

# Segmentation of human fingernails

Halil Suheyb Becerek

December 2020

## 1 Task Description

Our task in this project is to write an algorithm that allows the segmentation of human fingernail region without using any kind of deep learning. Task seems trivial at first however within the data set it turned out to be a non-trivial task

## 2 Literature Survey

During the literature reading done for this project I found few GitHub repositories[1] with the same goal however all most all of them used deep learning and in the forums most suggested thing was to use Haar cascades which is a form of Machine Learning.

In the paper "Characterization of Human Fingernails Using Iterative Thresholding Segmentation"[2] researchers claimed to achieve 80% accuracy in their data set using certain image processing methods their methodology involved shortly in their words: "Preliminary methods include intensity adjustment to reduce noise, contrast enhancement for edge detection, and morphological operation to improve finger region from noisy background. In stage two, iterative histogram-based thresholding of multispectral image (R, G, and B components) to binarize fingernail region from finger object is adapted." [2] Hence in the algorithm to be designed steps that are useful for our data set were applied.

## 3 Algorithm Description

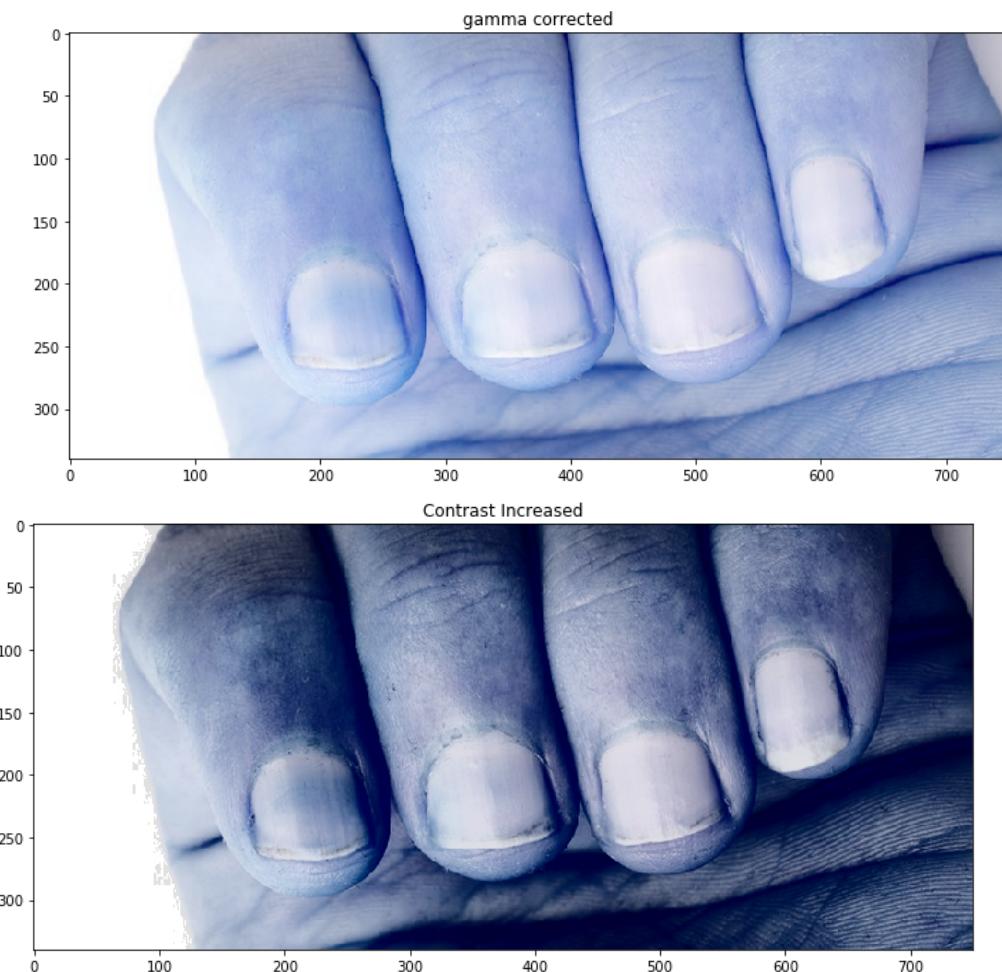
In our data set there are quite a variety between images and in all kinds of parameters hence first we need to try to normalize our images using image processing techniques.

First we apply gamma correction to the image to increase overall intensity in the image which helps to realize fingernails that have been under shadow then an increase in contrast was applied to enhance the detection of Proximal Nail Fold.

---

```
gammaCorrected = adjust_gamma(image,0.6)
eq_gammaCorrected = equalizeHistogram(gammaCorrected)
```

---



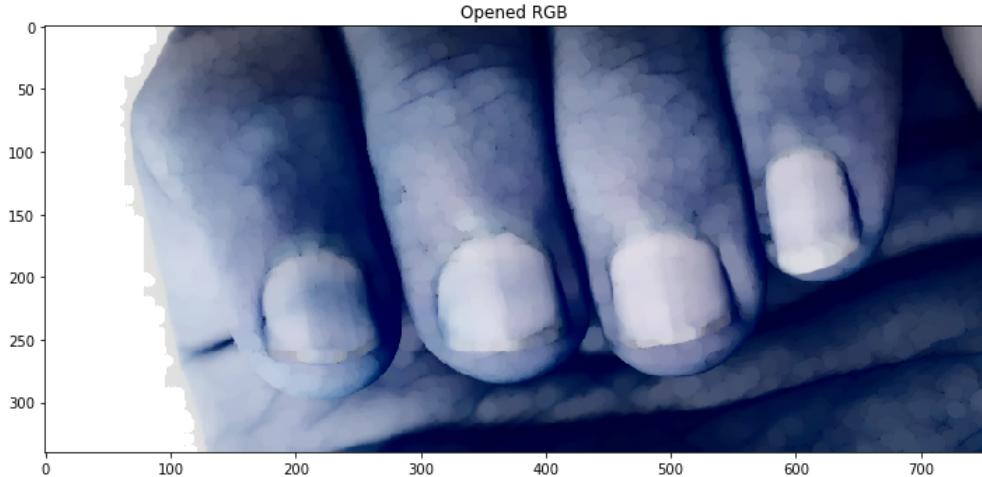
Then in-order to loose the background and pop the foreground we do morphological opening which is erosion followed by dilation with an ellipse kernel of size (11,11).

---

```
#Opening on original image
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))
eroded = cv2.erode(eq_gammaCorrected, kernel, iterations = 1)
#display(eroded, 'eroded')
dilated = cv2.dilate(eroded, kernel, iterations = 1)
```

---

below is the opened three channel image the title says RGB even though it is BGR image, don't let it trick you RGB was written to show there are 3 channels in the image.

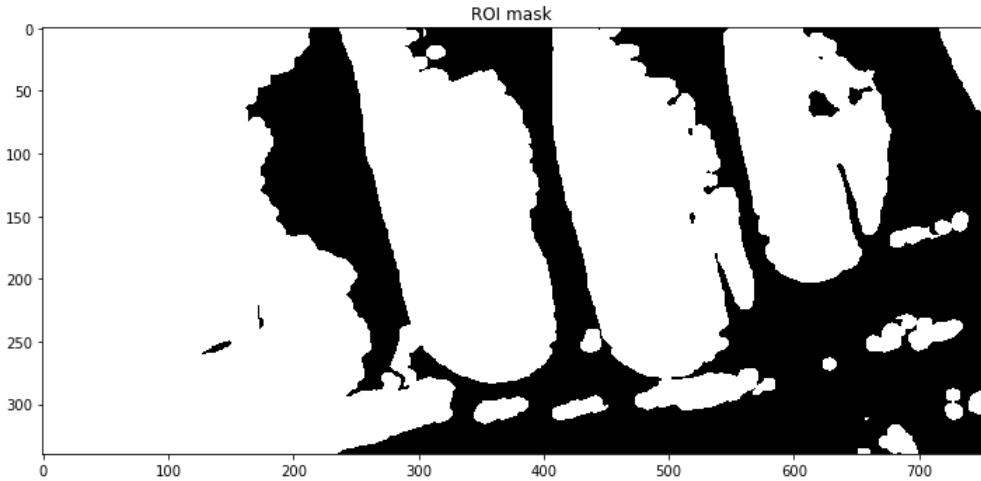


In the next step we calculate our mask to get rid of most of the background. During calculation of maskROI OTSU thresholding was used this method's automatic decision of threshold value made it easier through the data set.

---

```
hsv = cv2.cvtColor(dilated, cv2.COLOR_BGR2HSV)
saturation = hsv[:, :, 1]
#display(saturation, 'saturation')
blurred = cv2.medianBlur(saturation, 3)
ret, maskROI = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY_INV +
    cv2.THRESH_OTSU)
```

---

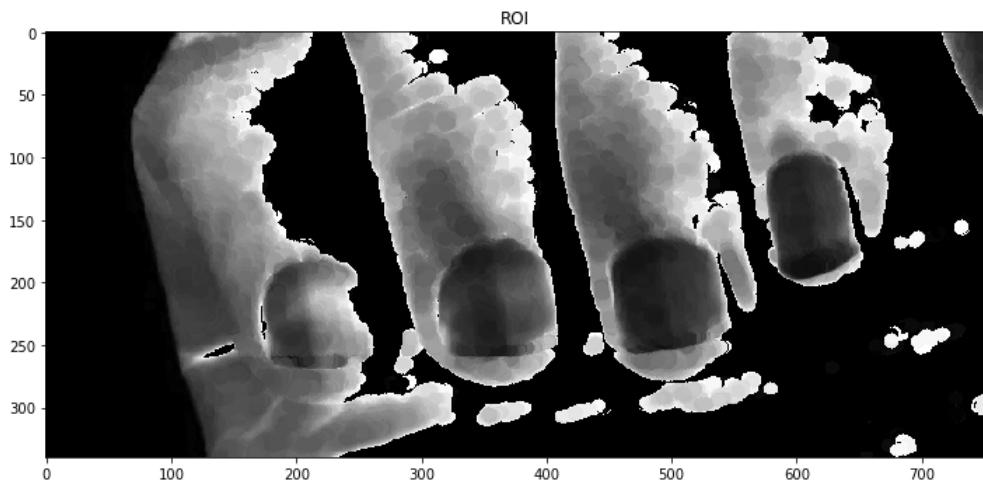


This calculated mask is applied to the saturation channel we were working on

---

```
ROI = cv2.bitwise_and(saturation,maskROI)*2
```

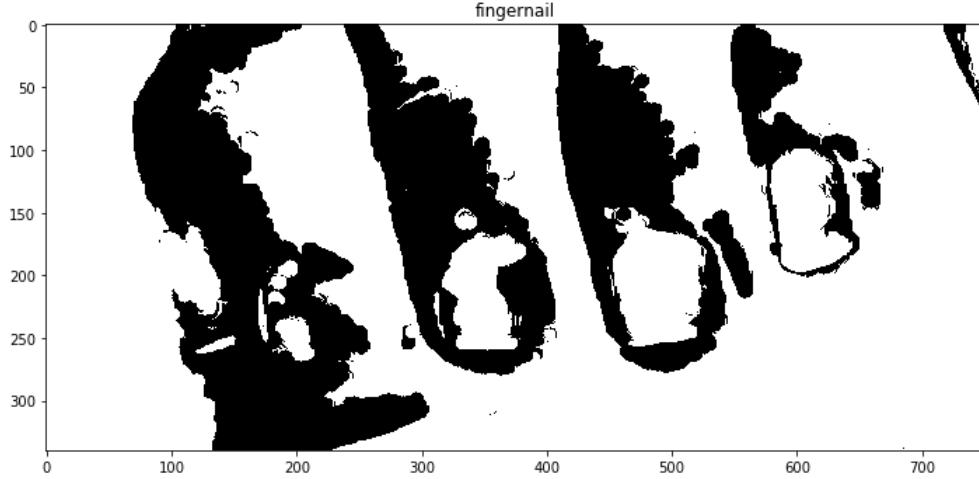
---



In our data set overall ROI calculations are accurate most of the time and even for the cases that fingernails were under shadows, since we did our masking in saturation channel in the images that has no nail polish or there are not significant difference in saturation between fingernail and finger failed.

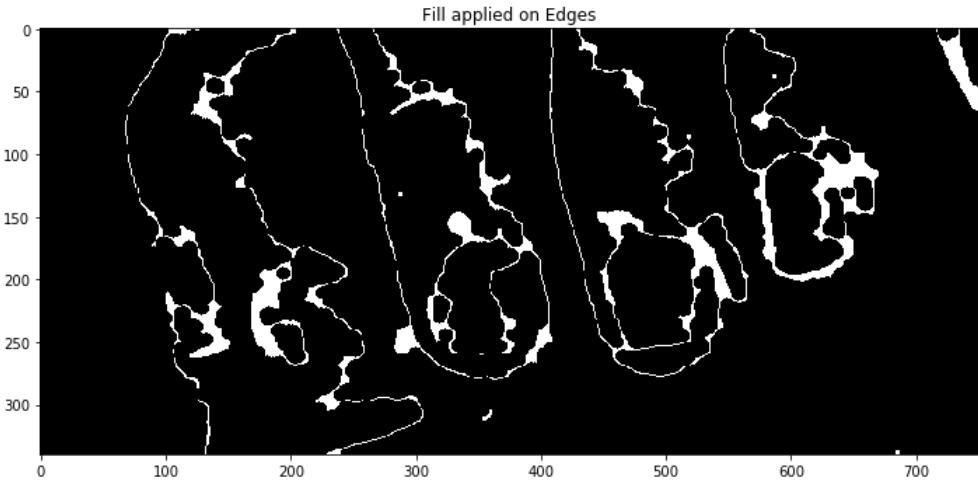
Given that fingernails usually have the high or low saturation values compared to fingers we use inRange function from cv2 to extract more of the finger nail

```
fingernail =  
cv2.bitwise_or(cv2.inRange(ROI,0,60),cv2.inRange(ROI,185,255))
```



On the latest result a Canny edge detection is run and the edges are closed with morphological operation using the kernel that was used before

```
med = cv2.mean(fingernail)[0]  
#print(med)  
lower = int(max(0,0.7*med))  
upper = int(min(255,1.3*med))  
edges = cv2.Canny(fingernail,lower-75,upper+75)  
display(edges,'Edges')  
  
filledEdges =  
cv2.morphologyEx(edges,cv2.MORPH_CLOSE,kernel,iterations = 1)  
display(filledEdges,'Fill applied on Edges')
```



After edge detection a contour detection is applied and fingernails are detected to be the internal contours

---

```

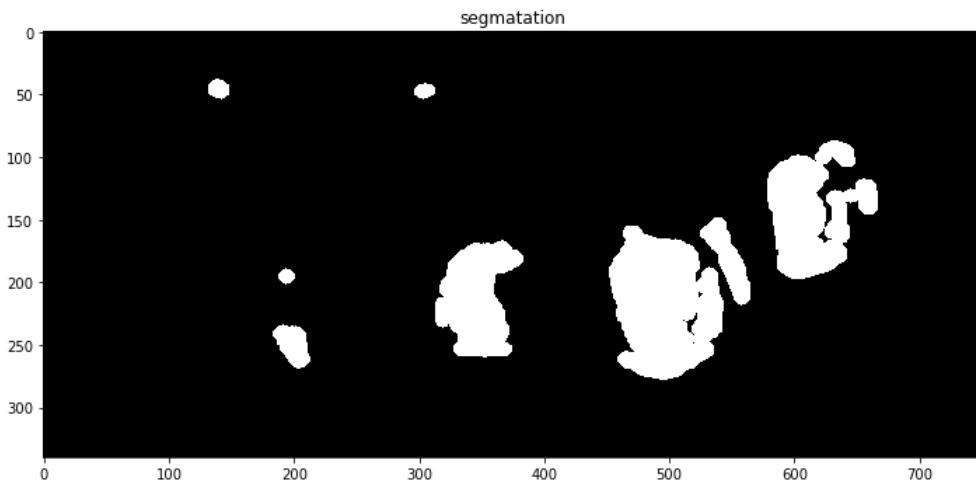
image, contours, hierarchy = cv2.findContours(filledEdges,
cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)

# Create empty array to hold internal contours
image_internal = np.zeros(image.shape)

# # Iterate through list of contour arrays
for i in range(len(contours)):
    if hierarchy[0][i][3] != -1:
        cv2.drawContours(image_internal, contours, i, 255, -1)

```

---



## 4 Ground Truth Assesment

with the manually selected data set we calculate the IoU score using below function

---

```
ratios = []
def IoU(calculated):
    mypath='labels'
    onlyfiles = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
    images = np.empty(len(onlyfiles), dtype=object)
    for n in range(0, len(onlyfiles)):
        images[n] = cv2.imread( join(mypath,onlyfiles[n]),0)
    for i in range(len(images)):
        #display(calculated[i],'Calculated')
        #display(images[i],'Label')
        ratio =
            np.sum(np.logical_and(images[i],calculatedSegmentation[i]))/np.sum(np.logical_or(images[i],
            ratios.append(ratio)
    print(np.average(ratios)*100 )
    print(np.max(ratios)*100)
    print(np.min(ratios)*100)
```

---

Calling the function with our calculated segmentations

---

```
IoU(calculatedSegmentation)
```

---

Gives values 17.40% for average, 76.35% for maximum and 0% for minimum even though average values is drastically low our maximum values shows that when the input images are entered with a certain assumption the average value increases to 30-40%. As mentioned before In the paper "Characterization of Human Fingernails Using Iterative Thresholding Segmentation" researchers were only able to achieve 80% accuracy with more uniform data set given that with such varying data set 17% could be considered a good enough segmentation.

## 5 Other Suggested Solutions

### 5.1 Deep Learning

Easiest and most painless method to automate such task would be with training a model since fingernails come in different size, shape, length and color. Similar to face detection Haar cascades could be used.

### 5.2 Other Image Processing Approaches

If we think more about image processing approaches we could improve on the Region of Interest we already calculated using the suggested algorithm using Blob detection on full color scale image with Region of Interest since all the

pixels belonging to fingernail will come out into the same blob this might help but again comes the question of how to mark the regions which actually belong to fingernail since fingernails come in various sizes in our data set

Another approach would be to use watershed algorithm on the ROI(in full scale) directly or on a gradient calculated from the Region of Interest to form the edges.

These suggested methods were tried however there were no observations that they gave better results hence finalization of the algorithm was done using the method suggested in the algorithm description section.

Note: Who's thesis work is this good luck to them

## **References**

- [1] V.Papenko - Segmentation of human fingernails
- [2] Characterization of Human Fingernails Using Iterative Thresholding Segmentation - N.S. Kumuda, M.S.Dinesh, G.Hemantha Kumar