BARCODE DETECTION

18CSE353T – DIGITAL IMAGE PROCESSING

Mini Project Report

Submitted by

Khushi Tiwari [Reg. No.: RA2111003011709]
B.Tech. CSE - Core
Aniket Srivastava [Reg. No.: RA2111003011754]
B.Tech. CSE - Core

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603 203
CHENGALPATTU DISTRICT

November 2023

COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603 203

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

rtified that Mini project report titled Barcode Detection is the bonafide work of Reg.No \2111003011709 and RA2111003011754 Name Khushi Tiwari and Aniket Srivastava who carried out e minor project under my supervision. Certified further, that to the best of my knowledge, the work ported here in does not form any other project report or dissertation on the basis of which a degree or vard was conferred on an earlier occasion on this or any other candidate.

[illegible]

Dr. M. L. Sworna Kokila

(ASSISTANT PROFESSOR)

[illegible]

Department of Comp Technolog Kattankulat 603 203 SRM IST

Dr. Pushpalatha M

(HEAD OF THE DEPARTMENT)

TABLE OF CONTENTS

1. Problem Statement

The task involves implementing a robust barcode detection system in Python. The system is designed to analyze an image file specified by its path, employing OpenCV for image processing and the pyzbar library for barcode detection and decoding. The image is first converted to grayscale to facilitate efficient analysis. Subsequently, the pyzbar library is employed to identify and decode barcodes present in the image. For each successfully detected barcode, the system outlines a rectangle around it on the original image, providing a visual representation of the recognized codes. The processed image, complete with drawn rectangles, is then displayed using Matplotlib

Furthermore, the system outputs essential information about the detected barcodes, including their types and data. This information is printed, providing a concise yet informative summary of the barcode contents. Overall, this barcode detection system combines image processing, library integration, and visualization techniques to offer an effective and comprehensive solution for identifying, decoding, and presenting barcode information from a given image file**.**

2. Methodology for Barcode detection:

- Import Required Libraries:
  Import the necessary libraries: OpenCV (cv2), pyzbar (decode function), Matplotlib (plt), and NumPy (np).

- Define Function detect_barcode:
  Define a function named detect_barcode that takes an image file path as an argument.

- Read the Image:
  Use OpenCV to read the input image specified by the provided file path.

- Convert Image to Grayscale:
  Convert the color image to grayscale using OpenCV's cvtColor function.

- Barcode Detection Using pyzbar:
  Use the decode function from pyzbar to detect barcodes in the grayscale image.
  The result is a list of barcode objects, where each object contains information about the detected barcode, including its type, data, and polygon points.

- Loop Over Detected Barcodes:
  Iterate over each detected barcode in the list.
  Extract the barcode data and type from the barcode object.

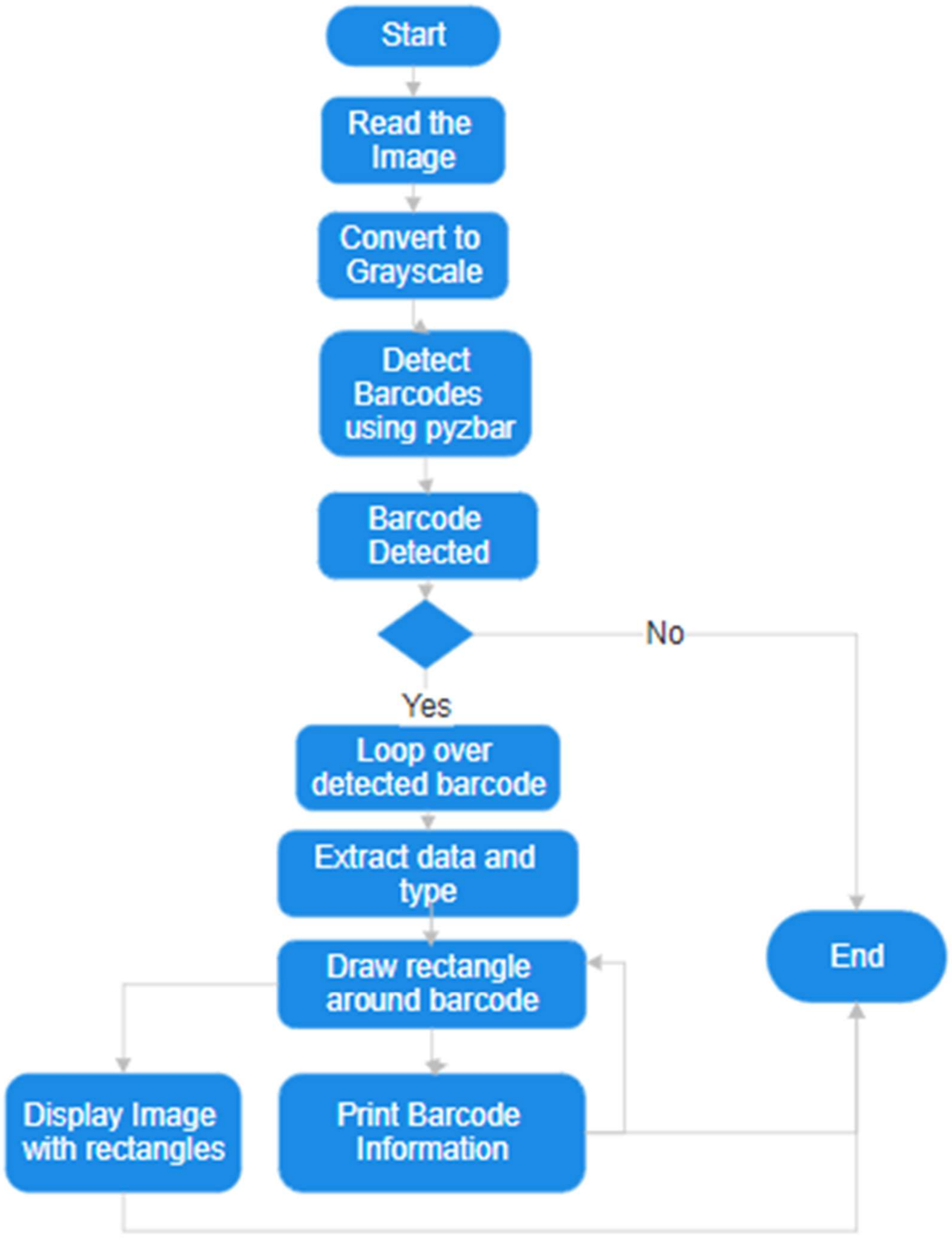
- Draw Rectangles Around Barcodes:
  For each barcode, extract the polygon points and draw a green rectangle around it.
  If the barcode has exactly 4 points (a rectangle), convert the points to NumPy array format and use OpenCV's polylines function to draw the rectangle on the original image.

- Print Barcode Information:
  Print the barcode type and data for each detected barcode in the console.

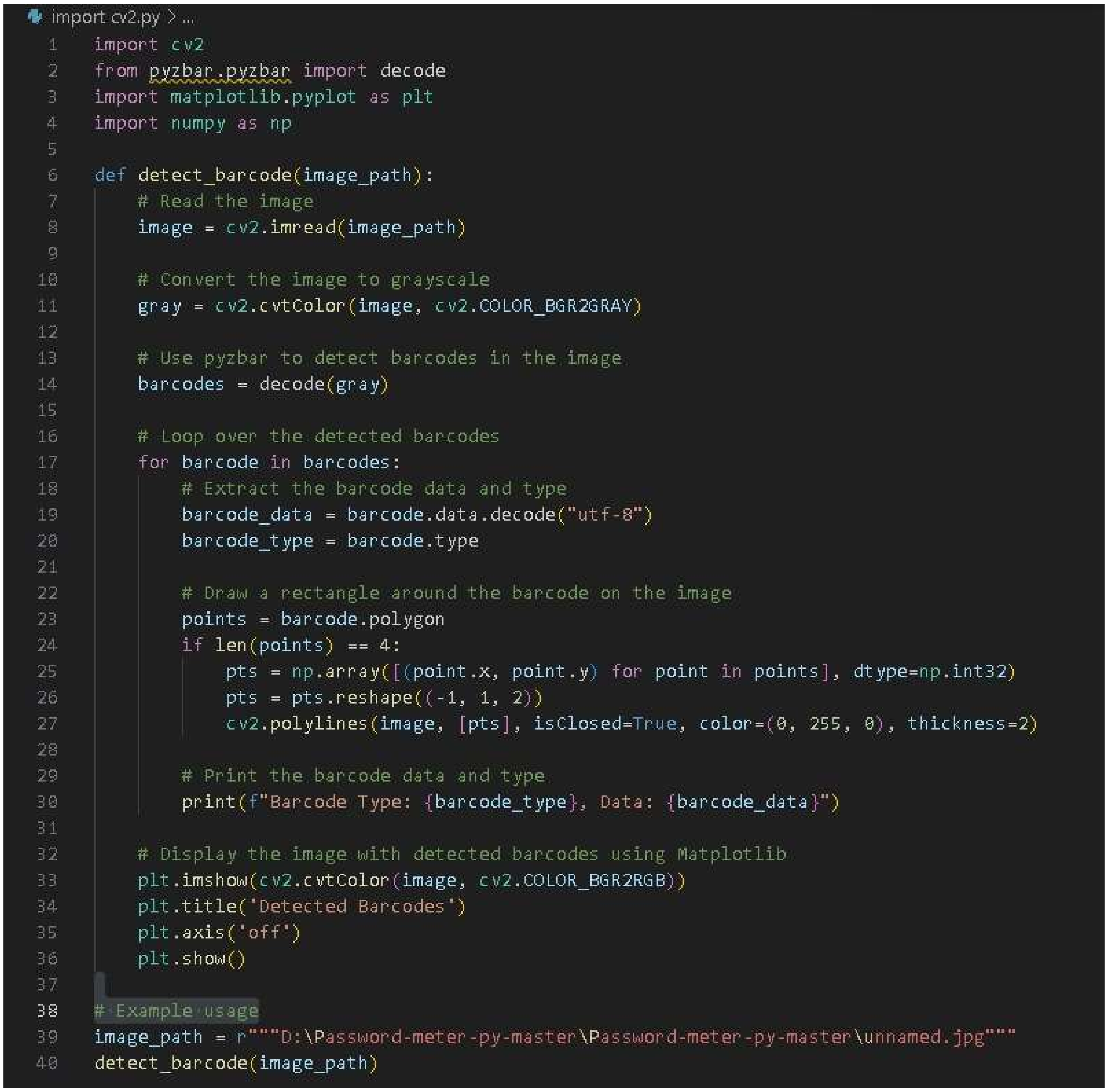
- Display Image with Detected Barcodes:
  Use Matplotlib to display the original image with the drawn rectangles around the detected barcodes.
  The image is displayed in RGB format, so convert it using cv2.cvtColor(image, cv2.COLOR_BGR2RGB).

- Example Usage:
  Provide an example usage of the function by specifying the file path of an image containing barcodes.

3. Flow chart


Start
Read the Image
Convert to Grayscale
Detect Barcodes using pyzbar
Barcode Detected
No
Yes
Loop over detected barcode
Extract data and type
Draw rectangle around barcode
End
Display Image with rectangles
Print Barcode Information

4. Coding (Python)



import cv2.py > ...

```python
import cv2
from pyzbar.pyzbar import decode
import matplotlib.pyplot as plt
import numpy as np

def detect_barcode(image_path):
    # Read the image
    image = cv2.imread(image_path)

    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Use pyzbar to detect barcodes in the image
    barcodes = decode(gray)

    # Loop over the detected barcodes
    for barcode in barcodes:
        # Extract the barcode data and type
        barcode_data = barcode.data.decode("utf-8")
        barcode_type = barcode.type

        # Draw a rectangle around the barcode on the image
        points = barcode.polygon
        if len(points) == 4:
            pts = np.array([(point.x, point.y) for point in points], dtype=np.int32)
            pts = pts.reshape((-1, 1, 2))
            cv2.polylines(image, [pts], isClosed=True, color=(0, 255, 0), thickness=2)

        # Print the barcode data and type
        print(f"Barcode Type: {barcode_type}, Data: {barcode_data}")

    # Display the image with detected barcodes using Matplotlib
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title('Detected Barcodes')
    plt.axis('off')
    plt.show()

# Example usage
image_path = r"""D:\Password-meter-py-master\Password-meter-py-master\unnamed.jpg"""
detect_barcode(image_path)
```

5. Modules of the proposed work

The proposed work involves implementing a barcode detection system in Python. The key modules used in the provided code are as follows:

1**. OpenCV (`cv2`):**
Used for reading and processing images. It provides functions for image manipulation, color space conversion, and basic computer vision operations.

pip install opencv-python

2. pyzbar:
Used for barcode detection and decoding. It supports various barcode types such as QR codes and EAN-13.

pip install pyzbar

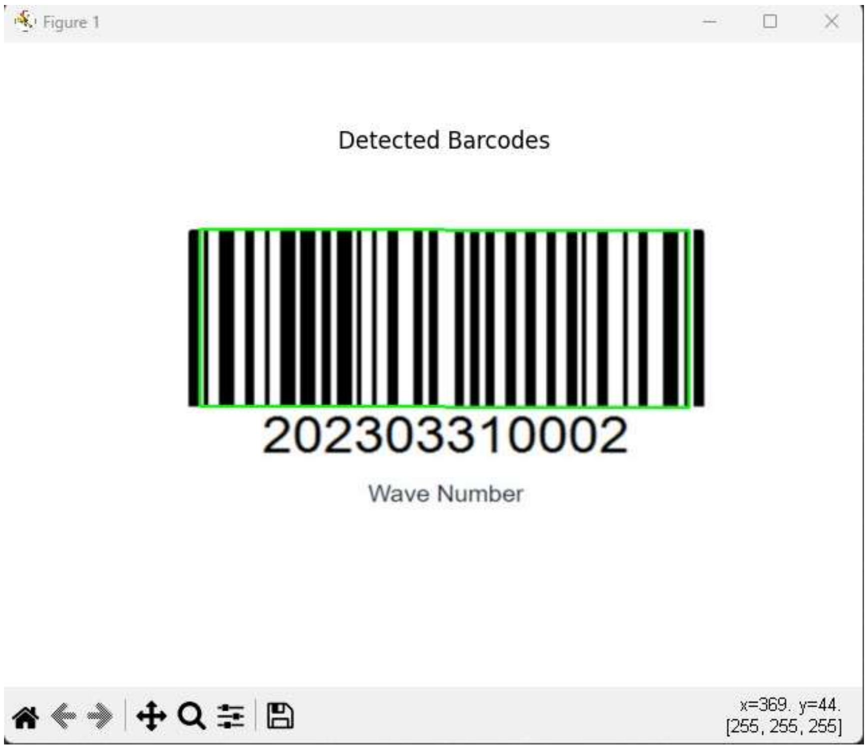
3. Matplotlib (`plt`):
Used for displaying the image with drawn rectangles. Matplotlib is a popular plotting library that can be used for various visualization tasks.

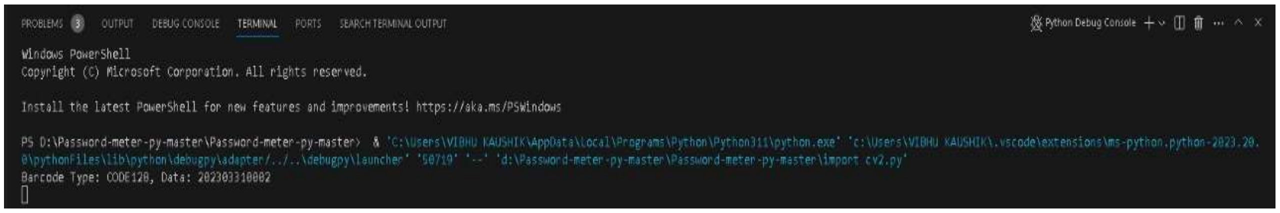
pip install matplotlib

4. NumPy (`np`):
Used for array operations. In this code, it is utilized to handle the polygon points of the detected barcodes.

pip install numpy

6. Results

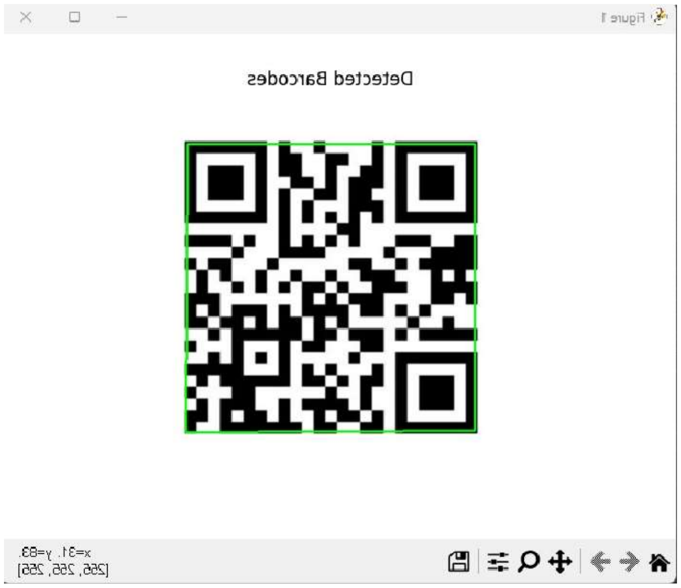


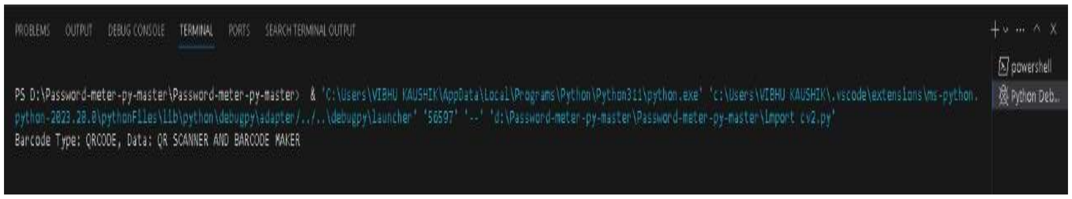


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Password-meter-py-master\Password-meter-py-master>  & 'C:\Users\VIBHU KAUSHIK\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\VIBHU KAUSHIK\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '50719' '--' 'd:\Password-meter-py-master\Password-meter-py-master\import cv2.py'
Barcode Type: CODE128, Data: 202303310002
```





```
PS D:\Password-meter-py-master\Password-meter-py-master>  & 'C:\Users\VIBHU KAUSHIK\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\VIBHU KAUSHIK\.vscode\extensions\ms-python.python-2023.28.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56597' '--' 'd:\Password-meter-py-master\Password-meter-py-master\import cv2.py'
Barcode Type: QRCODE, Data: QR SCANNER AND BARCODE MAKER
```

7. Conclusion

In conclusion, the implemented barcode detection system serves as an effective tool for extracting valuable information from images containing one or more barcodes. Leveraging the capabilities of OpenCV for image processing and the pyzbar library for barcode detection and decoding, the system successfully identifies barcode types and deciphers associated data within the provided image. The incorporation of NumPy facilitates efficient handling of polygon points, enabling the drawing of precise rectangles around detected barcodes.

The visual representation of the processed image, with rectangles highlighting each recognized barcode, enhances the interpretability of the results. Additionally, the printed output detailing the barcode type and data provides a succinct and informative summary. The successful integration of Matplotlib for image display ensures a seamless user experience, particularly in Jupyter Notebooks.

This barcode detection system offers a versatile solution applicable to diverse scenarios, from inventory management to document processing, where the quick and accurate extraction of barcode information is essential. The modular design, combining well-established libraries, promotes flexibility and ease of maintenance. Overall, this implementation demonstrates the efficacy of Python-based tools in barcode detection, contributing to the automation and efficiency of various applications.

8. References

1. https://developer.mozilla.org/en-US/docs/Web/API/Barcode_Detection_API
2. https://developers.google.com/ml-kit/vision/barcode-scanning
3. https://barcode.tec-it.com/en/MobileQRUrl
4. https://www.pyimagesearch.com/