MINI PROJECT

ON

Line Segmentation of Text Images using Open CV

SUBMITTED BY

Ashish Ramayee Asokan

SRN: PES1201801573

Table Of Contents

1. OVERVIEW OF PYTHON

Python is a general-purpose, object-oriented programming language approved by International Standards Organization (ISO). Python was developed as an interpreted high-level programming language by Guido Van Rossum.

Python is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

The following reasons make Python a widely used professional language:

- ❖ It is a general-purpose programming language
- ❖ It is object oriented.
- ❖ It is easy to learn.
- ❖ It is a structured language.
- ❖ It produces efficient programs.
- ❖ It offers low level memory manipulation features.

2. PROJECT SYNOPSIS

The aim of this project is to achieve line segmentation of text images using Open CV. Segmentation of images in this project has been done using the connected components and watershed algorithms.

Once the image has been read, it is pre-processed by passing it through thresholding, for which the Otsu's binarization technique has been used. Using the connected components and Euclidian distance functions, we determine the markers required for the watershed algorithm.

Once the result of the watershed function is obtained, we iterate over the result to find the contours of the image and construct rectangles around each line of the text image.

Components Used:

- **Lists:** A list in Python is a mutable data structure represented by '[]', which allows mixed type elements.

- **Functions:** A subroutine (also called a method or a subprogram) is a portion of code within a larger program that performs a specific task and is relatively independent of the remaining code.

- **Conditional Statements:** Used to check whether a specified condition is true or not, and act accordingly.

- **Loops:** Loops are used to perform repetitive tasks.

- **NumPy Arrays:** These arrays are defined in the numpy module which are designed to make scientific calculations easier.

3. MODULES

Modules contain definitions of **Functions and Variables**, which are imported into any Python program by using the import statement.
The following are the modules that were used in the development of the project:

- **Open CV:** The Open CV module is mainly used for image processing and computer vision. Here, it has been used for image segmentation.

- **Numpy:** The numpy module contains function definitions which make scientific calculations easier. In this project, numpy arrays are used to define kernels for dilation and erosion.

- **Scipy:** Scipy is an open source library in Python used for scientific computing and technical computing. Scipy.ndimage has been used for multidimensional operations on images.

- **Skimage:** Skimage is a python library used for image processing.

- **Os:** The OS module provides an interface to use the operating system functionalities. Here, it has been used to determine the file path based on the user input.

4. FUNCTIONS

A function is a group of statements that together perform a task. The following functions have been defined in the project:

- **LineSegment.connectComponents():** This instance function has been defined to apply the connected components algorithm on the pre-processed image.

- **LineSegment.Watershed():** This instance function has been defined to determine the markers for the watershed function and applying the watershed algorithm. This function also constructs the bounding rectangle for each line in the image.

- **LineSegment.Disp_image():** This function displays the final image after the segmentation process.

- **inputFind():** This function uses the functionalities of the OS module to determine the path of the file entered by the user. It raises a FileNotFound exception if the file has not been found.

5. SOURCE CODE

Segmentation.py:

```
# Importing all necessary modules
import cv2
from skimage.feature import peak_local_max
from skimage.morphology import watershed
from scipy import ndimage
import numpy as np

# Initialising Kernels to be used in the program
kernel = np.ones((3, 3), np.uint8)
kernel1 = np.ones((5, 600), np.uint8)


class LineSegment:
    # Reading and pre-processing image
    def __init__(self, file):
        img = cv2.imread(file)
        img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        height, width, channels = img.shape
        _, thresh = cv2.threshold(img2, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
        img3 = cv2.dilate(thresh, kernel1, iterations=1)
        img3 = cv2.erode(img3, np.ones((5, int(height/200)*100), np.uint8), iterations = 2)
        self.image = img
        self.img = img3
        cv2.imshow("Image", self.img)
        self.img2 = img2

    def connectComponents(self):
        _, marked = cv2.connectedComponents(self.img, connectivity=4)
        # Mapping each component to corresponding HSV values
        label_hue = np.uint8(179*marked/np.max(marked))
        blank_ch = 255*np.ones_like(label_hue)
        labeled_img = cv2.merge([label_hue, blank_ch, blank_ch])

        # Converting from HSV to BGR for display
        labeled_img = cv2.cvtColor(labeled_img, cv2.COLOR_HSV2BGR)
        labeled_img[label_hue==0] = 0

        # Eroding Image and converting to grayscale
        labeled_img = cv2.erode(labeled_img, kernel)
        image = cv2.cvtColor(labeled_img, cv2.COLOR_BGR2GRAY)
        self.img = image
```

```
    def Watershed(self):
        # Finding Eucledian distance to determine markers
        temp = ndimage.distance_transform_edt(self.img)
        Max = peak_local_max(temp, indices=False, min_distance=30, labels=self.img)
        markers = ndimage.label(Max, structure=np.ones((3, 3)))[0]

        # Applying Watershed to the image and markers
        res = watershed(temp, markers, mask = self.img)

        # If the elements are same, they belong to the same object (Line)
        for i in np.unique(res):
            if i==0:
                continue
            mask = np.zeros(self.img2.shape, dtype="uint8")
            mask[res == i] = 255

            # detect contours in the mask and grab the largest one
            cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]
            c = max(cnts, key=cv2.contourArea)

            # Drawing a rectangle around the object
            [x, y, w, h] = cv2.boundingRect(c)
            if((w * h) > 2000):
                cv2.rectangle(self.image, (x, y), (x+w, y+h), (0, 0, 255), 2)

    def disp_image(self):
        cv2.imshow("Segmented Image", self.image)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

Mainmodule.py:

```
import Segmentation

import os

# Defining function to determine file path

def inputFind():

    name = input("Enter the file name: ")

    lol = input("Enter file extension: ")

    stri = name+"."+lol
```

```
    # Setting variables to search for path
    path=['E:\\']; path1=[]; npath=" "
    for pathn in path:

        # Using os.walk() to walk through the directories
        for root, dirs, files in os.walk(pathn):
            for names in files:
                if names ==stri:
                    npath = os.path.join(root, names)
                    path1.append(npath)
                    break

    # Checking if file has been found
    if len(path1) == 0:
        # If file has not been found, raise exception
        raise FileNotFoundError
    else:
        return path1[0]

# Handling exception raised by inputfind()
try:
    path = inputFind()
    sample = Segmentation.LineSegment(path)
    sample.connectComponents()
    sample.Watershed()
    sample.disp_image()
except FileNotFoundError:
    print("Error! The specified file could not be found")
```

6. BIBLIOGRAPHY

- www.stackoverflow.com
- www.google.co.in
- www.opencv.org
- www.pythonprogramming.net
- www.pyimagesearch.com
- www.danvk.com