When scanning a document, a slight skew gets into the scanned image. If you are using the scanned image to extract information from it, detecting and correcting skew is crucial.

There are several techniques that are used to skew correction.

- Projection profile method
- Hough transform
- Topline method
- Scanline method

However, the projection profile method is the simplest and easiest way to determine skew in documents in the range ±5°. Let's take a part of the scanned image and see how to correct skew.

In this method, we will convert the image to black (absence of pixel) & white (presence of pixel). Now the image is projected vertically to get a histogram of pixels. Now the image is rotated at various angles and the above process is repeated. Wherever we find maximum difference between peaks, that will be the best angle.

```
def small_angle(img):

        #convert nparray data
        img = im.fromarray(img)
        image = img.convert("RGBA")


        #In this method, convert image to black (absence of pixel) & white (presence of pixel).
        wd, ht = image.size
        pix = np.array(image.convert('1').getdata(), np.uint8)
        bin_img = 1 - (pix.reshape((ht, wd)) / 255.0)

        #the image is projected vertically to get a histogram of pixels.
        def find_score(arr, angle):
        data = inter.rotate(arr, angle, reshape=False, order=0)
        hist = np.sum(data, axis=1)
        score = np.sum((hist[1:] - hist[:-1]) ** 2)
        return hist, score

        #Now the image is rotated at various angles (at a small interval of angles called Delta)
        delta = 1
        limit = 5
        #image is rotated at various angles and above process is repeated
        angles = np.arange(-limit, limit+delta, delta)
```

```
    scores = []

    # the difference between the peaks will be calculated
    #The angle at which the maximum difference between peaks (or Variance) is found,
    #that corresponding angle will be the Skew angle for the image.
    for angle in angles:
    hist, score = find_score(bin_img, angle)
    scores.append(score)

    best_score = max(scores)
    best_angle = angles[scores.index(best_score)]
    print('Best angle: '+str(best_angle))

    return (best_angle)
```

Let's go through each preprocessing technique mentioned above one-by-one

1. **Binarization:** In layman's terms Binarization means converting a coloured image into an image which consists of only black and white pixels (Black pixel value=0 and White pixel value=255). As a basic rule, this can be done by fixing a threshold (normally threshold=127, as it is exactly half of the pixel range 0–255). If the pixel value is greater than the threshold, it is considered as a white pixel, else considered as a black pixel.
   But this strategy may not always give us desired results. In the cases where lighting conditions are not uniform in the image, this method fails.

2. *Skew Correction:* While scanning a document, it might be slightly skewed (image aligned at a certain angle with horizontal) sometimes. While extracting the information from the scanned image, detecting & correcting the skew is crucial.

Several techniques are used for skew correction.

→ Projection profile method

→ Hough transformation method

→ Topline method

→ Scanline method

However, the *projection profile* method is the simplest, easiest and most widely used way to determine skew in documents.

In this method, First, we'll take the binary image, then

- project it horizontally (taking the sum of pixels along rows of the image matrix) to get a histogram of pixels along the height of the image i.e count of foreground pixels for every row.

- Now the image is rotated at various angles (at a small interval of angles called *Delta*) and the difference between the peaks will be calculated (*Variance* can also be used as one of the metrics). The angle at which the **maximum** difference between peaks (or *Variance*) is found, that corresponding angle will be the *Skew angle* for the image.
- After finding the Skew angle, we can correct the skewness by rotating the image through an angle equal to the skew angle in the ***opposite direction*** of skew.

3. *Noise Removal:* The main objective of the *Noise removal* stage is to smoothen the image by removing small dots/patches which have higher intensity than the rest of the image. Noise removal can be performed for both *Coloured* and *Binary images*.
One way of performing Noise removal is by using OpenCV *fastNlMeansDenoisingColored* function.

```
#removing dots and random patches from images
img = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 15)
```

4. Image Rotation using Pytesseract

```
def rotate_image_180(image,angle):

    #rotate by small angle
    image_center = tuple(np.array(image.shape[1::-1]) / 2)
    rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
    image = cv2.warpAffine(image, M=rot_mat, dsize=image.shape[1::-1],
            flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_CONSTANT,
             borderValue   = [255, 255, 255])

    #check if the docs needs to be rotated more than 180 degree
    #we have used pytesseract to find the rotation angle
    rot_data = pytesseract.image_to_osd(image);
    rot = re.search('(?<=Rotate: )\d+', rot_data).group(0)
    print(rot_data)
    angle = float(rot)
    print("Angle: {}".format(angle))

    #sometime we face incorrect rotation if angle is 90
    #hence we confirm again if the image is rotated well
    if angle == 90.0:
            image = imutils.rotate_bound(image, angle)
            rot_data = pytesseract.image_to_osd(image);
            rot = re.search('(?<=Rotate: )\d+', rot_data).group(0)
            angle = float(rot)
            print("Angle2: {}".format(angle))
```

```
        img_rotated = imutils.rotate_bound(image, angle)#added

        b, g, r = cv2.split(img_rotated)
        rotated2 = cv2.merge([r, g, b])

        img = im.fromarray(rotated2)
        img.save("file.jpg")
        return img
```

Sample Output from pytesseract.image_to_osd(image);

```
Page number: 0
Orientation in degrees: 270
Rotate: 90
Orientation confidence: 1.68
Script: Latin
Script confidence: 4.44
```

Code to run this program on a folder

```
import os
filenames = next(os.walk("/home/ds/Documents/cropped/"))[2]
for f in filenames:
        path="/home/ds/Documents/cit1/"+str(f)
        ## bulk alignment process goes here
```

References:

https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7
https://avilpage.com/2016/11/detect-correct-skew-images-python.html