

Signature Detection and Removal

Veer Singh

August 5, 2021

1 Introduction

This algorithm is able to detect signatures from scanned documents and remove it. It is based on Connected-component labelling. This was adapted from [1].

Connected components labelling scans an image and groups its pixels into components based on pixel connectivity, i.e. all pixels in a connected component share similar pixel intensity values and are in some way connected with each other.

2 Input

Read the image as a NumPy ndarray in grayscale mode and apply thresholding to binarize the image.

```
img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)[1]
```

Image without binarization will have too many colors and the algorithm will fail to group them. Now the pixels are either 0(black) or 255(white).



Figure 1: (a) Original Image (b) Image with Thresholding

3 Connected Component Analysis with scikit-learn Framework

First we will set all white pixels to True(1) and all black pixels to False(0)

```
blobs = img > 254
```

Next we will use `skimage.measure.label` which will label all the connected pixels. We set `background=1` which means it will assume the pixels which are 1 (or True) is the background and will not take them into consideration when labelling the connected pixels and set them to 0.

```
blobs_labels = measure.label(blobs, background=1,)
```

We can visualize the different components using `skimage.color.label2rgb()`. This will assign different colors to the different components. We can set `bg_label=0` which means the pixels with value 0 will not be coloured.

```
image_label_overlay = label2rgb(blobs_labels, bg_label=0)
```

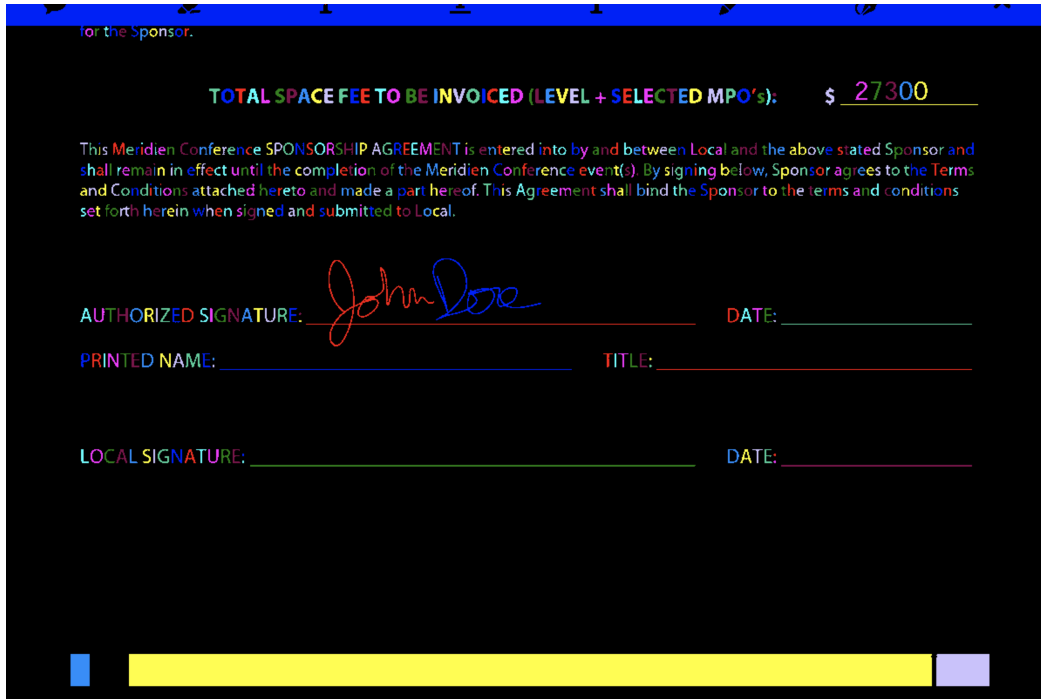


Figure 2: Visualization of Connected Pixel Regions

Now we need to identify the connected pixels regions which form the signature, so we need to remove the other regions. This involves removing the smaller connected regions which form the printed text and the much larger connected regions which form things like this bottom banner in the image and the underlines.

To find smaller regions: A.Özlü has defined some constants in [1] based on experimental evidence with A4 size scans. We use them with the average region area (eliminating super small regions) to get a threshold value. Any connected regions with area smaller than this are not considered as a potential signature.

```
small_parameter_1 = 84
small_parameter_2 = 250
small_parameter_3 = 100

total_area = 0
count = 0
for region in regionprops(blobs_labels):
    # To remove really small regions
    if region.area > 10:
        total_area = total_area + region.area
        count = count + 1
average = (total_area / count)

a4_small_size_threshold = ((average/small_parameter_1)*small_parameter_2) + small_parameter_3
```

To find larger regions: A.Özlü has defined a constant in [1] based on experimental evidence with A4 size scans. We use it with the small region threshold calculated earlier to get a threshold value. Any connected regions with area larger than this are not considered as a potential signature.

```
large_parameter_1 = 18

a4_large_size_threshold = a4_small_size_threshold * large_parameter_1
```

Now we need to remove these regions.

To remove the small regions we will use `skimage.morphology.remove_small_objects()` function which takes in the labelled image and the small threshold area and outputs the labelled image with the small connected components removed.

```
signature = morphology.remove_small_objects(blobs_labels, a4_small_size_threshold)
```

To remove the larger regions we use the following logic to set the connected regions with larger area than the large threshold to 0.

```
component_sizes = np.bincount(signature.ravel())
```

```

too_large = component_sizes > a4_large_size_threshold
too_large_mask = too_large[signature]
# Sets all large outliers to 0
signature[too_large_mask] = 0

```

We can now compare the total number of connected regions first identified versus the total number of connected regions which fall in our range:

```

total_connected_regions = len(regionprops(blobs_labels))
print(f'Number of total connected regions --> {total_connected_regions}')

connected_regions_in_given_range = len(regionprops(signature))
print(f'Number of connected regions in the given range--> {connected_regions_in_given_range}')

#####OUTPUT#####
Number of total connected regions --> 530
Number of connected regions in the given range--> 5
#####OUTPUT#####

```

4 Output

The `detect_signature()` method gives just the signature and the `get_image_without_signature()` method gives the image without the signature.

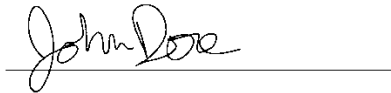


Figure 3: Extracted Signature



for the Sponsor.

TOTAL SPACE FEE TO BE INVOICED (LEVEL + SELECTED MPO's): \$ 27300

This Meridien Conference SPONSORSHIP AGREEMENT is entered into by and between Local and the above stated Sponsor and shall remain in effect until the completion of the Meridien Conference event(s). By signing below, Sponsor agrees to the Terms and Conditions attached hereto and made a part hereof. This Agreement shall bind the Sponsor to the terms and conditions set forth herein when signed and submitted to Local.

AUTHORIZED SIGNATURE: _____ DATE: _____

PRINTED NAME: _____ TITLE: _____

LOCAL SIGNATURE: _____ DATE: _____



Figure 4: Image with Signature Removed

References

- [1] A. Özlü, "Overlapped handwritten signature extraction from scanned documents," 2018. Available at https://github.com/ahmetozlu/signature_extractor.