**REPORT ON DATA MATRIX DETECTION.**

**Abstract:**

The 2-D bar code can hold large amounts of data with in a very tiny region, it also has strong ability for error correction and high safety. This allows it to be used extensively for different applications. This report presents a model that could locate a data matrix code based on its finder pattern and decode the information it contains. Data Matrix codes are printed in black on a transparent plastic roll and contain the corresponding location information and have a constant size of about ~ (200\*200) pixels. The model is then evaluated on a test dataset that contains 800 images. The metrics are discussed to arrive at the reasons of failing.

**Task Objective:**

A plastic roll (web) is fed into a roll-to-roll laser machine, it sometimes comes with defective regions. When there is a defective region in the web, the location information for the defective region will already be supplied. Manual identification of defective regions is tedious and quite error prone. Data matrices are printed on the web at regular intervals and contain the location information of the web.

Main objective for the task is to scan the data matrix codes and learn about the location information, so when the web is fed into the roll-to-roll laser system it is possible to identify the defective regions and lasering the web could be prevented. Therefore, additional machine features like monitoring and dynamic processing could be added.

**Introduction:**

Data Matrix is a two-dimensional code consisting of black and white dots or cells arranged in either a square or rectangular pattern. It is composed of two solid adjacent borders in an “L” shape (known as Finder Pattern as shown in the following [figure](https://www.researchgate.net/publication/258384910/figure/fig10/AS:669585825091584@1536653109265/Data-matrix-code-symbol_W640.jpg):1.0) and two other borders consisting of alternate dark and light cells (known as timing pattern). The finder pattern helps in locating the data matrix and find its orientation and the timing pattern provides a count of the number of rows and columns in the matrix.

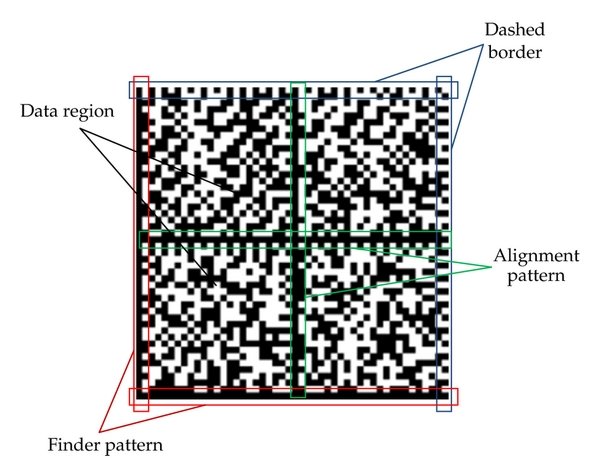


Fig: 1.0 Data Matrix

**Literature Review:**

Qiang Huang et al. in [1] presented the Finder Pattern detection method mainly based on the line segment detection (‘L’) when it is clearly distinguishable and continuous. However, it is challenging to find this approach helpful to detect matrix codes that does not have a continuous line on its finder pattern. Chutatape, O. et al. in [2] discussed about using Hough lines to detect the Horizontal and vertical lines in an image and there by finding about the matrix code within the image. However, it is computationally expensive. Dai, Y.G., Liu et al. in [3] discusses the detection of highly compact sized data matrix code on a metal surface using a digital image processing technique where binarization is achieved by improving the traditional Otsu algorithm. They used the Hough transformation to detect the four vertices of the Data Matrix code, which takes it long to recognize the matrix code. D.K. Hansen et al. [4] demonstrated a method to detect the data matrix by training a deep learning neural network. They described a way to adopt the state-of-art deep learning-based detector of YOLO (You Only Look Once) for the purpose of detecting barcodes in a fast and reliable way. However, training a deep learning neural network would require a lot of data and the accuracy provided in the paper is not so convincing to choose this method.

**1. CODE DEVELOPMENT:**

**Preprocessing the image:**

Development of code is centered on detecting the “L” shaped solid adjacent borders (finder pattern of data matrix). The image is run through several image processing steps to achieve this. We start with gray scaling the image, then we pass a 3\*3 kernel on the image to detect the horizontal lines that gives us gradX (fig 1.2, where only horizontal lines could be seen). A transpose of this kernel is passed over the grayscale image to detect the vertical lines and it gives out gradY (fig 1.3).

A picture containing text, dark, night sky

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Fig: 1.1 Original image Fig: 1.2 gradX Fig: 1.3 gradY

**Erosion and Dilation:**

A gradient image (fig 1.4) is then obtained by subtracting both horizontal and vertical outputs. (gradX and gradY respectively). This image is then converted into black and white by applying Otsu thresholding technique as shown in fig: 1.5. The thresholded image is then eroded and dilated in various sequence of steps to give a closed image(Fig:1.6). Contours are found in the closed image. The height and width of the matrix code is already a known variable which is about 200 pixels. Therefore, the detected contours which obey this size +/- 30 pixels are selected.

A picture containing application

Description automatically generatedA picture containing text, dark, night sky

Description automatically generatedA picture containing application

Description automatically generated

Fig: 1.4 Gradient Image Fig: 1.5 Thresholded Gradient Fig: 1.6 Closed image

**Decoding the Region of Interest:**

After finding the required contour, bounding rectangle for the contour is selected and a border of about 50 pixels is added on its each side to prevent any loss of matrix code. This gives us a new bounding box. Using the new bounding box coordinates we slice the original image to obtain region of interest (ROI) as shown in fig: 1.7. ROI which is a sliced original 3-dimensional image. It is then thresholded, following a erosion and dilation to make the separated dots connected into a continuous segment, we obtain closed\_ROI and pass it into the decoding function Pylibdmtx. The detected data matrix code is displayed on the console.

A picture containing text

Description automatically generatedQr code

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Fig:1.7 Detected contour Fig: 1.8 ROI Fig: 1.9 Closed\_ROI

**2. Metrics:**

803 sample images are collected and fed into the model for which the following data is observed.

Total number of sample images: 803

Number of files where data matrix is detected and decoded successfully: 734

Number of files where data matrix is not detected is: 69

Number of files where data is not present: 2

true positives = 734 (85.95%)

false positives = 0 (0%)

true negatives = 51 (5.97%)

false negatives = 69 (8.08%)

Chart, treemap chart

Description automatically generated

Chart, bar chart

Description automatically generated

Fig:2.1 Bar Plot Fig:2.2 Heat map of Confusion Matrix

From the Confusion matrix,

* Out of 804 images, the (734) or ~86% of the instances represents the successful detection and decoding of the data matrix codes.
* Out of remaining 120 instances, 51 represents inaccurate detections like QR-codes, or contours that doesn’t contain Matrix codes.
* 69 instances are accurate matrix code detection, but they could not be decoded for some reasons like code distortion, additional noise over or around the matrix code.

The images contain QR-code along with the data matrix code with a similar size, and these infer the detection as QR-codes also contain the ‘L’ shape with in them. However, as far as data matrix code is detected and decoded in an image QR-code does not hold any threat as they don’t return anything when passed into data matrix decoder (Pylibdmtx). The major issue of concern is he remaining 69 instances where data matrix is detected but could not be decoded.

**3.Evaluation:**

Successful detection and decoding of the matric code could be observed when the code is clearly visible, have little to no noise, little to no distortion, and the code is not destroyed. The following images from fig: 3.1 to fig:3.3 are some examples where there is a successful extraction of the code.

A picture containing text, indoor

Description automatically generated

Fig:3.1 Fig:3.2 Fig:3.3

The case of interest is where the matrix codes are detected but not decoded. This could happen for several reasons. The model could not detect codes from the following images

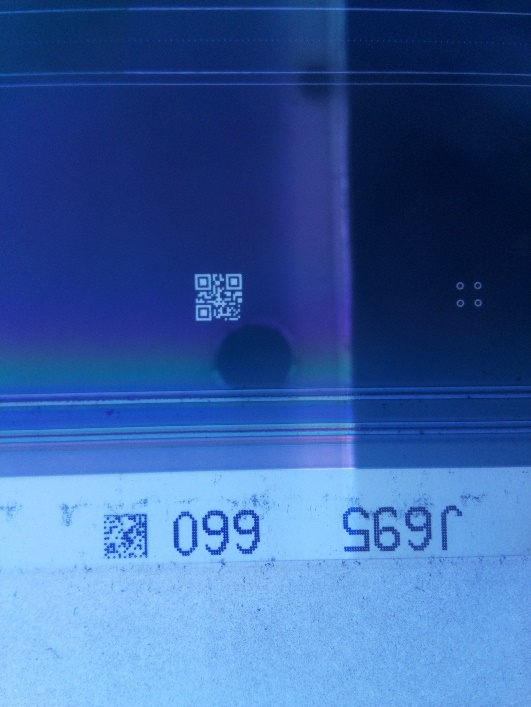
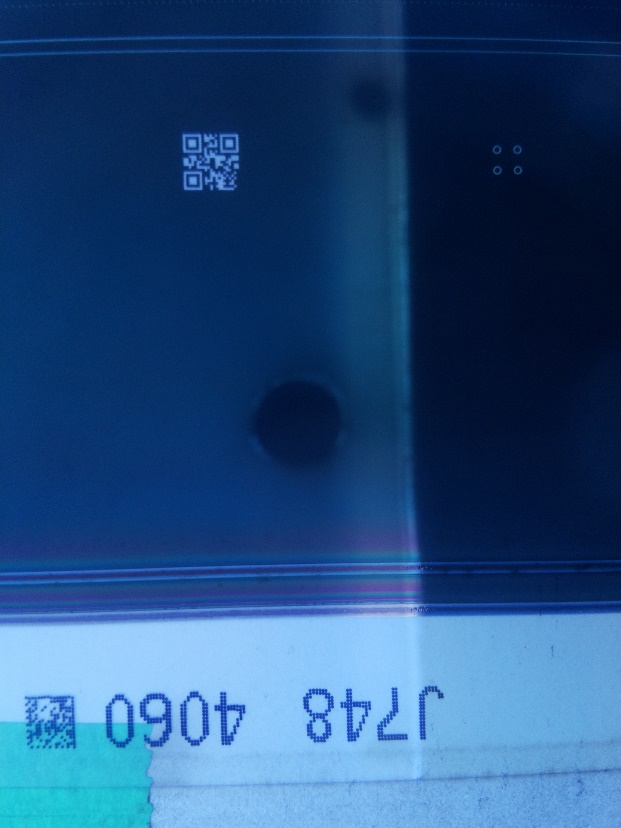


Fig:3.4 Fig:3.5 Fig:3.6

**4.Analysis**

**Reasons for non-detection:**

Among several reasons for a matrix code to not be detected, the major problems are

1. with the noise around or over the matrix code like the tape
2. Not properly printed/broken matrix code.
3. Distortion in the matrix code
4. Varied Lighting conditions

The following images (fig:4.1 and fig:4.2) demonstrates the cases where the model finds it difficult to decode the matrix code.

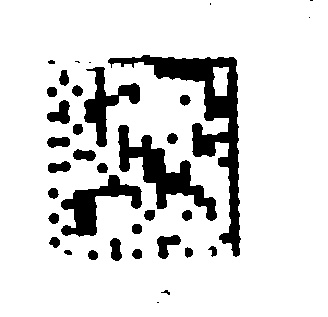
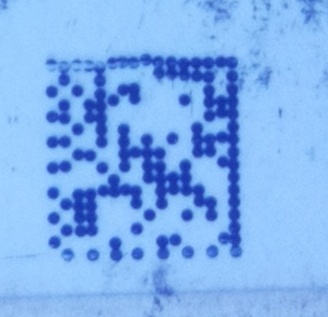
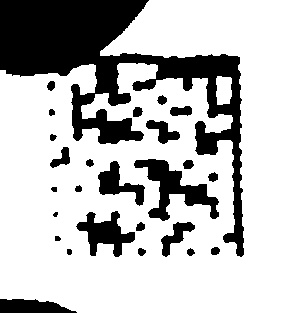
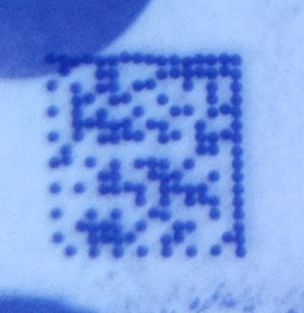
Images where the Matrix codes are not detected by the model:

Fig: 4.1 Tape on code Fig: 4.2 Not properly printed matrix code

**5.Further improvements:**

There is still a large scope for improvement.

1. One of the major things is improving the thresholding techniques to extract the code alone removing any extra noise to be able to use in any lighting conditions.
2. Improving the data matrix detection by refining the image preprocessing techniques.

**6.References:**

1. Huang, Q.; Chen, W.S.; Huang, X.Y.; Zhu, Y.Y. Data matrix code location based on finder pattern detection and bar code border fitting. Math. Probl. Eng. 2012, 2012, 515296, doi:10.1155/2012/515296.
2. Chutatape, O., Guo, L.F.: A modified Hough transform for line detection and its performance. J. Pattern Recognit.32, 181–192 (1999).

<https://www.researchgate.net/deref/https%3A%2F%2Fdoi.org%2F10.1016%2FS0031-3203(98)00140-X>

1. Dai, Y.G.,Liu,L.Z.,Song,W.,Chao,D.,Zhao,X.L.: The realization of identification method for Data Matrix code. In: 2017 International Conference on Progress in Informatics and Computing,pp.410–414.IEEEPress,Nanjing(2017). <https://www.researchgate.net/deref/https%3A%2F%2Fdoi.org%2F10.1109%2FPIC.2017.8359582>
2. D.K. Hansen and K. Nasrollahi. “Real-Time Barcode Detection and Classification Using Deep Learning”, in Proc. of 9th International Joint Conference on Computational Intelligence,vol.1,2017,pp.321-327

<https://vbn.aau.dk/en/publications/real-time-barcode-detection-and-classification-using-deep-learnin>