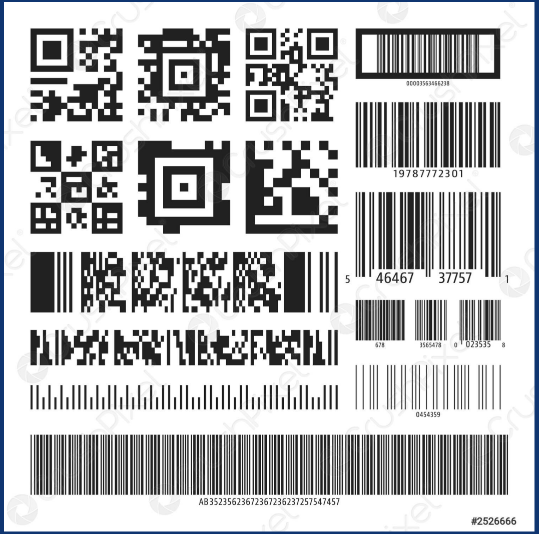
GENERATION AND DETECTION OF BARCODE AND QR-CODE

A Mini Project in Image Processing



PROJECT TEAM:

Team Members:
Bhimaraddy B Yarabandi - 201EC170
Saathwik T K - 201EC157

Under Guidance and Approval of:
A V Narasimhadhan Sir

AIM OF THE PROJECT:

To implement an efficient method for Barcode, QR-Code generator, detector and decoder using Python

INTRODUCTION

Implementation of

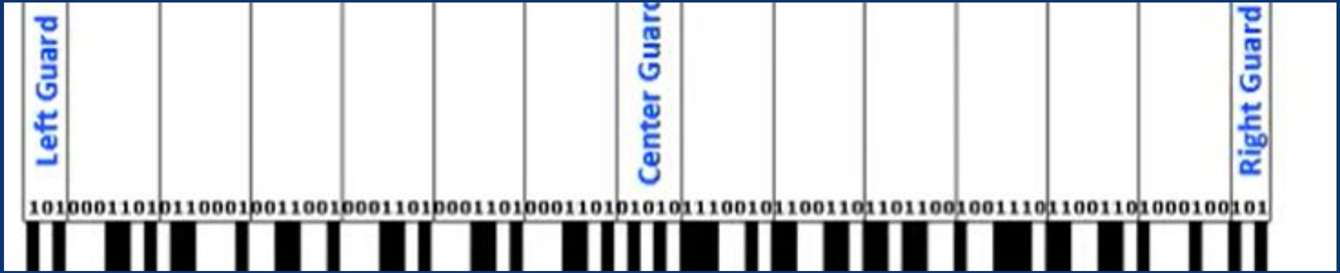
- Barcode Generator capable of encoding 13 integers (0-9) in EAN standard measurements
- Barcode/QR-code detection and decoding of (detecting the coordinates) in any given image

BARCODE GENERATION:

EAN13 STANDARD BITCODES

- Adding 6 values with left code values and 6 values with right code values
- Appending all those values and writing it into the image with left guard, right guard and centre guard

Left-Side Codes	Right-Side Codes
0001101 = 0	1110010 = 0
0011001 = 1	1100110 = 1
0010011 = 2	1101100 = 2
0111101 = 3	1000010 = 3
0100011 = 4	1011100 = 4
0110001 = 5	1001110 = 5
0101111 = 6	1010000 = 6
0111011 = 7	1000100 = 7
0110111 = 8	1001000 = 8
0001011 = 9	1110100 = 9



RESULTS OF GENERATING BARCODE

0011001155446

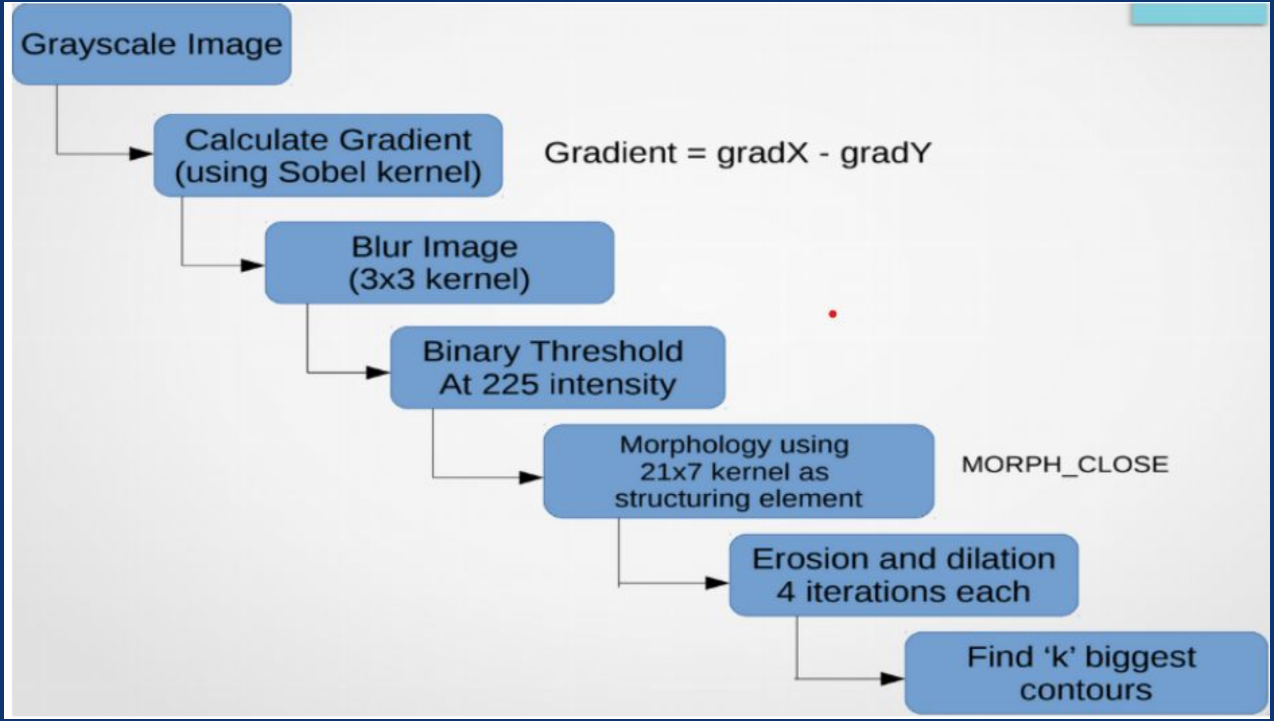
112233445566

0011001199228

BRIEF OVERVIEW OF THE APPROACH

- Conversion to grayscale image.
- Calculation of gradient using Sobel operator with kernel size =3.
- Blurring the image using 9*9 kernel and calculating the binary threshold setting the threshold value as 225.
- Applying the series of Morphological operations such as erode ,dilate to remove the white noise present in the threshold image.
- Finding the contours and drawing the contour with the help of opencv APIs

BLOCK DIAGRAM OF DETECTING BARCODDE


Grayscale Image
Calculate Gradient (using Sobel kernel)
Gradient = gradX - gradY
Blur Image (3x3 kernel)
Binary Threshold At 225 intensity
Morphology using 21x7 kernel as structuring element
MORPH_CLOSE
Erosion and dilation 4 iterations each
Find 'k' biggest contours

BARCODE/QR CODE DETECTION

Step 1:- Conversion to grayscale image

- ➢ Grayscaling is the process of converting an image from other color spaces e.g. RGB, CMYK, HSV, etc. to shades of gray.
- ➢ Importance of grayscaling in this project is ,to apply different edge detection techniques we need grayscale image and also we can reduce dimension of image(colored image has 3 dimensions where as grayscale image has only one dimension).
- ➢ Conversion to grayscaling function takes input as 3 channel image and gives 1 channel image as output
- ➢ RGB[A] to Gray: $Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$

Step 2 :- Applying gradients on image

- We use gradients for detecting edges in images, which allows us to find contours and outlines of objects in the images.
- Gradient of the image defines the different order of derivative of intensity versus pixel number plot.
- In our project we have used sobel high pass filter which finds approximate first order derivative using sobel operators which are nothing but kernels which various size and we have taken kernel size of 3 which gives optimum result.
- cv2.Sobel(gray, ddepth=ddepth, dx=1, dy=0, ksize=-1)
- dx=1 implies derivative in x direction keeping y constant and ksize=-1 implies the kernel size of 3 which is by default and ddpeth is desired depth of output.

- Calculate the gradient in both x and y direction and absolute value of subtraction of x gradient from y gradient give rough estimation where the barcode is present in the entire image.
- Sobel method uses two kernels one in x direction and other in y direction to evaluate the gradient of the image.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- We could also use the Scharr kernel instead of the Sobel kernel which may give us better approximations to the gradient and the function parameters for Scharr gradient remains same.

Step 3 :- Blurring and Applying Threshold to gradient image

- Blurring of the image helps in removal of noise.

- We implemented the averaging blurring function of 9*9 kernel which calculates the average of 81 near elements of particular pixel and replaces that particular pixel with average value calculated.

- Generally 9*9 kernel is used as standard in blurring operation because of the efficiency achieved in blurring so we took 9*9 kernel to calculate the blurred value of each pixel.

- Our implementation of threshold function takes two arguments one is source image and other is threshold value.

- In binary thresholding we slide through all the pixels and if pixel value is greater than the threshold value then the pixel will be replaced with 255 as value indicated white region else it will be replaced with zero indicating black region.

- This is necessary because the further operations on image requires image to be black and white image.

Step 4:- Applying Morphological operations

- Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images.

- It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation and in our case it is by default 3*3 kernel.

- Our implementation of erosion function takes source image as input and erodes the white noise presented in image and return the eroded image.

- Logic of erosion is simple , the minimum value of 8 adjacent pixel values of particular pixel is taken and the pixel under consideration is replaced with the minimum value calculated.

- Here the white patches other than barcode region are eroded away and boundaries of barcode region also eroded.

- In dilation function the logic implemented is exactly opposite to that of erosion in which we replace maximum of 8 adjacent pixel values of particular pixel is taken and the pixel under consideration is replaced with the maximum value calculated.
- The closed operation is dilation followed by erosion.
- For erosion the method is erosion(src_img)
- For dilation the method is dilation(src_img).
- Iterations defines the number of times the operation must be done .
- The important thing is the object must be white and background must be black and we got this configuration from previous operations.

Step 5:- Finding the contours and drawing them

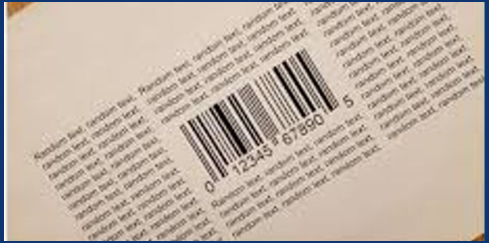
- ➢ Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

- ➢ there are three arguments in cv.findContours() function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy. Contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

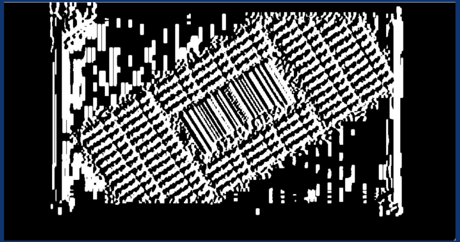
- ➢ We are using retrieval mode as External because it only returns the contours of hierarchy 0 in the hierarchy list and we used Simple as the approximation because to draw rectangle only 4 points are required and like this we are saving memory.

- To draw the contours, cv2.drawContours( ) function is used.
- Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.
- If there are many contours in hierarchy 0 then we sort according to the area and take the contour with maximum area because we assume that after filtering the barcode region or QR code region will be having more area compared to other contours.
- Pass this as argument to cv2.minAreaRect( ) function which returns the midpoint ,height and width of rectangle and passing this as argument to boxPoints( ) method gives the four coordinate of rectangle.
- Now give this result as Argument to cv2.drawContours( ) method and we finally get the required result where the red rectangle covers the barcode region of the input image.

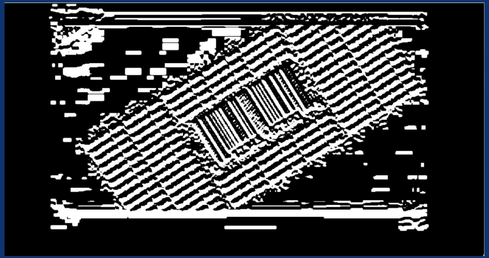
RESULTS OBTAINED IN DETECTING



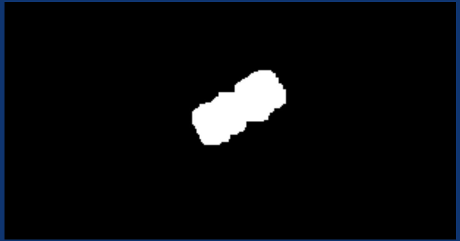
Input image containing Barcode



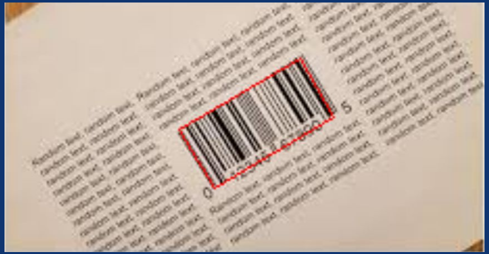
Gradient in X direction



Gradient in Y direction

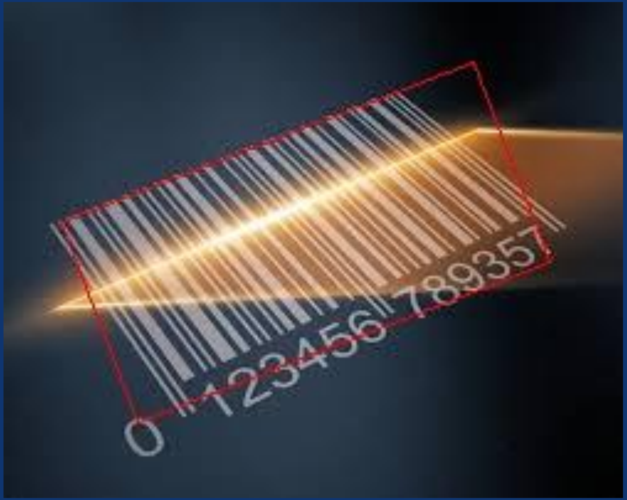


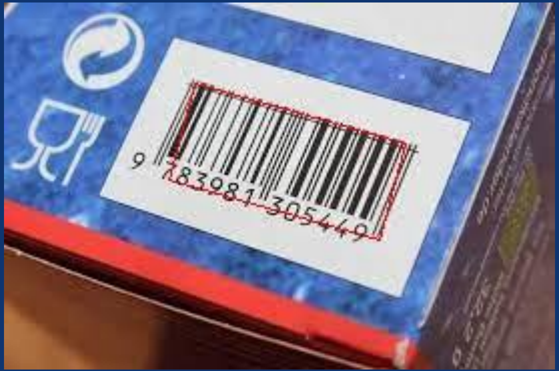
Result after applying morphological operations

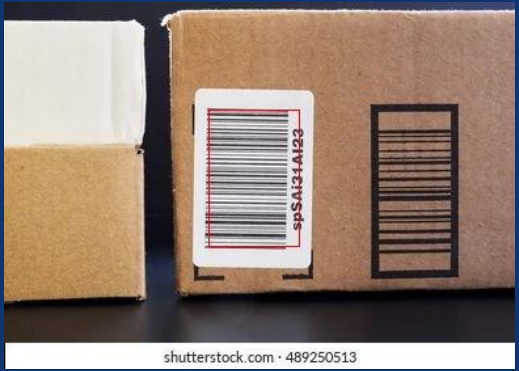


Final detected barcode

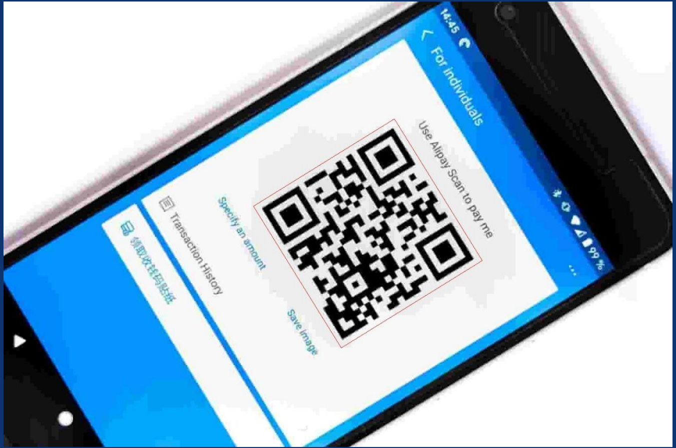
RESULTS OF DETECTING BARCODE

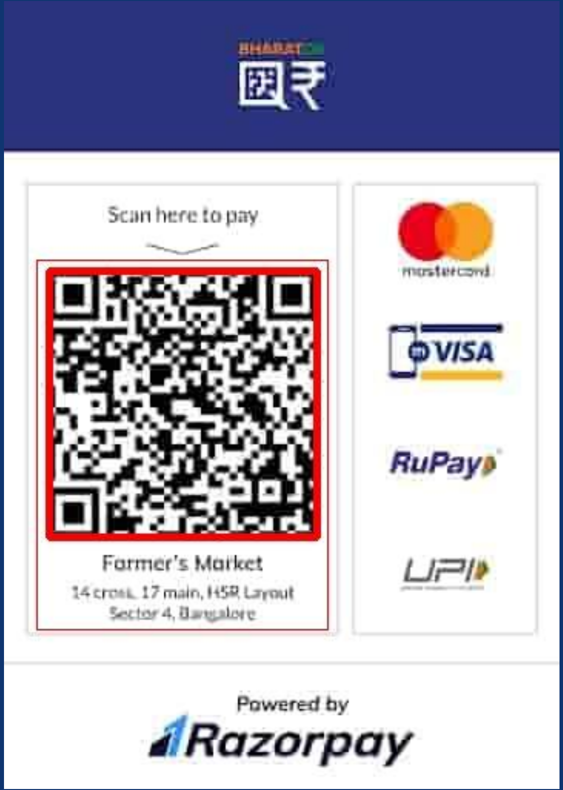






RESULTS OF DETECTING QR CODE

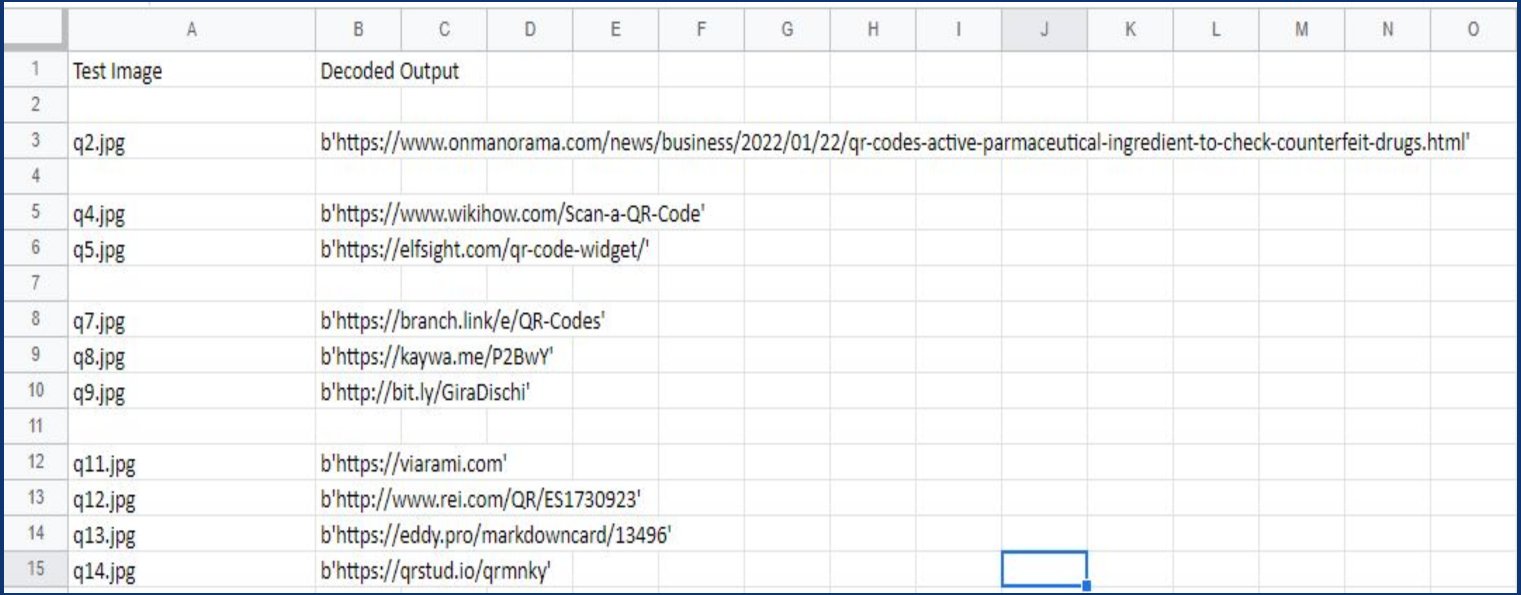




BARCODE/QR CODE DECODING

➢ Decode function in the pyzbar library is used for decoding the information stored in the barcode and QR code .

➢ The decoded data is extracted and written into the excel file as we have shown below.



	A	B
1	Test Image	Decoded Output
2		
3	q2.jpg	b'https://www.onmanorama.com/news/business/2022/01/22/qr-codes-active-parmaceutical-ingredient-to-check-counterfeit-drugs.html'
4		
5	q4.jpg	b'https://www.wikihow.com/Scan-a-QR-Code'
6	q5.jpg	b'https://elfsight.com/qr-code-widget/'
7		
8	q7.jpg	b'https://branch.link/e/QR-Codes'
9	q8.jpg	b'https://kaywa.me/P2BwY'
10	q9.jpg	b'http://bit.ly/GiraDischi'
11		
12	q11.jpg	b'https://viarami.com'
13	q12.jpg	b'http://www.rei.com/QR/ES1730923'
14	q13.jpg	b'https://eddy.pro/markdowncard/13496'
15	q14.jpg	b'https://qrstud.io/qrmnky'

REFERENCES

[1]. Mr.B.Nagaraju, N.Venkatesh, G.Dhanalakshmi, N.Sai.Chand, D.Haritha. (2022) QR Code Generator and Detector using Python.

[2]. Adeel U., Yang S., McCann, J. A. (2014). Self-Optimizing Citizen-centric Mobile Urban Sensing Systems. Proceedings of the 11th International Conference on Autonomic computing (pp.161-167).

[3]. Badra M., and Badra R. B. (2016). A Lightweight Security Protocol for NFC-based Mobile Payments. Procedia Computer Science, 83(Ant), 705–711.

[4]. Frank I., Samuel J., and Emmanuel A. (2011). Online Mobile Phone Recharge System in Nigeria. European Journal of Scientific Research, 60(2), 295–304