Skew Detection and Correction Of Scanned Document Image

Div: C

Batch: EC1

Team Members:

- Esha Shah: C093
- Sachi Shah: C094

Many a times, when the documents are scanned, the scanning skews the image i.e tilts the image, making it a little inconvinient to read. So we are performing skew detection and correction with 2 methods implemented below in order to align the image properly.

Methods used:

1. Deskewing text with OpenCV and Python
2. Deskewing using Hough Transform

1. Deskewing text with OpenCV and Python

Given an image containing a rotated block of text at an unknown angle, we need to correct the text skew by:

1.Detecting the block of text in the image.

2.Computing the angle of the rotated text.

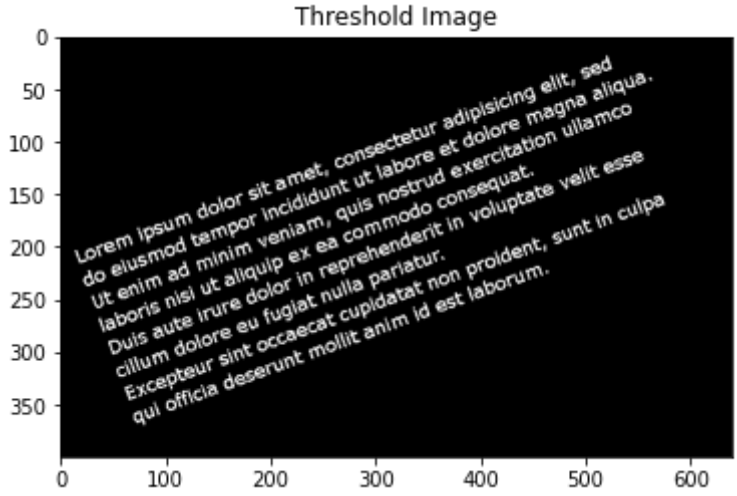
3.Rotating the image to correct for the skew.

```
# import the necessary packages
import numpy as np
import argparse
import cv2
from matplotlib import pyplot as plt

#Read image
image = cv2.imread("/content/SkewImage2.jpeg")
```

```
# convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
gray = cv2.bitwise_not(gray)
# threshold the image, setting all foreground pixels to 255 and all background pixels t
threshold_image = cv2.threshold(gray, 0, 255,cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

plt.imshow(threshold_image, cmap='gray')
plt.title('Threshold Image')
```

```
Text(0.5, 1.0, 'Threshold Image')
```



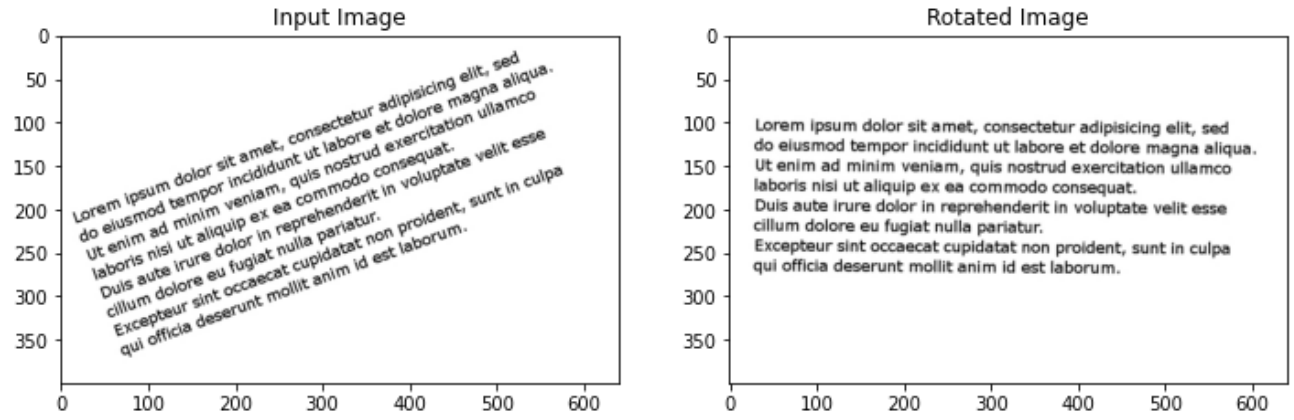

```
# grab the (x, y) coordinates of all pixel values that
# are greater than zero, then use these coordinates to
# compute a rotated bounding box that contains all coordinates
coords = np.column_stack(np.where(threshold_image > 0))
angle = cv2.minAreaRect(coords)[-1]
# the `cv2.minAreaRect` function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

print(angle)
```

```
-20.418426513671875
```

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

plt.figure(figsize=(12,12))
plt.subplot(121)
plt.imshow(image, cmap='gray')
plt.title('Input Image')
plt.subplot(122)
```

```
plt.imshow(rotated, cmap='gray')
plt.title('Rotated Image')
```

```
Text(0.5, 1.0, 'Rotated Image')
```



2. Using Hough transform

Hough transform is a feature extraction technique that converts an image from Cartesian to polar coordinates which is how it got “transform” in its name. It can be used to detect lines or a set of collinear points on the image.

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc.

The basic idea is:

1.Convert the image to grayscale

2.Apple Canny or Sobel filter

3.Find Hough lines between 0.1 to 180 degree angle.

4.Round the angles from line peaks to 2 decimal places.

5.Find the angle with the highest occurrence.

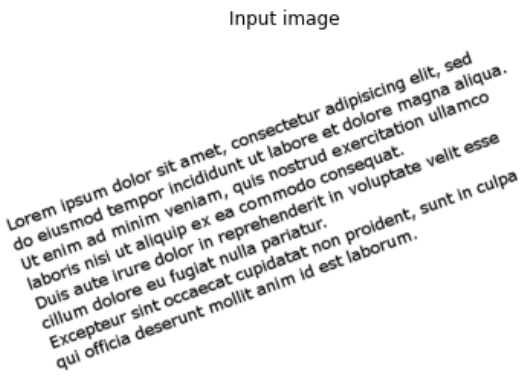
6.Rotate the image with that angle

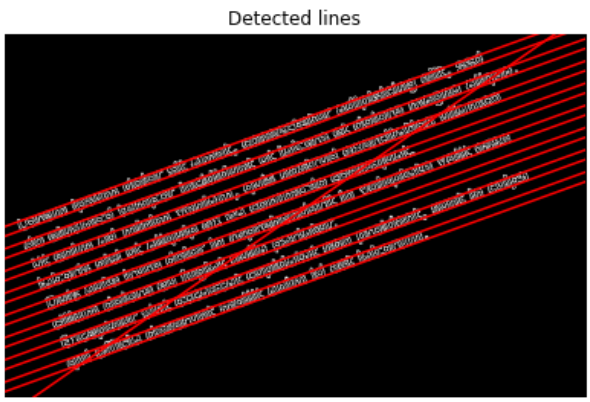
```
#import libraries
import numpy as np
from skimage.transform import hough_line, hough_line_peaks
from skimage.transform import rotate
from skimage.feature import canny
from skimage.io import imread
from skimage.color import rgb2gray
import matplotlib.pyplot as plt
```

```
from scipy.stats import mode

#Load image and conver to gray scale
image = rgb2gray(imread("/content/SkewImage2.jpeg"))

#Convert to edges
edges = canny(image)
# Classic straight-line Hough transform
tested_angles = np.deg2rad(np.arange(0.1, 180.0))
```

```
h, theta, d = hough_line(edges, theta=tested_angles)
# Generating figure 1
fig, axes = plt.subplots(1, 2, figsize=(15, 16))
ax = axes.ravel()

ax[0].imshow(image, cmap="gray")
ax[0].set_title('Input image')
ax[0].set_axis_off()

ax[1].imshow(edges, cmap="gray")
origin = np.array((0, image.shape[1]))

for _, angle, dist in zip(*hough_line_peaks(h, theta, d)):
    y0, y1 = (dist - origin * np.cos(angle)) / np.sin(angle)
    ax[1].plot(origin, (y0, y1), '-r')

ax[1].set_xlim(origin)
ax[1].set_ylim((edges.shape[0], 0))
ax[1].set_axis_off()
ax[1].set_title('Detected lines')
```

```
Text(0.5, 1.0, 'Detected lines')
```

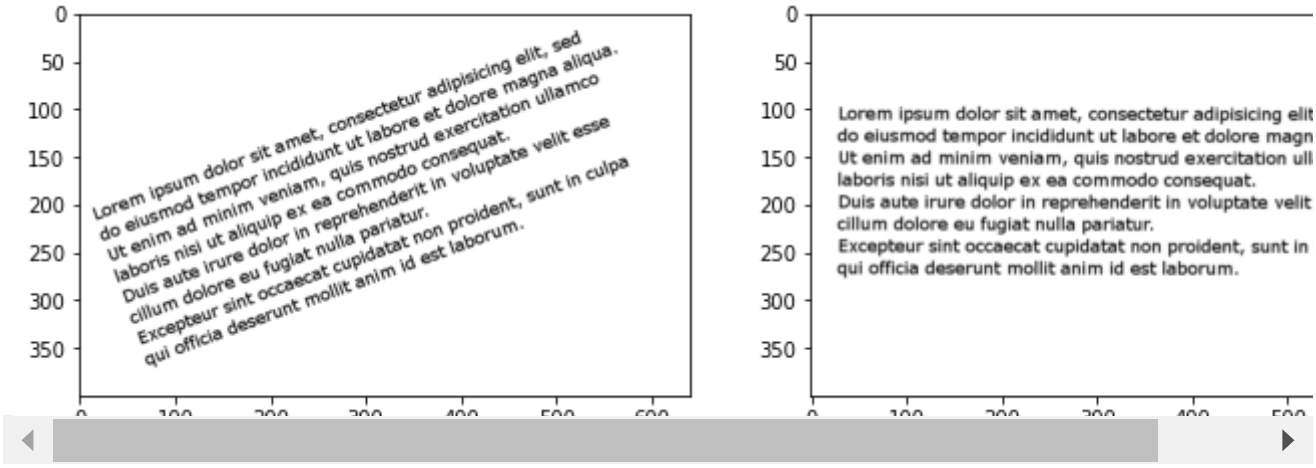





```
def skew_angle_hough_transform(image):
    # convert to edges
    edges = canny(image)
    # Classic straight-line Hough transform between 0.1 - 180 degrees.
    tested_angles = np.deg2rad(np.arange(0.1, 180.0))
    h, theta, d = hough_line(edges, theta=tested_angles)

```

```
    # find line peaks and angles
    accum, angles, dists = hough_line_peaks(h, theta, d)

    # round the angles to 2 decimal places and find the most common angle.
    most_common_angle = mode(np.around(angles, decimals=2))[0]

    # convert the angle to degree for rotation.
    skew_angle = np.rad2deg(most_common_angle - np.pi/2)
    print(skew_angle)
    return skew_angle

fig, ax = plt.subplots(ncols=2, figsize=(12,12))
ax[0].imshow(image, cmap="gray")
ax[1].imshow(rotate(image, skew_angle_hough_transform(image), cval=1), cmap="gray")
plt.show()
```

```
[-20.09914899]
```



Conclusion

1. We have implemented the code for Deskewing text with OpenCV and Python
2. Read the image and converted the image to grayscale
3. Then grabed 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
4. We rotated the image to deskew it.
5. Then we implemented the Deskewing using Hough Transform
6. We Converted the image to edges and generated Classic straight-line Hough transform
7. We found line peaks and angles
8. We found the most common angle and converted the angle to degree for rotation.
9. Thus we obtained a skew corrected image using both the methods.

Colab paid products - Cancel contracts here