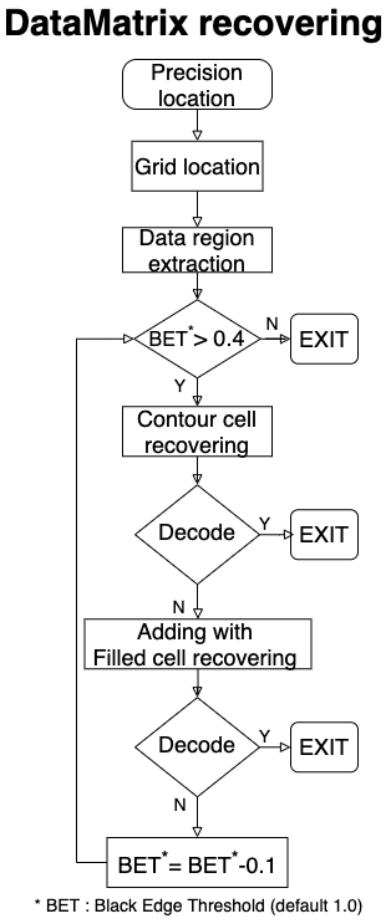
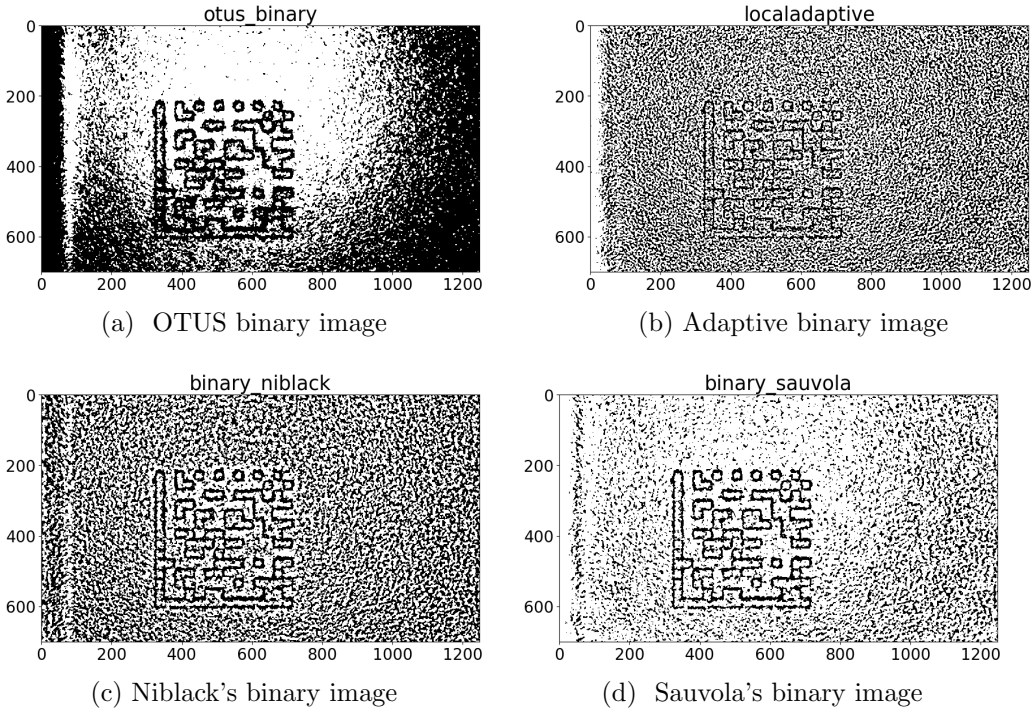


Figure 2.26: DataMatrix recovering flow

2.2 Pre-processing results

Pro-processing is the first step to process the image. Using the method introduced in the last subsection, after grey processing and blurred image processing. Comparing different binarization methods, I select Sauvola's algorithm to binary the image (Figure 2.27). Because Sauvola's algorithm decreases the number of noises and makes a better distinction between DataMatrix contour and noise.



(a) OTUS binary image

(b) Adaptive binary image

(c) Niblack's binary image

(d) Sauvola's binary image

Figure 2.27: Different binarization

2.3 Position localization results

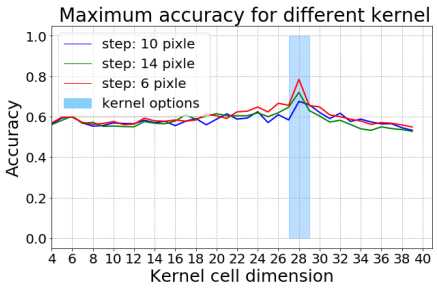
1) Auto position localization

Auto positioning is adopted in the factory in which the camera is fixed on the machine. In this condition, we do not consider the orientation problem, but the scanning area should be the total image.

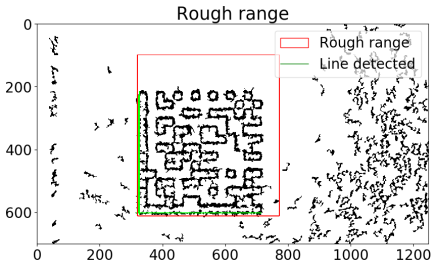
- **Rough range selection**

In the study case, after image binarization and noise area filtering. Using L-detection algorithm we successfully shrink the options range from 40 numbers to 3 numbers (Figure 2.28a), In the optional range ([27;28;29] pixel/cell) with the default sliding step (10 pixel), select the most representative cell

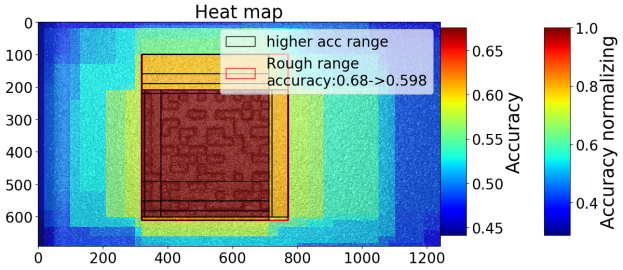
dimension which has the largest maximum accuracy, in our case it is 28 pixels per cell. Using normalize threshold (defaults 0.7) select some kernel positions to constitute a rough range, the heat map rough range result is shown in Figure 2.28c, In addition from Figure 2.28b , it is easy to see relations between DataMatrix and rough range.



(a) Kernel selection from options



(b) Rough range

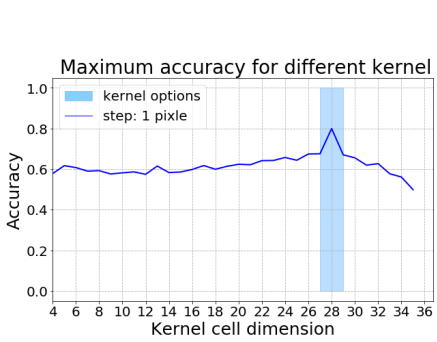


(c) Heat map with kernel cell 28 pixels per cell

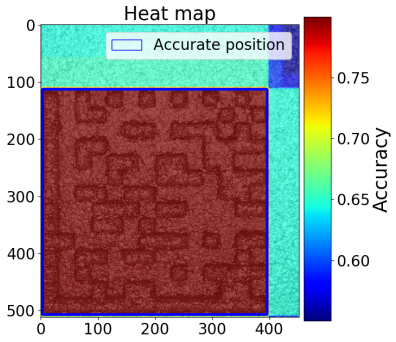
Figure 2.28: DataMatrix code auto positioning rough range

- **Precision location selection**

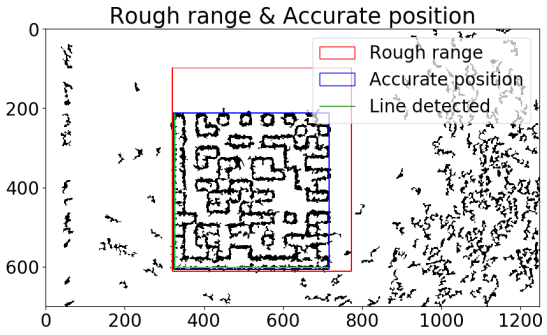
Based on the rough range, we reset the sliding step equal to one pixel, redoing the "sliding searching process" and selecting the most representative cell from the options (in our case from set [27;28;29] select 28 pixel/cell) (Figure 2.29a). In the selected heat map (Figure 2.29b), the DataMatrix position is the kernel position which has the maximum matching accuracy. After these processes, we get the DataMatrix location automatically (Figure 2.29c). The square looks near to the actual DataMatrix position.



(a) Kernel selection from options



(b) Heat map with kernel cell 28 pixels per cell



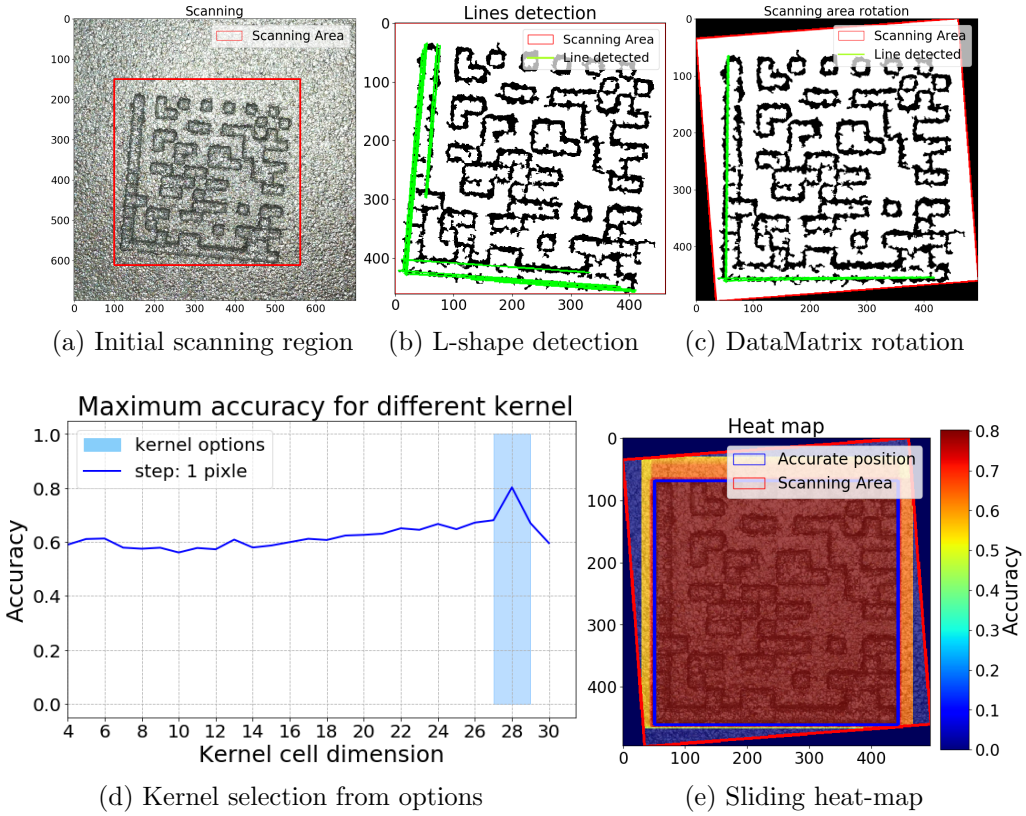
(c) Precision location

Figure 2.29: DataMatrix code auto positioning precision location

2) Manually positioning

In the experiment, we simulate the scenario that using the square as an initial scanning region, to scan the oblique DataMatrix. The DataMatrix has been rotated 5 degrees in advance (Figure 2.30a). Processing the image with the analysis flow introduced in chapter 2.1.3 (manually positioning). Detecting lines in the binary image (Figure 2.30b), selecting the L-shape from several lines. In the study case two L-shapes were detected, the average rotation angle of L-shapes is 4.4°, and the optional range of kernel cell length is [27,28,29] pixels per cell. Rotating DataMatrix to "vertical-horizontal direction" (Figure 2.30c) and using "sliding searching algorithm" to select the suitable cell dimension. Obviously, in the curve the maximum matching accuracy is around 0.8 when the cell equals 28 pixels (Figure 2.30d), thus the DataMatrix position is selected in the heat map when the kernel cell has 28 pixels and kernel position has the maximum matching accuracy

(Figure 2.30e).



(a) Initial scanning region (b) L-shape detection (c) DataMatrix rotation

(d) Kernel selection from options (e) Sliding heat-map

Figure 2.30: DataMatrix code manually positioning

The result of DataMatrix location and matching accuracy is similar in manually positioning process and auto positioning process, it demonstrates that the rotation algorithm and manually positioning analysis flow performs well. The oblique DataMatrix positioning has the same performance with ideal orientation DataMatrix positioning.

3) Comparing

For now, the thesis adopts two methods to locate the DataMatrix. Auto position localization is used for manufacture and manually position localization is used for the mobile phone application (Figure 2.31). The way to measure the positioning process is the calculation time.

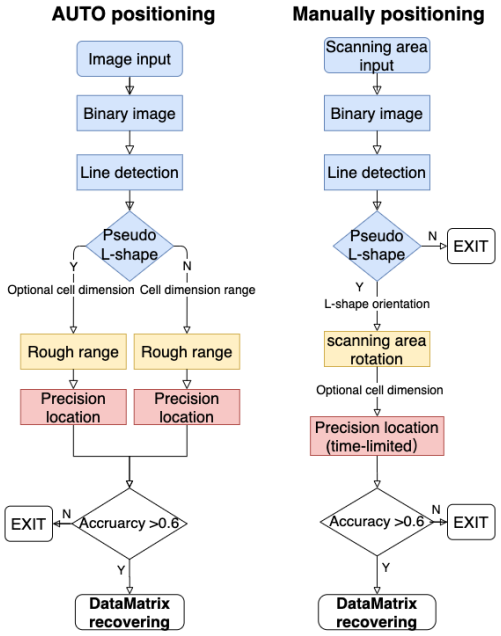


Figure 2.31: Position Localization

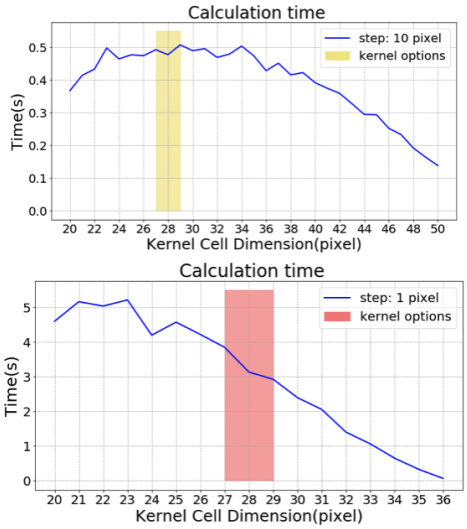


Figure 2.32: Auto positioning time

In the auto positioning process, since it requests two times sliding, the total calculation time should be rough range searching time plus precision location searching time, However, in the manually positioning process, the calculation time is equal to precision location searching time.

In the study case, after obtaining some kernel options, the auto positioning needs 1.4 seconds to obtain the rough range and 11 seconds to obtain the precision, the total time is around 12 seconds (Figure 2.32). For manually positioning, using the square as the initial scanning region, the sliding time is around 2.4 seconds (Figure 2.17).

4) Conclusion

In conclusion, these two positioning processes have the similar positioning result and matching accuracy, demonstrate the suitability of these two algorithms. However, manually positioning can decrease the calculation time a lot, because it gives a possible position range in advance, and we can control the time by controlling the initial scanning region. For the auto positioning process, the time is hard to predict, because it is hard to predict the rough range area, the larger rough range area costs more calculation time, however it can scan every part of the image, locate the position automatically.

2.4 DataMatrix code recovering results

From the DataMatrix positioning algorithm, we obtain a precision location. Since we know the type of DataMatrix, it is easy to separate the location into many cells. The recovering method aims to recover each cell, in order to obtain a complete DataMatrix.

Following the binarization process and using morphology algorithm to omit noise, we have gridding contour DataMatrix image (Figure 2.33). Extract the data region in the central and compare with the standard cell. Since we already know this DataMatrix sample's content, we create our ground truth (Figure 2.34).

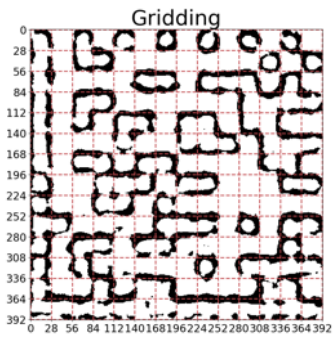


Figure 2.33: Gridding contour DataMatrix

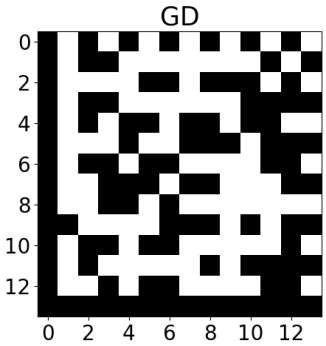


Figure 2.34: Sample ground truth

- **Contour cell recovering**

In this study case, the standard cell has black edge which occupied a quarter of width. using the color map to represent the black accuracy comparing result of the DataMatrix data region (Figure 2.35), higher accuracy means this cell has a larger probability to be predicted as the black cell. For instance, select 0.6 on the normalizing bar as the threshold to classify the cell, we obtain the recovering result (Figure 2.36).

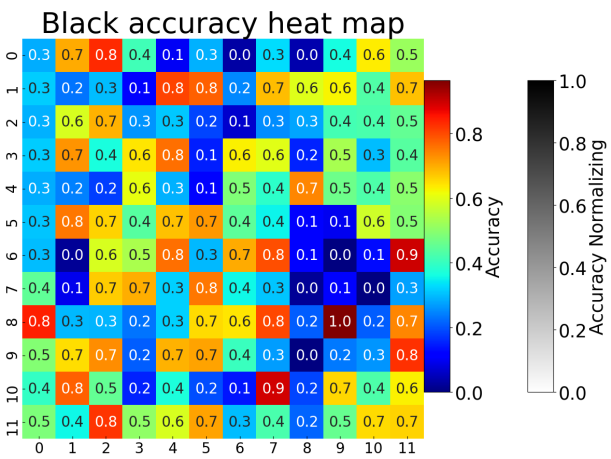


Figure 2.35: Black accuracy color map

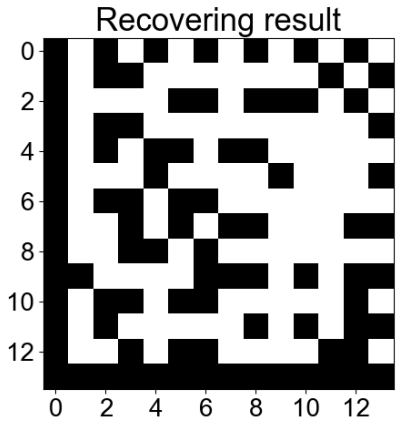


Figure 2.36: Contour cell recovering

Compared with the ground truth (Figure 2.34), the recovering result represents most cells in the DataMatrix. However, the connected cell part which is in the top right of the DataMatrix is not recovered, because the cell in the connection part has few black edges around, therefore cells in the cell connection part are predicted as white. From the confusion matrix (Figure 2.37), the total mistake is over the baseline threshold which accepts 12 mistakes totally, therefore this result cannot be decoded.

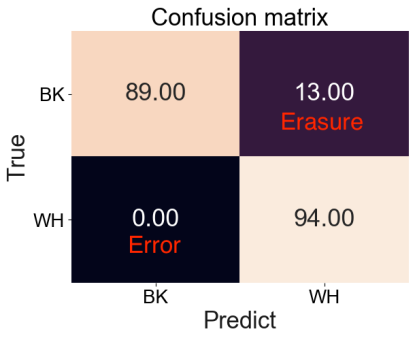
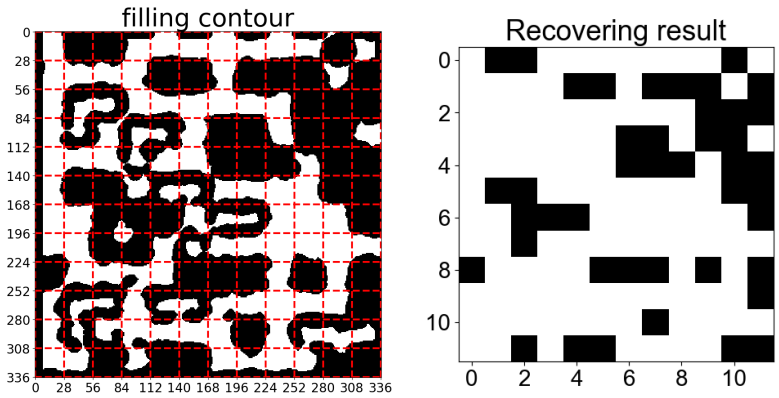


Figure 2.37: Contour cell recovering confusion matrix

- **Filled cell recovering**

To avoid the cell connection part cannot be recovered, after the contour cell recovering process, using the filled cell recovering process to improve the DataMatrix recovering result.

Filling the closed contours in the DataMatrix data region. In the image (Figure 2.38) obviously the complete contours are filled and recovered, other incomplete contours still cannot be recovered.



(a) Cell filling (b) Filled cell recovering

Figure 2.38: Filled cell recovering

- **Recovering result addition**

The final step is to compensate black pixels between the first method with the second method, obtaining the final recovering result, which not only has single cells but also has cell connection parts.

The image shows the final recovering result (Figure 2.39), it seems good. Compare the result with the ground truth (Figure 2.34), using a confusion matrix to represent the matching statistic. In the confusion matrix (Figure 2.40), the total black pixels are 102 including L-shape, and we predict 101 of them, only one black pixel gets a mistake that is predicted as white. While all of the white pixels are correctly predicted. Finally, we have only one erasure mistake, the result satisfied by the baseline request which accepts totally 12 mistakes and can be decoded to “ABCDE12345” as we expect.

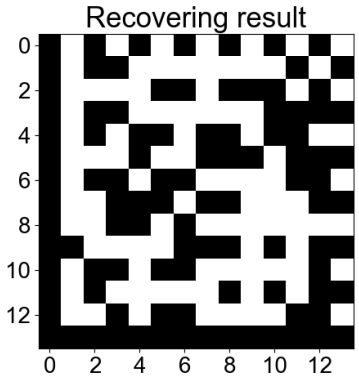


Figure 2.39: Recovering result

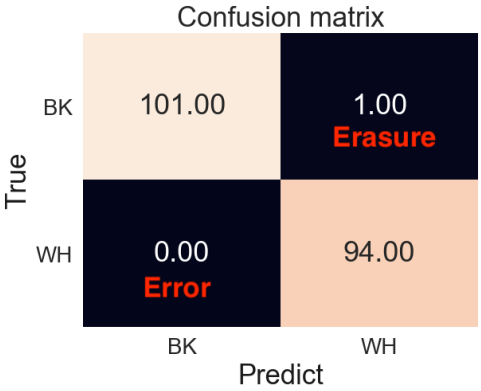


Figure 2.40: Confusion matrix

2.5 Additional (Neural Network recovering)

The thesis also proposes to use a Neural Network to recover the DataMatrix in "position localization process". Firstly, Let's see what the Neural Network is, what is the basic structure and components in the Neural network.

1) Background

Artificial neural networks (ANNs), simply called neural networks (NNs) is a technical algorithm of machine learning, the principle of the neural network is inspired by the physiological structure of our brain-neurons that are interconnected with each other. However, unlike a neuron in the brain that can connect to any neuron within a certain distance, an artificial neural network has discrete layers, connections, and data propagation directions. Same as machine learning, the Neural network is also classified into three classes: unsupervised learning, supervised learning and reinforcement learning.

- Unsupervised learning using because we lack sufficient prior knowledge, it is necessary to solve pattern recognition problems based on training samples with unknown categories (not labeled). The typical algorithm is like the K-means algorithm and principal component analysis (PCA).
- Supervised learning using samples with known or specific characteristics as the training set to establish a mathematical model (weight model in artificial neural network method, etc.), and then use the established

model to predict unknown samples. It is the most common machine learning method. In supervised learning, each example is a pair consisting of an input object and labeled output value. The supervised learning algorithm analyzes the training data and produces an inferred function, which can be used to map new examples. An optimal model would allow the algorithm to correctly determine the class label when the label does not exist.

- Reinforcement Learning is one of machine learning paradigms and methodologies. It is used to describe and solve the learning process of the agents in the interaction environment. Aims to maximize returns or achieve specific goals.

In this part, the thesis uses a neural network to train a model that can classify the cell according to their attribute. The training process is a supervised learning process since we have the ground truth DataMatrix at the beginning, the ground truth cells are the certainly known labels. Then using trained models to predict other cells' attributes.

2) Neural network structure introduction

- **Single layer (node)**
  In a neural network, the most basic structure is the single layer network (Figure 2.41). In single layer NN. The neural receives input signals from other n input neurons. The input signal is transmitted through a weighted connection, the output of the neural is the summation of all weighted single plus a bias value. Active the neural output to obtain the final output of the single layer.

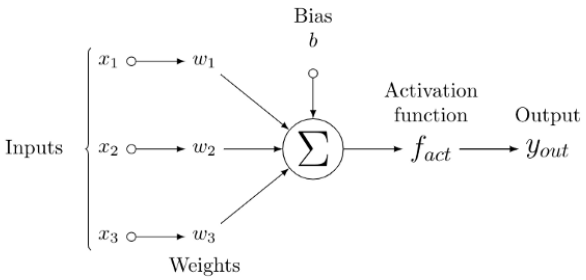


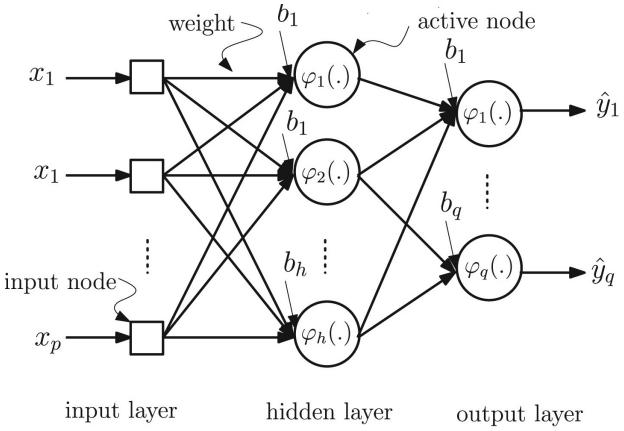
Figure 2.41: Single layer NN

The activation function adds a nonlinear factor to the neuron, so that the neural network can approximate any nonlinear function arbitrarily, therefore the neural network can be applied to many nonlinear models.

$$\text{output} = \text{g}\left(\sum_{i=0}^{n} x_i w_i + b\right)$$

g:activation function

- **Multi-layer feedforward neural network**
  Multi-layer feedforward NN (Figure 2.42 [4]) contains three layers: input layer, hidden layer and output layer. The input layer neurons receive external input signals, the hidden layer and the output layer process the signals, and the output layer neurons process the final result. Feedforward means that external signals reach the output layer from the input layer through the hidden layer, and there is no reverse propagation of the signal.



Three-layer feedforward neural network

Figure 2.42: Three-layer feedforward NN

Noticed that the Hidden layer could store many other layers and nodes, the function of the hidden nodes is the same as before, however, mostly activation function is only used in the last layer before the output layer. Because activation function is used for decreasing the algorithm

[4]Image from "Metaheuristic design of feedforward neural networks"[15]

complexity, and one time is enough, more activation using will decrease the output result accuracy.

- **Back-propagation**

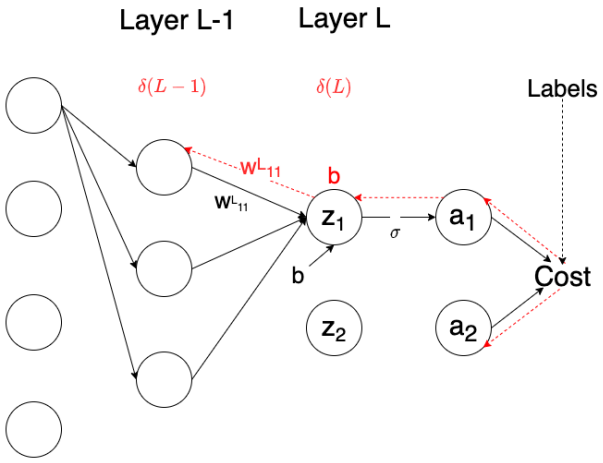


Figure 2.43: Back-propagation NN

BP (back-propagation) neural network is a multilayer feedforward neural network, trained according to the error back propagation algorithm, and it is the most widely used neural network. The purpose of error back-propagation is to minimize the NN's cost function by updating the weight and bias of the nodes in the hidden layer (Figure 2.43). The back-propagation process is represented using four BP equations.

$$\delta_j{}^L = \frac{\partial c}{\partial a_j{}^L}\sigma'(z_j{}^L) \tag{BP.1}$$

$$\delta^{L-1} = \left((w^L)^T\delta^L\right) \odot \sigma'(z^{L-1}) \tag{BP.2}$$

$$\frac{\partial c}{\partial b_j{}^{L-1}} = \delta_j{}^{L-1} \tag{BP.3}$$

$$\frac{\partial c}{\partial w_{jk}{}^{L-1}} = a_k{}^{L-2}\delta_j{}^{L-1} \tag{BP.4}$$

BP.1 represents an auxiliary variable ('error') δ of the node j in layer L, an auxiliary variable related to the cost (c), and the activation result (a).

BP.2 demonstrates how the auxiliary matrix updates from the deeper auxiliary matrix, with the weight(w) and node(z).

BP.3 and BP.4 are the processes to update the bias(b) and weight(w) from the auxiliary variable in the same layer

- **CNN**
  Convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery, that can recognize and classify features in images (Figure 2.44).

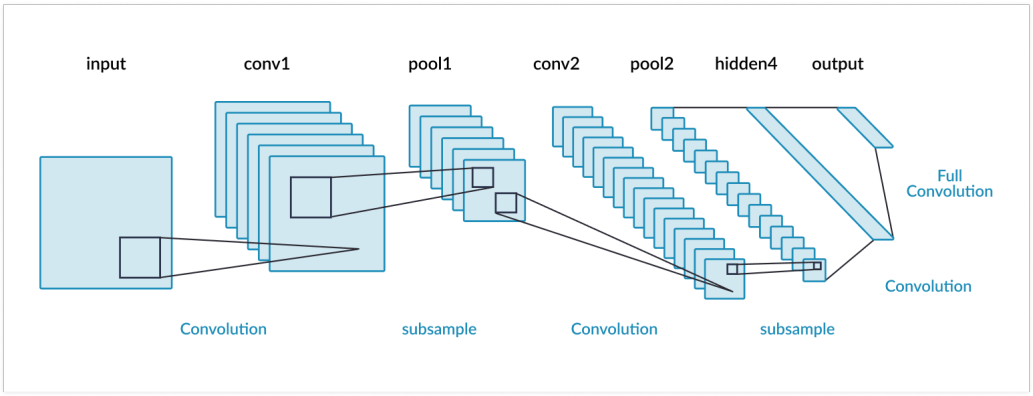


Figure 2.44: Convolution Neural Network

A CNN is composed of several kinds of layers:

  - Convolutional layer: Each convolutional layer in a convolutional neural network is composed of several convolutional units, and the parameters of each convolutional unit are optimized through the back-propagation algorithm. The purpose of the convolution operation is to extract different features of the input. The first layer of the convolutional layer extracts some low-level features such as edges, lines, and corners. More layers of networks can iteratively extract more complex features from low-level features.
  - Pooling layer: scales down the amount of information the convolutional layer generates for each feature and maintains the most essential information. Specifically cut the image into several regions and the maximum or average value is taken to obtain a new feature with a smaller dimension. (the process of the convolutional and pooling layers usually repeats several times).

– Fully connected input layer: "flattening" the output generated by the previous layer to convert them into a single vector, in other words combine all local features into axial features and used in the next layer.
– Fully connected layer: After the input layer, the structure and function is the same as the back-propagation neural network which introduced above, applies weights over the input generated by the feature analysis to predict an accurate label.

In the last, CNN generates the final probabilities for different classes, to classify the input images.

3) Neural network recovering

Thesis using the back-propagation algorithm to recover the rough contour DataMatrix cell by cell, with the NN structure named 'Spatial Pyramid Pooling'(SPP). SPP neural network is similar to convolution neural network(CNN). However, CNN requests the input image have the same dimension because the input of the structure is pre-defined. Spp neural network improves this restriction, it accepts different dimension image input[16]. SPP neural network structure is shown in the image (Figure 2.45), and this is the structure I adopt in the thesis.

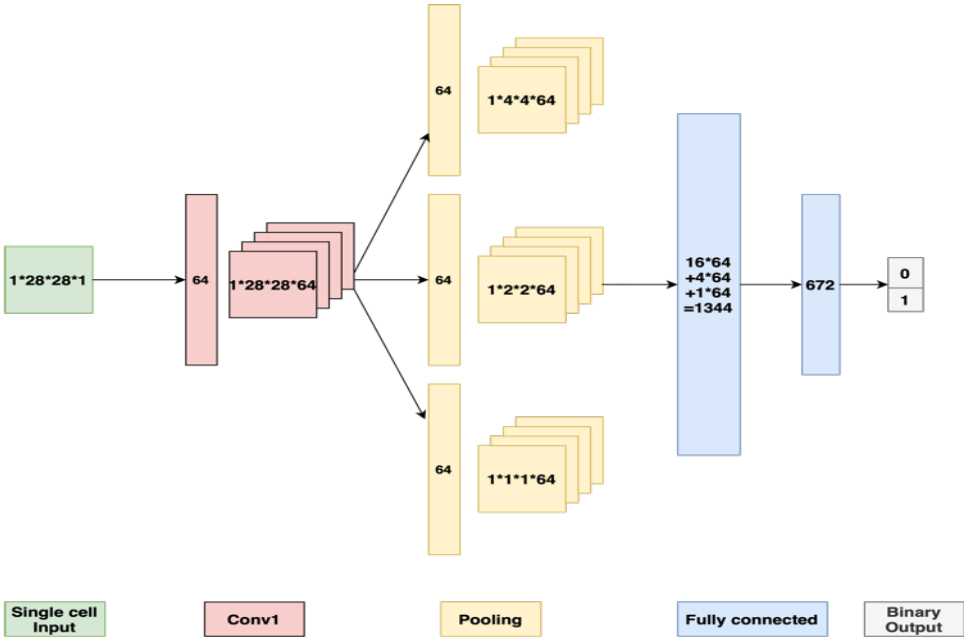
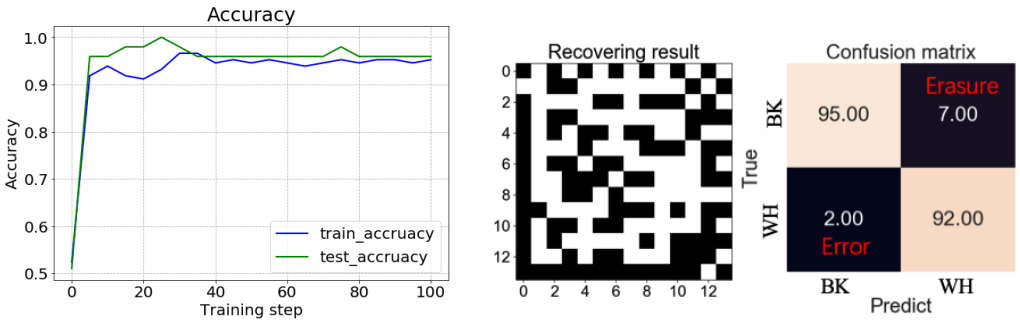


Figure 2.45: Spatial Pyramid Pooling

- Convolution layer: for each image, the convolutional layers use sliding filters, and their outputs have the same aspect ratio as the inputs. These outputs are known as features. Using the convolution method to extend the depth of the image. Since a feature map is extracted from different angles, it is robust for different images.
- Pooling layer: fully-connected layers require a fixed-length vector, SPP can maintain spatial information by pooling in local spatial bins. These spatial bins have sizes proportional to the image size, so the number of bins is fixed regardless of the image size.
- Fully connected layer: using fixed length vectors to represent any size of the image, it is possible to train the features in the vector with the back-propagation algorithm by updating the weight and bias, to minimize the loss in the cost function.

In the thesis, since DataMatrix has different dimensions, SPP is the most suitable structure. The input image of SPP is the single separated cell of DataMatrix, the output is the probability of black (0) and white (1). The number of depths extent and the number of pooling bins is the literature[16] proposition and adoption.

Since we have only one image and the ground truth (totally 196 cells), we select three-quarters of them as the training set others are test set. Because the training set is not huge, therefore it is not necessary to train many times. After training a hundred times, the training set accuracy and test set accuracy is around 0,96 and trend to stable (Figure 2.46a).



(a) SPP training accuracy

(b) Recovering result

Figure 2.46: SPP Neural network recovering

In specific, one of the recovering results demonstrated in Figure 2.46b, the result has nine mistakes totally, the accuracy is around 0.96 of all pixels.

Actually, the real mistake is seven not nine, because we focus on the data region, L-shape mistake can be corrected in the last. Anyhow, the ratio of error and erasure is satisfied with the Baseline (5/7), the result can be decoded.

In conclusion, the thesis verifies that the SPP neural network is suitable for the uncertain dimension cell recovering. The recovering accuracy seems good, but it still needs to be improved when the dataset becomes more complete. Since the problem is lack of dataset, the thesis does not go more in-depth to modify some specific parameters.

2.6 Recovery conclusion

In Conclusion, from the input rough contour DataMatrix code, the recovery process using the binarization and smooth approach process the input image to obtain the denoising image. And using two steps to recover and decode it. Two steps are **position localization** and **recovering**. Firstly, according to the different requests, the thesis put forward two position localization methods. One is **auto positioning** for the factory that accepts more seconds to process and is requested to scan the whole image, positioning the DataMatrix code automatically. Another one is **manually positioning** for the mobile phone application, which needs a quickly positioning process with small area scanning. Second, the recovering process following the position localization process, if the position matches DataMatrix more than 70%. Thesis proposes a **"contour DataMatrix code recovering"** analysis flow to recover the gridded DataMatrix code cell by cell with two succession processes that focus on cell recovering and cell connection part recovering. Transforming the rough contour DataMatrix code to regular shapes. Moreover, generate an idea to use Spatial Pyramid Pooling neural network to recover the gridded DataMatrix code, and verify the SPP suitability (Figure 2.47).

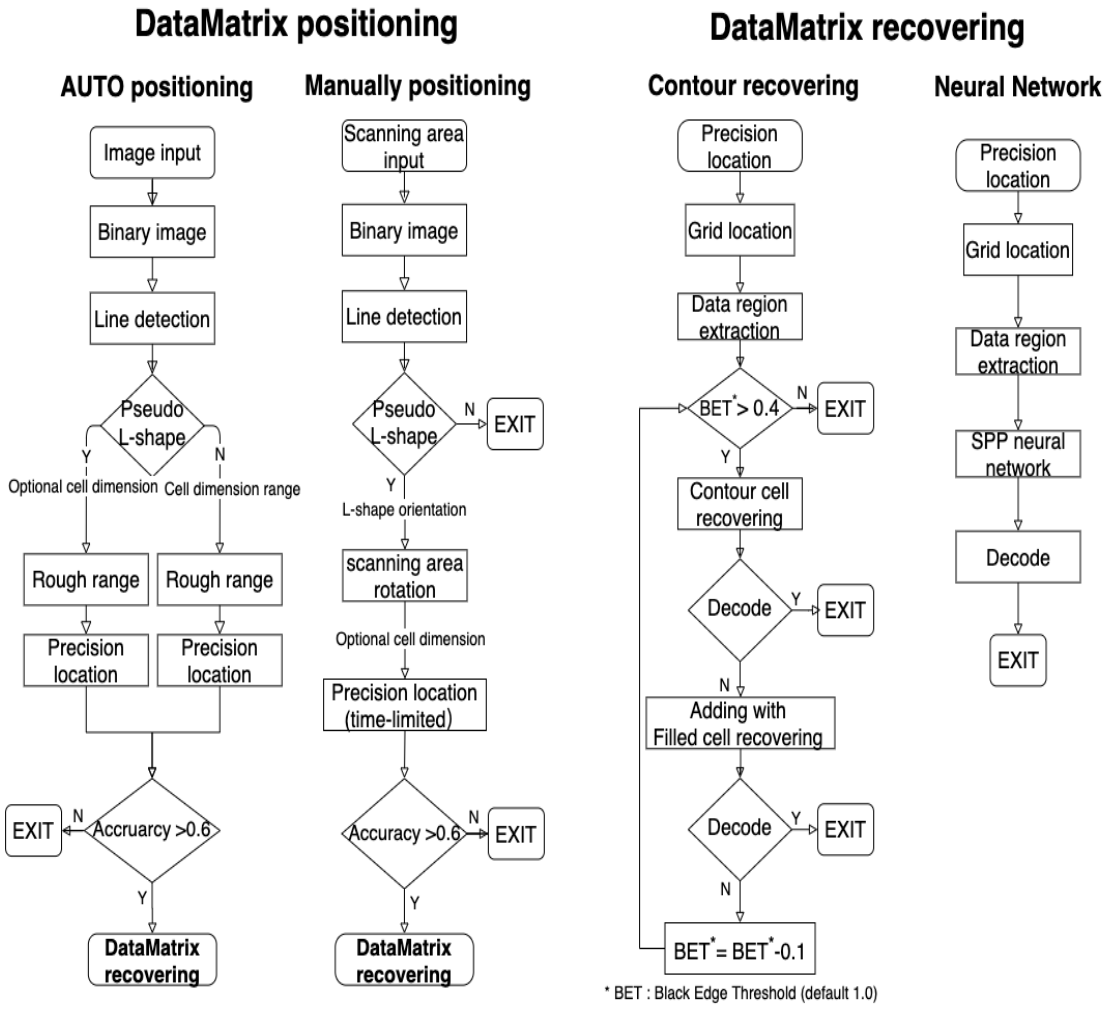


Figure 2.47: Recovery flowchart

Chapter 3

Traceability Platform

Chapter 3 is an intelligent manufacturing system (IMS) development, aims obtain a reliable Database to achieve I4.0. Section 3.1 introduces the IMS logic, and protocols adopted in the platform. Section 3.2 and 3.3 are the IMS database and platform simulation in order to share the agents' information.

The company is constructing a manufacturing execution system (MES) system, In MES system all detection agents can share the product's information and track the product state. Therefore, it requests an intelligent way to store the data in the database rather than manually recording. A smart platform needs to be developed, aims to collect the data from the machine with the technical sensors, update the product state after the multi detection agents process, and record the product analysis report.

The purpose of platform development is to build a complete and reliable dataset, analyze the relationship between the production process and product quality, in order to achieve product management and predictive maintenance.

3.1 Protocol introduction

Since the manufactory has technique to obtain the production process information from sensors, the thesis focuses on the detection results aggregating, to mark the product with the quality label in the dataset.

MQTT

MQTT is a light publish/subscribe network protocol, It is designed for connections with remote locations with small code capacity, or the network

bandwidth is limited. The MQTT protocol defines two types of network entities: a message broker and several clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client is any device that runs an MQTT library and connects to an MQTT broker over a network. (Figure 3.1)

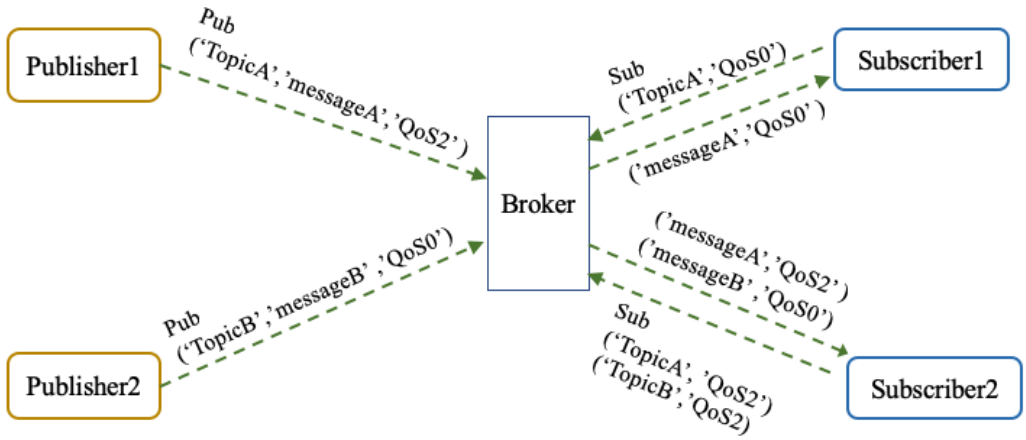


Figure 3.1: MQTT message delivering

In MQTT, information is organized with topics. When a publisher needs to publish the message, it sends a message with the topic to the broker. The broker then distributes the information to any clients who subscribe to the same topic. The publisher and subscriber do not need to be configured with each other. Using the topic, any publisher can publish to any subscriber with the same broker.

Publish/subscribe is an asynchronous communication paradigm. It allows the development of loosely-coupled event-based systems. It removes the dependencies between producer and consumer of information.

Publish also can define Quality of Service (QoS). The QoS defines how hard the broker/client will try to ensure that a message is received. Higher levels of QoS are more reliable but involve higher latency. QoS of Message related to the subscription, moreover it depends on the published message, meaning lower subscripts QoS must have lower QoS message however higher subscripts QoS might receive lower QoS message.

CherryPy

A web service is a service offered by an electronic device to another electronic device communicating with each other via the World Wide Web.

CherryPy is an object-oriented web application framework, it can act as a web server. REST (Representational State Transfer) is an architectural style that is well-suited to implementation in CherryPy. Both REST and CherryPy heavily leverage the HTTP protocol but otherwise carry minimal requisite overhead. In CherryPy REST is defined by four interface constraints:

- GET: retrieves the state of a specific resource.
- PUT: creates or replaces the state of a specific resource.
- POST: passes information to a resource to use at it sees fit.
- DELETE: removes resources.

Based on Rest architectural and CherryPy framework, we can build our own web server,deliver information to other users.

3.2 Database creation

Database schema and content

The database is a collection of organized, shareable, and unified management of large amounts of data stored in the computer for a long time. Moreover, the database is a data collection that is stored together in a certain way, can be shared with multiple users, has as little redundancy as possible, and is independent of the application. It can be regarded as an electronic file cabinet place for storing electronic files. Users can add, query, update, delete and other operations on the data in the file

In the MES system, since the contents have not been confirmed, we create an initial database, the database schema (Figure 3.2) contains the following contents:

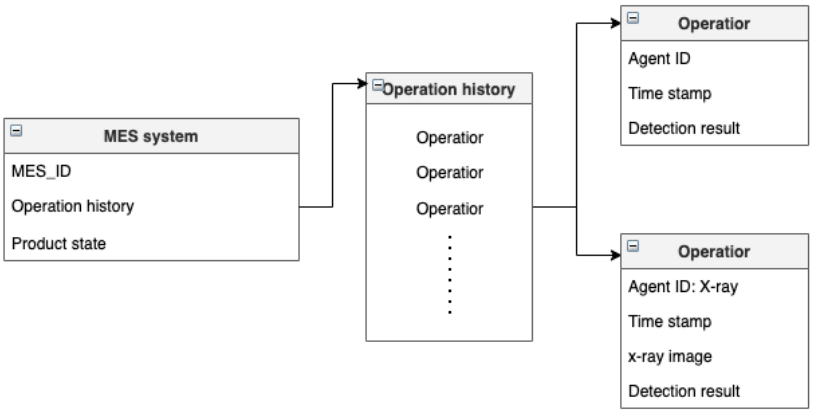


Figure 3.2: Database schema

- MES ID: the unique product's identity, stored in DataMatrix code and printed on the product.
- Operation history: an array to record different agency operations. In the array, each object records an agent's ID, detection time and detection result. Particularly, the x-ray agency also uploads the x-ray detection image.
- Product state: represent the product's current state, it depends on the previous state and the last agency detection result.

Our database simulates the MES system information update, moreover, the database has the schema to verify the information validation. For instance, the database accept the specific agents' ID, the detection result and product state must be "ACCEPT","REJECT" or "NOTSURE", other results will be reject in database.

3.3 Platform development

Platform structure

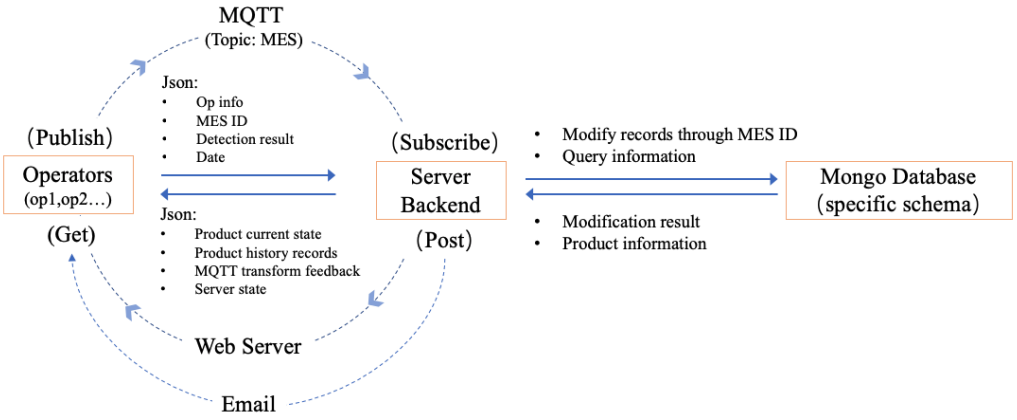


Figure 3.3: MES detection agencies platform

The MES platform uses MQTT protocol and HTTP protocol to achieve information delivering between detection agencies; MES central server and database. MQTT adoption is used to solve the bandwidth limitation for the detection machine, while HTTP protocol is widely used in information transform through a web server.

Since the MES process is constructing, the components are not finalized yet, the thesis develops a platform demo to simulate the MES process. The platform demo contains **some agency operators; a central server and a database**(Figure 3.3). As publishers in MQTT protocol, all agencies achieve one-way communication with the central server using the common topic, this single topic simplifies the central server's MQTT configuration. Moreover the QoS level of MQTT is "level 2", meaning the broker/client will deliver the message exactly once by using a four steps handshake. In the platform, central server likes an information aggregator and manager, has the authority to connect with the Mongo database, query and modify the objects, complete the database, moreover obtain the feedback from the database. A part of feedback information can be delivered from the central server to agencies with a web server. Web server posts the information, and agencies get the information using different URLs. Moreover, the central server could send some notice and news to the specific agency by email. In the platform, detection agencies receive the product's current state which belongs to feedback information from the database, but without central server management, they cannot communicate with the database directly.

Main function introduction

Using the platform, agencies can cooperate with each other, obtain the product current state and history records.The product's history information, for instance, the disqualified reason could help agents to make the decision whether to detect this product.

Some main functions that the platform can achieve are:

1. Current product state updating:
   The current product state depends on the previous state and current detection result. According to the MES-ID, the platform uses the "MES detection algorithm" to update the current product state until the last agency (Algorithm 5;6;7 in Annex).

2. Querying from database and uploading to the database:
   Scanning the DataMatrix code, the agencies could obtain the current product state, and the previous operator's detection result. Moreover, after detection, upload the detection information to the database, in order to update the product's current state.

3. Critical condition alarm:
   Supposing the central server has a problem, for instance, shutdown or disconnected with the mongo database. In that case, server will automatically send an email to every detection and inform the agencies to break off the detection process.

Chapter 4

Conclusion

4.1 Tasks and achievements

In order to achieve I4.0 product quality management and predictive maintenance. The thesis proposes making a reliable database which contains the product quality state, production information, and unqualified reason. With the completely reliable dataset, we can achieve predictive model creation by supervised machine learning methods and summarize the relation between the quality and production process to improve the production technique.

A complete dataset creation needs a smart platform to collect the information from the production process and different quality detection agents. The collected product information should be stored in the database. According to product identity, it is easy to make the features and labels correspond with each other. However, the DataMatrix code which stores the product's identity cannot be decoded, since it lost the modules, only reserve the module's contour.

Thus, the principal task to achieve a smart platform is to recover the DataMatrix code, to obtain the encoded contents which record the product identity. I worked in the following tasks to recover the shot-blasted DataMatrix code:

1) Understand the DataMatrix code structure, components and attributes.

2) Understand the algorithm, analysis flow and performance measurement in previous literature.

3) Recover the shot-blasted DataMatrix code and create a baseline.
   The recovery process contains two stages, DataMatrix code positioning and DataMatrix code recovering. For the positioning stage, I create two

positioning algorithms to satisfy different requests in factory or the mobile phone application. For the recovering stage, two methods compensate each other, transforming the rough contour DataMatrix code to regular shapes. Moreover, propose an idea to use the SPP neural network to recover the DataMatrix code cell by cell, avoiding the uncertain image dimension.

Since this kind of DataMatrix code recovery is not mentioned in any other literature, it is a new raised project and a baseline needs to be created for the new project.

4) Using calculation time to measure the positioning performance, while using the number of mispredicted cells and a baseline to measure the final recovering performance.

After recovery the DataMatrix code, the thesis develops a platform demo to simulate the MES process and creates an initial database to store the information. Platform development requests us to understand the following fields:

1) MES system operation logic and requests.

2) Internet techniques and protocols, which are suitable for MES processing.

3) Database structure and validation.

With the tasks above, an initial MES platform has been built, the platform contains functions from product recognition to products' quality detection result upload. However, this platform construction is a long-term project, the algorithm and structure should be improved according to the requests.

4.2 Question and answer

As for the questions which we put forward at the beginning:

a) **How can we decode shot-blasted DataMatrix code with computer vision?**
Ans: DataMatrix code decoding requires the code have a regular format. We recover the "Rough Contour DataMatrix code" to traditional type by positioning the DataMatrix code location and recovering the separated cells.

b) **Which recovery techniques perform better and are better suitable for a real case scenario?**
Ans: Different scenarios have different requests. We have two different **positioning algorithms** adapt different scenarios. i) Manufacturer accepts more seconds to process the image, but it requests to scan the whole image. In this case we adopt the "auto positioning process" to locate the DataMatrix code automatically in the image.

ii) Mobile phone application, which needs a quickly positioning process with small area scanning, should adopt the "manually positioning process" with the predefined scanning region. However, these two positioning processes have the similar positioning result and sliding matching accuracy.

Moreover, to ensure the decoded success rate and response time, after positioning algorithm two **recovering methods** are in series, and the recovering threshold should be iterated, once decoded successfully the program should break.

c) **How can we create a traceability system? What's the structure of the platform and Database?**
Ans: The traceability system requests to build a database, and according to the product identity to update the records in the Database. The platform contains some agent operators, a central server and a database. Agents are in charge of uploading the detection report, the central server is in charge of modifying the records in the database and sending database feedback information to agents.

d) **What functions should the platform achieve?**
Ans: the main functions are: Current product state updating; Querying from database and uploading to the database; Critical condition alarm.

4.3 Future tasks suggestion and expectation

Since the thesis proposes some suggestions about I4.0 platform construction and DataMatrix code recovery, future tasks can be divided into three stages:

Stage 1 DataMatrix code recovery program:
Waiting for more samples to verify the DataMatrix code recovery algorithm, according to the recovery performance, improve the algorithm to achieve a better recovery result.

Stage 2 Platform improvement:
This stage parallel with the first stage, improves and updates the platform function to satisfy the MES new requests. Achieve a complete platform which can process and analyze the data, while can deal with different critical situations.

Stage 3 Dataset analyzation:
A realiable dataset is an excellent beginning to analyze the relationship between the production process and product quality performance. With supervised machine learning, we can give the company an analysis report, help the company understand the influence of production factors on product quality. Moreover, create a prediction model to predict the product quality in the production simulation process. A useful prediction model also can be integrated into the platform to verify the detection agents' quality detection results.

With intelligent platforms and intelligent predictive maintenance, the company can achieve product optimization and upgrades. Product information can be stored and fully utilized in a smart way. The production and detection system improves work efficiency significantly. Every improvement is useful to smart manufacturing and smart society.

Reference

[1] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

[2] V. Alcácer and V. Cruz-Machado. Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems, 2019.

[3] Lucas Santos Dalenogare, Guilherme Brittes Benitez, Néstor Fabián Ayala, and Alejandro Germán Frank. The expected contribution of Industry 4.0 technologies for industrial performance. *International Journal of Production Economics*, 204(July):383–394, 2018.

[4] Alejandro Germán Frank, Lucas Santos Dalenogare, and Néstor Fabián Ayala. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210(January):15–26, 2019.

[5] Manuel Oliveira, Emrah Arica, Marta Pinzone, Paola Fantini, and Marco Taisch. *Human-Centered Manufacturing Challenges Affecting European Industry 4.0 Enabling Technologies*, volume 1. Springer International Publishing, 2019.

[6] Ray Y. Zhong, Xun Xu, Eberhard Klotz, and Stephen T. Newman. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering*, 3(5):616–630, 2017.

[7] Tatiana A. Ryabchik, Elvira E. Smirnova, Marina I. Lukashova, and Hasan Haydar. Manufacturing processes quality control as a main factor of performance enhancement in industrial management. *Proceedings of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2019*, pages 1463–1466, 2019.

[8] Matheus E. Leusin, Mirko Kück, Enzo M. Frazzon, Mauricio U. Maldonado, and Michael Freitag. Potential of a Multi-Agent System Approach for Production Control in Smart Factories. *IFAC-PapersOnLine*, 2018.

[9] Marc Benhaim. GS1 DataMatrix Guideline. page 76, 2016.

[10] GS1. GS1 DataMatrix Guideline Overview and technical introduction to the use of GS1 DataMatrix. pages 8–12, 2018.

[11] Ladislav Karrach and Elena Pivarčiová. Options to Use Data Matrix Codes in Production Engineering. *Management Systems in Production Engineering*, 26(4):231–236, 2018.

[12] Feng Liu, Anan Liu, Meng Wang, and Zhaoxuan Yang. Robust and fast localization algorithm for data matrix barcode. *Proceedings - 2010 International Conference on Optoelectronics and Image Processing, ICOIP 2010*, 2:356–359, 2010.

[13] Qiang Huang, Wen Sheng Chen, Xiao Yan Huang, and Ying Ying Zhu. Data matrix code location based on finder pattern detection and bar code border fitting. *Mathematical Problems in Engineering*, 2012, 2012.

[14] Yange Dai, Lizhen Liu, Wei Song, Chao Du, and Xinlei Zhao. The realization of identification method for DataMatrix code. *Proceedings of 2017 International Conference on Progress in Informatics and Computing, PIC 2017*, pages 410–414, 2017.

[15] Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60(January):97–116, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8691 LNCS(PART 3):346–361, 2014.

Appendices

ASCII conversion

Table 1: ASCII conversion

Decimal-Binary-Hexadecimal Conversion Chart

This chart shows all of the combinations of decimal, binary and hexadecimal from 0 to 255 decimal. When making a change in a CV this chart will show the conversion for different numbering systems. Some decoders split the CV into two parts. When you modify a CV you need to write back all 8 bits. This chart will help determine the correct bit value a CV.

Decimal	Binary	Hex	Decimal	Binary	Hex	Decimal	Binary	Hex	Decimal	Binary	Hex
Bit No.>	76543210			76543210			76543210			76543210	
0	00000000	0	64	01000000	40	128	10000000	80	192	11000000	C0
1	00000001	1	65	01000001	41	129	10000001	81	193	11000001	C1
2	00000010	2	66	01000010	42	130	10000010	82	194	11000010	C2
3	00000011	3	67	01000011	43	131	10000011	83	195	11000011	C3
4	00000100	4	68	01000100	44	132	10000100	84	196	11000100	C4
5	00000101	5	69	01000101	45	133	10000101	85	197	11000101	C5
6	00000110	6	70	01000110	46	134	10000110	86	198	11000110	C6
7	00000111	7	71	01000111	47	135	10000111	87	199	11000111	C7
8	00001000	8	72	01001000	48	136	10001000	88	200	11001000	C8
9	00001001	9	73	01001001	49	137	10001001	89	201	11001001	C9
10	00001010	A	74	01001010	4A	138	10001010	8A	202	11001010	CA
11	00001011	B	75	01001011	4B	139	10001011	8B	203	11001011	CB
12	00001100	C	76	01001100	4C	140	10001100	8C	204	11001100	CC
13	00001101	D	77	01001101	4D	141	10001101	8D	205	11001101	CD
14	00001110	E	78	01001110	4E	142	10001110	8E	206	11001110	CE
15	00001111	F	79	01001111	4F	143	10001111	8F	207	11001111	CF
16	00010000	10	80	01010000	50	144	10010000	90	208	11010000	D0
17	00010001	11	81	01010001	51	145	10010001	91	209	11010001	D1
18	00010010	12	82	01010010	52	146	10010010	92	210	11010010	D2
19	00010011	13	83	01010011	53	147	10010011	93	211	11010011	D3
20	00010100	14	84	01010100	54	148	10010100	94	212	11010100	D4
21	00010101	15	85	01010101	55	149	10010101	95	213	11010101	D5
22	00010110	16	86	01010110	56	150	10010110	96	214	11010110	D6
23	00010111	17	87	01010111	57	151	10010111	97	215	11010111	D7
24	00011000	18	88	01011000	58	152	10011000	98	216	11011000	D8
25	00011001	19	89	01011001	59	153	10011001	99	217	11011001	D9
26	00011010	1A	90	01011010	5A	154	10011010	9A	218	11011010	DA
27	00011011	1B	91	01011011	5B	155	10011011	9B	219	11011011	DB
28	00011100	1C	92	01011100	5C	156	10011100	9C	220	11011100	DC
29	00011101	1D	93	01011101	5D	157	10011101	9D	221	11011101	DD
30	00011110	1E	94	01011110	5E	158	10011110	9E	222	11011110	DE
31	00011111	1F	95	01011111	5F	159	10011111	9F	223	11011111	DF
32	00100000	20	96	01100000	60	160	10100000	A0	224	11100000	E0
33	00100001	21	97	01100001	61	161	10100001	A1	225	11100001	E1
34	00100010	22	98	01100010	62	162	10100010	A2	226	11100010	E2
35	00100011	23	99	01100011	63	163	10100011	A3	227	11100011	E3
36	00100100	24	100	01100100	64	164	10100100	A4	228	11100100	E4
37	00100101	25	101	01100101	65	165	10100101	A5	229	11100101	E5
38	00100110	26	102	01100110	66	166	10100110	A6	230	11100110	E6
39	00100111	27	103	01100111	67	167	10100111	A7	231	11100111	E7
40	00101000	28	104	01101000	68	168	10101000	A8	232	11101000	E8
41	00101001	29	105	01101001	69	169	10101001	A9	233	11101001	E9
42	00101010	2A	106	01101010	6A	170	10101010	AA	234	11101010	EA
43	00101011	2B	107	01101011	6B	171	10101011	AB	235	11101011	EB
44	00101100	2C	108	01101100	6C	172	10101100	AC	236	11101100	EC
45	00101101	2D	109	01101101	6D	173	10101101	AD	237	11101101	ED
46	00101110	2E	110	01101110	6E	174	10101110	AE	238	11101110	EE
47	00101111	2F	111	01101111	6F	175	10101111	AF	239	11101111	EF
48	00110000	30	112	01110000	70	176	10110000	B0	240	11110000	F0
49	00110001	31	113	01110001	71	177	10110001	B1	241	11110001	F1
50	00110010	32	114	01110010	72	178	10110010	B2	242	11110010	F2
51	00110011	33	115	01110011	73	179	10110011	B3	243	11110011	F3
52	00110100	34	116	01110100	74	180	10110100	B4	244	11110100	F4
53	00110101	35	117	01110101	75	181	10110101	B5	245	11110101	F5
54	00110110	36	118	01110110	76	182	10110110	B6	246	11110110	F6
55	00110111	37	119	01110111	77	183	10110111	B7	247	11110111	F7
56	00111000	38	120	01111000	78	184	10111000	B8	248	11111000	F8
57	00111001	39	121	01111001	79	185	10111001	B9	249	11111001	F9
58	00111010	3A	122	01111010	7A	186	10111010	BA	250	11111010	FA
59	00111011	3B	123	01111011	7B	187	10111011	BB	251	11111011	FB
60	00111100	3C	124	01111100	7C	188	10111100	BC	252	11111100	FC
61	00111101	3D	125	01111101	7D	189	10111101	BD	253	11111101	FD
62	00111110	3E	126	01111110	7E	190	10111110	BE	254	11111110	FE
63	00111111	3F	127	01111111	7F	191	10111111	BF	255	11111111	FF

Binary Number System for one byte

```
Bit Number| 7| 6| 5| 4|3|2|1|0|
Bit Weight|128|64|32|16|8|4|2|1|
```

Some Commonly used CVs

CV-1 Short Address
CV-2 Start Voltage
CV-3 Acceleration Rate
CV-4 Deceleration Rate
CV-5 Maximum Voltage
CV-6 Mid Point Voltage
CV-7 Ver Number
CV-8 Maker ID
CV-17/18 Long Address
CV-19 Consist Address

CV-21 Advance Consist function control
CV-22 Advance Consist headlight control
CV-23 Advance Consist acceleration rate
CV-24 Advance Consist deceleration rate

SEE YOUR DECODER MANUAL FOR ALL OF THE CVs IT USES AND THE RANGE OF VALUES.

CV-29 Configuration Register
Bit 0= Direction of travel
Bit 1=Speed step 28
Bit 2=d.c. enable
Bit 3= Advance acknowledgment
Bit 4= Alternate speed table
Bit 5= Long address.

CV-66 Forward Trim
CV-67 to 94 Speed Table
CV-95 Reverse Trim

DCF 25April02

DataMatrix attributes

Table 2: DataMatrix attributes

Symbol size*		Data region		Mapping matrix size	Total codewords		Maximum data capacity		% of Codewords used for error correction no.	Max. correctable codewords error/erasure
							Num	Alpha - num.		
Row	Col	Size	No		Data	Error	Cap.	Cap.		
10	10	8x8	1	8x8	3	5	6	3	62.5	2/0
12	12	10x10	1	10x10	5	7	10	6	58.3	3/0
14	14	12x12	1	12x12	8	10	16	10	55.6	5/7
16	16	14x14	1	14x14	12	12	24	16	50	6/9
18	18	16x16	1	16x16	18	14	36	25	43.8	7/11
20	20	18x18	1	18x18	22	18	44	31	45	9/15
22	22	20x20	1	20x20	30	20	60	43	40	10/17
24	24	22x22	1	22x22	36	24	72	52	40	12/21
26	26	24x24	1	24x24	44	28	88	64	38.9	14/25
32	32	14x14	4	28x28	62	36	124	91	36.7	18/33
36	36	16x16	4	32x32	86	42	172	127	32.8	21/39
40	40	18x18	4	36x36	114	48	228	169	29.6	24/45
44	44	20x20	4	40x40	144	56	288	214	28	28/53
48	48	22x22	4	44x44	174	68	348	259	28.1	34/65
52	52	24x24	4	48x48	204	84	408	304	29.2	42/78
64	64	14x14	16	56x56	280	112	560	418	28.6	56/106
72	72	16x16	16	64x64	368	144	736	550	28.1	72/132
80	80	18x18	16	72x72	456	192	912	682	29.6	96/180
88	88	20x20	16	80x80	576	224	1152	862	28	112/212
96	96	22x22	16	88x88	696	272	1392	1042	28.1	136/260
104	104	24x24	16	96x96	816	336	1632	1222	29.2	168/318
120	120	18x18	36	108x108	1050	408	2100	1573	28	204/390
132	132	20x20	36	120x120	1304	496	2608	1954	27.6	248/472
144	144	22x22	36	132x132	1558	620	3116	2335	28.5	310/590

MES processing algorithm

Algorithm 5 Current state updating algorithm

if previous state == *"NOTSURE"* & detection result == *"ACCEPT"* **then**
  return product current state=*"NOTSURE"*
else if previous state == *"NOTSURE"* & detection result == *"REJECT"* **then**
  return product current state=*"REJECT"*
else if previous state == *"REJECT"* **then**
  operator do not detect, do not upload message
  return current state do not update
else
  return current state do not update
end if

Algorithm 6 Final state updating algorithm

if previous state == *"NOTSURE"* & final detection result == *"ACCEPT"* **then**
  return product final state=*"ACCEPT"*
else if previous state == *"NOTSURE"* & final detection result == *"REJECT"* **then**
  return product final state=*"REJECT"*
else if previous state == *"REJECT"* **then**
  operator do not detect, do not upload message
  return current state do not update
else
  return current state do not update
end if

Algorithm 7 MES detection algorithm

Input:
1.MES detection sequence
2.Detection agent(op) output information
3.Current state update algorithm
4.Final state update algorithm

Output:
Product final state

if op is the first in the sequence **then**
  create new MES-ID object or update existing MES-ID records
  if op detection result:*"ACCEPT"* **then**
    return product current state=*"NOTSURE"*
  else if op detection result:*"REJECT"* **then**
    return product current state=*"REJECT"*
  end if
end if
if op is not the first or last in the sequence **then**
  if previous op uploaded **then**
    current state updating algorithm(Algorithm 5)
    upload product info
    return current state update
  else
    operation do not detect, do not upload message
    return current state do not update
  end if
end if
if op is the last in the sequence **then**
  if previous op uploaded **then**
    final state updating algorithm(Algorithm 6)
    upload product info
    return final state
  else
    operation do not detect, do not upload message
    return current state do not update
  end if
end if
if op is the x-ray operator **then**
  accept both *"ACCEPT"* and *"REJECT"*
  upload product info include x-ray image
end if