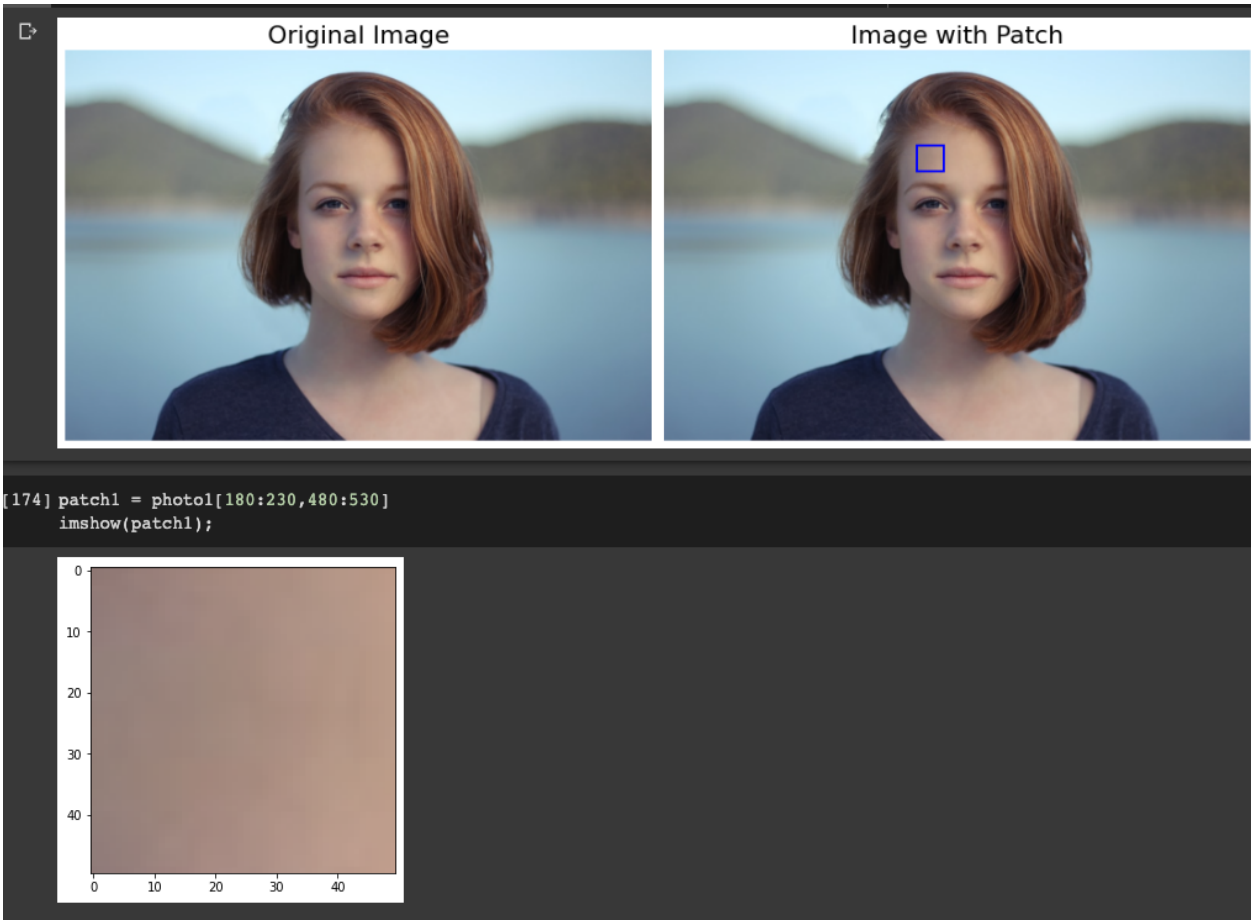
Skin Color Detection with RG Chromaticity

Pan Wang

Methods

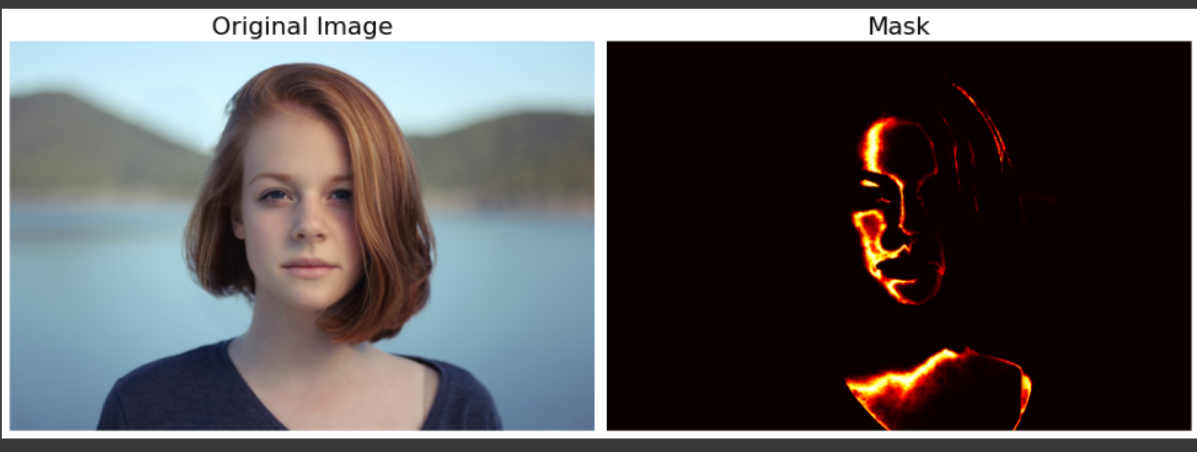
To develop the algorithm that can detect skin color for people with a wide variety of skin tones, we need to collect more skin samples so that we can detect more skin colors, and I used RG Chromaticity

1. First, we need to get a patch of skin color from the image for later skin detection



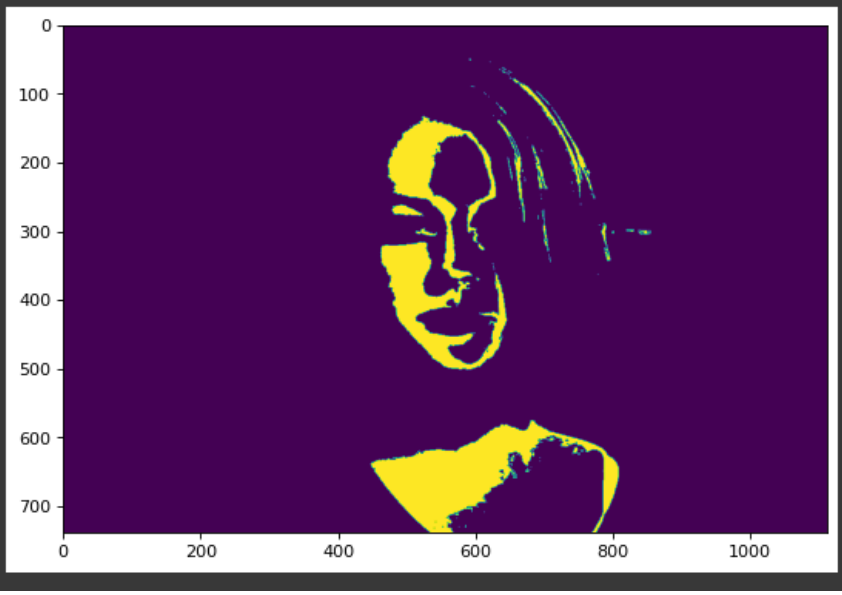
2. Then, We'll make a mask that takes the patch's characteristics as input to generate a Gaussian distribution. This mask will determine the location of the patch's pixels in the original image.

```python
def gaussian(p,mean,std):
    return np.exp(-(p-mean)**2/(2*std**2))*(1/(std*((2*np.pi)**0.5)))
def rg_chroma_patch(image, patch, mean = 1, std = 1):
    image_r = image[:,:,0] /image.sum(axis=2)
    image_g = image[:,:,1] /image.sum(axis=2)

    patch_r = patch[:,:,0] / patch.sum(axis=2)
    patch_g = patch[:,:,1] / patch.sum(axis=2)

    std_patch_r = np.std(patch_r.flatten())
    mean_patch_r = np.mean(patch_r.flatten())
    std_patch_g = np.std(patch_g.flatten())
    mean_patch_g = np.mean(patch_g.flatten())
    masked_image_r = gaussian(image_r, mean_patch_r, std_patch_r)
    masked_image_g = gaussian(image_g, mean_patch_g, std_patch_g)
    final_mask = masked_image_r * masked_image_g
    fig, ax = plt.subplots(1,2, figsize=(15,7))
    ax[0].imshow(image)
    ax[0].set_title('Original Image', fontsize = 22)
    ax[0].set_axis_off()

    #clean the mask using area_opening
    ax[1].imshow(final_mask, cmap = 'hot');
    ax[1].set_title('Mask', fontsize = 22)
    ax[1].set_axis_off()
    fig.tight_layout()

    return final_mask
final_mask1 = rg_chroma_patch(photo1, patch1)
```



3. Then, we can binarize our mask for a better result later

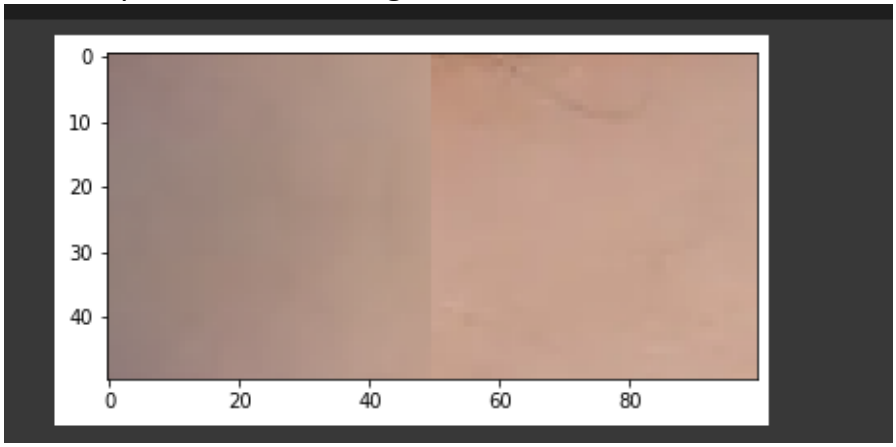
```
[176] binarized_mask1 = final_mask1 > final_mask1.mean()
      plt.figure(num=None, figsize=(8, 6), dpi=80)
      plt.imshow(binarized_mask1)
      plt.show()
```



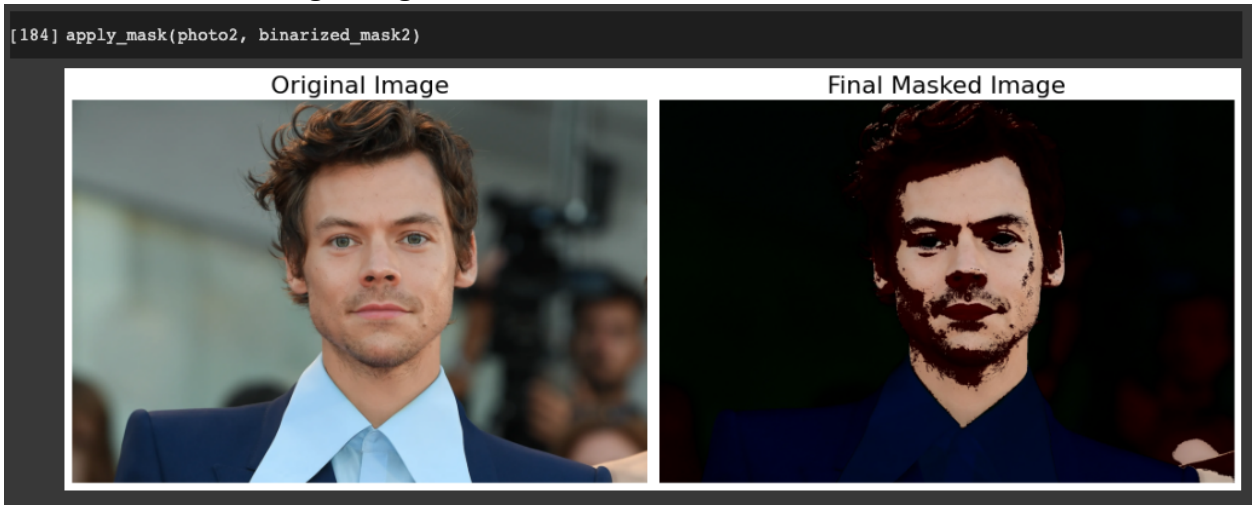
4. With the final binarized mask, we can finally apply it to the original photo

```
def apply_mask(image,mask):
    yuv_image = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)
    yuv_image[:,:,0] = yuv_image[:,:,0] * mask

    masked_image = cv2.cvtColor(yuv_image, cv2.COLOR_YUV2RGB)

    fig, ax = plt.subplots(1,2, figsize=(15,7))
    ax[0].imshow(image)
    ax[0].set_title('Original Image', fontsize = 22)
    ax[0].set_axis_off()

    ax[1].imshow(masked_image);
    ax[1].set_title('Final Masked Image', fontsize = 22)
    ax[1].set_axis_off()
    fig.tight_layout()
apply_mask(photo1, binarized_mask1)
```

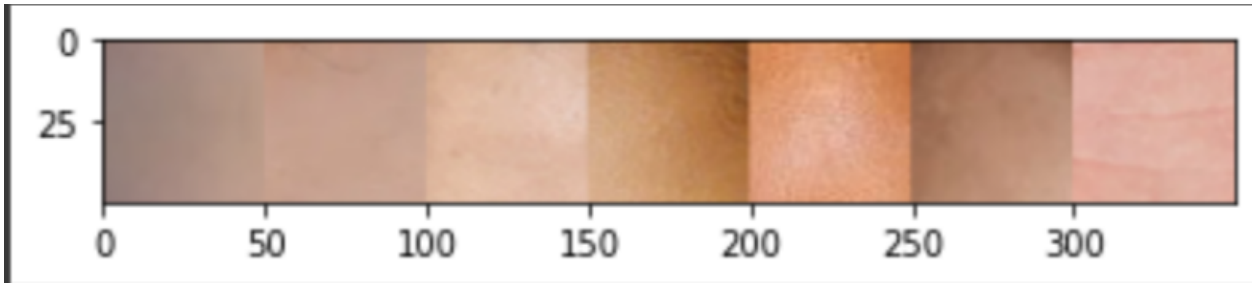
5. For the 2nd photo, the only different is that we combine the new patch and the patch of the first together to have a larger patch set so that we can accept more color ranges



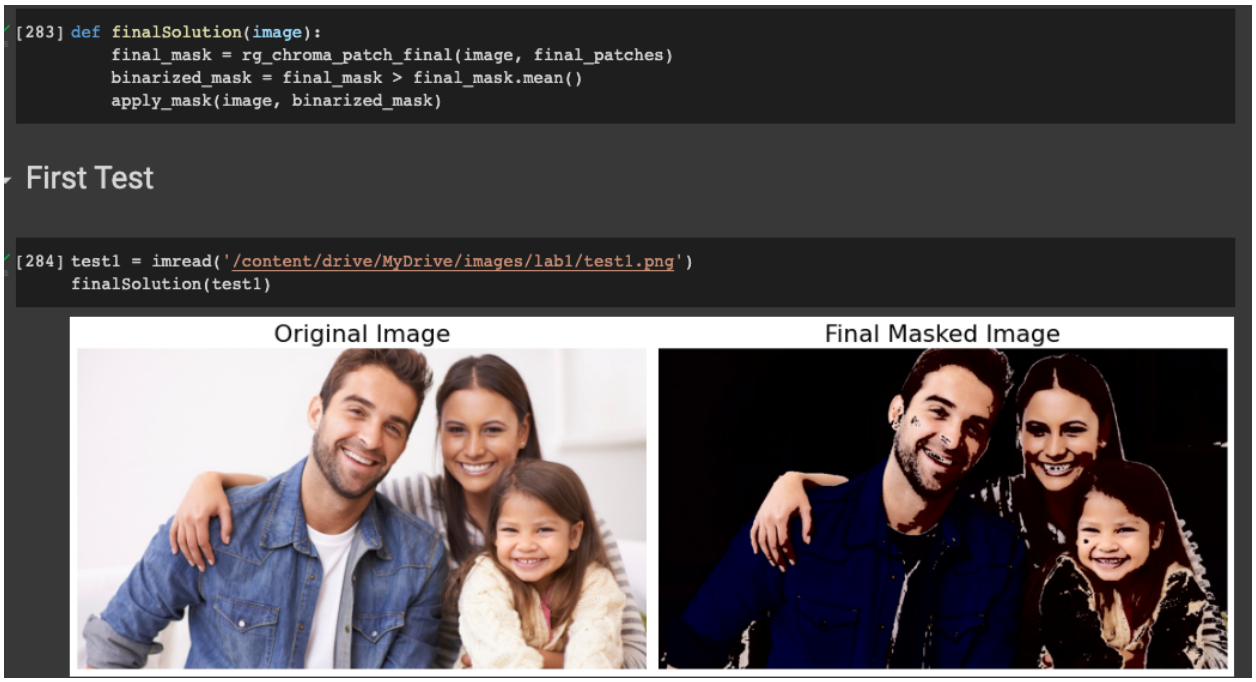
And the result is getting better



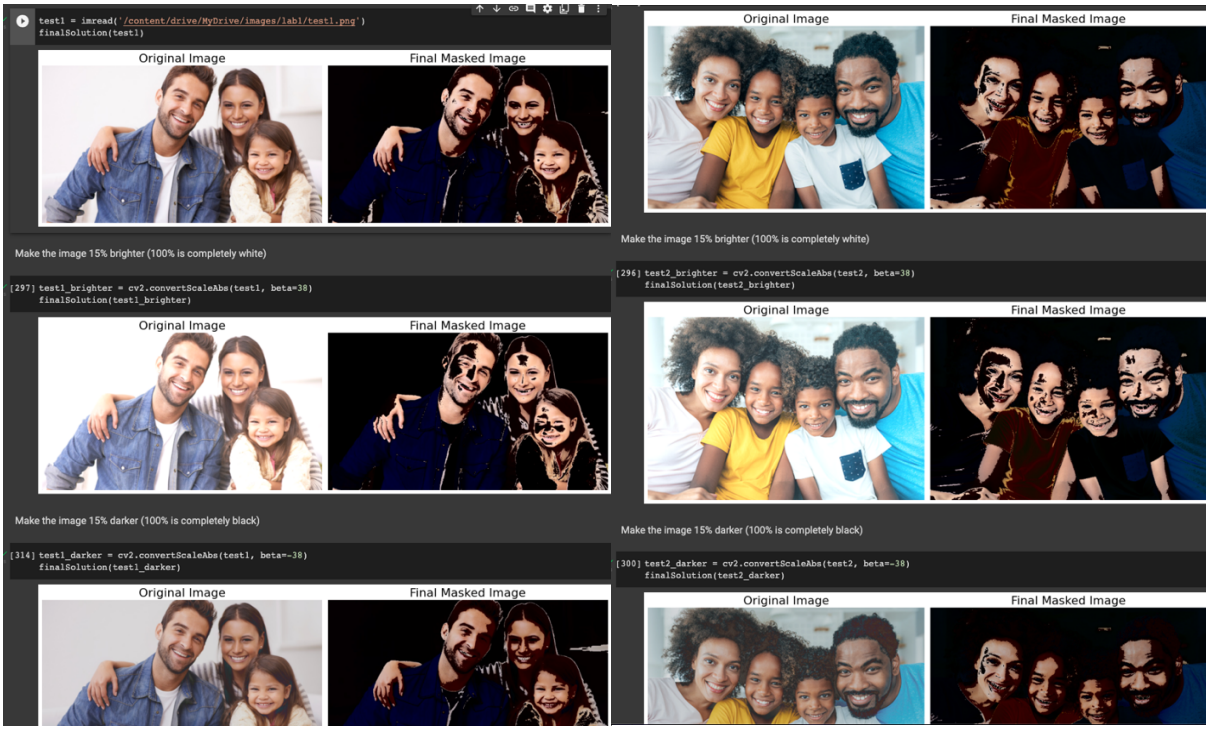
6. We do the same procedures for all the 7 photos as a dataset, and got the final patches, which accepts 7 photos

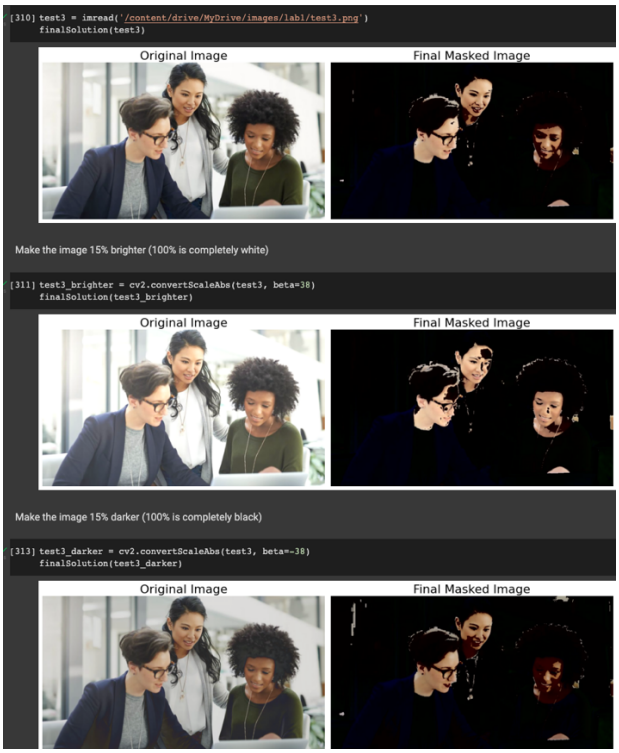


7. With the final patches, we can get the final algorithm that can directly detect the skin color



Results and Discussion





As we can see, the skin detector can accurately detect most of the skin even when there are a group of people, and even the people have various colors of skin. And if we make the image brighter or darker, the result will become a little not that accurate due to lack of dataset for more scenarios.

One limitation of our method is that I didn't use background distribution to get rid of the background, sometimes the detector will accidentally detect the background that is similar to skin color

Conclusion

In order to develop a more accurate skin color detector, we used 7 photos of people which include people with a variety of skin colors, and we can find that that the result is becoming more and more accurate with the collection of more patches. And finally, we can even test our algorithm with photos of a group of people with a variety of skin colors. The result may be less accurate with the change of brightness, in the future we can find more methods or train more datasets in order to make our detector more accurate.

References

[1] Tonichi Edeza. Image Processing with Python — Using RG Chromaticity. https://towardsdatascience.com/image-processing-with-python-using-rg-chromaticity-c585e7905818