GOA COLLEGE OF ENGINEERING

IMAGE PROCESSING & VISSION

IPV Mini Project

TOPIC: BARCODE DETECTION **DATE**: / /2023

AIM:

Basic implementation of barcode detection using image processing techniques.

THEORY:

Barcode:

- A barcode is a pattern of parallel lines arranged by black bars and white bars with vastly different reflectivity.
- A barcode is a pattern that machines can read and is often affixed to items, packaging, or individual components.
- It includes information that may be utilized for marketing and informative reasons, and it can also be used to monitor items during their entire lifespan.
- In various commercial applications, such as manufacturing, healthcare, and advertising, barcodes are used to embed essential information for product identification, expiry date, batch number, and other purposes.
- Bar code recognition is a process used to scan the bar code in the horizontal direction to get a string of binary codes composed of bars of different widths and colors, that is, the code information of the bar code.

METHOD:

1. Grayscale

It is an image conversion technique in image processing that eliminates every form of color information and only leaves different shades of gray. In order to perform barcode detection, we need to convert the image into grayscale.

2. Sobel function

The Sobel filter is used for edge detection. It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction.

The bar code is identified by detecting the lines that conform the bar code, as these lines can be easily identified using the Sobel operator.

3. Blurring

Blurring an image is a process that makes the image less sharp and reduces its level of detail. Blurring the image filters out the noise in the image and focus solely on the barcode region.

On applying an average blur to the gradient image using a *9 x 9* kernel, it will help smooth out high frequency noise in the gradient representation of the image

4. Binary threshold

Thresholding is one of the segmentation techniques that generates a binary image from a given grayscale image by separating it into two regions based on a threshold value.

On thresholding the blurred image, any pixel in the gradient image that is not greater than 225 is set to 0 (black). Otherwise, the pixel is set to 255 (white).

5. Morphology

Morphology is a set of image processing operations that process images based on shapes. After thresholding, there are gaps between the vertical bars of the barcode. In order to close these gaps and make it easier for the algorithm to detect the region of the barcode, we'll need to perform some basic morphological operations.

6. Erosion and Dilation

Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image.

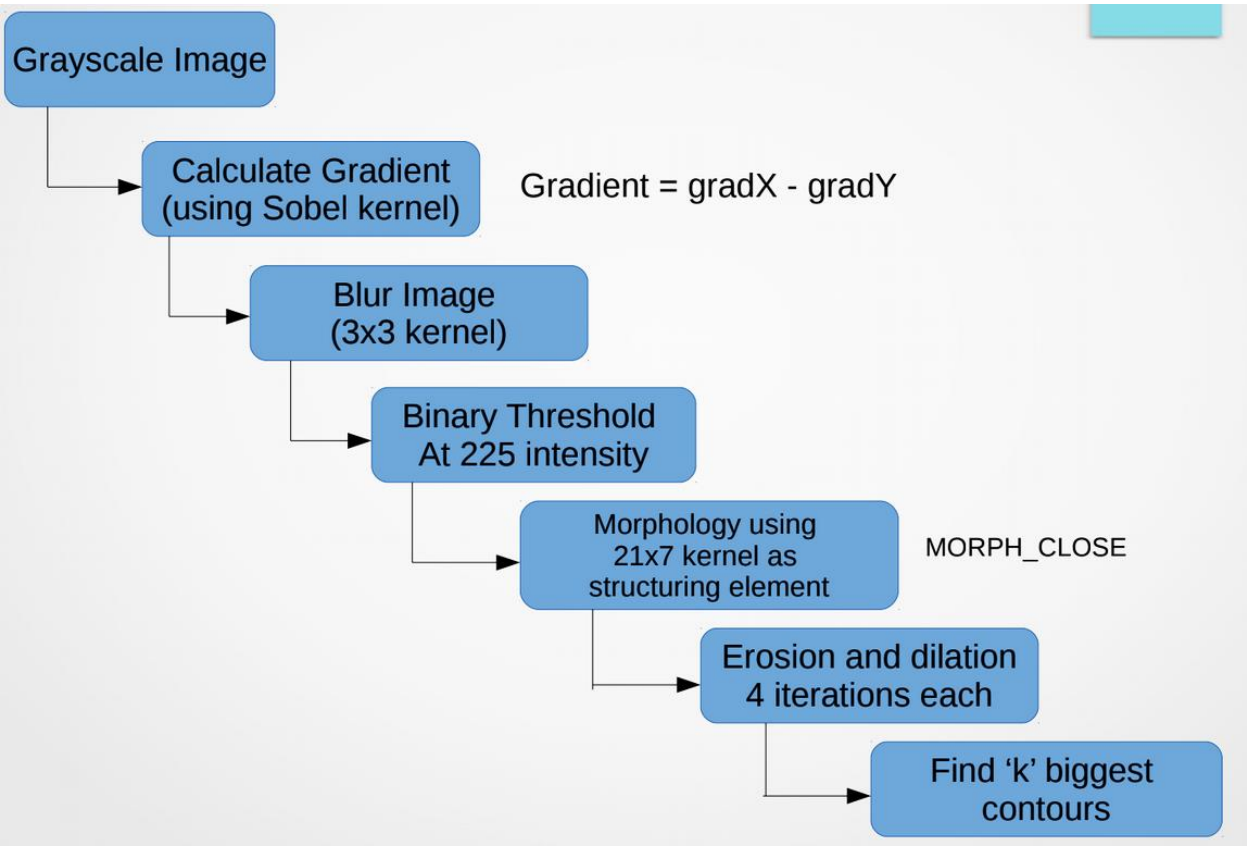
An erosion will "erode" the white pixels in the image, thus removing the small blobs, whereas a dilation will "dilate" the remaining white pixels and grow the white regions back out.

7. Finding contours

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come handy in shape analysis, finding the size of the object of interest, and object detection.

To detect the contours of the barcode region of the image, we find the largest contour in the image, which correspond to the barcoded region.

FLOWCHART:



CODE:

```
# import the necessary packages
import numpy as np  # for Numeric processing
import imutils
import cv2
# ------------ INPUT IMAGE -------------
```

```
image_name = 'C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\
IPV Project\\Project\\Img1.jpeg'

image = cv2.imread(image_name)
image = cv2.resize(image, (1000, 600))
cv2.imshow('Original Image',image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ------------- 1. GRAYSCALE --------------

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imwrite('C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\IP
V Project\\Project\\Grayscale.jpeg', gray)
cv2.imshow('Gray Scale', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ------------- 2. GRADIENT --------------

# Compute the Scharr gradient magnitude representation of
the images
# In both the x and y direction

ddepth = cv2.cv.CV_32F if imutils.is_cv2() else cv2.CV_32F
gradX = cv2.Sobel(gray, ddepth=ddepth, dx=1, dy=0, ksize=-1)
gradY = cv2.Sobel(gray, ddepth=ddepth, dx=0, dy=1, ksize=-1)

# Subtract the y-gradient from the x-gradient
gradient = cv2.subtract(gradX, gradY)
gradient = cv2.convertScaleAbs(gradient)

cv2.imwrite('C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\IP
V Project\\Project\\Gradient.jpeg', gradient)
cv2.imshow('Gradient',gradient)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
# ------------ 3. BLUR IMAGE -------------

# Blur the image
blurred = cv2.blur(gradient, (9, 9))

cv2.imwrite('C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\IP
V Project\\Project\\Blurred.jpeg', blurred)
cv2.imshow('Blurred',blurred)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ------------ 4. BINARY THRESHOLD -------------

# Threshold the image

(_, thresh) = cv2.threshold(blurred, 225, 255,
cv2.THRESH_BINARY)

cv2.imwrite('C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\IP
V Project\\Project\\Threshold.jpeg', thresh)
cv2.imshow('Threshold',thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ------------ 5. MORPHOLOGY -------------

# Construct a closing kernel and apply it to the thresholded
image
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (21, 7))
morphed = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)

cv2.imwrite('C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\IP
V Project\\Project\\Morphed.jpeg', morphed)
cv2.imshow('Morphed',morphed)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ------------ 6. EROSION AND DILATION -------------
```

```
# Perform a series of erosions and dilations
eroded = cv2.erode(morphed, None, iterations = 4)
eroded_and_dilated = cv2.dilate(eroded, None, iterations =
4)

cv2.imwrite('C:\\Users\\Admin\\Desktop\\Siddhi\\Programs\\IP
V Project\\Project\\Eroded_Dilated.jpeg',
eroded_and_dilated)
cv2.imshow('Eroded and Dilated',eroded_and_dilated)
cv2.waitKey(0)
cv2.destroyAllWindows()

# ------------ 7. FINDING CONTOURS -------------

# Find the contours in the thresholded image
# Then sort the contours by their area
# Keep only the largest one

contours = cv2.findContours(eroded_and_dilated.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(contours)
c = sorted(contours, key = cv2.contourArea, reverse =
True)[0]

# Compute the rotated bounding box of the largest contour
rect = cv2.minAreaRect(c)
box = cv2.cv.BoxPoints(rect) if imutils.is_cv2() else
cv2.boxPoints(rect)
box = np.int0(box)

# Draw a bounding box around the detected barcode and
display the image
cv2.drawContours(image, [box], -1, (0, 255, 0), 3)
cv2.imshow("Image", image)
cv2.waitKey(0)
```

OUTPUT:

1. Original Image

2. Grayscale Image

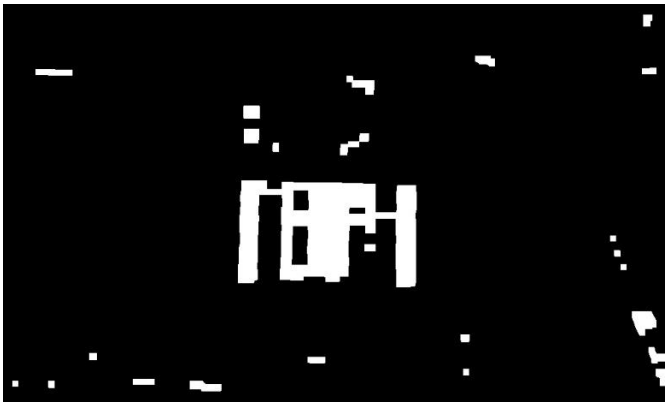
3. Gradient Image

4. Blurred Image

5. Binary Threshold

6. Morphed Image

7. Eroded and Dilated Image



8. K contour

CONCLUSION: In this project, we have successfully detected the barcode region using the above mentioned image processing operations.