Q2> **skin detection algorithm using Python and image processing techniques.**
Colab Link: https://colab.research.google.com/drive/1BsoYLq0-9R3sgse7eUhLi0Gc1B7MNTBi?usp=sharing

Introduction:

Skin detection is central to the foundation of computer vision and is important in many applications, from biomet ric authentication to image enhancement. The main purpose of skin detection is to detect and identify areas of h uman skin in images, thereby extracting important features and attributes for further analysis and management.

procedure of detection:

Installing Important Library: cv2

Open-Source Computer Vision Library, which is widely used for image recognition or identification.

The RGB image value is converted to HSV as well as YCbCr value, the HSV and YCbCr value of each pixel is compared to the standard values of a skin pixel and the decision is made whether the pixel is a skin pixel or not depending on whether the values lie in a range of predefined threshold values for each parameter.

The ranges for a skin pixel used in this algorithm are as follows:

0<=H<=17 and 15<=S<=170 and 0<=V<=255
And
0<=Y<=255 and 135<=Cr<=180 and 85<=Cb<=135

Functions used in code:

The preprocess_image function takes an image path as input and reads the image using OpenCV's imread function. It then converts the image to two different color spaces: HSV (Hue, Saturation, Value) and YCrCb (Luminance, Chrominance). This results in a list of three images: the original image, the HSV color space image, and the YCrCb color space image

The create_skin_mask function generates a skin mask using the cv2.inRange function, which filters out pixels within a specified color range. The lower and upper range values are defined based on the respective color space (HSV or YCrCb) and empirically determined thresholds. Additionally, the cv2.morphologyEx function is applied to the mask to perform morphological opening, which helps to remove noise and small artifacts from the mask.

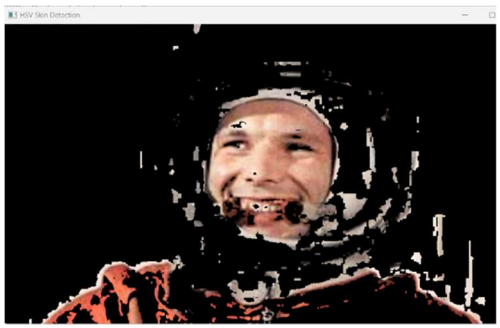
The combine_masks function combines the skin masks generated from the HSV and YCrCb color spaces. It uses the cv2.addWeighted function to create a weighted combination of the two masks. The resulting combined mask is further refined using Gaussian blurring and erosion.

The apply_skin_detection function applies the skin detection mask to the original image. It uses NumPy's array operations to selectively extract the skin regions from the original image based on the mask values.

The main function is the entry point of the script. It processes the image using the defined functions, including preprocessing the image in different color spaces, creating skin masks, combining masks, and applying skin detection. The processed results for each step are displayed using OpenCV's imshow function. The loop at the end of the function listens for the 'q' key to be pressed to close the displayed images and exit the script.

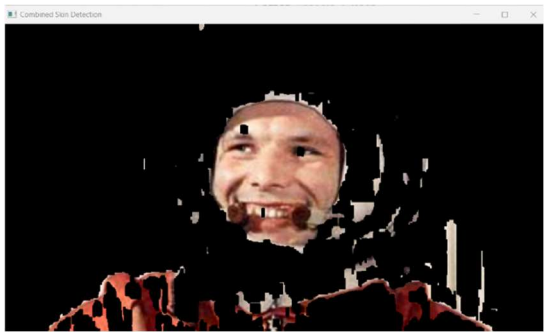
Output Visualization:

Original Image

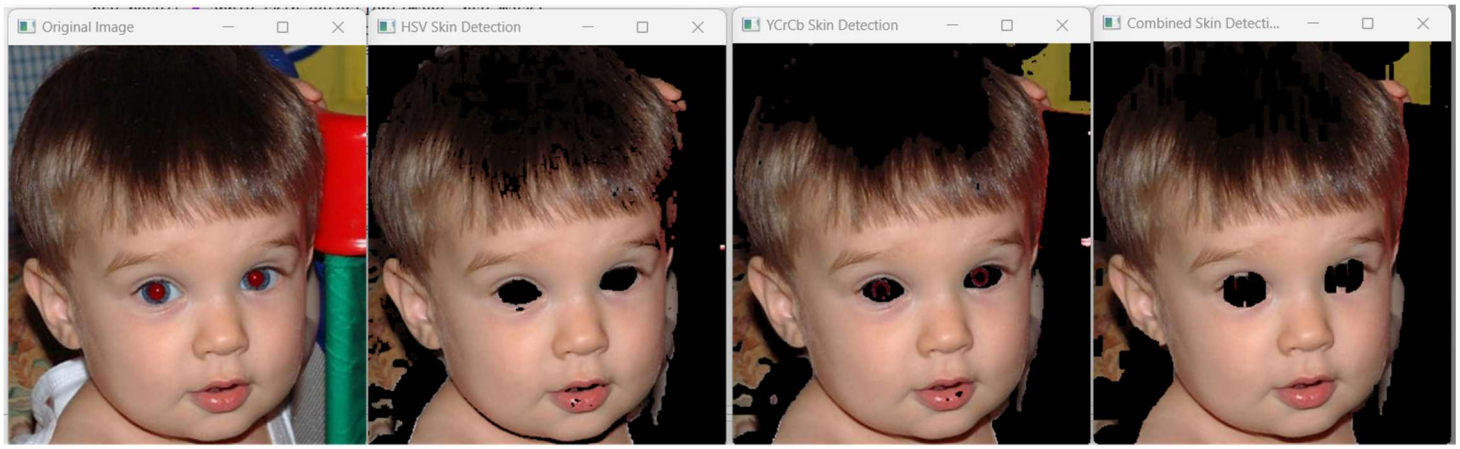


HSV Skin Detection

YCrCb Skin Detection



Combined Skin Detection



Reference: skindetection/README.md at main · noorkhokhar99/skindetection · GitHub

2. https://www.codeproject.com/Tips/5320792/Akinator-Vision

3. https://github.com/CHEREF-Mehdi/SkinDetection/blob/master/SkinDetection.py

4. https://www.codespeedy.com/skin-detection-using-opencv-in-python/#:~:text=Python%20Program%20For%20Skin%20Detection&text=Numpy%20used%20for%20mathematical%20operation,our%20maxRange%20and%20minRange%20variable.

Q1> **Develop a Python program that detects red-eye in images.**

Colab Link:
https://colab.research.google.com/drive/1kxfcz3dM_PD9jO_DE7LG8ae_odK62Ilq?usp=sharing

A "haarcascade classifier" is an effective machine learning based approach for object detection. To train a haarcascade classifier for eye detection, the algorithm initially needs a lot of positive images (images of eyes) and negative images (images without eyes). Then the classifier is trained from these positive and negative images. It is then used to detect eyes in other images. We can use already trained haarcascades for eye detection.

For eye detection in the input image, we need two haarcascades one for face detection and other for eye detection.

- haarcascade_frontalface_alt.xml
- haarcascade_eye_tree_eyeglasses.xml

haarcascades link: https://github.com/opencv/opencv/tree/master/data/haarcascades

Image Loading:
The input image is loaded using the OpenCV library. It's then converted from the standard BGR color space to grayscale.

Face Detection:

Utilizing the HaarCascade classifier for frontal face detection, the algorithm searches for potential face regions in the grayscale image.

Eye Detection:

Within each detected face region, the algorithm employs a separate HaarCascade classifier designed for eye detection. This further narrows down the analysis to identify potential eye regions within the previously detected faces.

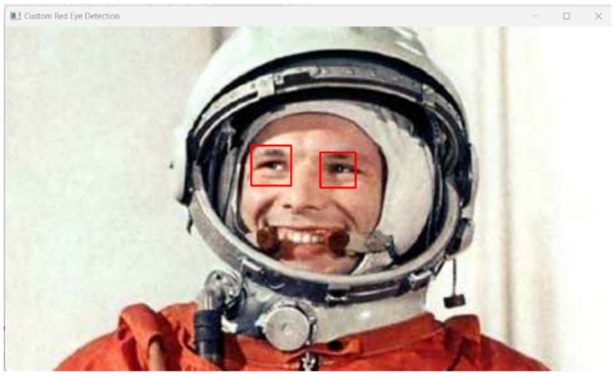
Red Color Filtering:

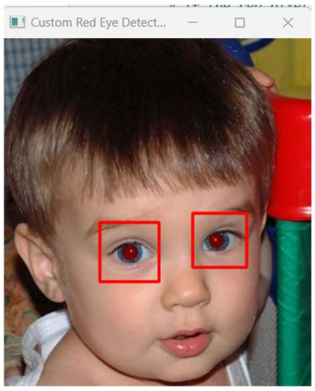
For each detected eye region, the algorithm converts it to the HSV (Hue, Saturation, Value) color space. A mask is then created to isolate pixels falling within a predefined range of red hues. The algorithm calculates the proportion of red pixels in the eye region.

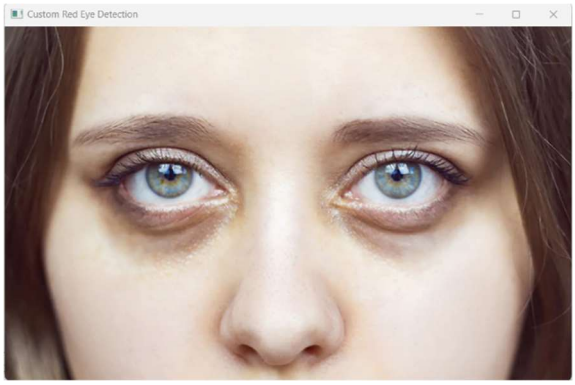
Red-Eye Identification:

If the calculated percentage of red pixels within an eye region surpasses a set threshold (2% in this case), the algorithm identifies it as exhibiting red-eye. Consequently, a red rectangle is drawn around the region to highlight the presence of red-eye.

output:







Red Eyes **Red Eyes** **No Red Eyes**

The algorithm draws red rectangles around the identified red eyes and generates a final output image. This image is displayed using the OpenCV function imshow.

References:

1. https://reintech.io/blog/how-to-create-a-face-detection-system-with-opencv-tutorial
2. https://www.tutorialspoint.com/how-to-detect-eyes-in-an-image-using-opencv-python
3. https://stackoverflow.com/questions/59026026/how-to-get-separate-face-from-image
4. https://www.researchgate.net/publication/237415739_Eye_Detection_Using_Morphological_and_Color_Image_Processing
5. https://www.hindawi.com/journals/acisc/2018/1439312/
6. https://aanilkayy.medium.com/extract-eyes-in-face-i%CC%87mages-with-dlib-and-face-landmark-points-c45ef480c1