ImageEnhancementProject

September 13, 2020

[ ]:
```
import cv2
```

0.0.1 Module 1: Code to find the screen resolution so that image can be resized as per users screen

[ ]:
```
import ctypes

def screen_resolution():
    user32 = ctypes.windll.user32
    screensize = user32.GetSystemMetrics(0), user32.GetSystemMetrics(1)
    width, height = screensize[0], screensize[1]
    # print(f'width = {width} and height = {height}')
    return round(width-(0.2*width)), round(height-(0.4*height))
```

0.0.2 Module 2: Code to load the image

[ ]:
```
import os



def load_images_and_files_path():
    #  path of the directory outside the main project directory
    mainProjectPath = os.path.normpath(os.getcwd() + os.sep + os.pardir)
    #  print(f 'mainProjectPath= {mainProjectPath}')

    projectPath = os.getcwd()
    print(projectPath)

    imagesDir = "passbook.jpg"
    imageDirPath = os.path.join(projectPath, imagesDir)
    # print(imageDirPath)

    imagesInDirectory = os.listdir(imageDirPath)
    imageUsed = imagesInDirectory[1].split('.')  # path to select the image for␣
 ↪testing
    # print(imageUsed)
    imageName = imageUsed[0]
    imageExtension = imageUsed[1]
    completeImageName = imageName + '.' + imageExtension

    readImagePath = os.path.join(imageDirPath, completeImageName)
    # print('readImagePath')
    return mainProjectPath, projectPath, imageDirPath, readImagePath,␣
 ↪imageName, imageExtension, completeImageName
```

0.0.3 Module 3: Code to flip the image Horizontal/Vertical

[ ]:

```
import cv2

flippedImagesCollection = []



def image_flip(image):
    flippedImagesCollection.append(image)
    flipMethod = str(input(
        'Enter H to flip image HORIZONTALLY and Enter V to flip image␣
 ↪VERTICALLY and Q to quit this function ').lower())

    if flipMethod.startswith('h'):
        flippedImage = cv2.flip(image, 1)
        user_input = str(input('Do you want to flip the image again? Press Y/y␣
 ↪for YES and N/n for NO ').lower())
        if user_input.startswith('y'):
            image_flip(flippedImage)
        elif user_input.startswith('n'):
            # flippedImageName = load_images_and_files_path()[4] + '_flip.' +␣
 ↪load_images_and_files_path()[5]
            # print(flippedImageName)
            # pathToStoreImage = load_images_and_files_path()[2] + '\\' +␣
 ↪flippedImageName
            # print(pathToStoreImage)
            # cv2.imwrite(pathToStoreImage, flippedImage)
            # print('flip operation sucessfull')
            flippedImagesCollection.append(flippedImage)
        else:
            print('Please enter a valid input')
            image_flip(flippedImage)

    elif flipMethod.startswith('v'):
        flippedImage = cv2.flip(image, 0)
        user_input = str(input('Do you want to flip the image again? Press Y/y␣
 ↪for YES and N/n for NO ').lower())
        if user_input.startswith('y'):
            image_flip(flippedImage)
        elif user_input.startswith('n'):
            # flippedImageName = load_images_and_files_path()[4] + '_flip.' +␣
 ↪load_images_and_files_path()[5]
            # print(flippedImageName)
            # pathToStoreImage = load_images_and_files_path()[2] + '\\' +␣
 ↪flippedImageName
            # print(pathToStoreImage)
            # cv2.imwrite(pathToStoreImage, flippedImage)
            # print('flip operation sucessfull')
            flippedImagesCollection.append(flippedImage)
        else:
            print('Please enter a valid input')
            image_flip(flippedImage)

    elif flipMethod.startswith('q'):
        print('Quiting the function as No operations is chosen')

    else:
        print('Please enter a valid entry')
        image_flip(image)

    print('-------------------- end of image Mirror Horizontal/Vertical␣
 ↪Module--------------------------')
```

0.0.4 Module 4: Code to rotate the image

[ ]:

```
import cv2

rotatedImagesCollection = []



def image_rotation(image):
    rotatedImagesCollection.append(image)
    print('images inside rotatedImagesCollection is/are ',␣
 ↪len(rotatedImagesCollection))
    while True:
        imgRotationRequired = str(
            input("Do you want to rotate this image? Please reply with Y/y for␣
 ↪YES or N/n for NO ").lower())

        if imgRotationRequired.startswith('y'):
            rotationOrientation = str(input(
                "Please enter your input as 'R' for Right rotation or 'L' for␣
 ↪Left rotation or 'Q' to exit ").lower())

            if rotationOrientation.startswith('r'):
                imageRotated = cv2.rotate(rotatedImagesCollection[-1], cv2.
 ↪ROTATE_90_CLOCKWISE)
                rotatedImagesCollection.append(imageRotated)
                continue
            elif rotationOrientation.startswith('l'):
                imageRotated = cv2.rotate(rotatedImagesCollection[-1], cv2.
 ↪ROTATE_90_COUNTERCLOCKWISE)
                rotatedImagesCollection.append(imageRotated)
                continue
            elif rotationOrientation.startswith('q'):
                break
            else:
                print('please enter a valid input')
                image_rotation(rotatedImagesCollection[-1])

        elif imgRotationRequired.startswith('n'):
            break
        else:
            print('please enter a valid input')
            image_rotation(rotatedImagesCollection[-1])

    print('-----------------------end of image rotation␣
 ↪module----------------------------')
```

0.0.5 Module 5: Correct the skewness of the image

[ ]:

```
# import the necessary packages
import numpy as np
import cv2
from utils.loadDocumentsAndImages import load_images_and_files_path


def image_skewness(skewedImage):

    # convert the image to grayscale and flip the foreground and background to␣
 ↪ensure foreground is now "white" and
    # the background is "black"
    gray = cv2.cvtColor(skewedImage, cv2.COLOR_BGR2GRAY)
    gray = cv2.bitwise_not(gray)

    # threshold the image, setting all foreground pixels to 255 and all␣
 ↪background pixels to 0
```

```
    thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

    # grab the (x, y) coordinates of all pixel values that are greater than zero, then use these coordinates to
    # compute a rotated bounding box that contains all coordinates
    coords = np.column_stack(np.where(thresh > 0))
    # print(f'coords= {coords}')
    angle = cv2.minAreaRect(coords)[-1]
    print(f'angle = {angle}')

    # the `cv2.minAreaRect` function returns values in the range [-90, 0); as the rectangle rotates clockwise the
    # returned angle trends to 0 -- in this special case we  need to add 90 degrees to the angle
    if angle < -45:
        angle = -(90 + angle)
    # otherwise, just take the inverse of the angle to make  it positive
    else:
        angle = -angle

    # rotate the image to deskew it
    (h, w) = skewedImage.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotatedImage = cv2.warpAffine(skewedImage, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)

    # Saving the output image
    flippedImageName = load_images_and_files_path()[4] + '_skewnessCorrected.' + load_images_and_files_path()[5]
    pathToStoreImage = load_images_and_files_path()[2] + '\\' + flippedImageName
    cv2.imwrite(pathToStoreImage, rotatedImage)

    # draw the correction angle on the image so we can validate it
    # cv2.putText(rotatedImage, "Angle: {:.2f} degrees".format(angle), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # show the output image
    print("[INFO] angle: {:.3f}".format(angle))
    print('-------------------------end of Image Skewness Module----------------------------')
    return rotatedImage
```

0.1 Main Module to run all the codes

[ ]:

```python
"""Hierarchy of functions to be applied
1. resize
2. Mirror Horizontal / Vertical
3. Deskew
4. Rotate 180, Rotate 90 Right and Rotate 90 Left
5. Black Border Removal

Author : Parul Sharma @ internship Project iNeuron
"""

import cv2

# importing utils functions
# from utils.imageResize import screen_resolution
# from utils.loadDocumentsAndImages import load_images_and_files_path
# from utils.mirrorHorizontalVertical import image_flip, flippedImagesCollection
# from utils.imageRotation import image_rotation, rotatedImagesCollection
# from utils.correct_skew import image_skewness


# Loading the image
img = cv2.imread(load_images_and_files_path()[3])
# print(img.shape)

# Resizing the image as per user's screen resolution
imgResize = cv2.resize(img, (screen_resolution()[0], screen_resolution()[1])) # (width, height)
cv2.imshow("Original Image", imgResize)

# print(imgResize.shape)



#  calling the util functions

#  Performing the Image Mirror/Flip operation
image_flip(imgResize)
imgFlip = flippedImagesCollection[-1]
print(imgFlip.shape)
cv2.imshow("Flipped Image", imgFlip)


#  Removing the skewness of the image
imgSkewnessRemoved = image_skewness(imgFlip)
cv2.imshow("Skewness Removed Image", imgSkewnessRemoved)
```

```
#  Performing the image rotation
image_rotation(imgSkewnessRemoved)
imgRotated = rotatedImagesCollection[-1] # imgRotated is the input for next␣
 ↪module
cv2.imshow("Rotated Image app", imgRotated)
while(True):
    k = cv2.waitKey(33)
    if k == -1:  # if no key was pressed, -1 is returned
        continue
    else:
        break
cv2.destroyWindow('Rotated Image app')


print('-------------------------------------end of␣
 ↪code-----------------------------------------')
```

[ ]:

```
```

[ ]:

```
```