1 document cleanup

March 1, 2021

Candidate: André Oliveira Françani

0.1 Description

Many image processing applications make use of digitalized textual data. However, the presence of any type of noise can create difficulties in post-processing information, such as on OCR detection. To improve the information manipulation on such data, a previous image processing step is required.

In light of this idea, a set of text paragraphs containing plain English language was collected. Different font styles, size, and background noise level were arranged to simulate the a variety of scenarios.

0.2 Objective

The objective of this test is to evaluate the possible image processing methods that could fix the text samples. Note that the samples have a different type of background noise and present a set of text fonts. Therefore, the candidate should provide a flexible algorithm that can correctly detect what is text characters and background noise, offering a clean version of each text paragraph as result.

0.3 Important details

- As a common pattern, the text must be represented by BLACK pixels and the background by WHITE pixels. Therefore, the output image MUST be in binary format (i.e. 0 pixel values for text and 255 pixel values for background)
- This test does not require a defined image processing algorithm to be used. The candidate is free to choose any kind of image processing pipeline to reach the best answer.
- The candidate will receive only the noisy data, as clean data is rarely provided on real-case scenarios, and no annotations are provided. Thus, creativity is needed if the candidate chooses to use supervised learning algorithms.
- The perfect correct result is reached with: 1) white background, 2) black text characters, 3) horizontal text alignment, 4) text block centered in the image and 5) straight text font (not itallic formatting).
- Do not change the filename when applying your image processing methods. The filename is important for data comparison purposes.
- The output image file will be only accepted on the following formats: .png, .tif, .jpg

0.4 Implementation

Since we have different background noise, the main idea is to remove this noise using image processing techniques. The first thing I will do is explore the data and try some of these techniques.

```
[1]: #import libraries
     import numpy as np
     from PIL import Image
     import matplotlib.pyplot as plt
     import os
     import glob
     from scipy import signal
     from skimage.transform import probabilistic_hough_line
     from skimage.feature import canny
     import cv2
     ### Functions ###
     def load_image(image_path):
         111
         This function loads and normalizes the image given by the 'image_path'.
         return np.asarray(Image.open(image_path))/255.0
     def remove_background(image):
         This function removes the background noise of the image by applying a_{\sqcup}
      \hookrightarrow Median Filter, and returns
         a binary denoised image.
         111
         #Background image
         background_image = signal.medfilt2d(image, kernel_size=13)
         #Foreground mask
         mask = image < background_image - 0.2</pre>
         #make binary image
         denoised_image = np.where(mask, image, 1.0)
         return denoised_image
     def image_cleanup(filename):
         111
         This function reads a image given by the path 'filename', removes its \sqcup
      \hookrightarrow background noise and
         corrects its skew, making an horizontal text alignment.
         111
```

```
#read and filter background noise
   image = load_image(filename)
   denoised_img = remove_background(image)
   #Canny edges detector
   edges = canny(denoised_img)
   #Hough lines
   hough_lines = probabilistic_hough_line(edges)
   #slopes of the line segments
   slopes = [(y2 - y1)/(x2 - x1)] if (x2-x1) else 0 for (x1,y1), (x2, y2) in
→hough_lines]
   #the angle, in degrees, in a circle by which the line is offset
   angles = [np.degrees(np.arctan(theta)) for theta in slopes]
   #most common angle
   hist, bins = np.histogram(angles, bins=180)
   max_idx = np.argmax(hist)
   rotate_angle = bins[max_idx]
   #angle correction for the rotation
   if rotate_angle > 45:
       rotate_angle = -(90-rotate_angle)
   elif rotate_angle < -45:</pre>
       rotate_angle = 90 - abs(rotate_angle)
   #image dimensions
   (h, w) = denoised_img.shape[:2]
   center = (w // 2, h // 2)
   #affine matrix of 2D rotation
   M = cv2.getRotationMatrix2D(center, rotate_angle, 1.0)
   rotated_image = cv2.warpAffine(denoised_img.astype('uint8'), M, (w, h), u
→flags=cv2.INTER_CUBIC,borderMode=cv2.BORDER_REPLICATE)
   return rotated_image
```

```
[2]: #list all images in "noisy_data" folder
images_path = sorted(glob.glob('noisy_data/*.png'))
```

Let's visualize three images:

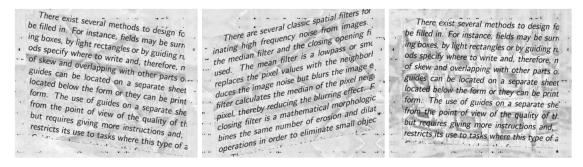
```
[3]: #loading 3 images
img1 = load_image(images_path[0])
img2 = load_image(images_path[1])
```

```
img3 = load_image(images_path[2])

#plot images
plt.figure(figsize=(20,20))
plt.subplot(1,3,1)
plt.imshow(img1, 'gray')
plt.axis('off')

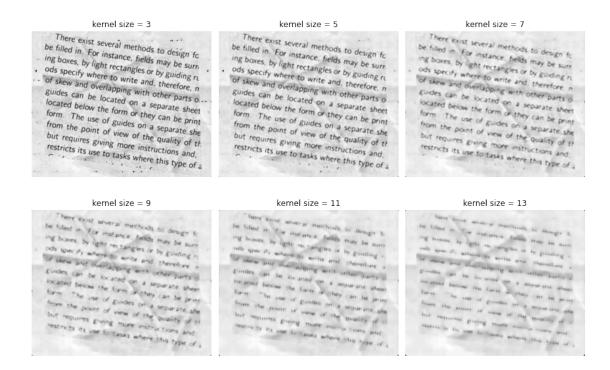
plt.subplot(1,3,2)
plt.imshow(img2, 'gray')
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(img3, 'gray')
plt.axis('off')
plt.tight_layout()
```



Now I will use a Median Filter to estimate the background of the image 1. The first parameter that I will explore is the kernel_size of the filter.

```
[4]: #background image using a median filter
kernel_size = [3, 5, 7, 9, 11, 13]
plt.figure(figsize=(12,8))
for i in range(len(kernel_size)):
    background_img1 = signal.medfilt2d(img1, kernel_size[i])
    plt.subplot(2,3,i+1)
    plt.imshow(background_img1, 'gray')
    plt.axis('off');
    plt.title('kernel size = '+ str(kernel_size[i]))
plt.tight_layout()
```

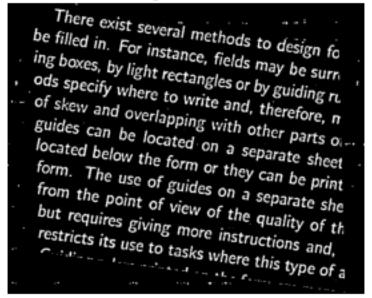


From the images above, it can be seen that a kernel (13,13) is good enough to extract the background information without losing significant text information. The next step is to compute the foreground mask, that is, everythig that is significantly darker than the background.

```
[5]: #Foreground mask
mask = img1 < background_img1 - 0.2

plt.imshow(mask, 'gray')
plt.axis('off');
plt.title('Foreground Mask');</pre>
```

Foreground Mask



Now in the previous image, the black color is a boolean False and the white, True. Therefore I will invert the colors just by making it binary as the following:

```
[6]: denoised_img1 = np.where(mask, img1, 1.0)

plt.figure(figsize=(12,12))
plt.subplot(1,2,1)
plt.imshow(img1, 'gray')
plt.axis('off');
plt.title('Original Image');

plt.subplot(1,2,2)
plt.imshow(denoised_img1, 'gray')
plt.axis('off');
plt.title('Denoised Image');
```

Original Image

There exist several methods to design for be filled in. For instance, fields may be surning boxes, by light rectangles or by guiding runds specify where to write and, therefore, not skew and overlapping with other parts of skew and overlapping with other parts of skew and overlapping with other parts of suides can be located on a separate sheet located below the form or they can be print from the point of view of the quality of the but requires giving more instructions and, restricts its use to tasks where this type of a

Denoised Image

There exist several methods to design for be filled in. For instance, fields may be surrounded be filled in. For instance, fields may be surrounded by light rectangles or by guiding recodes specify where to write and, therefore, not skew and overlapping with other parts of skew and overlapping with other parts of suides can be located on a separate sheet located below the form or they can be print from. The use of guides on a separate sheet form the point of view of the quality of the but requires giving more instructions and, restricts its use to tasks where this type of a

```
[7]: #Display 10 images
plt.figure(figsize=(16,16))
for i in range(1,11):
    img = load_image(images_path[i])
    denoised_img = remove_background(img)
    plt.subplot(2,5,i)
    plt.imshow(denoised_img, 'gray')
    plt.axis('off');
    image_number = images_path[i].split('\\')[1].split('.')[0]
    plt.title('image '+ image_number)
    plt.tight_layout()
```

There are several classic spatial filters for instead by the frequency noise from images, the state of the classic spatial filters for the median filter and the classing special filters. The mean state of the classic spatial filters with the state of t

image 104

There exist several methods to design for being from the contraction of several movement of the contract below the form of the point of several movement of several movement

image 105

mating high frequent classic spatial filters for the many high frequent for the classic spatial filters and the classic spatial filters and the classic spatial filters as observed that the classic spatial filters as observed the many classic spatial filters as observed the many classic spatial filters are classic spatial filters and the many classic spatial filters are classic spatial filters and the many classic spatial filters are many classic spatial filters and classic spatial filters are many classic spatial filters are spatial fil

image 10

Then exit several methods to design 6 be filled in. For instance, fields may be surring beact, by "gint regions exit policy where to write and, therefore a filled with the property where to write and, therefore a few and overlapping with other parts see the second of the second policy where the property with other parts see the second policy with the property with the property of the second policy with the sec

image 108

image 11

There exist served inclines to design forms with fields to folia to folia to get accurated by locality loses. In field to the field to the control of the field to the field to the control of the control of the field to the fie

image 110

There exist several methods to design be filled in. For instance, fields may be solve filled in. For instance, fields may be good to be filled in. For instance, field may be good to several filled f

image 111

inter are several casaic splatal inters so: landing high frequency noise from images. the median filter and the closing opening file seed. The mean filter is a lowpsy-or sem replaces the pixel values with the neighbort duces the image noise but blust the image efficer calculates the median of the pixel neighpoxt, thereby reducing the bluring effect. Fi closing filter is a mathematical merphologic bines the same number of erosion and dilat operations in order to eliminate small objectimage 113

There exist several methods to design for be filled in. For instance, fields my being fing boxes, by gift rectangles or by guiding, and ods specify where to write and, before, or slew and overlapping with other parts of guides can be located on a separate of total below the form or they can be print form. The use of guides on a separate of the from the point of view of the quality side. The print of the print of the print of the but requires giving more instructions and, sortics its use to tasks where the top of a image 114

There are several classic spatial filters for inating high frequency nose from images the median filter and the closing opening 6₂ used. The mean filter is a loopass or smerplaces the piesel values with the neighborh duces the image noise but blars the image of siter calculates the median of the pied neith, pied, thereby reducing the burning effect. Following filter is a mathematical morphologic bines the same number of erosion and district operations in order to eliminate small object.

Assuming that the background is removed, the next step is the horizontal text alignment.

```
[9]: #read and filter background noise
filename = images_path[0]
image = load_image(filename)
denoised_img = remove_background(image)
denoised_image[denoised_img<0] = 0.

#Canny edges detector
edges = canny(denoised_img)

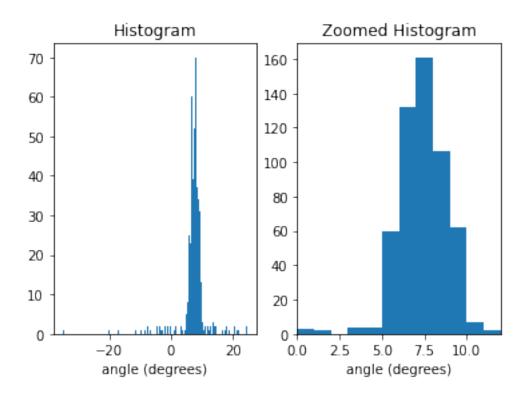
plt.imshow(edges,'gray')
plt.axis('off');</pre>
```

```
There exist several methods to design to be filled in. For instance, fields may be surning boxes, by light rectangles or by guiding rules of specify where to write and, therefore, not skew and overlapping with other parts of guides can be located on a separate sheet form. The use of guides on a separate sheet from the point of view of the quality of the restricts its use to tasks where this type of a
```

be falled in. For instance, fields may be sum, ing bases, by light rectangle or by grading nods specify where to write and, therefore, no of she and explaining with other parts of the fall of the fa

```
[11]: #slopes of the line segments
      slopes = [(y2 - y1)/(x2 - x1)] if (x2-x1) else 0 for (x1,y1), (x2, y2) in
      →hough_lines]
      #the angle, in degrees, in a circle by which the line is offset
      angles = [np.degrees(np.arctan(theta)) for theta in slopes]
      #which of these degree values is most common?
      #histogram of the angles
      plt.subplot(1,2,1)
      plt.hist(angles, bins=180)
      plt.title('Histogram')
      plt.xlabel('angle (degrees)')
     plt.subplot(1,2,2)
      plt.hist(angles, bins=[x for x in range(360)])
      plt.xlim([0,12])
      plt.title('Zoomed Histogram')
     plt.xlabel('angle (degrees)')
      #most common angle
     hist, bins = np.histogram(angles, bins=180)
      max_idx = np.argmax(hist)
      rotate_angle = bins[max_idx]
     print('Most common angle: ', rotate_angle)
```

Most common angle: 7.708511579515523



It can be seen that the most frequent angle for this image sample is around 7 degrees. This means that we can rotate the image by this angle to correct the text skew.

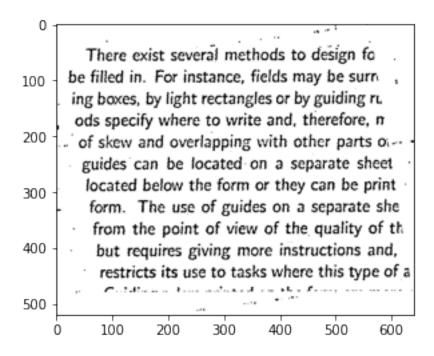
```
[12]: #angle correction for the rotation
if rotate_angle > 45:
    rotate_angle = -(90-rotate_angle)
elif rotate_angle < -45:
    rotate_angle = 90 - abs(rotate_angle)

#image dimensions
(h, w) = denoised_img.shape[:2]
center = (w // 2, h // 2)

#affine matrix of 2D rotation
M = cv2.getRotationMatrix2D(center, rotate_angle, 1.0)
rotated_image = cv2.warpAffine(denoised_img.astype('uint8'), M, (w, h), U
--flags=cv2.INTER_CUBIC,borderMode=cv2.BORDER_REPLICATE)

plt.imshow(rotated_image,'gray')</pre>
```

[12]: <matplotlib.image.AxesImage at 0x21e969b2c08>



Now, I will apply a final function called image_cleaunp that receives a path to the image, reads and removes its background and correct the skew of the image.

Notes: I was not able to straight the text font (not itallic formatting). I tried quickly to apply an unwrap method but had no success. However, comparing the obtained cleaned documents with the original given ones, it can be seen that the cleaned versions are much better than the dirty ones.