AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
International Credit Hours Programs (ICHEP)
Mechatronics and Automation

Machine Vision (CSE480)

Lab 3 Report

Name	ID	Section
Mahmoud Elsayd Abdelqader Labib Eldwakhly	21P0017	1

Submitted to Dr Hossam Hassan & Eng. Dina Zakaria

Fall 2025

Python Code

```python
# ================================
# TASK 1
# ================================




import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# ============================================
# 1. Load image
# ============================================
img = cv2.imread("/content/Task1.jpg")   # <-- change file name
if img is None:
    raise ValueError("Image not found!")

print("Original Image:")
cv2_imshow(img)

# ============================================
# 2. Convert to grayscale
# ============================================
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print("Grayscale Image:")
cv2_imshow(gray)

# ============================================
# 3. Histogram Equalization (Contrast Enhancement)
# ============================================
equalized = cv2.equalizeHist(gray)
print("After Histogram Equalization:")
cv2_imshow(equalized)

# ============================================
# 4. Edge Detection — Laplacian
# ============================================
laplacian = cv2.Laplacian(equalized, cv2.CV_64F)
laplacian = cv2.convertScaleAbs(laplacian)  # convert float → uint8

print("Laplacian Edges:")
cv2_imshow(laplacian)

# ============================================
# 5. Edge Detection — Canny
# ============================================
canny = cv2.Canny(equalized, 50, 150)

print("Canny Edges:")
cv2_imshow(canny)

```

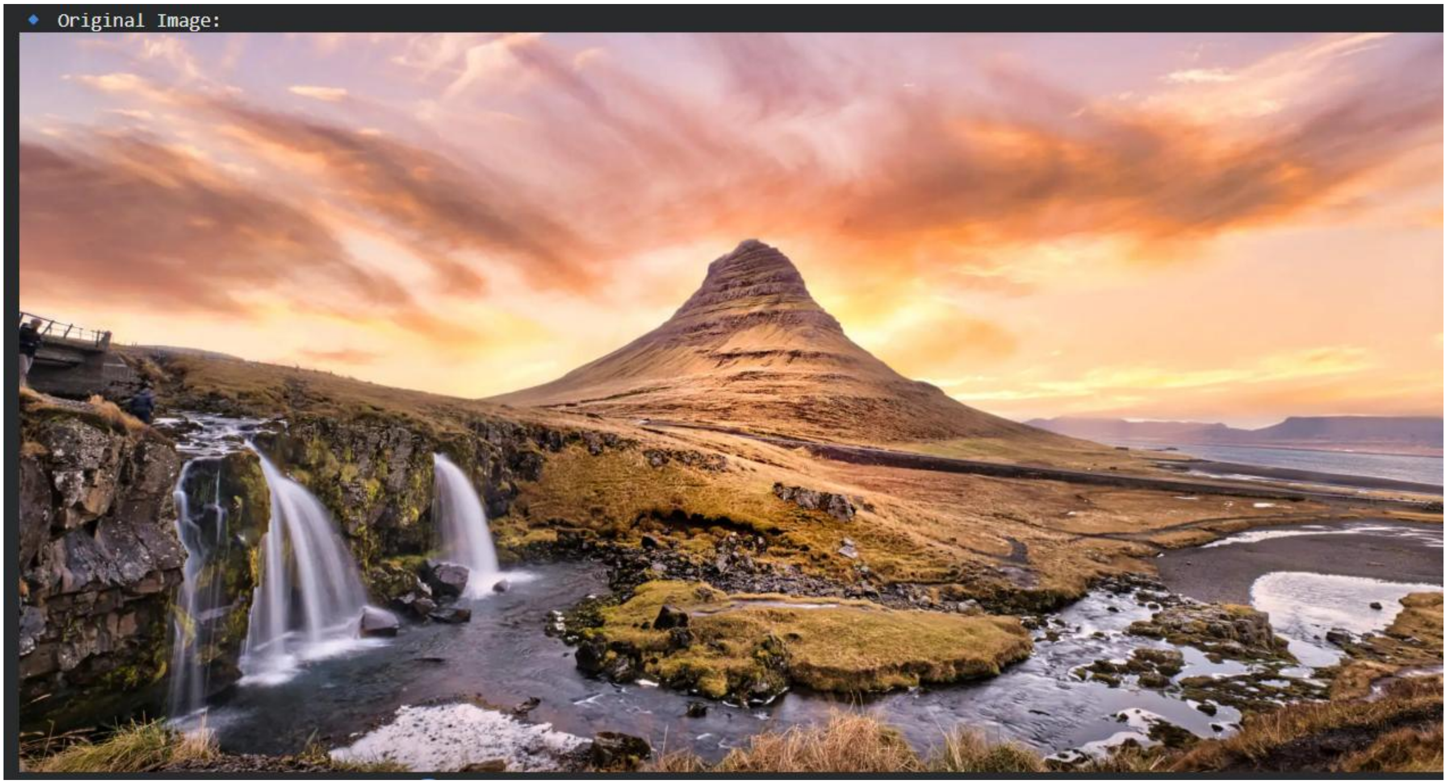
```python
# ============================================
# 6. Artistic blending — soften edges + combine
# ============================================

# soften edges
lap_blur = cv2.GaussianBlur(laplacian, (7, 7), 0)
canny_blur = cv2.GaussianBlur(canny, (7, 7), 0)

# combine edges (weight them)
combined_edges = cv2.addWeighted(lap_blur, 0.6, canny_blur, 0.4, 0)

# invert edges for sketch look
edges_inv = cv2.bitwise_not(combined_edges)

print("Combined Soft Edges (Inverted):")
cv2_imshow(edges_inv)

# convert original image to grayscale for blending
gray3 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# make into 3-channel for blending
edges_color = cv2.cvtColor(edges_inv, cv2.COLOR_GRAY2BGR)

# blend with original
artistic_sketch = cv2.addWeighted(img, 0.5, edges_color, 0.5, 0)

print("Final Artistic Sketch-Style Image:")
cv2_imshow(artistic_sketch)

# save result
cv2.imwrite("/artistic_sketch_result.jpg", artistic_sketch)
print("Saved as /artistic_sketch_result.jpg")


import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# ======================================================
# 1. Load Image
# ======================================================
img = cv2.imread("/content/mosalah.jpg")
if img is None:
    raise ValueError("Image not found!")

print("Original Image:")
cv2_imshow(img)

# ======================================================
# 2. Convert to HSV for Skin Detection
# ======================================================
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Skin color range (tuned for many lighting conditions)
lower_skin = np.array([0, 40, 40], dtype=np.uint8)
```

```python
upper_skin = np.array([25, 255, 255], dtype=np.uint8)

# Create skin mask
skin_mask = cv2.inRange(hsv, lower_skin, upper_skin)

print("Skin Mask (raw):")
cv2_imshow(skin_mask)

# ======================================================
# 3. Clean the Mask (Morphology)
# ======================================================
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))

# Remove noise
skin_mask = cv2.erode(skin_mask, kernel, iterations=2)
skin_mask = cv2.dilate(skin_mask, kernel, iterations=2)

# Smooth the mask
skin_mask = cv2.GaussianBlur(skin_mask, (7, 7), 0)

print("Skin Mask (cleaned):")
cv2_imshow(skin_mask)

# ======================================================
# 4. Find the Face Region (Largest Skin Blob)
# ======================================================
contours, _ = cv2.findContours(skin_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

if len(contours) == 0:
    raise ValueError("No skin region detected.")

# Choose largest contour = face region
largest_contour = max(contours, key=cv2.contourArea)
x, y, w, h = cv2.boundingRect(largest_contour)

# Expand the bounding box a little (for natural framing)
pad = 20
x = max(x - pad, 0)
y = max(y - pad, 0)
w = min(w + pad * 2, img.shape[1] - x)
h = min(h + pad * 2, img.shape[0] - y)

# Draw rectangle for debugging
face_area = img.copy()
cv2.rectangle(face_area, (x, y), (x + w, y + h), (255, 0, 0), 2)
print("Detected Face Region:")
cv2_imshow(face_area)

# ======================================================
# 5. Blur background but keep face sharp
# ======================================================

# 5.1 Create blurred version of entire image
blurred = cv2.GaussianBlur(img, (45, 45), 0)
```

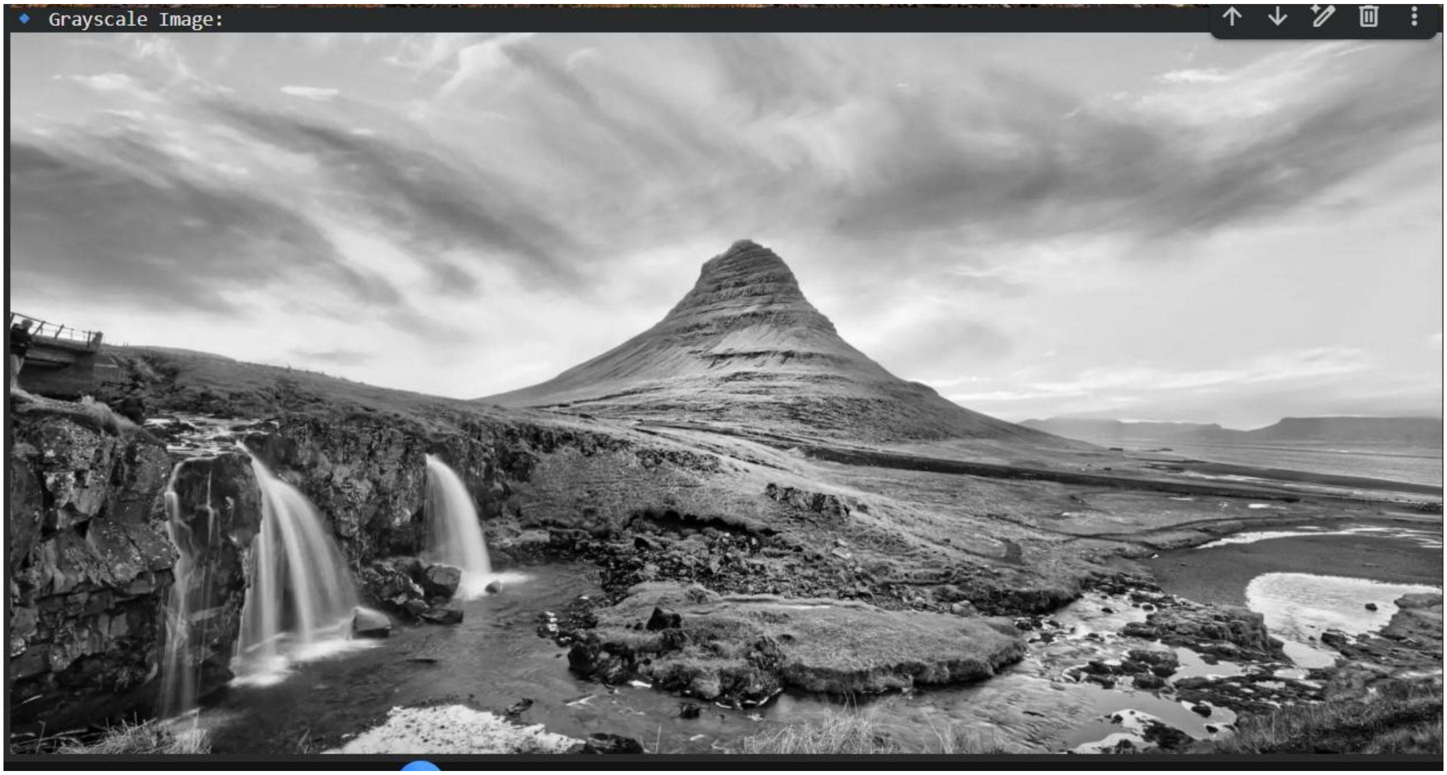
```

# 5.2 Create mask where face = white, background = black
mask = np.zeros(img.shape[:2], dtype=np.uint8)
mask[y:y+h, x:x+w] = 255

# Smooth mask for soft blending
mask = cv2.GaussianBlur(mask, (41, 41), 0)

# Convert mask to 3 channels
mask3 = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)

# Normalize mask to 0–1
mask_float = mask3.astype(float) / 255.0

# Blended result:
# face = img * mask  + blurred * (1 - mask)
result = (img * mask_float + blurred * (1 - mask_float)).astype(np.uint8)

print(" Final Result (Face Sharp, Background Blurred):")
cv2_imshow(result)

# Save output
cv2.imwrite("/face_blur_result.jpg", result)
print("Saved as: /face_blur_result.jpg")

```

Output:

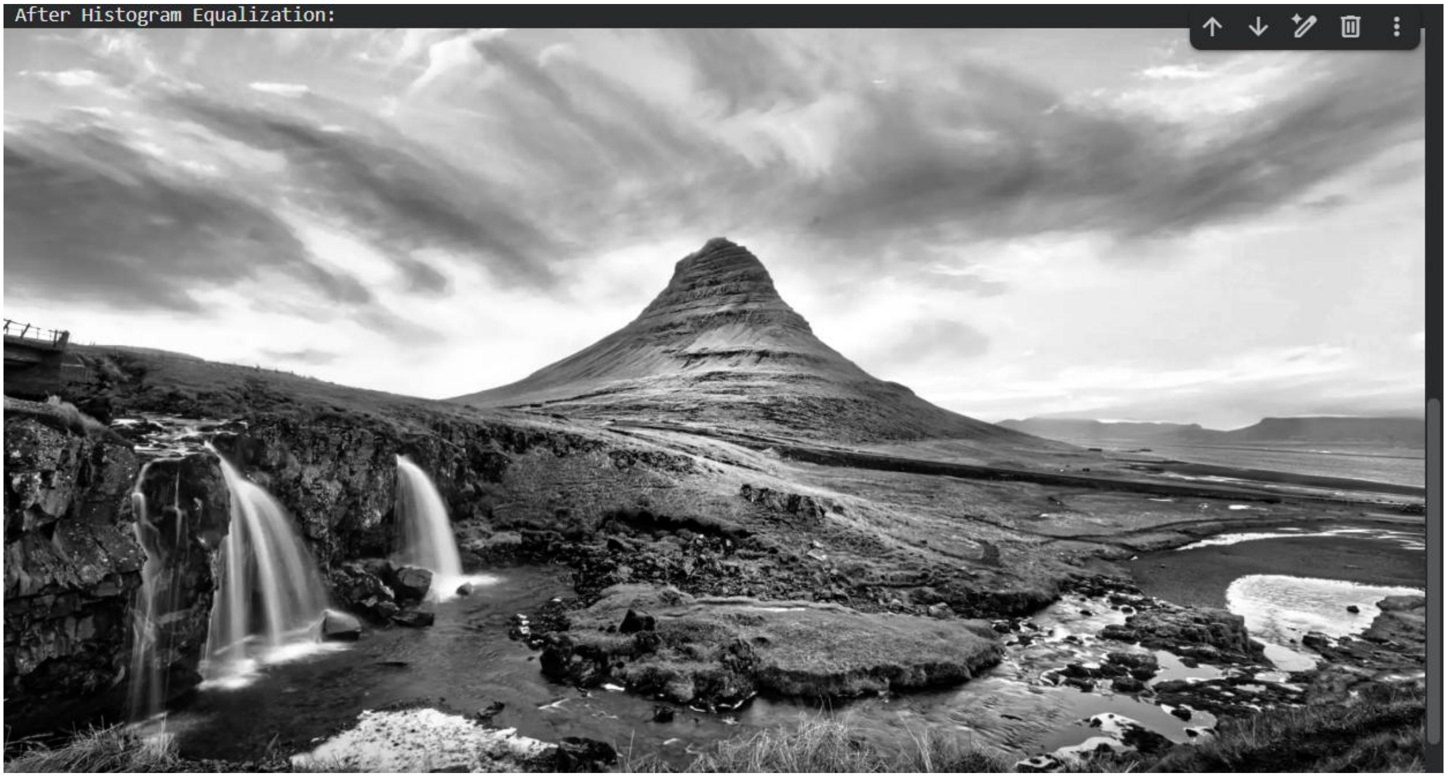
Task 1 :



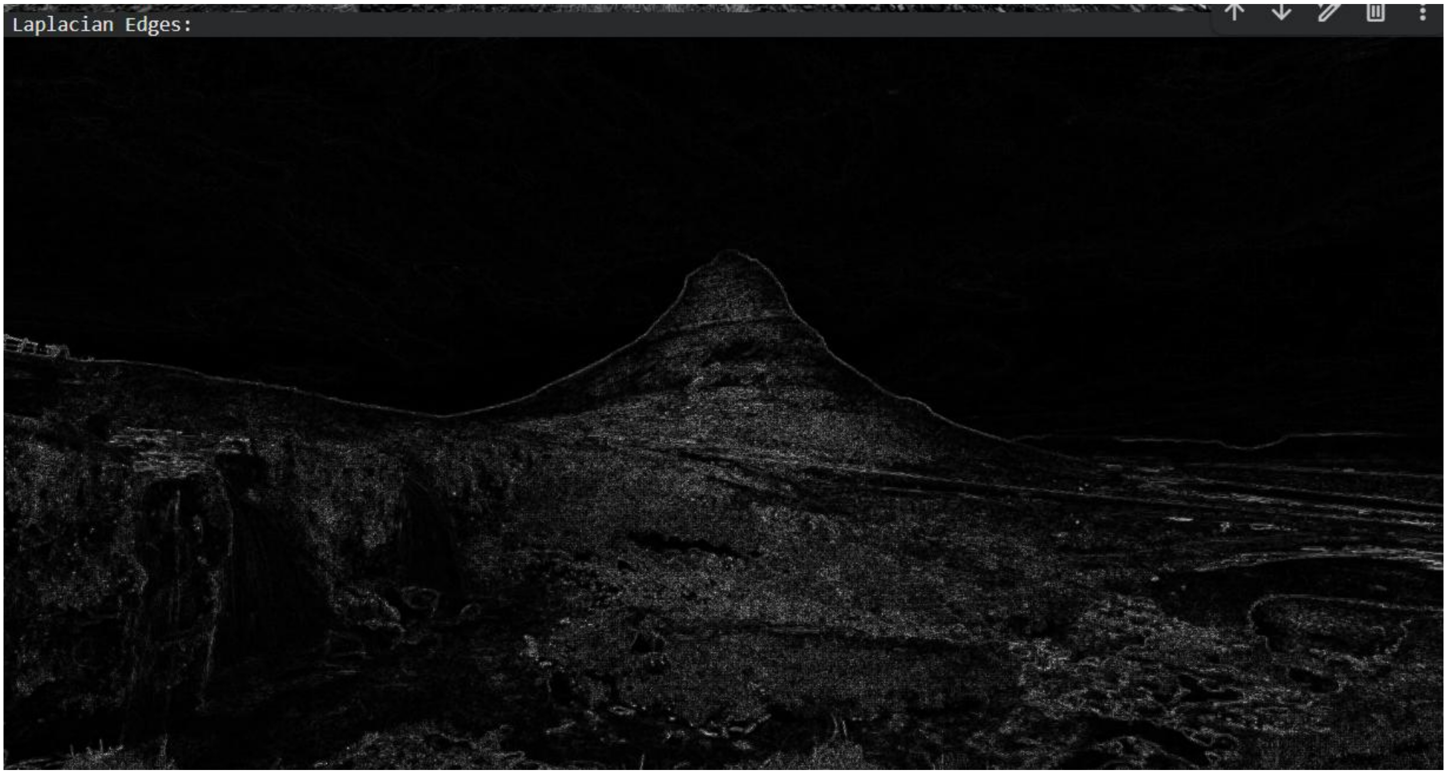
```
Original Image:
```



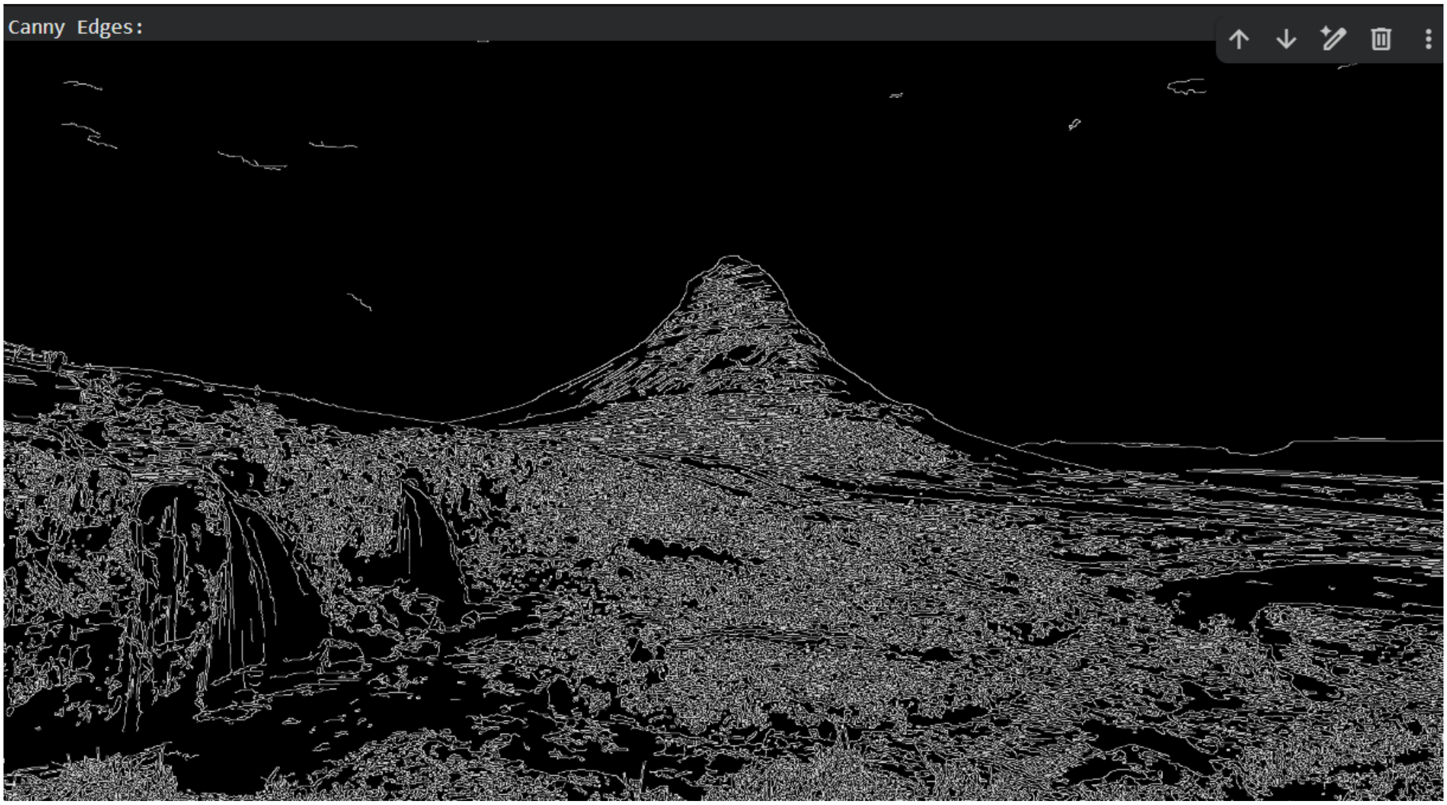
```
Grayscale Image:
```



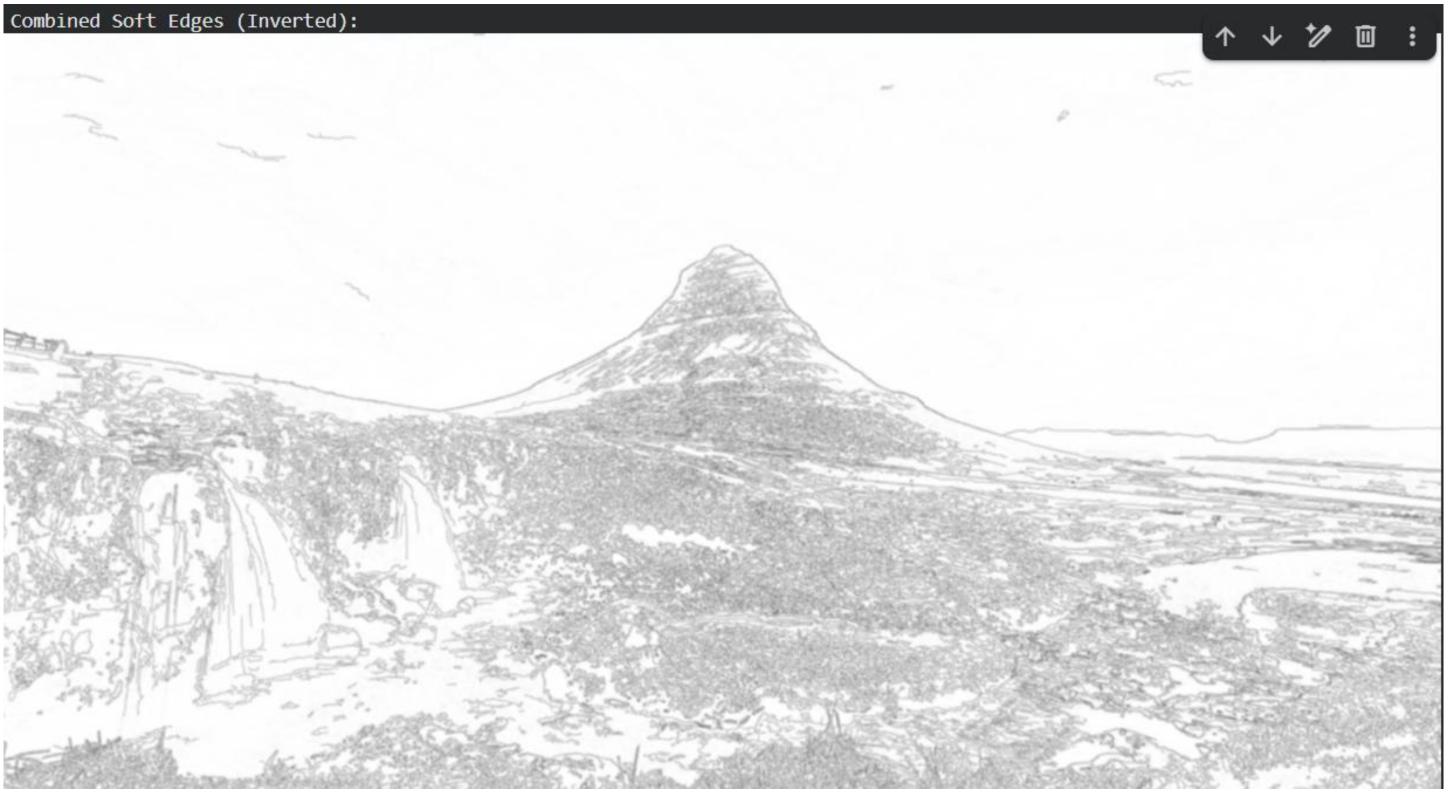
```
After Histogram Equalization:
```



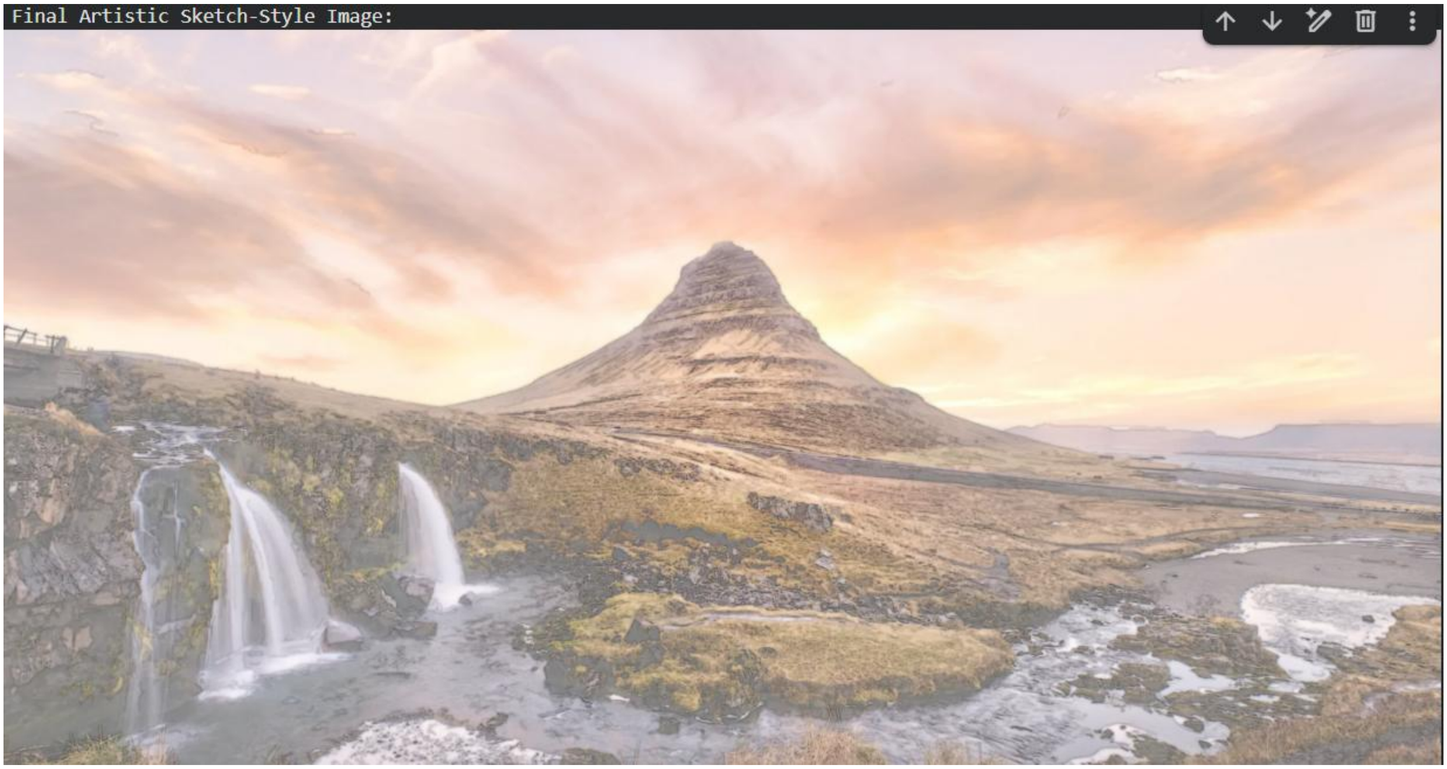
```
Laplacian Edges:
```



```
Canny Edges:
```

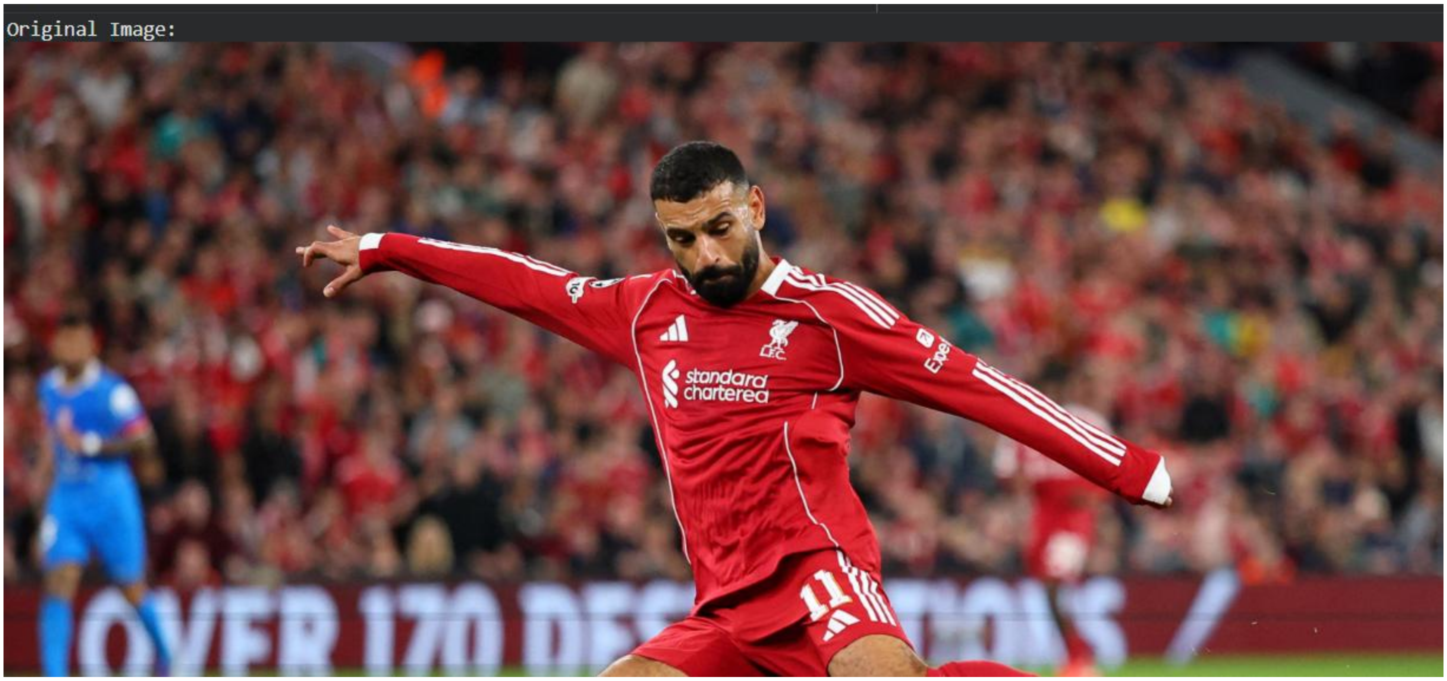


```
Combined Soft Edges (Inverted):
```

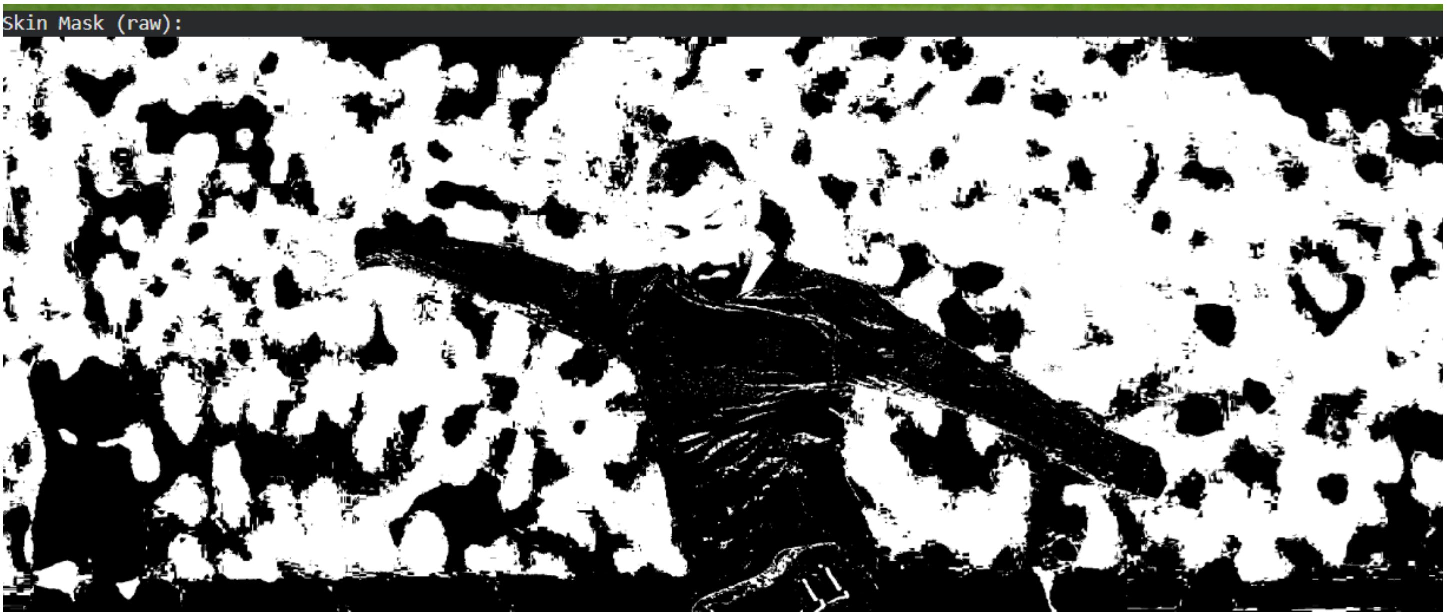


```
Final Artistic Sketch-Style Image:
```

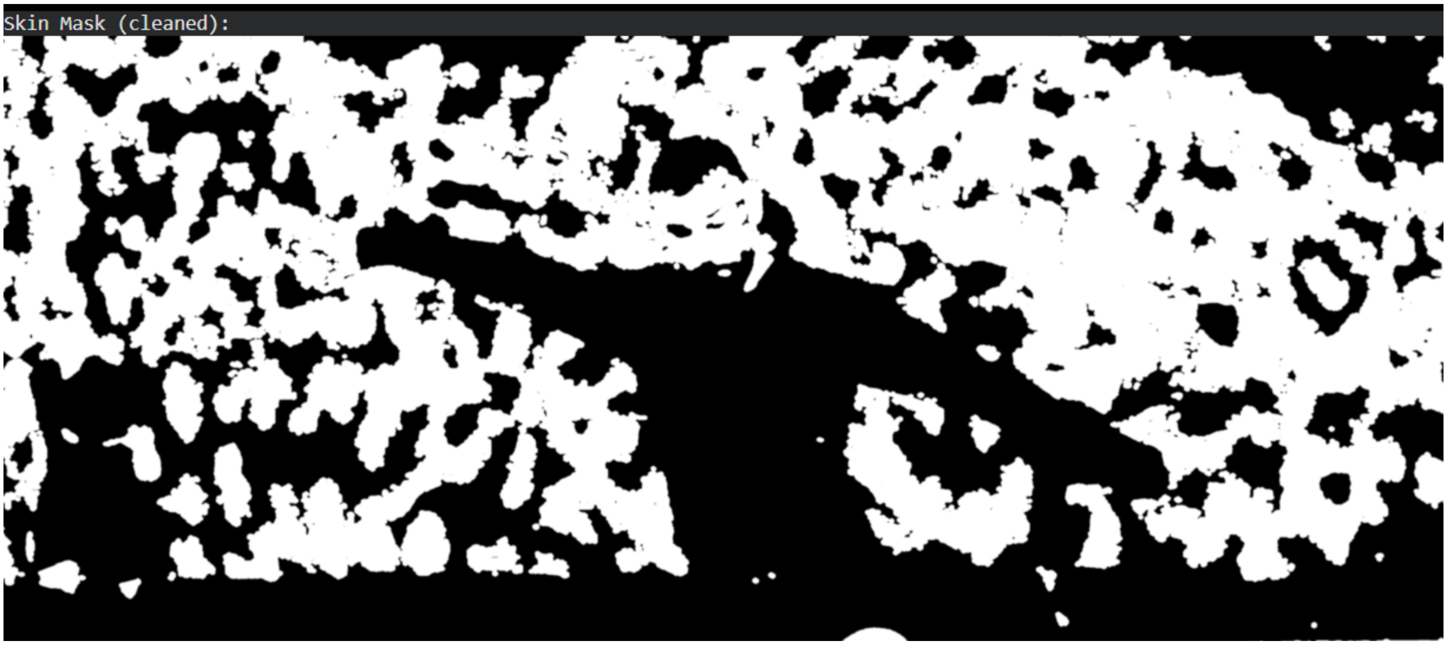
Task 2 :



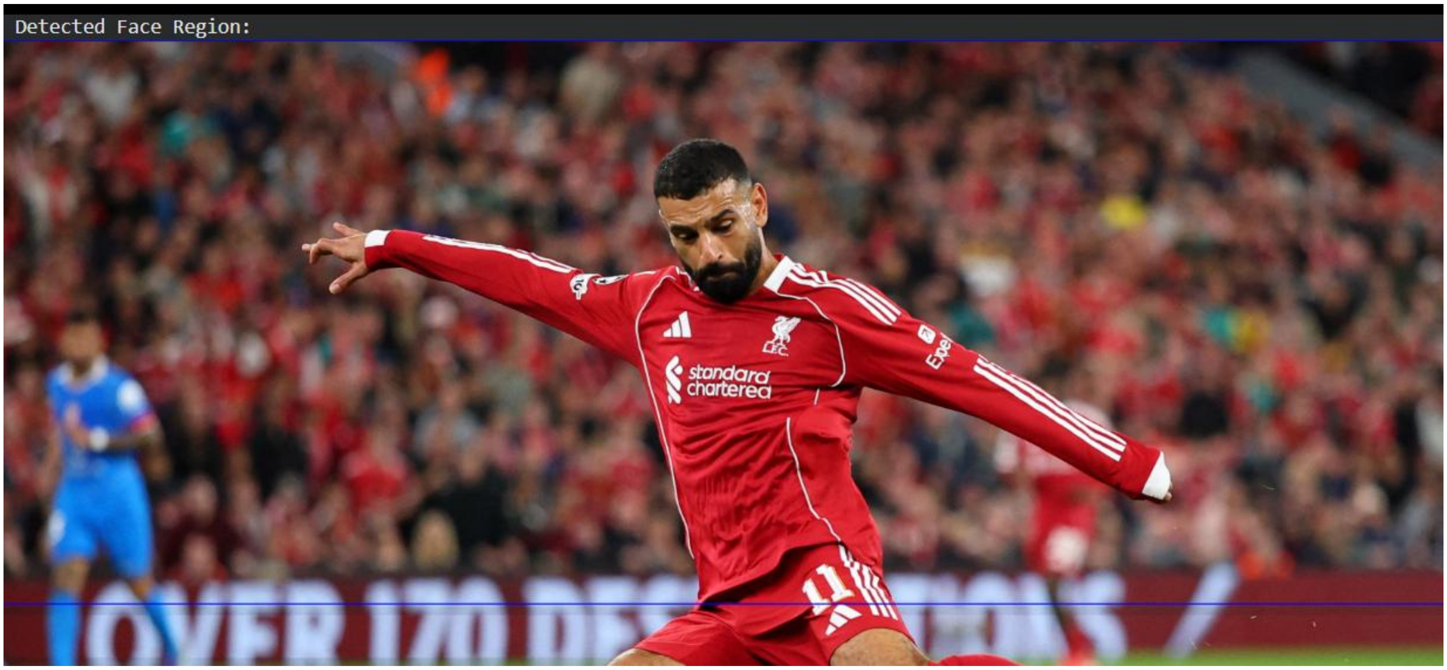
```
Original Image:
```



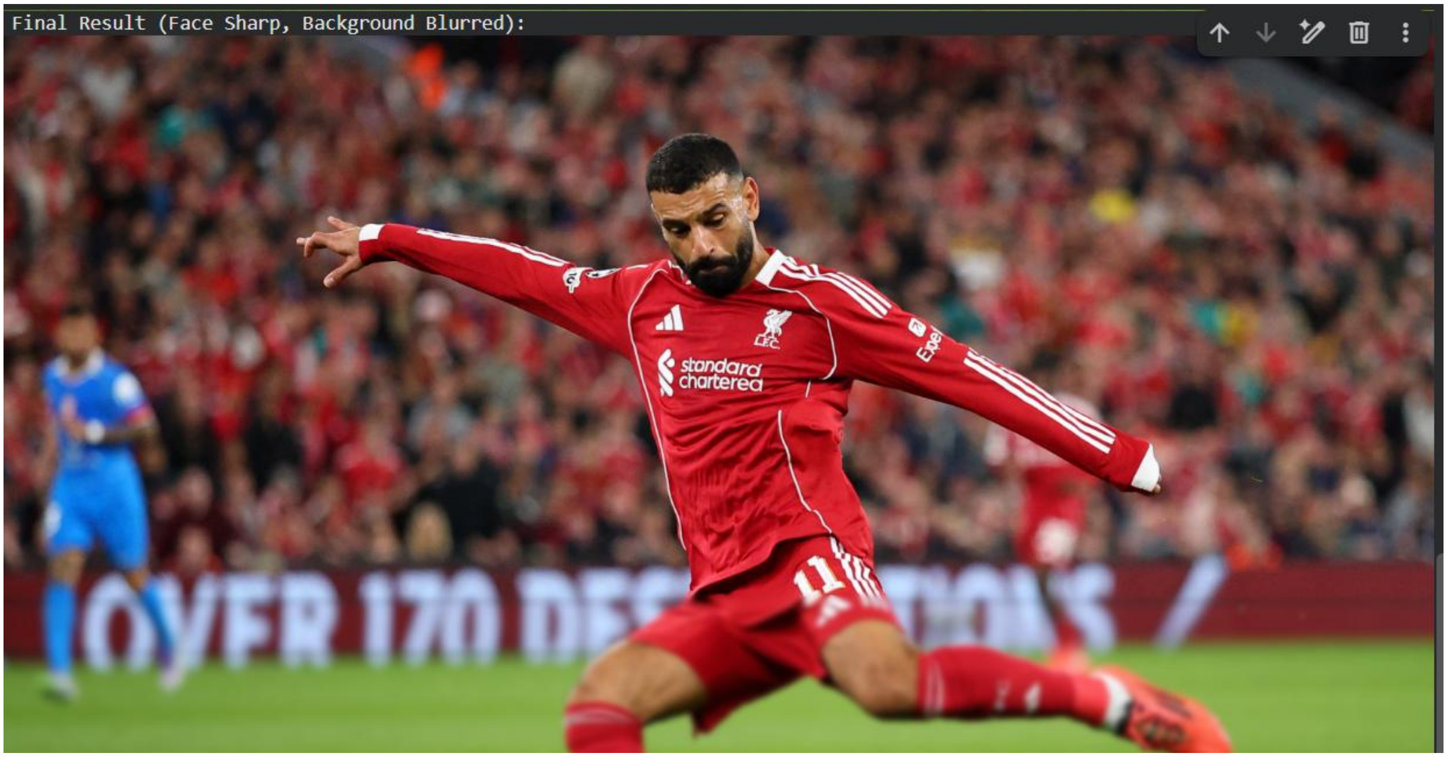
```
Skin Mask (raw):
```



```
Skin Mask (cleaned):
```



Detected Face Region:



Final Result (Face Sharp, Background Blurred):