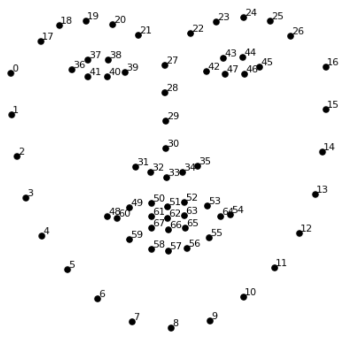
OpenCV 105 Computer Vision Applications II Python course
Project 1: Virtual Makeup — Apply lipstick

In this document I will describe the steps followed to apply lipstick to a face identifying the right location and size of the lips to modify the color using a LUT.

We use the provided landmark detector and calculated points as well as the girl's image as starting point.

STEP 1: DETECT LANDMARKS

We just need to run the provided cells in the notebook to get an array of points for the landmark calculation, we will get points as per the image below.

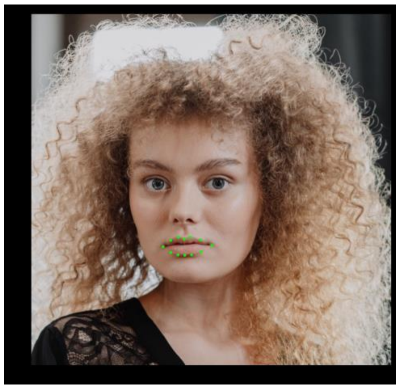


STEP 2: GET THE LIPS POINTS

Once we have the complete array of landmarks, let's identify which are the ones corresponding to the outer shape of the lips. These points are **points[48:60]**.

Let's print the image with the lips related landmarks.

```
# Print lips landmarks
# Select the landmarks that correspond to the lips.

im_landmark = imDlib.copy()

# Outer lips
outer_lips = points[48:60]
for p in outer_lips:
        cv2.circle(im_landmark, (p[0], p[1]), 3, (0, 255, 0), -1)

plt.imshow(im_landmark)
```

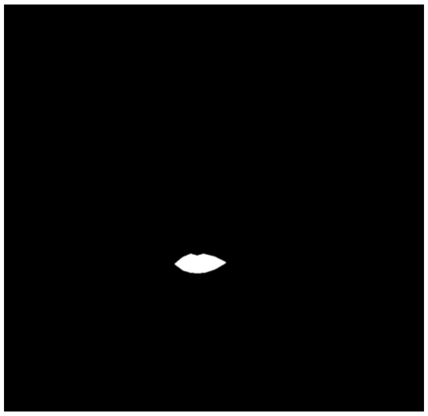


STEP 3: CREATE A LIPS MASK

Next step is to create a mask based on the lips shape. We are using the cv2.fillConvexPoly() function to create a polygon and we refine the final touch by using some morphological operations like closing and opening.

We display the final mask result

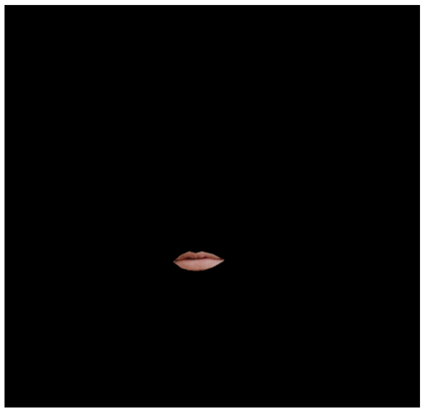
```
# Create initial black mask with the same size as im input
mask = np.zeros(im.shape[:2], dtype=np.uint8)

# Create white mask for outer lips
cv2.fillConvexPoly(mask, np.int32(outer_lips), 255);

# Refine the shape using morphological operations
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

# Display final mask
plt.imshow(mask)
```



STEP 4: MASK THE ORIGINAL IMAGE

We now mask the original image to extract just the ROI with the lips that we want to change the color.

```
rgb_mouth = cv2.bitwise_and(imDlib, imDlib, mask=mask)
plt.imshow(rgb_mouth)
```



STEP 5: APPLY LUT TO CHANGE THE COLOR

We define a function **apply_color_mouth()** which is just a modified LUT for the red and blue color channels that will allow us colorize the masked lips region

Let's apply a green color effect and plot the result

```
def apply_color_mouth(rgb_mouth):
    img = np.copy(rgb_mouth)
    originalValue = np.array([0, 50, 100, 150, 200, 255])

    #green
    bCurve = np.array([0, 20, 40, 75, 150, 255])
    rCurve = np.array([0, 20,  40,  75, 150, 255])

    fullRange = np.arange(0,256)
    rLUT = np.interp(fullRange, originalValue, rCurve )
    bLUT = np.interp(fullRange, originalValue, bCurve )

    bChannel = img[:,:,0]
    bChannel = cv2.LUT(bChannel, bLUT)
    img[:,:,0] = bChannel

    rChannel = img[:,:,2]
    rChannel = cv2.LUT(rChannel, rLUT)
    img[:,:,2] = rChannel

    return img
```

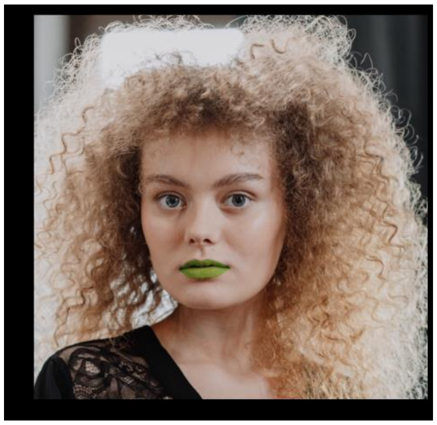
```
coloured_mouth = apply_color_mouth(rgb_mouth)
plt.imshow(coloured_mouth)
```

Step 6: Combine the modified lips with the original image

Then, by performing bitwise operations, we subtract the original lips from the image and perform a merging operation (addition) to place the colored lips on top of the original image.

Let's print the final result

```
background = cv2.bitwise_and(imDlib, imDlib, mask=(255-mask))
lipstick = background + coloured_mouth
plt.imshow(lipstick)
```



By modifying the LUT coefficients we can colorize the lips differently to get a wide range of colour variations.