**COMSATS University Islamabad, Lahore Campus**

**Fall 2024 – Assignment 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Course Title: | Computer Vision | Course Code: | | CSC455 | Credit Hours: | 3(3,0) |
| Course Instructor/s: | Dr. Zulfiqar Habib, Professor | Programme Name: | | BS Computer Science | | |
| Topic | Segmentation  Section: | | | Max Marks: | 10 | |
| Out Date: | 12-10-24 | **Due Date:** | | | **16-10-24** | |
| Student’s Name: | **SHAFIN-UZ-ZAMAN** | Reg. No. | SP22-BCS-063 | | | |
| **Instructions:**   1. You may use AI tools to help understand the concepts. 2. However, the answers must be in your own words and show your understanding of the topics. 3. Copying answers directly from any source, including AI tools, is not allowed. 4. Your assignment will be evaluated & graded through a leading Quiz   **Submission Guidelines:**  Submit your assignment on this sheet via Google Classroom. | | | | | | |

**Problem:**

The goal of this assignment is to understand and implement skin segmentation techniques by using a personal image. Capture a picture of yourself with at least your hands clearly visible. Save the image as yourname\_skin\_image.jpg. You will use the YCbCr color space, which separates the luminance (Y) from the chrominance (Cb and Cr). Your task is to identify and segment the skin pixels from the image using predefined thresholds for the Cr and Cb channels.

**Program Code**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Step 1: Load the image

img = cv2.imread("shafin\_skinimage.png")

# Step 2: Convert the image from BGR to YCbCr color space

img\_YCrCb = cv2.cvtColor(img, cv2.COLOR\_BGR2YCrCb)

# Step 3: Define the skin color range in YCbCr color space

# Cr: 135-180, Cb: 85-135

lower\_skin = np.array([0, 135, 85], dtype=np.uint8)  # Lower bound for skin color

upper\_skin = np.array([255, 180, 135], dtype=np.uint8)  # Upper bound for skin color

# Step 4: Create a binary mask for skin colors

skin\_mask = cv2.inRange(img\_YCrCb, lower\_skin, upper\_skin)

# Step 5: Apply morphological operations to reduce noise

# Applying morphological opening to remove small noise in the mask

kernel = np.ones((3, 3), np.uint8)

skin\_mask = cv2.morphologyEx(skin\_mask, cv2.MORPH\_OPEN, kernel)

# Step 6: Apply morphological closing to fill small holes

skin\_mask = cv2.morphologyEx(skin\_mask, cv2.MORPH\_CLOSE, kernel)

# Step 7: Segment the skin area from the original image using the mask

skin\_segmented = cv2.bitwise\_and(img, img, mask=skin\_mask)

# Step 8: Convert the original BGR image to RGB for displaying with Matplotlib

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

img\_YCrCb\_rgb = cv2.cvtColor(img\_YCrCb, cv2.COLOR\_YCrCb2RGB)  # Convert YCbCr to RGB

skin\_segmented\_rgb = cv2.cvtColor(skin\_segmented, cv2.COLOR\_BGR2RGB)

# Step 9: Create subplots to display the original image, YCbCr image, mask, and segmented skin image

fig, axs = plt.subplots(1, 4, figsize=(20, 5))

# Step 10: Display the original image

axs[0].imshow(img\_rgb)

axs[0].set\_title("Original Image")

axs[0].axis('off')  # Hide the axes for a cleaner look

# Step 11: Display the YCbCr image

axs[1].imshow(img\_YCrCb\_rgb)

axs[1].set\_title("YCbCr Image")

axs[1].axis('off')

# Step 12: Display the skin mask result

axs[2].imshow(skin\_mask, cmap='gray')  # Use a grayscale colormap

axs[2].set\_title("Skin Mask")

axs[2].axis('off')

# Step 13: Display the segmented skin image

axs[3].imshow(skin\_segmented\_rgb)

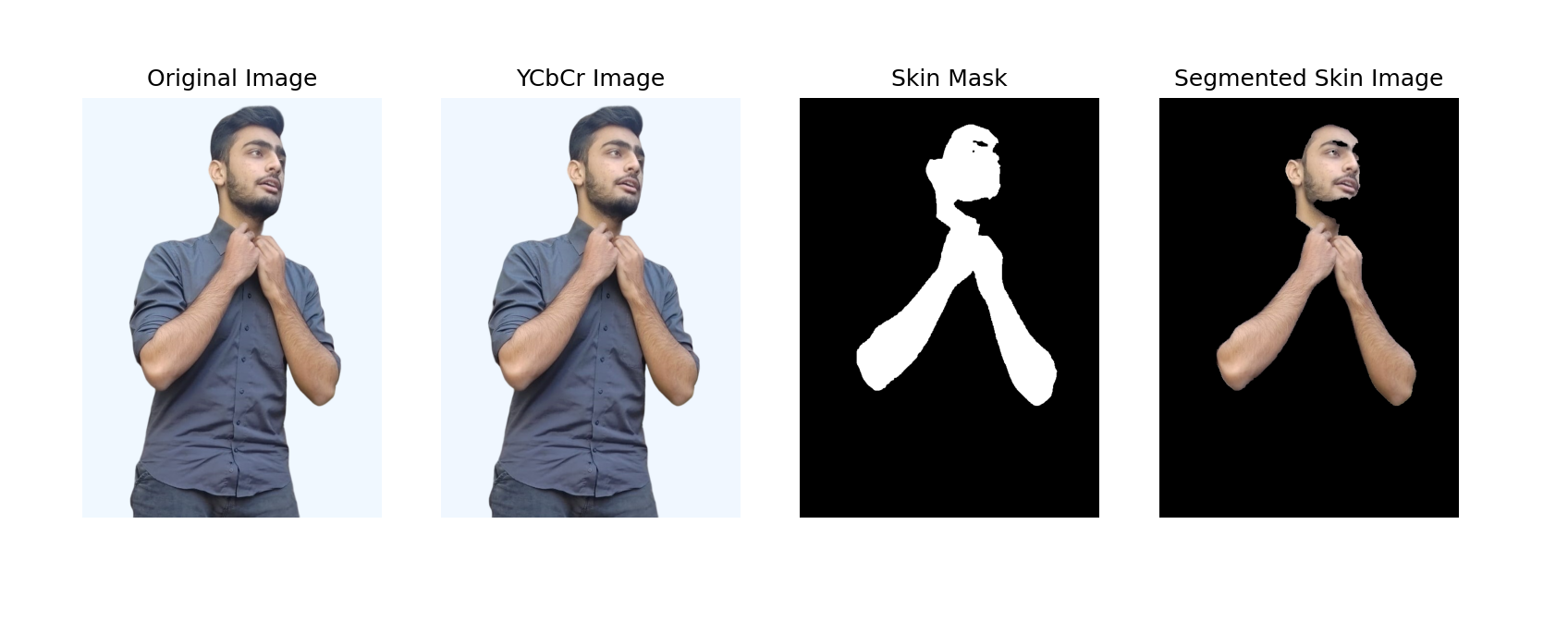
axs[3].set\_title("Segmented Skin Image")

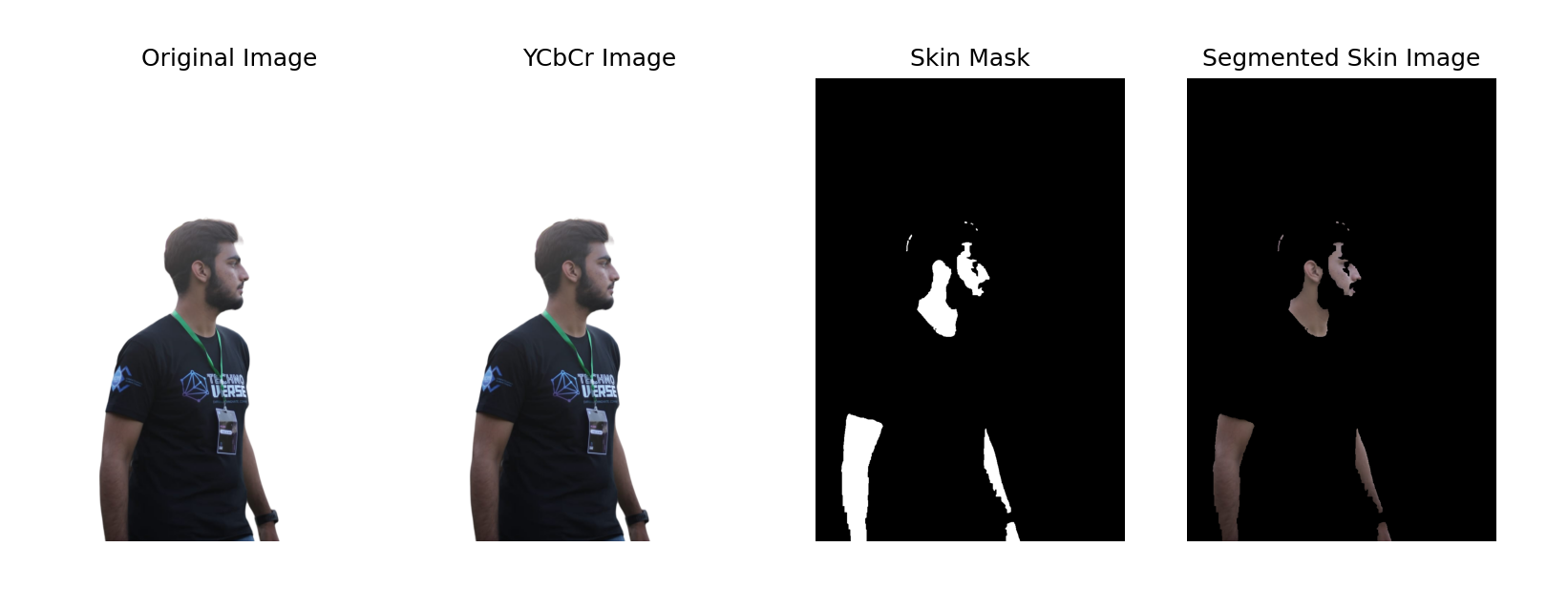
axs[3].axis('off')

# Step 14: Show all plots

plt.show()

**Output**

****

****