**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):1.1.0) 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.

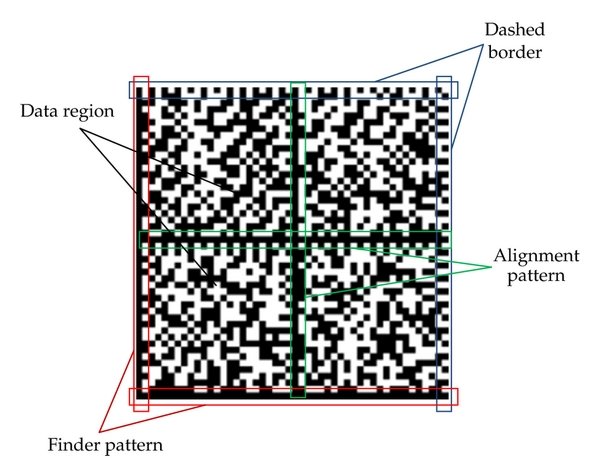


Fig: 1.1.0 Data Matrix

**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. Ladislav Karrach et al. in [5] compared the impact of various cameras on detection and decoding of data matrix codes in real scene images by locating the finder pattern using adaptive thresholding techniques and connecting neighboring points to continuous regions. They proceeded onto finding the right angles isosceles triangle (that has the ‘finder patter’ as equal sides) which is not ideal in this case as the finder pattern sometimes is distorted. Ion-Cosmin Dita et al.[6] provided a method to localize the modules of Industrial Data Matrix Code (IMDC) marked on curved surfaces, where an imaginary grid that has same orientation of the IMDC is projected onto the image and accordingly the grid modules of the code are scanned. However, if the surface is not a sphere the results are not accurate with this method.

**1. Code Development:**

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. First, we start with reading the image.



Next, the image is changed into grayscale to reduce the number of channels. A 3x3 kernel as shown in the code below is passed on to the gray image. This gives us gradX (figure:1.1.2) where only horizontal lines are detected. Transpose of this kernel is passed again over the grayscale image. This gives us gradY (figure:1.1.4), where only vertical lines are seen. Gradient image (Fig:1.1.4) is obtained as we subtract gradY from gradX. This helps us in obtaining the objects that contain both horizontal and vertical lines in them and get rid off the rest of the objects that are circular, rhomboid etc.



Graphical user interface, text

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

Fig: 1.1.2 gradX

A picture containing text, dark, night sky

Description automatically generated

Fig: 1.1.3 gradY

Fig: 1.1.1 original Image

**Closing operation:**

Thresholded gradient (figure 1.1.5) image is obtained after applying Otsu thresholding on Gradient image (figure 1.1.4) to convert it into black and white. Closing operation is performed on the thresholded image to cover the small gaps. Now, erosion and dilation are applied on the closed image to reduce the gaps between the dots in the matrix code. The data matrix contour is

prominently visible in final closed image (figure 1.1.6).



A picture containing application

Description automatically generatedA picture containing text, dark, night sky

Description automatically generatedA picture containing application

Description automatically generated

Figure 1.1.6 Closed Image

Figure 1.1.5 Closed Image

Figure 1.1.4 Gradient Image

**Finding Contours and selecting Region of Interest:**

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+/- 50 pixels are selected. After selecting the required contour, bounding rectangle for the contour is considered. To prevent any loss of code, an additional border of about 50 pixels is added on each edge of the rectangle, which 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: 1.1.8. ROI is a sliced original 3-dimensional image. It is then converted into grayscale, then it is thresholded using OTSU thresholding technique, It is then eroded and dilated to connect the separate dots in the threshold into continuous lines and we obtain closed ROI as shown in figure 1.1.9.



Finally, we pass the closed image into pylibdmtx decode function to obtain scan the data matrix and display the code on the console.

A picture containing text

Description automatically generatedQr code

Description automatically generated

Figure 1.1.8 ROI

Graphical user interface

Description automatically generated with medium confidence

Figure 1.1.7 Detected Contour

Figure 1.1.9 Closed ROI

**2. Metrics:**

To evaluate the performance of the model, a sample test data that contains 803 images is taken.

The model is run on the test data, for which the following results are 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: 65

Number of files where data is not present: 4

True positives = 734 (85.95%)

False positives = 0 (0,00%)

True negatives = 51 (5.97%)

False negatives = 69 (8.08%)

Chart, bar chart

Description automatically generated

Figure 1.2.2 Heat Map for confusion matrix

Chart, treemap chart

Description automatically generated

Figure 1.2.1 Bar Plot

From the Confusion matrix,

* Out of 803 images, the (734) or ~86% of the instances represents True positives, the successful detection and decoding of the data matrix codes.
* Out of remaining 120 instances, 51 represents inaccurate detections like QR-codes, or other contours that doesn’t contain Matrix codes.
* 69 instances are matrix code detections, but they could not be decoded for various reasons like matrix code distortion or noise over or around the matrix code.

The images also contain similar sized QR-codes along with the data matrices, and these come in the way of detection as QR-codes also contain the ‘L’ shape with in them. QR-code does not return anything when passed into the decoding function Pylibdmtx. Therefore, as far as data matrix code is detected and decoded in an image QR-code does not impose any threat. However, the major issue of concern is when the data matrix is detected but not decoded.

**3.Evaluation:**

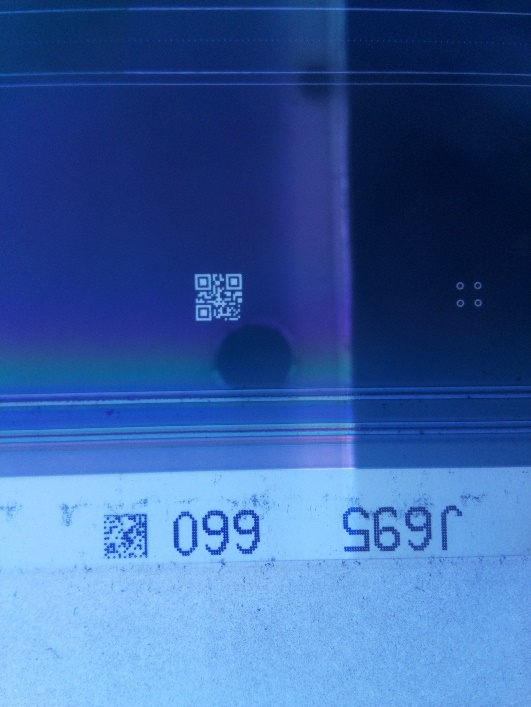
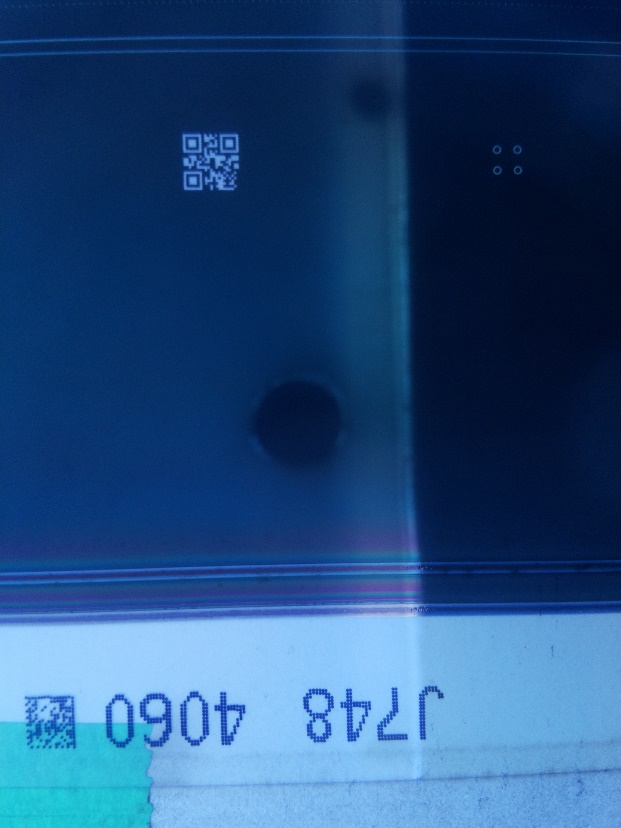
Successful detection and decoding of the matric code could be observed when the code is clearly visible, have little to no noise, little to no distortion, and the code is not destroyed. The following images from fig: 3.1 to fig:3.3 are some examples where there is a successful extraction of the code.

Figure 1.3.1 Successfully detected images



The case of interest is where the matrix codes are detected but not decoded. The following images show the instances where the code is not successfully extracted.

Figure 1.3.2 Unsuccessful detection



**Possible reasons for unsuccessful detection:**

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

1. Presence of noise around or over the matrix code like the tape, additional gray/black pixels
2. Not properly printed or a broken matrix code.
3. Distortion in the matrix code
4. Varied Lighting conditions

The following images (figure 1.3.3 and figure 1.3.4) demonstrates the cases where the model finds it difficult to decode the matrix code.

Figure 1.3.4 improperly printed matrix code

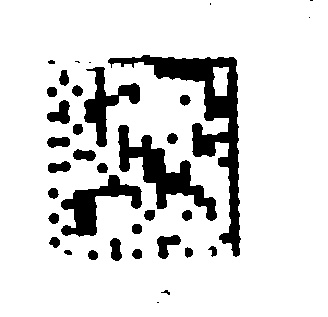
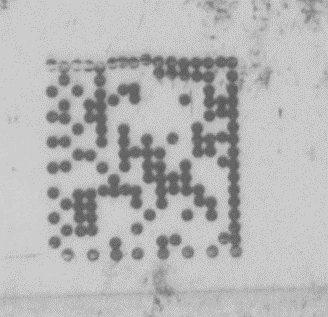
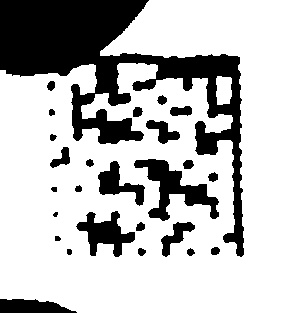
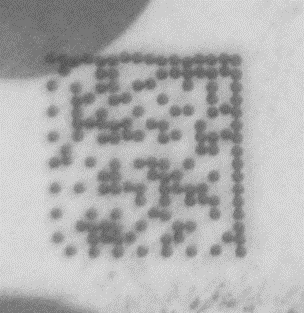


Figure 1.3.3 Tape on the matrix code



**Further improvements:**

A different approach for preprocessing the ROI made the instances where the code extraction was unsuccessful proved to be successful. However, it does not work very well with other images where the code was successfully detected.



There is still a large scope of improvement.

1. One of the major things is improving the thresholding techniques to extract the code 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.
3. There is a slight improvement when a different approach for preprocessing the ROI is followed.

**4.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-learning>

1. Ladislav Karrach, Elena Pivarciova, Y. Nikitin “Comparing the impact of different cameras and image resolution to recognize the data matrix codes.” <https://www.researchgate.net/publication/327863572_Comparing_the_impact_of_different_cameras_and_image_resolution_to_recognize_the_data_matrix_codes>
2. Ion-Cosmin Dita, Marius Otesteanu, Franz Quint, Nagaraju Annamdevula “A scanning method for industrial data matrix codes marked on spherical surfaces” <https://www.researchgate.net/publication/261860699_A_scanning_method_for_industrial_data_matrix_codes_marked_on_spherical_surfaces>