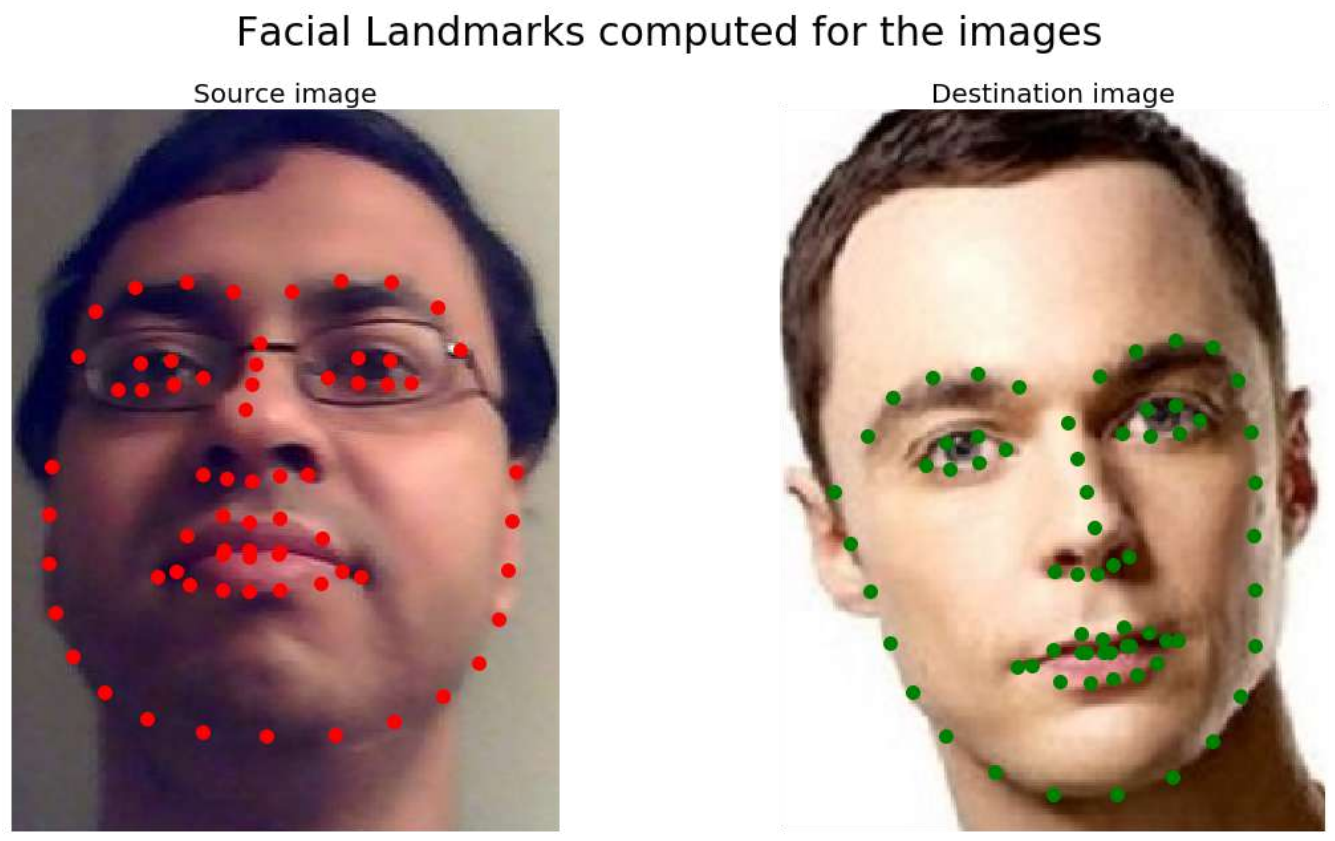
Chapter 7: Facial Image Processing

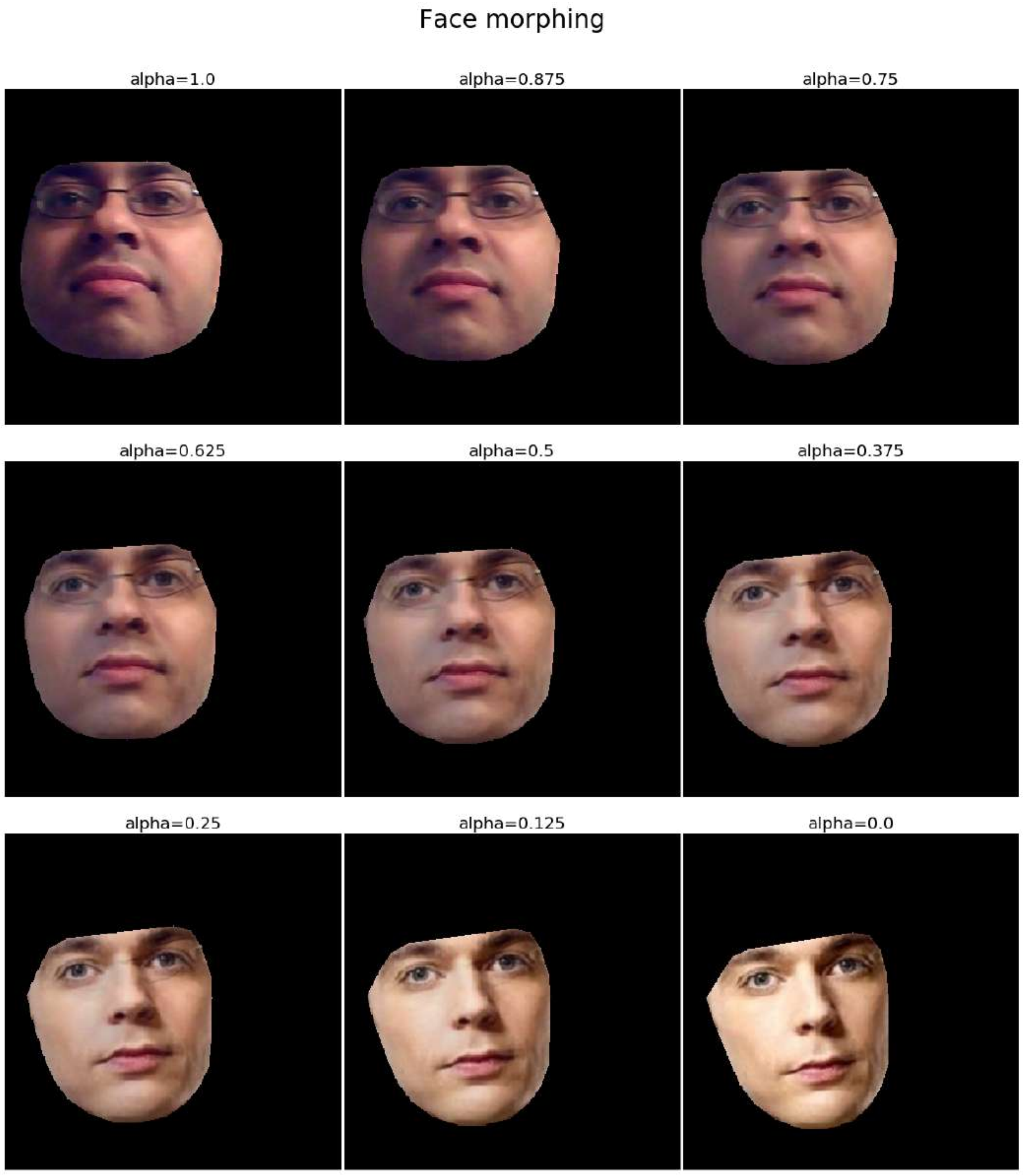
Problems

1. Face morphing with dlib, scipy.spatial and opencv-python

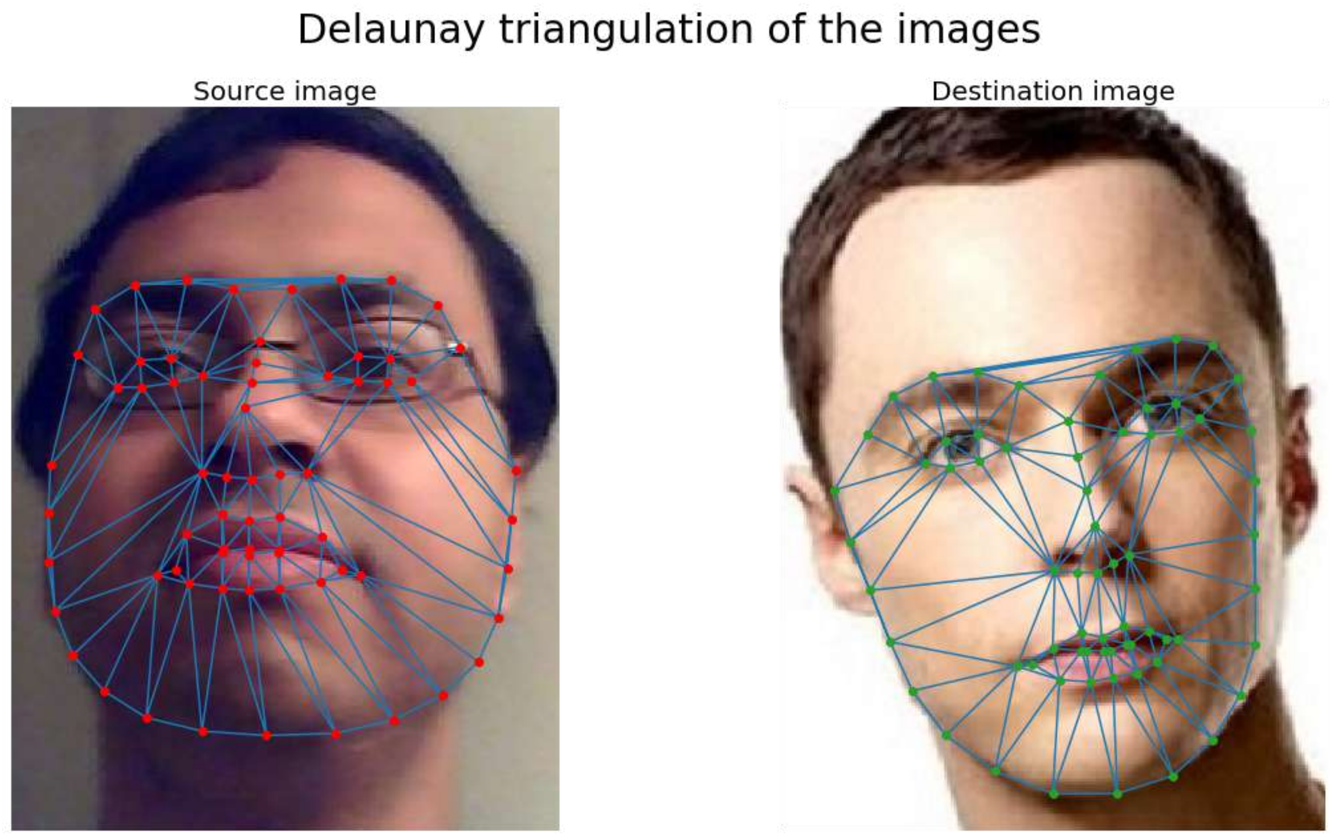
In [26]:
```
fig = plt.figure(figsize=(20,10))
plt.subplot(121)
plt.imshow(src_img)
for i in range(68):
    plt.plot(src_points[i,0], src_points[i,1], 'r.', markersize=20)
plt.title('Source image', size=20)
plt.axis('off')
plt.subplot(122)
plt.imshow(dst_img)
for i in range(68):
    plt.plot(dst_points[i,0], dst_points[i,1], 'g.', markersize=20)
plt.title('Destination image', size=20)
plt.axis('off')
plt.suptitle('Facial Landmarks computed for the images', size=30)
fig.subplots_adjust(wspace=0.01, left=0.1, right=0.9)
plt.show()
```



```
In [33]: fig = plt.figure(figsize=(20,20))
         # Produce morph frames!
         i = 1
         for percent in np.linspace(1, 0, 9):
             points = weighted_average_points(src_points, dst_points, percent)
             src_face, src_d = warp_image(src_img, src_points, points, (350,350))
             end_face, end_d = warp_image(dst_img, dst_points, points, (350,350))
             average_face = weighted_average_points(src_face, end_face, percent)
             plt.subplot(3, 3, i)
             plt.imshow(average_face)
             plt.title('alpha=' + str(percent), size=20)
             plt.axis('off')
             i += 1
         plt.suptitle('Face morphing', size=30)
         fig.subplots_adjust(top=0.92, bottom=0, left=0.075, right=0.925, wspace=0.01, hspace=0.
         plt.show()
```



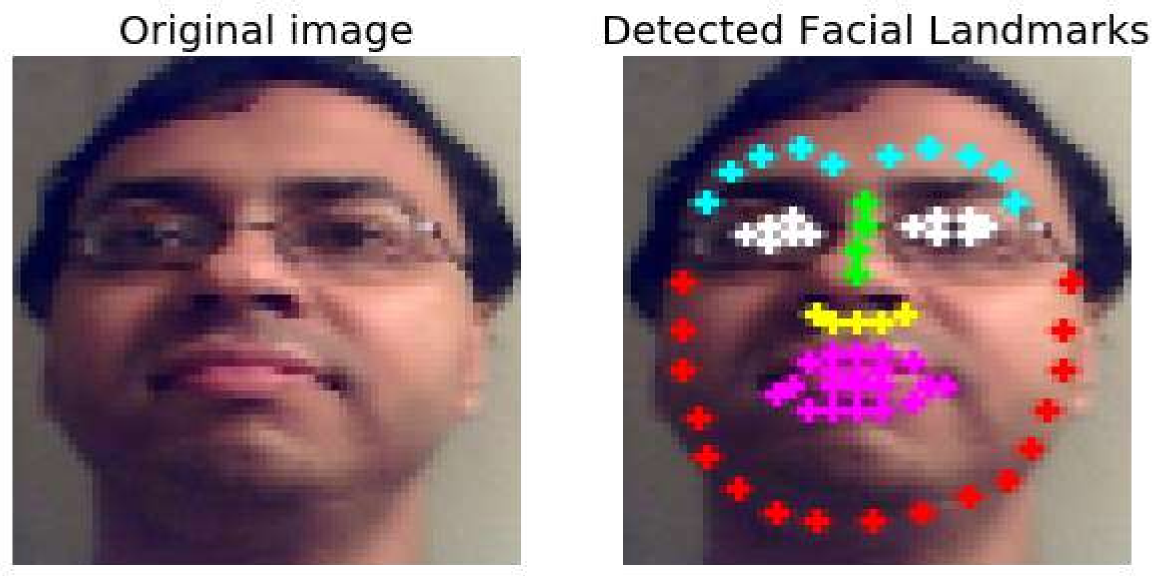
In [34]:
```
fig = plt.figure(figsize=(20,10))
plt.subplot(121)
plt.imshow(src_img)
plt.triplot(src_points[:,0], src_points[:,1], src_d.simplices.copy())
plt.plot(src_points[:,0], src_points[:,1], 'o', color='red')
plt.title('Source image', size=20)
plt.axis('off')
plt.subplot(122)
plt.imshow(dst_img)
plt.triplot(dst_points[:,0], dst_points[:,1], end_d.simplices.copy())
plt.plot(dst_points[:,0], dst_points[:,1], 'o')
plt.title('Destination image', size=20)
plt.axis('off')
plt.suptitle('Delaunay triangulation of the images', size=30)
fig.subplots_adjust(wspace=0.01, left=0.1, right=0.9)
plt.show()
```



2. Facial Landmark Detection with Deep Learning Models

2.1 Facial Landmark Detection with Keras

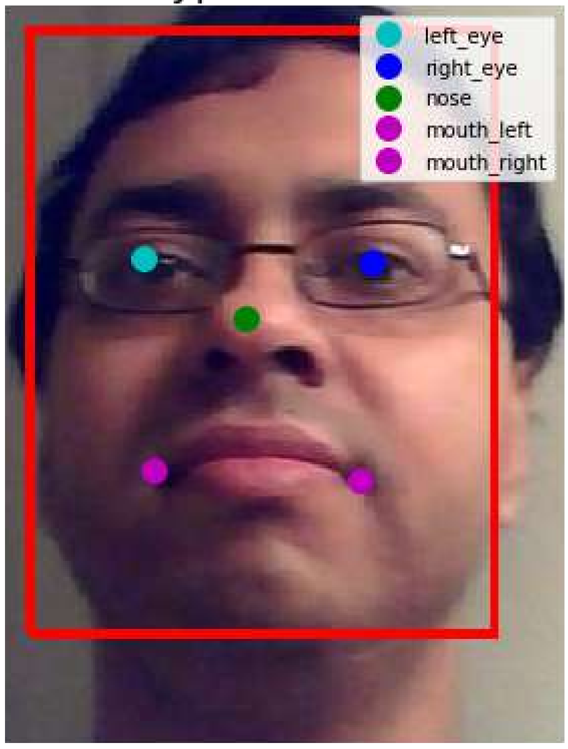
In [57]:
```
image = image_color.copy()
plt.figure(figsize=(10,5))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)), plt.axis('off'), 
for mark in marks[:17]:
    cv2.circle(image, (int(w*mark[0]), int(h*mark[1])), 1, (0,0,255), -1, cv2.LINE_AA)
for mark in marks[17:27]:
    cv2.circle(image, (int(w*mark[0]), int(h*mark[1])), 1, (255,255,0), -1, cv2.LINE_AA
for mark in marks[27:31]:
    cv2.circle(image, (int(w*mark[0]), int(h*mark[1])), 1, (0,255,0), -1, cv2.LINE_AA)
for mark in marks[31:36]:
    cv2.circle(image, (int(w*mark[0]), int(h*mark[1])), 1, (0,255,255), -1, cv2.LINE_AA
for mark in marks[36:48]:
    cv2.circle(image, (int(w*mark[0]), int(h*mark[1])), 1, (255,255,255), -1, cv2.LINE_
for mark in marks[48:]:
    cv2.circle(image, (int(w*mark[0]), int(h*mark[1])), 1, (255,0,255), -1, cv2.LINE_AA

plt.subplot(122), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)), plt.axis('off'), 
plt.show()
```



2.2 Facial Landmark Detection with MTCNN

In [10]:
```
plt.figure(figsize=(5,8))
box = res[0]['box']
rr, cc = rectangle_perimeter(box[0:2][::-1], end=box[2:4][::-1], shape=img.shape)
for k in range(-2,2):
    img[rr+k, cc+k] = np.array([255,0,0])
kp = res[0]['keypoints']
plt.plot(kp['left_eye'][0], kp['left_eye'][1], 'co', markersize=12, label='left_eye')
plt.plot(kp['right_eye'][0], kp['right_eye'][1], 'bo', markersize=12, label='right_eye'
plt.plot(kp['nose'][0], kp['nose'][1], 'go', markersize=12, label='nose')
plt.plot(kp['mouth_left'][0], kp['mouth_left'][1], 'mo', markersize=12, label='mouth_le
plt.plot(kp['mouth_right'][0], kp['mouth_right'][1], 'mo', markersize=12, label='mouth_
plt.imshow(img)
plt.legend(loc='best')
plt.axis('off')
plt.title('Facial keypoints with MTCNN', size=20)
plt.show()
```

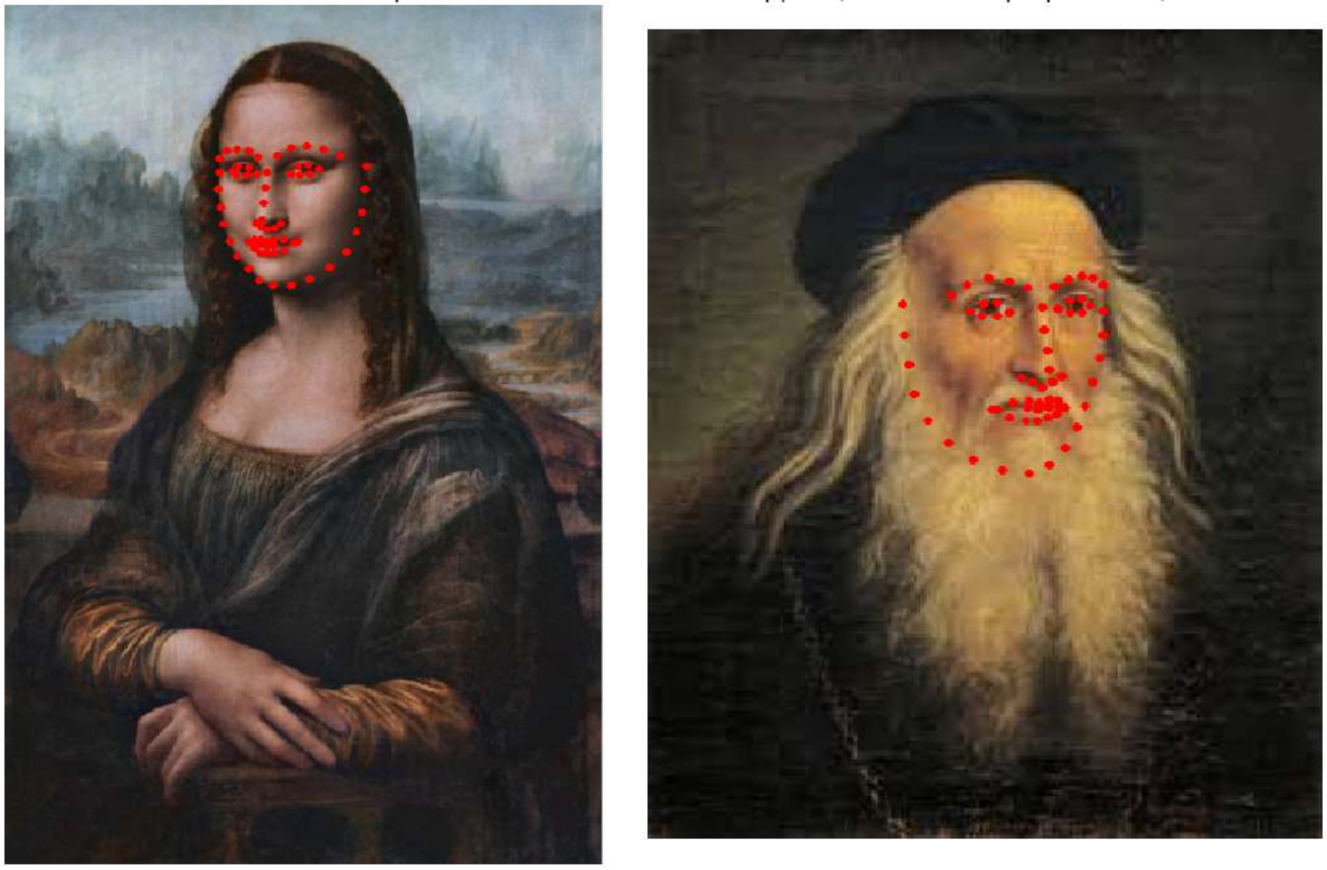
Facial keypoints with MTCNN

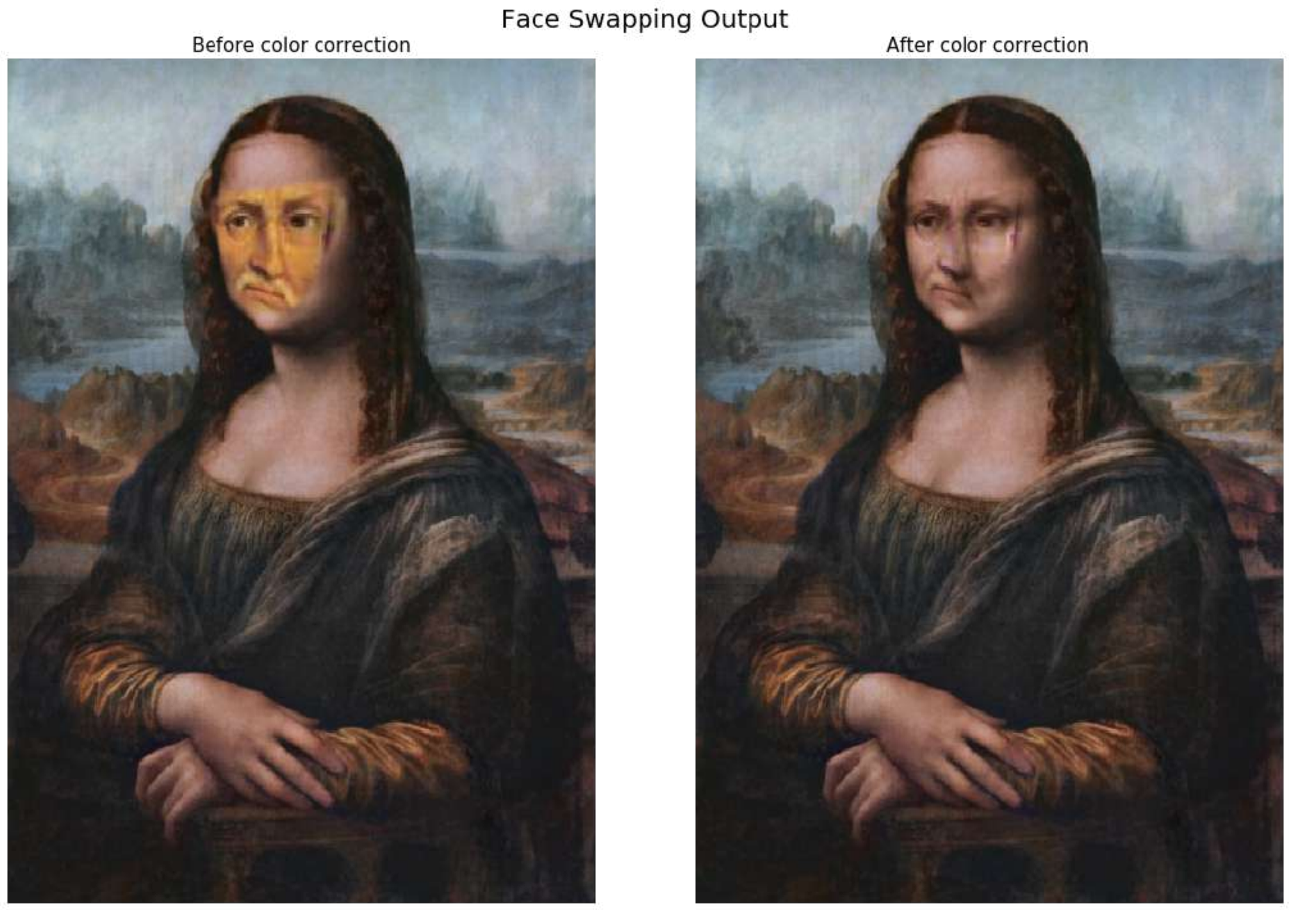


3. Implement Face Swapping

```
In [21]: output_img, output_img_corrected = face_swap_filter('images/Img_07_05.jpg', 'images/Img_

         plt.figure(figsize=(15,10))
         plt.subplots_adjust(0,0,1,0.925,0.01,0.01)
         plt.subplot(121), plt.imshow(cv2.cvtColor(output_img.astype(np.uint8), cv2.COLOR_BGR2RGB
         plt.title('Before color correction', size=15)
         plt.subplot(122), plt.imshow(cv2.cvtColor(output_img_corrected.astype(np.uint8), cv2.COL
         plt.title('After color correction', size=15)
         plt.suptitle('Face Swapping Output', size=20)
         plt.show()
```

Facial landmarks computed for the faces to be swapped (with dlib shape-predictor)

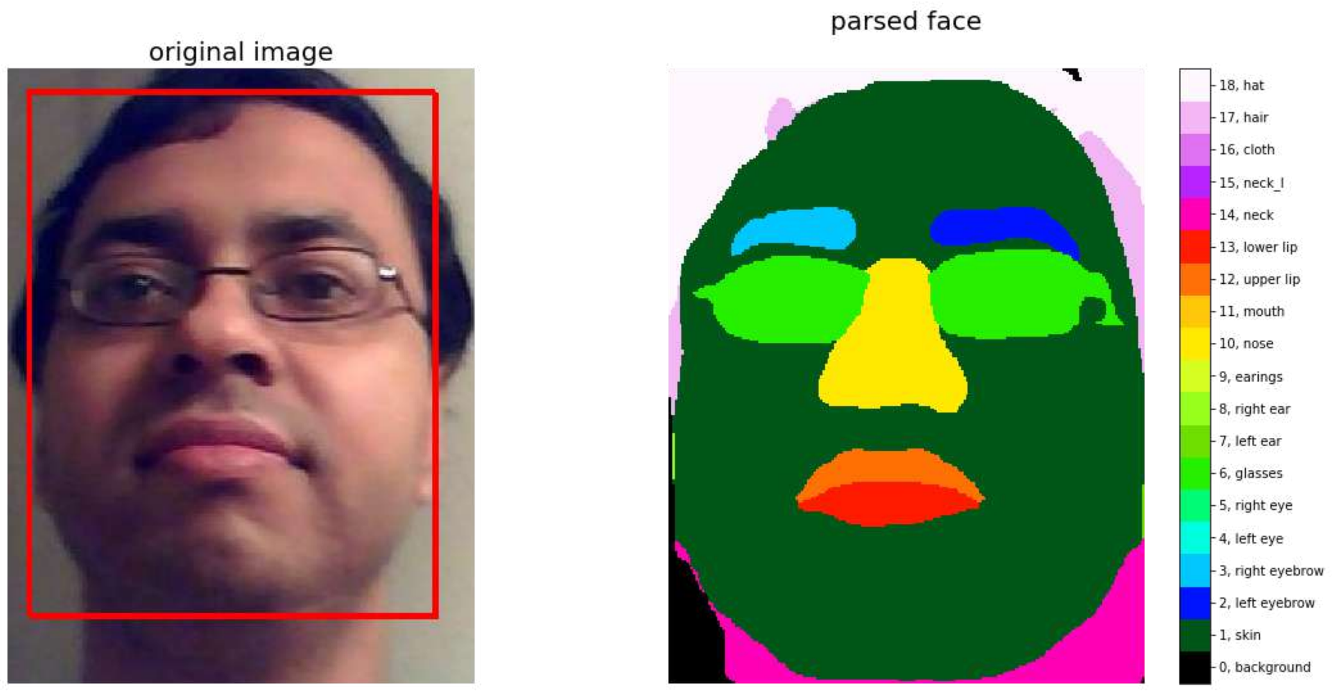




4. Implement Face Parsing

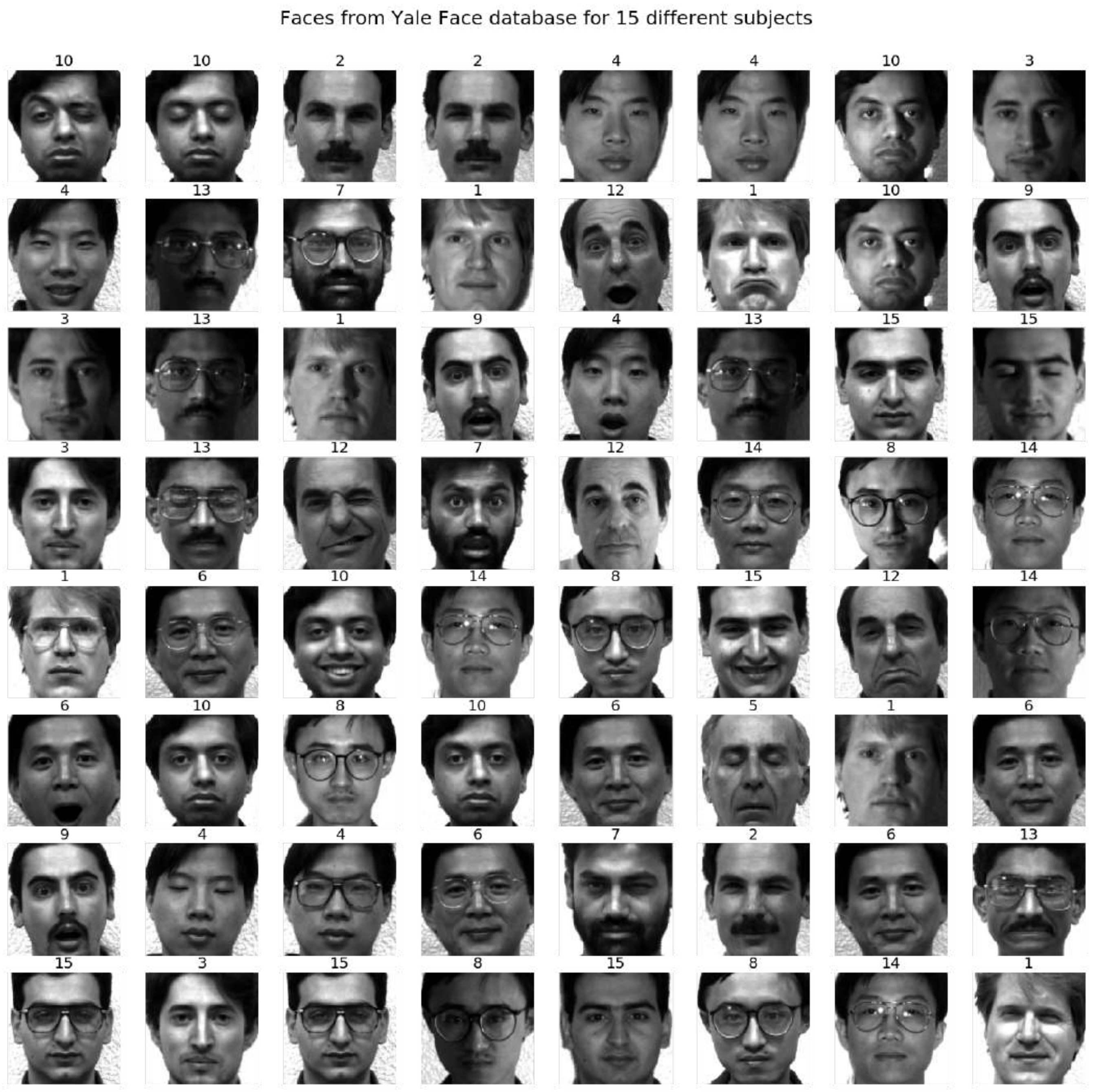
In [35]:

```python
img = imread("images/Img_07_02.jpg")
h, w = img.shape[:2]
detector = MTCNN()
faces = detector.detect_faces(img)
for face in faces:
    bb = face['box']
    face = crop(img,((bb[1],h-(bb[1]+bb[3])),(bb[0],w-(bb[0]+bb[2])),(0,0)))
    rr, cc = rectangle_perimeter((bb[1], bb[0]), extent=(bb[3], bb[2]), shape=img.shape
    for k in range(-1,2):
        img[rr+k, cc+k,:] = [255,0,0]
    # Preprocess input face for parser networks
    orig_h, orig_w = face.shape[:2]
    inp = 255*resize(face, (512,512))
    inp = normalize_input(inp)
    inp = inp[None, ...]
    # Parser networks forward pass
    # Do NOT use bilinear interp. which adds artifacts to the parsing map
    out = model.predict([inp])[0]
    parsing_map = out.argmax(axis=-1)
    parsing_map = cv2.resize(
        parsing_map.astype(np.uint8),
        (orig_w, orig_h),
        interpolation=cv2.INTER_NEAREST)
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,7))
    plt.subplots_adjust(0,0,1,0.95,0.01,0.01)
    ax1.imshow(img), ax1.axis('off'), ax1.set_title('original image', size=20)
    show_parsing_with_annos(parsing_map, f, ax2), ax2.set_title('parsed face', size=20)
    plt.show()
```



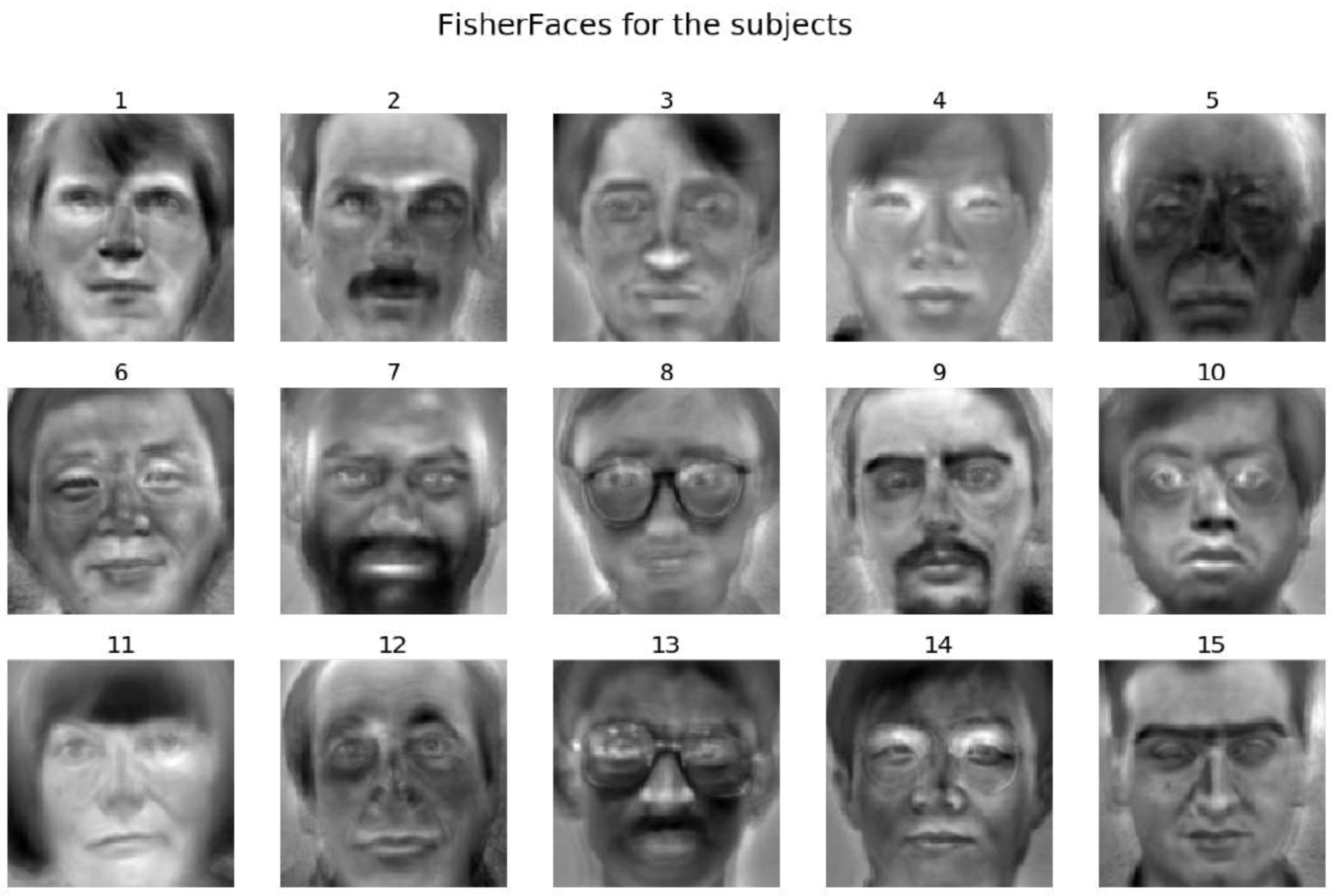
5. Face Recognition with FisherFaces

In [10]:
```python
indices = np.random.choice(165, 64)
plt.figure(figsize=(20,20))
plt.gray()
plt.subplots_adjust(0,0,1,0.925,0.05,0.15)
for i in range(len(indices)):
    plt.subplot(8,8,i+1), plt.imshow(np.reshape(images[indices[i],:], (160,160))), plt.a
    plt.title(labels[indices[i]], size=20)
plt.suptitle('Faces from Yale Face database for 15 different subjects', size=25)
plt.show()
```



```
In [18]: compute_fisherfaces(X_train, X_test, y_train, y_test)
```

```
Fitting data into LDA..
[0.234745   0.17225883 0.12595579 0.08408923 0.06826283 0.05255275
 0.05051213 0.04835622 0.04042571 0.03456516 0.03072084 0.0218708
 0.01971916 0.01596556]
```



```
Data fitting into LDA finished.
Training process finished in 1.90104 seconds.
Data prediction finished.
Classification report
              precision    recall  f1-score   support

           1       0.50      1.00      0.67         1
           2       1.00      1.00      1.00         1
           3       0.50      1.00      0.67         1
           4       1.00      0.50      0.67         2
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         2
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         1
          10       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1
          15       1.00      1.00      1.00         1

   micro avg       0.88      0.94      0.91        16
   macro avg       0.93      0.96      0.93        16
weighted avg       0.94      0.94      0.92        16

0.8823529411764706
```

6. Face Detection and Recognition with Microsoft Cognitive Vision APIs

In [68]:
```
detect_face_age_geneder('images/Img_04_02.jpg')
```

In [59]:
```
recognize_celeb_face("images/Img_07_20.jpg")
```

```
{'result': {'celebrities': [{'faceRectangle': {'top': 96, 'left': 415, 'width': 87,
'height': 87}, 'name': 'Albert Einstein', 'confidence': 0.9985173344612122}, {'face
Rectangle': {'top': 58, 'left': 83, 'width': 85, 'height': 85}, 'name': 'Rabindrana
th Tagore', 'confidence': 0.9770157933235168}]}, 'requestId': '6c078547-4da7-4f37-9
d4b-ccfb1d20a52b', 'metadata': {'width': 654, 'height': 500, 'format': 'Jpeg'}}
```