Utilizing Machine Learning, Perspective Correction, and String Matching for Detecting QRIS Counterfeits

Maulvi Ziadinda Maulana – 13522122[1]
Department of Informatics Engineering
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jl. Ganesha 10, Bandung 40132, Indonesia
[1]13522122@std.stei.itb.ac.id

***Abstract*— The rapid adoption of Quick Response Code Indonesian Standard (QRIS) for electronic transactions in Indonesia has been accompanied by a rise in fraudulent activities involving counterfeit QRIS codes. This paper presents an approach to detecting counterfeit QRIS codes through the integration of machine learning, perspective correction, and string-matching algorithms. The proposed method begins with edge detection to isolate and normalize the QRIS image to ensure the image is accurate. Machine learning techniques are then use for text extraction. The authenticity of the QRIS is verified using a string-matching algorithm to compare the extracted data against the expected values. Testing showed that the method effectively corrected the perspective of QRIS images captured at various angles and accurately distinguished between genuine and counterfeit QRIS codes, though the execution time was relatively slow. Further optimization is needed to improve the efficiency of the perspective correction, text extraction, and QR decoding processes.**

Keywords*— QRIS; Machine Learnin; Perspective Correction; String Matching.

I. Introduction

In recent years, the use of Quick Response Code Indonesian Standard (QRIS) has gained significant traction in Indonesia, offering a very convenient and efficient method for electronic transactions. However, the rise in QRIS usage has also led to an increase in fraudulent activities, where counterfeit QRIS codes are used to deceive users and misappropriate funds. This paper aims to address this pressing issue by exploring an approach to detect counterfeit QRIS codes through the integration of machine learning, perspective correction, and string-matching algorithms.

The motivation behind this research arises from several reported cases of QRIS counterfeiting in Indonesia, which pose a significant threat to both consumers and businesses. For instance, incidents of QRIS fraud have been reported where fake QR codes were used to divert payments intended for legitimate merchants to fraudulent accounts. Additionally, the increasing prevalence of such cases has prompted concerns about the security of QRIS transactions, highlighting the urgent need for a reliable detection mechanism to safeguard against fraudulent activities.

This paper detection process begins with perspective correction to isolate and normalize the QRIS image, ensuring that only the relevant QR code is processed, devoid of any irrelevant background. This step is crucial for enhancing the accuracy of following analyses. Following normalization, machine learning techniques are used to extract text from the QRIS, specifically targeting the retrieval of the name and ID associated with the QR code.

To verify the authenticity of the QRIS, this paper utilizes a string-matching algorithm to compare the reconstructed string against the expected name and ID from the QR code. If a match is found, the QRIS is considered authentic; otherwise, it is classified as counterfeit. This multi-step approach leverages the strengths of various computational techniques to provide a comprehensive solution for detecting counterfeit QRIS codes.

II. Fundamental Theory

By combining machine learning, perspective correction, and string matching, this paper aims to contribute a method for enhancing the security and reliability of QRIS transactions in Indonesia, ultimately protecting users from the disadvantageous effects of QRIS counterfeiting.

A. Machine Learning and Text Detection

Machine Learning is a subset of artificial intelligence that involves the development of algorithms and statistical models enabling computers to perform specific tasks without explicit instructions. Instead, the systems learn from data and improve their performance over time.[1]

Machine learning's ability to learn from data and improve over time has made it a powerful tool in the field of computer vision, particularly in tasks such as text detection. By training machine learning models on large datasets of images with labeled text regions, these models can learn to identify and localize text within images or video frames. The detected text regions can then be processed further by text recognition

systems, creating a solution for automated text extraction from images.

Text detection itself is a process in computer vision that involves identifying and localizing text within images or video frames. This task is crucial for various applications such as document analysis, automatic number plate recognition, and assisting visually impaired individuals. Text detection algorithms scan through images to detect regions that likely contain text, which can then be extracted and processed further by text recognition systems (often referred to as Optical Character Recognition, or OCR).[2]

In this study, this paper will utilize PyTesseract for text detection. PyTesseract is an Optical Character Recognition (OCR) tool for Python that recognizes and reads the text embedded in images. It is a wrapper for Google's Tesseract-OCR Engine. By using PyTesseract, it can accelerate the research process as it eliminates the need to develop a new model from scratch. This tool has been widely used and validated in the field, providing reliable and efficient text detection for a variety of applications. Its use in this context allows to focus on the application and evaluation of our approach, rather than the time-consuming task of model development.

The text detection process in this study will specifically target the Quick Response Code Indonesian Standard (QRIS) images. The text elements of interest within these images are the 'name' and 'id'. These pieces of information are important as they are unique identifiers that can be used to verify the authenticity of the QRIS.

B. Perspective Correction

Perspective correction is necessary for ensuring QRIS images are properly aligned and easy to read. The process begins with preprocessing the QRIS image by resizing it, converting it to grayscale, and applying transformations such as dilation, erosion, and Gaussian blur to enhance the edges. Grayscale conversion simplifies processing by removing color information, while dilation and erosion emphasize boundaries and remove noise. Gaussian blur reduces noise and detail, improving edge detection accuracy.

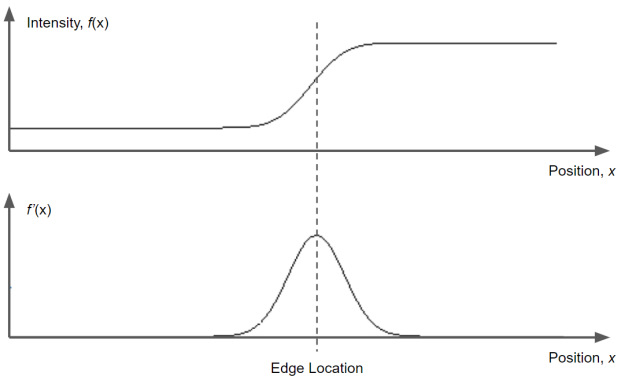


Figure 1. Canny Edge Detection Illustration (Source: *towardsdatascience.com*)

Edge detection is performed using the Canny edge detection algorithm, which identifies areas with rapid intensity changes. This algorithm uses two thresholds to detect strong and weak edges, ensuring that weak edges connected to strong edges are also considered. Then, the Hough Line Transformation is going to be used to detect straight lines in the edge-detected image by transforming points in the image space to the parameter space and identifying lines where many points converge.

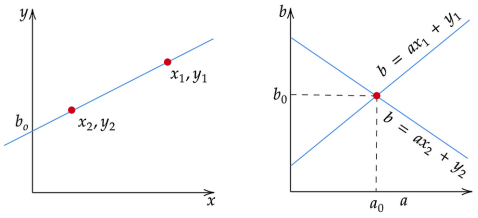


Figure 2. Line Detection Illustration (Source: *towardsdatascience.com*)

Detected lines often include duplicates and irrelevant lines, so it needs to be filtered to retain only the most relevant ones and calculates their intersections. Line filtering removes duplicates based on predefined thresholds, and intersection points are calculated for lines forming angles within a specified range, indicating potential corners of the QRIS image. From these intersection points, the best quadrilateral, which forms the QRIS code boundary, is selected based on the maximum area of the convex hull.

This process, the perspective correction, is going to be performed using the OpenCV library in Python, transforms the selected quadrilateral into a flat, rectangular image. Using the four corner points, a transformation matrix is computed to map the quadrilateral to a rectangle. This matrix is then applied to obtain the corrected image, ensuring a top-down, flat view of the QRIS code. OpenCV is chosen for this task because it contains all the necessary functions and algorithms for image processing, including edge detection, line detection, and perspective transformation, making it a efficient tool for such operations.

C. QR Code Decoding

This study utilizes ZBar, a versatile open-source library designed for reading barcodes from various sources including video streams, image files, and raw intensity sensors. ZBar's proficiency in decoding QR codes is largely attributed to its implementation of the Reed-Solomon error correction algorithm, which significantly enhances the quality of the decoding process. Reed-Solomon is a powerful error-correcting code capable of detecting and correcting multiple symbol errors. In the context of QR codes, this ensures that even if parts of the QR code are damaged or obscured, the embedded information can still be accurately decoded. This error correction capability is critical for maintaining the integrity of the data extracted from QRIS codes, as it ensures the reliability of the decoded information even under suboptimal conditions.

Moreover, the ZBar library employs a unique approach akin to traditional "wand" and "laser" scanners used for linear (1D) bar codes. These traditional scanners decode bar codes by passing a light sensor over the light and dark areas of a symbol, interpreting the reflected light to decode the data. Similarly, ZBar performs linear scan passes over an image, treating each

pixel as a sample from a single light sensor. This method allows the data to be scanned, decoded, and assembled on the fly, bypassing the need for complex image processing techniques that require significant CPU and memory resources. By adopting this approach, ZBar simplifies the decoding process and reduces sensitivity to various filter parameter configurations, which can often be difficult for end-users to understand and set up correctly.

Furthermore, ZBar abstracts this linear scanning approach into a layered streaming mode, allowing for efficient real-time processing of barcode data. This approach not only enhances the decoding speed but also maintains high accuracy, making it particularly effective for applications where rapid and reliable barcode reading is needed. The combination of Reed-Solomon error correction and ZBar's linear scanning method ensures that QR codes, including QRIS codes, are decoded accurately and efficiently, providing a great solution for detecting counterfeit QRIS codes. By using ZBar's efficient decoding capabilities and the robust error correction provided by Reed-Solomon, this system can reliably extract and verify the embedded information in QRIS codes.

D. String Matching Algorithm

The fundamental theory behind string matching algorithms revolves around efficiently finding a pattern within a larger text. The two primary approaches discussed are the Knuth-Morris-Pratt (KMP) algorithm and the Boyer-Moore Algorithm.

1. Knuth-Morris-Pratt (KMP) Algorithm

The Knuth-Morris-Pratt (KMP) algorithm is another highly efficient string matching algorithm. It avoids redundant comparisons by preprocessing the pattern to determine the longest prefix which is also a suffix (LPS array). This preprocessing allows the algorithm to skip sections of the text that have already been matched.

The algorithm preprocesses the pattern to create the LPS array, which stores the length of the longest prefix that is also a suffix for each sub-pattern of the pattern. This step ensures that the algorithm does not re-examine characters that have already been matched.

Index	0	1	2	3
p	A	A	A	B
lps	0	1	2	

i

Figure 3. LPS in KMP Algorithm (Source: *medium.com*)

After doing the preprocessing, the algorithm then switch to matching phase. During the matching phase, the algorithm compares characters of the pattern with characters of the text. If a mismatch occurs, it uses the LPS array to shift the pattern efficiently without re-examining previously matched characters.

The KMP algorithm is particularly effective for patterns with repetitive sub-patterns and performs well even with small alphabets. Its time complexity is O(m + n), where m is the length of the pattern and n is the length of the text, making it faster than brute force methods .

2. Boyer-Moore Algorithm

The Boyer-Moore string matching algorithm is known for its efficiency and is widely used in various applications. This algorithm utilizes two primary techniques: the looking-glass heuristic and the character-jump heuristic. These heuristics allow the algorithm to skip sections of the text, thereby reducing the number of comparisons and enhancing its performance.

When a mismatch occurs, the algorithm uses a precomputed table to determine how far the pattern can be shifted. This table, called the last occurrence function, maps each character in the pattern to its last occurrence in the pattern. If the mismatched character in the text does not exist in the pattern, the pattern is shifted completely past the mismatched character.

	pattern: foxtrot				
Character	f	o	x	t	r
Last Index	0	5	2	6	4

Figure 4. Example of Last Occurrence Table (Source: *www.semanticscholar.org*)

The Boyer-Moore algorithm's efficiency comes from its ability to skip large sections of the text, especially when dealing with large alphabets. However, it performs poorly with small alphabets and in the worst-case scenario, where the pattern and text are composed of repetitive characters.

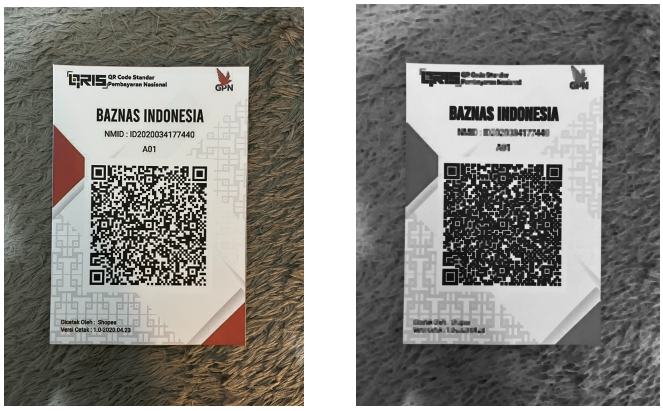
3. Algorithm Choice for QRIS Counterfeit Detection

For detecting QRIS counterfeits, where the QR codes contain a significant amount of data, the Boyer-Moore algorithm is better. QR codes encode large strings of characters, and the Boyer-Moore algorithm's ability to skip large sections of the text using the last occurrence array makes it well-suited for this task. Given the need for efficient and reliable detection, the Boyer-Moore algorithm's heuristics ensure that even large QR codes can be processed quickly and accurately by minimizing unnecessary comparisons and having more efficient pattern shifts.

III. METHODOLOGY

Based on the explanations in the fundamental theory, the research process will be divided into several parts. The first part involves perspective correction of the image to ensure that the QR code is properly aligned and isolated from its background. The second part focuses on text extraction from both the image and the QR code by using the machine learning text detection. Finally, the third part is the text matching stage, where the extracted text from the image is compared with the text from the QR code to verify authenticity and detect any potential counterfeit QRIS codes.

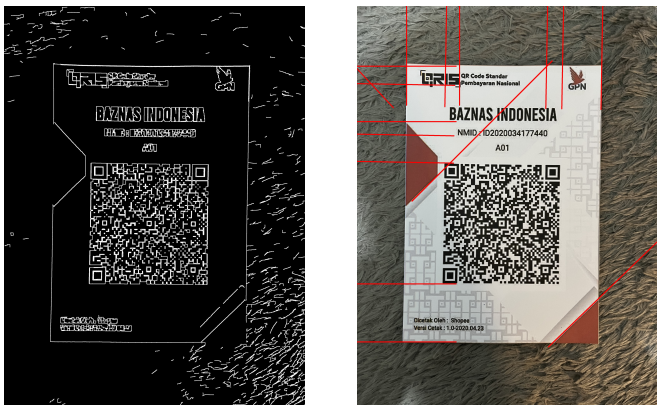
A. Perspecive Correction



(a) (b)

Figure 5. (a) Test Image for Perspective Correction and (b) Image after some filtering (Source: Author's Documentation)

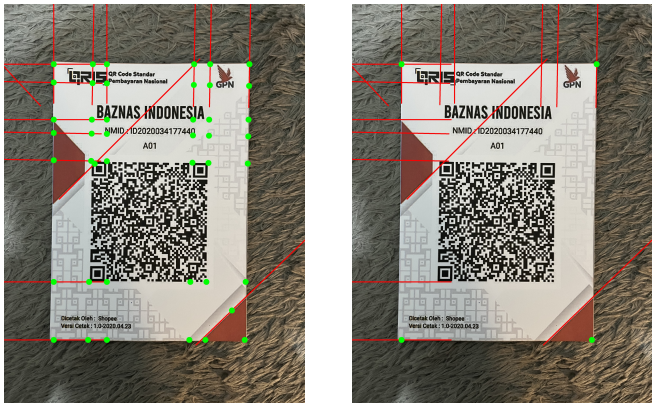
The first step of this process is converting the image to grayscale and applying dilation, blurring, and erosion. These are important preprocessing steps for edge detection and line detection. Converting to grayscale simplifies the image by reducing the color channels to one, making it easier to process and analyze. Dilation is applied to enhance and connect edges, making noticeable features more visible. Blurring, that is done using a Gaussian blur, helps reduce noise and detail. Erosion then refines the edges by removing small, irrelevant details and noise, improving the accuracy of following edge detection steps. These preprocessing steps collectively prepare the image, enhancing the important features while minimizing noise, to facilitate more accurate detection of lines and intersections in the following processing stages.



(a) (b)

Figure 6. (a) Result of Canny Edge Detection and (b) The Result of Hough Line Transformation (Source: Author's Documentation)

The next step is performing edge detection using the Canny Edge Detection Algorithm and then applying the Hough Transform to detect line. The Canny algorithm detects edges by identifying areas of rapid intensity change, creating a binary image where edges are highlighted. This edge map is then processed using the Hough Transform, which detects straight lines by transforming edge points into a parameter space (rho and theta) and identifying lines that accumulate sufficient sizes. These detected lines represent the structural boundaries of the document or object in the image. The next step is to identify the intersection points of these lines and correct the perspective of the image based on these points.



(a) (b)

Figure 7. (a) Intersection Points of the lines and (b) The Best 4 Point that created the biggest quadrilateral (Source: Author's Documentation)

The next step if finding the intersection points of the detected lines and identifying the four best points that form the largest quadrilateral. After detecting lines using the Hough Transform, the next step is to calculate the intersection points of these lines. These points represent where two different lines intersect, likely indicating the corners of the document or object of interest. From all the intersection points found, an algorithm then searches for the four points that form the largest quadrilateral. This is done by examining combinations of intersection points and calculating the area of the quadrilateral they form. The four points that form the largest area are most likely the corners of the document or object, which are then used for perspective correction to make the image appear straight and properly aligned.

Figure 8. Result Image after Perspective Correction

Finally, after identifying the four best points that form the largest quadrilateral, the perspective correction is applied. Using these points, a perspective transformation is performed to map

the identified quadrilateral to a rectangular shape, effectively "straightening" the QRIS image. This corrected image appears as if it was taken from a direct overhead view, making it properly aligned and easier to read or analyze further. The result of this perspective correction process is a clear, properly oriented image that accurately represents the QRIS in its corrected form.

B. Image and QR Code Text Extraction

The process involves two main tasks: QR code detection and decoding and text extraction from the image. This process utilizes several libraries. OpenCV (cv2) is used for most image processing tasks, including loading images, converting them to grayscale, applying Gaussian blur to reduce noise, and using Otsu's thresholding to create a binary image that highlights significant contrast areas. OpenCV also detects and filters contours based on their area, aspect ratio, and shape to identify and extract the QR code region for decoding.

For QR code decoding, the pyzbar library is used. This library is specifically designed to decode barcodes and QR codes from images, providing the functionality to read and interpret the encoded data within the QR code.

For text extraction, the Pillow (PIL) library and pytesseract are used. Pillow is a powerful imaging library that allows to open and manipulating image files. Tesseract OCR, accessed through the pytesseract wrapper, processes the image to detect and extract textual content, converting it into a readable string format. Pytesseract is a machine learning-based OCR tool, having a trained models to recognize and decode text from images accurately. This combination of libraries enables complete image processing, QR code decoding, and text extraction capabilities within the code.

Below is the result of the text extraction from the image in figure 8 using these libraries:

```
QR Code Standar
Ein [ = )/Pembayaran Nasional N
BAZNAS INDONESIA
NMID : ID2020034177440
Dicetak Oleh: Sh
Versi Cetak : 1.0-2020.04,23
```

Given that the extracted text may contain some unnecessary information, a filtering process is required to remove the unwanted text, leaving only the name and ID from the given QRIS. Below is the final result of the text extraction.

```
BAZNAS INDONESIA
ID2020034177440
```

Below is the result of the QR decoding:

```
00020101021126590016ID.CO.SHOPEE.WWW01189360091800
0016725302061672530303UBE51440014ID.CO.QRIS.
WWW0215ID20200341774400303UBE52048398530336058O2ID
5916BAZNAS Indonesia6013JAKARTA
TIMUR61051315062070703A0163041C0Cbbb
```

C. String Matching Phase

Once the name and ID from the QRIS have been obtained, a string-matching process will be done using these as patterns, with the text being the decoded result from the QR. If both the name and ID are found within the text, it can be concluded that the QRIS is authentic. However, if either or both are not found in the QRIS, it can be concluded that the QRIS is fake.

According to fundamental theory, the Boyer-Moore algorithm will be used for this task. The algorithm will be executed twice: once to search for the name and once to search for the ID.

D. Program Testing

The testing process will be conducted in two parts. The first part will be carried out on the perspective correction program. The second part will involve testing the string-matching process to determine the authenticity of QRIS.

1. Perspective Correction Testing

To test the perspective correction, images will be captured at various angles. The following illustration shows the angles at which the images will be taken. The angle value θ will change from 0, 30, and 60 degrees. At 0 degrees, the image is captured directly from the front, resulting in a straight and flat view of the QRIS code. At 30 degrees, the image is taken with a slight tilt, which may introduce some perspective distortion but remains relatively clear for recognition. At 60 degrees, the image is captured with a greater tilt, introducing more significant perspective distortion and making it harder to recognize directly without perspective correction. These angle variations are crucial for testing the ability of the perspective correction algorithm to handle different levels of distortion that might occur under various image capture conditions.

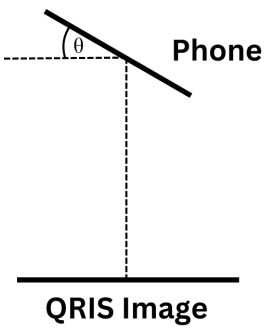
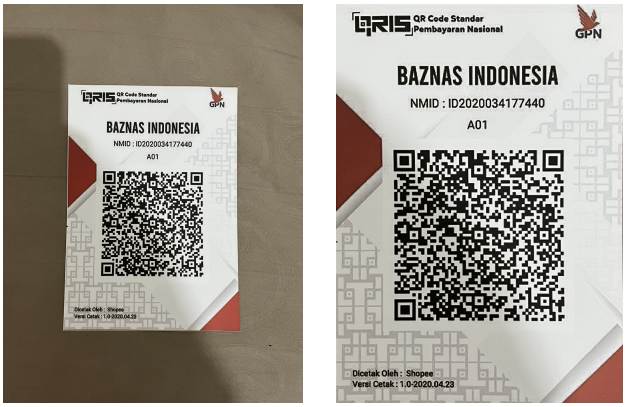


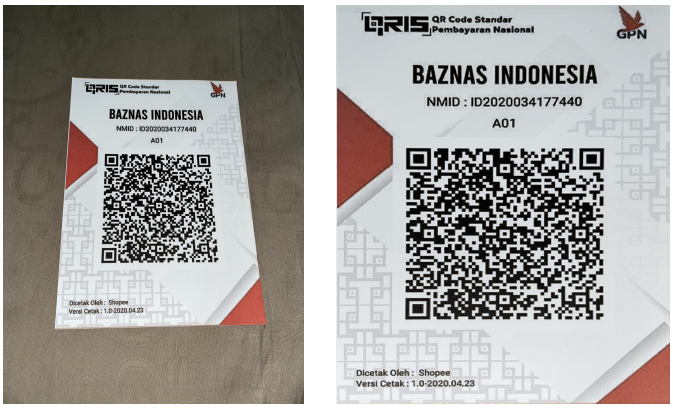
Figure 9. Test Image for Perspective Correction

And below is the result test for each angle.



(a) (b)

Figure 10. (a) Image with 0 degree tilt and (b) The corrected image result (Source: Author's Documentation)



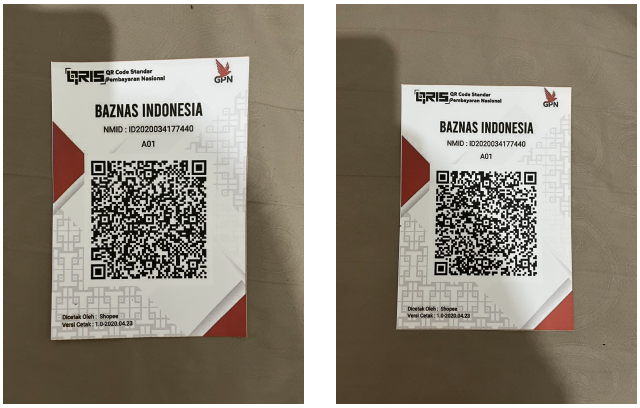
(a) (b)

Figure 11. (a) Image with 30 degree tilt and (b) The corrected image result (Source: Author's Documentation)

(a) (b)

Figure 12. (a) Image with 30 degree tilt and (b) The corrected image result (Source: Author's Documentation)

2. String Matching testing



(a) (b)

Figure 13. (a) Genuine QRIS Image and (b) Counterfeit QRIS Image (Source: Author's Documentation)

For testing purposes, two QRIS codes will be used: one genuine (QRIS (a)) and one fake (QRIS (b)). The testing will be conducted for each type, where the program will be used to verify the authenticity of the QRIS. Each type of QRIS will be captured in images three times with different angles (θ) of 0, 30, and 60 degrees, like the previous tests. The results of the text extraction and the time used will be displayed in the following table.

Table 1. QRIS Authenticity Test Results

Type / θ	Real		Fake	
	Result	Time Used	Result	Time Used
0°	Real	4,60 s	Fake	3,96 s
30°	Real	6,00 s	Fake	4,81 s
60°	Real	5,72 s	Fake	4,77 s

IV. RESULT & ANALYSIS

The testing process was divided into two parts: perspective correction testing and string-matching testing. In the perspective correction testing phase, images of QRIS codes were captured at various angles (0°, 30°, and 60°) to evaluate the accuracy of the perspective correction algorithm. The results showed that at 0° and 30°, the perspective correction was great, with the QRIS images being accurately corrected and fully captured. However, at a 60° angle, while the algorithm managed to include the entire QRIS image, some background elements were also included, indicating a decrease in accuracy.

In the string-matching testing phase, the process was tested to verify the authenticity of the QRIS codes using one genuine QRIS (a) and one fake QRIS (b). Images were captured at angles of 0°, 30°, and 60°, and the results of text extraction and QR decoding were analyzed. The program successfully distinguished between genuine and fake QRIS codes in all test cases. The performance of text extraction and QR decoding was consistent and accurately identified the authenticity of the QRIS

codes. However, the execution time was relatively slow, with most tests taking over 4 seconds, suggesting possible inefficiencies in the process. This indicates that despite using the Boyer-Moore string matching algorithm, which is known for its efficiency, the time-consuming steps might be in perspective correction, text extraction, or QR decoding.

V. Conclusion

In conclusion, the testing demonstrated that the perspective correction and text extraction processes are effective in handling QRIS images captured at various angles. Although the perspective correction at a 60° angle showed reduced accuracy by including some background elements, it did not affect the subsequent processes, and the QRIS codes were correctly identified. The string-matching tests confirmed the program's ability to differentiate between genuine and fake QRIS codes. However, the overall performance could be improved as the execution time for these processes was notably slow. Further research is needed to pinpoint whether the time-consuming steps are in perspective correction, text extraction, or QR decoding, and to optimize these processes accordingly.

Github Repository

https://github.com/maulvi-zm/DetectQRCounterfeits

Video Link at Youtube

https://youtu.be/kgaGwsbFUjo

Acknowledgment

Praise and gratitude are only to Allah Swt., for it is through His blessings and abundant grace that the author has been able to complete this paper successfully. Special thanks are also extended to Ir. Rila Mandala, M.Eng., Ph.D., and Mr. Monterico Adrian, S.T., M.T., as the lecturer for the IF2211 Algorithm Strategy course, Class K-03, for the knowledge imparted to the author, enabling the successful completion of this paper. Additionally, heartfelt thanks are conveyed to the parents for their constant support and motivation provided to the author.

References

[1] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

[2] Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2016). Reading Text in the Wild with Convolutional Neural Networks. *International Journal of Computer Vision*, 116(1), 1-20.

[3] Calvin, F., Olajuwon, J. K., & Kusuma, G. P. (2022). QR Code Detection a`nd Rectification Using Pyzbar and Perspective Transformation. *Journal of Theoretical and Applied Information Technology*, 100(21), 6408-6414.

[4] Munir, Rinaldi. (2021). *Pencocokan String*. Institut Teknologi Bandung. Retrieved from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Accessed on 10 June 2024, 20.51, GMT+7.

Statement

I hereby declare that this paper I have written is my work, not a translation or reproduction of someone else's paper, and it is not plagiarized.

Bandung, June 12nd 2024

Maulvi Ziadinda Maulana
13522122