Image Processing Group Project

Project Name: Skew correction of Text Image

Group

Name	Batch	Roll Number
Praval Pattam	CS-04	B220057CS
Pranav Sai Sarvepalli	CS-04	B220055CS
Potluri Theenesh	CS-04	B221121CS

Implementation

Jupyter notebooks were used for the execution and implementation of the codes.

Importing the required libraries and reading the image from the directory:

In [1]:
```
import cv2
import math
import numpy as np
from scipy import ndimage
```

In [2]:
```
# Read image from directory
image = cv2.imread('images\houghtransform.jpg')
```

Applying the pre-processing and the Probabilistic Hough line transform through the following code.

In [2]:
```
# Read image from directory
image = cv2.imread('images\houghtransform.jpg')
# image = cv2.imread('images\houghtransform_2.jpg')
img_copy = image.copy()
img_copy_1 = image.copy()
height,width = image.shape[:2]
center = (width//2, height//2)
original_image = image.copy()
# converting image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
# blur image
blur = cv2.GaussianBlur(gray, (5, 5), 0)
# threshold image
_, threshed = cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+
cv2.THRESH_OTSU)
# erode image
eroded = cv2.erode(threshed,(3,3),1)
# dilate image
dilate = cv2.dilate(eroded, (35, 35), iterations=3)
```

In [3]:
```
lines = cv2.HoughLinesP(dilate,1,np.pi/180,200,None,150,10)
```

In [5]:
```
if lines is not None:
    horizontal_lines = []
    angles = []
    for i,line  in enumerate(lines):
        x1 = line[0][0]
        y1 = line[0][1]
        x2 = line[0][2]
        y2 = line[0][3]
        cv2.line(img_copy, (x1,y1), (x2,y2), (0, 0, 255), 1, cv2.LINE_AA)
        diff_x = x2-x1
        diff_y = y2-y1
        if abs(diff_y) < 30 and abs(diff_x) != 0:
            horizontal_lines.append((x1, y1, x2, y2))
            try:
                slope = diff_y / diff_x
                angle = math.degrees(math.atan(slope))
                print(angle)
                angles.append(angle)
            except Exception as e:
                print(e)
                continue

    cv2.imshow('All Extracted Lines', img_copy)
    for line in horizontal_lines:
        # print(line)
        cv2.line(img_copy_1, (line[0],line[1]), (line[2],line[3]), (0,0,255), 1, cv2.LINE_AA)
        cv2.imshow('Formated lines',img_copy_1)
    rotation_angle = sum(angles) / len(angles)
```

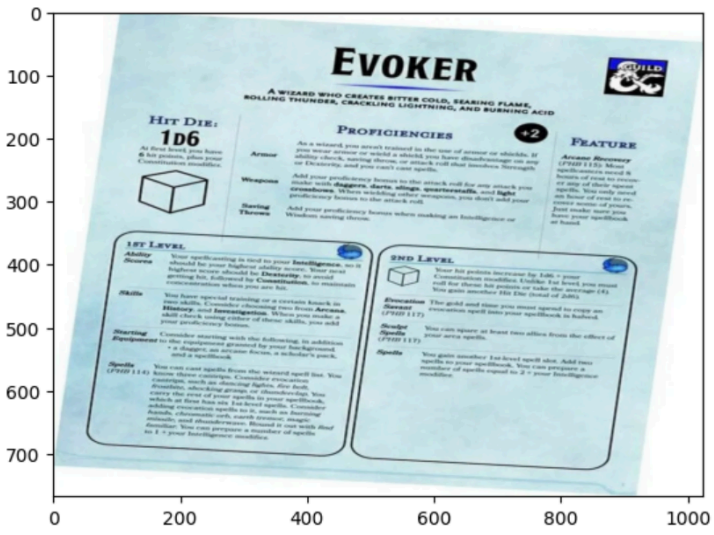
Rotation of original image to show transformation:

In [6]:
```
img_rotated = ndimage.rotate(image, rotation_angle, reshape=True)
img_rotated = img_rotated.astype(np.uint8)
```

In [7]:
```
rotation_matrix = cv2.getRotationMatrix2D(center, rotation_angle, 1)
# rotate orignal image to show transformation
rotated_image = cv2.warpAffine(original_image, rotation_matrix, (width, height), borderValue=(255, 255,
```

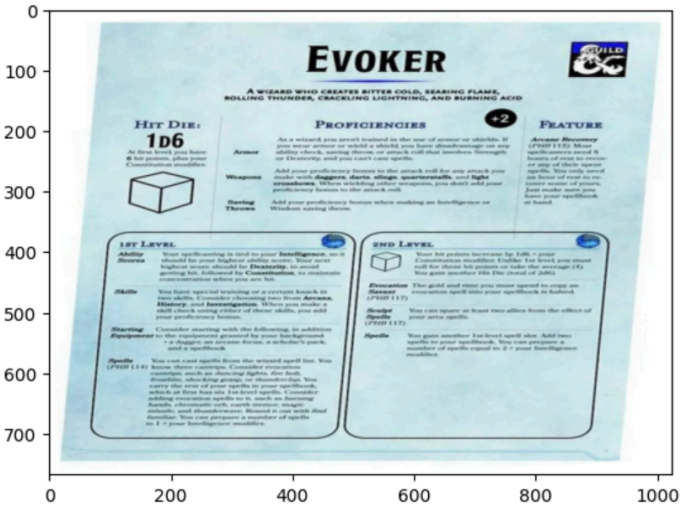
The original image:

In [8]:
```
import matplotlib.pyplot as plt
plt.imshow(image)
plt.show()
```



The corrected result image:

In [9]:
```
plt.imshow(rotated_image)
plt.show()
```

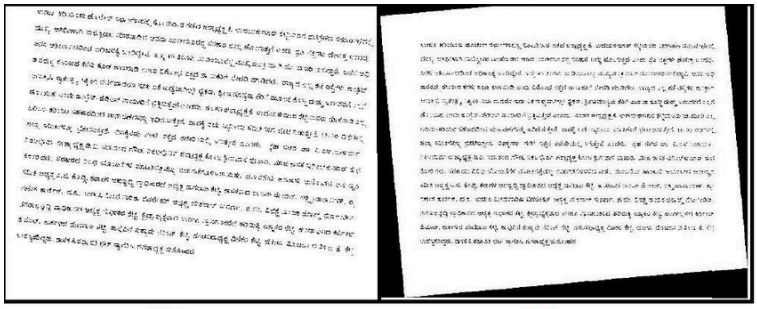


Comparison with alternate methods

1. Radon Transform:

More Complex and Computationally Intensive: Radon Transform is powerful but can be overkill for simple document skew correction, making it less efficient for this specific task.

Example of Skew correction using Radon transform.



(a) (b)

2. Projection Profile Analysis:

Quick and too Simple: Projection Profile Analysis is efficient but might struggle with documents that don’t have evenly spaced, well-defined text lines, reducing accuracy in more complex or noisy images.

Example of Skew correction using Projection profile analysis:

Conclusion:

The Hough Line Transform offers a sweet spot: It’s more robust and accurate for a variety of document types compared to Projection Profile Analysis, and less complex than Radon Transform, making it our preferred approach for skew correction of document images.