گزارش تمرین پنجم مبانی بینایی کامپیوتر

سروش جهانیان

983112034

این تمرین را به دو شکل انجام دادم

نحوه اول در فایل face_detection.ipynb :

لود کردن مدل آماده :

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
print(face_cascade)
```

✓ 0.2s

```
< cv2.CascadeClassifier 00000206E1C7A6D0>
```

لود کردن عکس :

```
image = cv2.imread('class-pic.jpg')
```

✓ 0.1s

تبدیل به سطح خاکستری :

```
# conversion to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = np.array(gray, dtype='uint8')
```

✓ 0.0s

پیدا کردن صورت توسط detectMultiScale :

```
    # detect faces is grayScale image
    faces = face_cascade.detectMultiScale(gray, 1.1, 5)
[18]  ✓ 0.5s
```

خط کشیدن دور صورت ها و نمایش تک تک آنها :

```
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
    plt.imshow(roi_gray)
    plt.show()
✓ 4.7s
```

می توان detectMultiScale را با پارامتر های متفاوت اجرا کرد که نتایج مختلفی به همراه دارد (چهره های متفاوتی را تشخیص می دهد)

```
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
    plt.imshow(roi_gray)
    plt.show()
✓ 4.7s
```

نحوه دوم در فایل hsv_detection :

لود کردن عکس و تبدیل به grayscale مشابه حالت قبل است.

سپس در فضای رنگی hsv یک ریت رنگی بالا و پایین برای پوست در نظر میگیریم:

```
# Define the bounds of the skin color in HSV using numpy array
first_lower_skin = np.array([0, 10, 60], dtype="uint8")
first_upper_skin = np.array([9, 255, 255], dtype="uint8")
```

سپس با توجه به آن یک ماسک باینری می سازیم :

```
# Create binary mask by applying the color range threshold
first_skin_mask = cv2.inRange(hsv, first_lower_skin, first_upper_skin)
```

سپس دوباره همین دو مرحله را تکرار می کنیم ولی این بار با طیف رنگی متفاوت:

```
# Define bounds of the skin color in HSV using numpy array
second_lower_skin = np.array([262/2, 0, 0], dtype="uint8")
second_upper_skin = np.array([360/2, 255, 255], dtype="uint8")

# Create the second binary mask by applying the color range threshold
second_skin_mask = cv2.inRange(hsv, second_lower_skin, second_upper_skin)
```

سپس دو ماسک تشکیل شده را با هم bitwise_or می کنیم:

```
# Combine the two binary masks using bitwise OR(performs a bitwise OR operation)
skin_mask_combined = cv2.bitwise_or(first_skin_mask, second_skin_mask)
```

سپس برای حذف نویز و اصلاح ماسک از عملیات مورفولوژیکی استفاده می کنیم (opening و closing)

```
# Apply morphological operations to remove noise and refine the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
skin_mask_combined = cv2.morphologyEx(skin_mask_combined, cv2.MORPH_OPEN, kernel)
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
skin_mask_combined = cv2.morphologyEx(skin_mask_combined, cv2.MORPH_CLOSE, kernel)
```

مشخص کردن کانتورها و تشخیص حالت دایره ای صورت ها:

```
for contour in contours:
    perimeter = cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, 0.02 * perimeter, True)
    if len(approx) > 7:
        # Apply area threshold to filter out small contours
        area = cv2.contourArea(contour)
        if  200 < area < 1200:
            # Find the bounding box of the contour
            x, y, w, h = cv2.boundingRect(contour)
            faces.append((x, y, w, h))
```

خط کشیدن دور صورت ها و نمایش تک تک آنها و نمایش عکس کلی:

```
# Draw rectangles around the detected faces
# separate the face
for (x, y, w, h) in faces:
    cv2.rectangle(copy_image, (x, y), (x+w, y+h), (0, 255, 0), 3)
    # detect each face
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
    plt.imshow(roi_gray)
    plt.show()


im_rgb = cv2.cvtColor(copy_image, cv2.COLOR_BGR2RGB)

plt.imshow(im_rgb)
plt.axis('off')
plt.show()
```

خروجی :