MINI PROJECT

19CSE432 Pattern Recognition

Prepared by

CH Hemanth – AM.EN.U4EAC20018
G Venkatesh – AM.EN.U4EAC20023
N Jaya Viswadutt – AM.EN.U4EAC20049
P Sri Sai Ashish Varma - AM.EN.U4EEE20041

PATTERN RECOGNITION PROJECT

Text Recognition

Abstract

Text recognition, also known as optical character recognition (OCR), is a process of converting printed or handwritten text into machine-encoded text. It has become an essential technology in various industries such as finance, healthcare, and legal, where the need to digitize large volumes of paper documents is necessary.

The project aims to develop a text recognition system that uses state-of-the-art techniques to improve OCR accuracy. The system takes an input image containing text and uses image processing algorithms to preprocess the image and enhance the quality of the text. Then, the text is extracted from the image using an OCR engine.

The project focuses on exploring different OCR engines such as Tesseract, OCRopus, and Google Vision API, to identify the best-suited OCR engine for the given use case. The system is tested on different types of input images, such as scanned documents, handwritten text, and images with complex backgrounds.

The project also explores the impact of various pre-processing techniques on OCR accuracy. Techniques such as image binarization, noise reduction, and skew correction are used to preprocess the input image and improve the OCR accuracy.

The system's output can be used for various applications such as text-to-speech conversion, automatic translation, and information retrieval. It can also be integrated with other technologies such as natural language processing (NLP) to analyze the extracted text and derive meaningful insights.

Overall, the project provides a useful tool for digitizing text from images and provides insights into the best practices and techniques for achieving high OCR accuracy.

Code (Forming Bounding Box and Letter Detection)

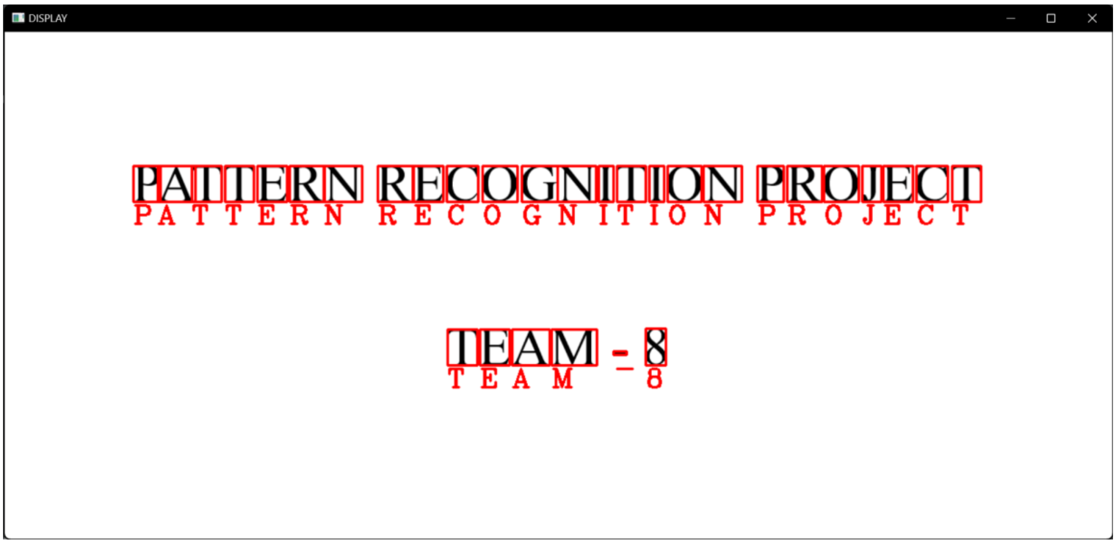
```
# importing open cv library and pytesseract
import cv2
import pytesseract

# giving path of executable file
pytesseract.pytesseract.tesseract_cmd = "C:\\Program
Files\\Tesseract-OCR\\tesseract.exe"

# read image
image = cv2.imread('DEMO.png')

# pytesseract accepts only RGB values open cv is in
BGR, so we need to convert before we load in
pytesseract library
image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)

# get raw info about image
print(pytesseract.image_to_string(image)) #image ->
prints converted text
print(pytesseract.image_to_boxes(image)) # prints the
x,y,w,h(pixels) of bounding box of text

# Detecting characters
himg,wimg,_ = image.shape
boxes = pytesseract.image_to_boxes(image) #saving
x,y,w,h(pixels) of bounding box of text in list

for b in boxes.splitlines():
    b = b.split(' ') #splitting each value based on
free space and converting the data to list
    print(b)
    x,y,w,h = int(b[1]),int(b[2]),int(b[3]),int(b[4])
    cv2.rectangle(image,(x,himg-y),(w,himg-
h),(0,0,255),2)

    # label characters around the box
    cv2.putText(image,b[0],(x,himg-
y+25),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)
cv2.imshow('DISPLAY',image) #display
cv2.waitKey(0) #infinite delay
```

Reference Text

PATTERN RECOGNITION PROJECT

TEAM - 8

Output



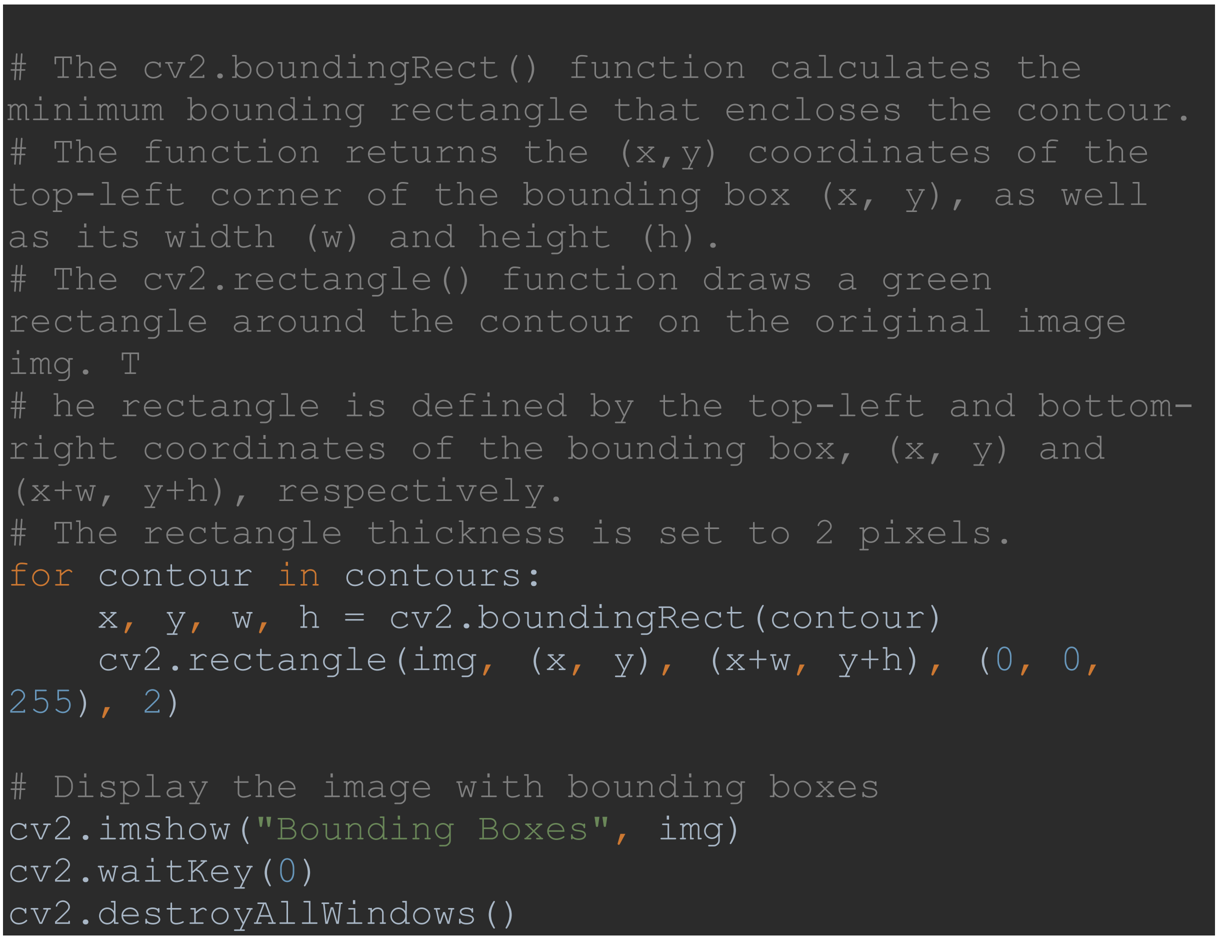
PATTERN RECOGNITION PROJECT
P A T T E R N R E C O G N I T I O N P R O J E C T

TEAM - 8
T E A M - 8

Code-2 (Forming Bounding Box without using tesseract)

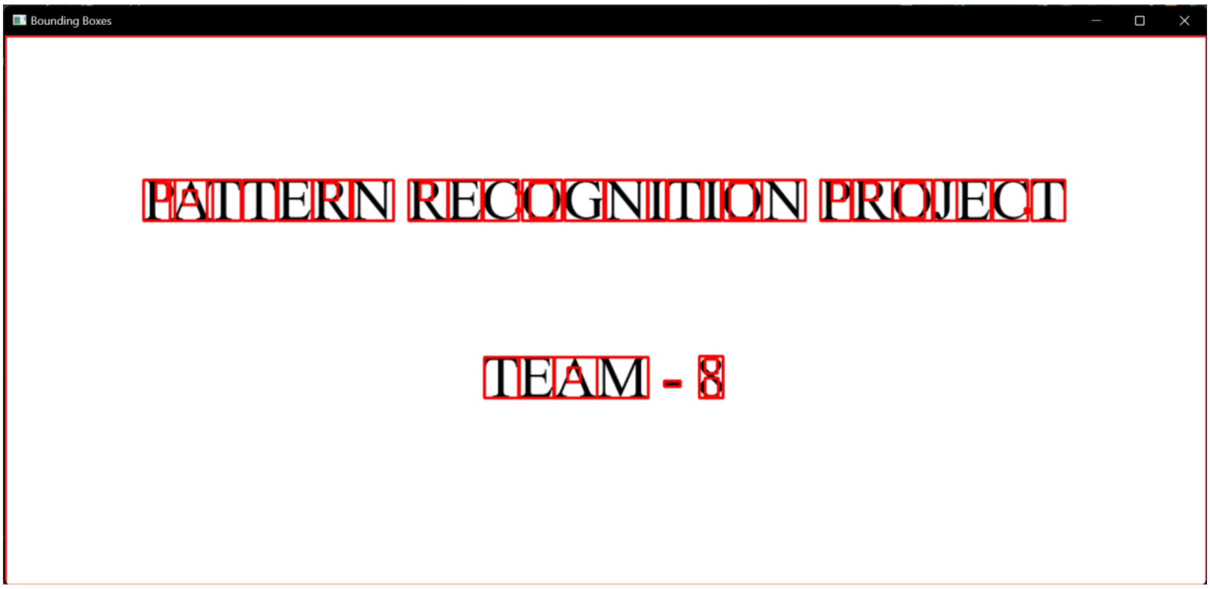
```
# Import the OpenCV library for image processing
import cv2

# Read in the input using the cv2.imread() function
and store it in the variable img
# This image will be used as input for the subsequent
image processing steps
img = cv2.imread("DEMO.png")
```

```
# Convert the image BGR (Blue-Green-Red) color space
to grayscale
# Grayscale images only contain intensity values,
while BGR images also contain color information
# Converting the image to grayscale simplifies the
image processing
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Thresholding is a technique that separates the text
from the background by setting pixel values above a
certain threshold to white and pixel values below the
threshold to black
# The cv2.threshold() function takes the grayscale
image as input, a threshold value of 127, and a
maximum value of 255.
# This means that all pixel values below 127 will be
set to 0 (black), and all pixel values above 127 will
be set to 255 (white).
# The underscore (_) in the code indicates that we
don't need the first value returned by
cv2.threshold(), which is the threshold value used.
_, binary = cv2.threshold(gray, 127, 255,
cv2.THRESH_BINARY)

# Find contours in the binary image

# Contours are continuous lines or curves that form
the boundaries of objects in an image.
# The cv2.findContours() function takes the binary
image as input, a retrieval mode of cv2.RETR_TREE,
and a contour approximation method of
cv2.CHAIN_APPROX_SIMPLE.
# The retrieval mode cv2.RETR_TREE retrieves all of
the contours and reconstructs a full hierarchy of
nested contours.
# The contour approximation method
cv2.CHAIN_APPROX_SIMPLE compresses horizontal,
vertical, and diagonal segments and leaves only their
end points, which speeds up the contour processing.
contours, hierarchy = cv2.findContours(binary,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Loop through the contours and draw bounding boxes
around each one
```



```
# The cv2.boundingRect() function calculates the
minimum bounding rectangle that encloses the contour.
# The function returns the (x,y) coordinates of the
top-left corner of the bounding box (x, y), as well
as its width (w) and height (h).
# The cv2.rectangle() function draws a green
rectangle around the contour on the original image
img. T
# he rectangle is defined by the top-left and bottom-
right coordinates of the bounding box, (x, y) and
(x+w, y+h), respectively.
# The rectangle thickness is set to 2 pixels.
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0,
255), 2)

# Display the image with bounding boxes
cv2.imshow("Bounding Boxes", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output



PATTERN RECOGNITION PROJECT

TEAM - 8