Artificial intelligence interaction technology 2018-2019(2)The final report

Team Number:11
Team Member:Bingnan Chen

Student Innovation Center
Jul 2019

Contents

1 Team introduction and division of labor

1.1 The name of the project: You-Language theme game

1.2 The function of the program

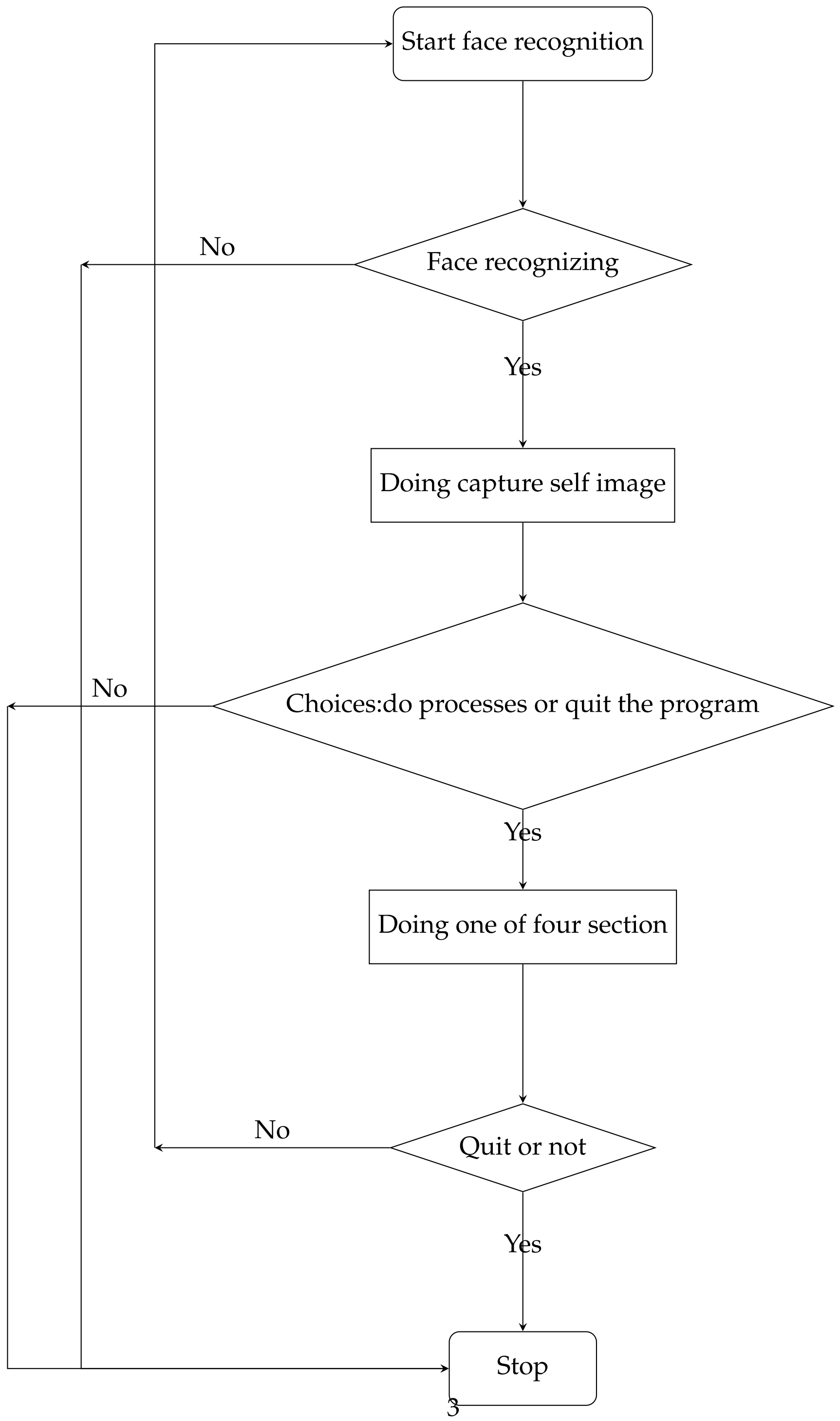
(1) face recognition
(2) you-language learning
(3) make decorations
(4) swap human face
(5) position recognition

2 System architecture design

2.1 The innovation points

The game is a combination of technological recognition part and funny kernel about the animation “Balala fairy”, which plays as its attractive point for kids.

2.2 System architecture design (flow charts provided and explained)



At the beginning, there are some model instantiations, finding device function, voice playing stream settings and initializing the faces and tags lists for later face recognition.

First of all, it's face recognition. If someone fails enter it, he may want to access it without parents permission which will force him to quit. If he passes it, one image will be saved and then the program will lead him to choice for doing processes or quitting the program. Finally, if the section part is finished(You-language learning, make decoration, sign the feature points and frame the face, position recognition(unaccomplished)), you have to choice quitting or not.

2.3 Functions design (give screenshots of main functions and corresponding codes, brief explanation)

```
translate = input("level1:")
pygame.mixer.init()
track = pygame.mixer.Sound("yunvwugua.wav")
time.sleep(1)
track.play()
time.sleep(2)
if(translate == "xxxx"):
        t.say(text = "xxxx", voice = "xiaofeng")
        flag += 1
```

This part plays the You-language voice, game participant has to translate it into mandarin and input it. Then the points will be calculated by variable flag.

```
def model_recognize():
    picture = input("picture_name:(xxx.xxx):")
    img = cv2.imread(picture)
    hat_img = cv2.imread("hat.png", -1)
    r,g,b,a = cv2.split(hat_img)
    rgb_hat = cv2.merge((r,g,b))
    cv2.imwrite("hat_alpha.jpg",a)
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor("shape_predictor_5_face_landmarks.dat")
    faces = detector(img, 1)
    print("face_number:", len(faces))
    for i, d in enumerate(faces):
        x,y,w,h = d.left(), d.top(), d.right()-d.left(), d.bottom()-d.top()
        print("Face", i+1, "at ",
              "left:", d.left(), "right:", d.right(), "top:", d.top(), "bottom:", d.bottom())
        #cv2.rectangle(img, tuple([d.left(), d.top()]), tuple([d.right(), d.bottom()]), (0, 255, 255), 2)
        shape = predictor(img, faces[i])
        points=shape.parts()
        for i in range(shape.num_parts):
            point=(points[i].x,points[i].y)
            print(point)
            #print(point.__doc__)
            #cv2.circle(img, point, 1, (0, 0, 255), 4)
        point1 = shape.part(2)
        point2 = shape.part(4)
        eyes_center = ((point1.x+point2.x)//2,(point1.y+point2.y)//2)
        #cv2.circle(img,eyes_center,3,color=(0,255,0))
        #cv2.imshow("image",img)
        #cv2.waitKey()
        factor = 1.5
        resized_hat_h = int(round(rgb_hat.shape[0]*w/rgb_hat.shape[1]*factor))
        resized_hat_w = int(round(rgb_hat.shape[1]*w/rgb_hat.shape[1]*factor))
        if resized_hat_h > y:
            resized_hat_h = y-1
        resized_hat = cv2.resize(rgb_hat,(resized_hat_w,resized_hat_h))
        mask = cv2.resize(a,(resized_hat_w,resized_hat_h))
        mask_inv = cv2.bitwise_not(mask)
        dh = 20
        dw = 0
        # bg_roi = img[y+dh-resized_hat_h:y+dh, x+dw:x+dw+resized_hat_w]
        bg_roi = img[y+dh-resized_hat_h:y+dh,(eyes_center[0]-resized_hat_w//3):(eyes_center[0]+resized_
        bg_roi = bg_roi.astype(float)
        mask_inv = cv2.merge((mask_inv,mask_inv,mask_inv))
        alpha = mask_inv.astype(float)/255
        alpha = cv2.resize(alpha,(bg_roi.shape[1],bg_roi.shape[0]))
        # print("alpha size: ",alpha.shape)
        # print("bg_roi size: ",bg_roi.shape)
        bg = cv2.multiply(alpha, bg_roi)
        bg = bg.astype('uint8')
        hat = cv2.bitwise_and(resized_hat,resized_hat,mask = mask)
        hat = cv2.resize(hat,(bg_roi.shape[1],bg_roi.shape[0]))
        add_hat = cv2.add(bg,hat)
        # cv2.imshow("add_hat",add_hat)
        img[y+dh-resized_hat_h:y+dh,(eyes_center[0]-resized_hat_w//3):(eyes_center[0]+resized_hat_w//3
        cv2.imshow("Output", img)
    cv2.imwrite("withHat.jpg", img)
    cv2.waitKey(0)
```

This part is given by Tea.Weiming. The next faceswap part will be the main illustrated one.

Thinking transplant other people's face to mine will be pretty fun, so with the help of Google and Github I try to work out this part. It does turn out to be an interesting function.

```
import os
import cv2
import dlib
import numpy as np
cur_path = "/Users/apple/Desktop/sjtu"
models_folder_path = cur_path
faces_folder_path = cur_path
predictor_path = os.path.join(models_folder_path, 'shape_predictor_68_face_landmarks.dat')
bereplaced = input("target_picture:")
image_face_path = os.path.join(faces_folder_path, bereplaced)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(predictor_path)
```

At the beginning, it's the imported about the data set and the swap face.

```
def get_image_size(image):
    image_size = (image.shape[0], image.shape[1])
    return image_size
```

This function gets the size of size in tuple (vertical, horizontal).

```
def get_face_landmarks(image, face_detector, shape_predictor):
    faces = face_detector(image, 1)
    num_faces = len(faces)
    if num_faces == 0:
        print("No_face:")
        exit()
    shape = shape_predictor(image, faces[0])
    face_landmarks = np.array([[p.x, p.y] for p in shape.parts()])
    return face_landmarks
```

This function returns the feature 68 points with its x,y value.

```
def get_face_mask(image_size, face_landmarks):
    mask = np.zeros(image_size, dtype=np.uint8)
    points = np.concatenate([face_landmarks[0:16], face_landmarks[26:17:-1]])
    cv2.fillPoly(img=mask, pts=[points], color=255)
    return mask
```

This function collect the points which circles the frame of human's face an fill the space in the circle with the mask, then returns the mask.

```
def get_affine_image(image1, image2, face_landmarks1, face_landmarks2):
    three_points_index = [18, 8, 25]
    M = cv2.getAffineTransform(face_landmarks1[three_points_index].astype(np.float32),
                                face_landmarks2[three_points_index].astype(np.float32))
    dsize = (image2.shape[1], image2.shape[0])
    affine_image = cv2.warpAffine(image1, M, dsize)
    return affine_image.astype(np.uint8)
```

This function is mainly about the Affine transformation, using transformation matrix to acquire the Affine transformed picture after type transformed.

```
def get_mask_center_point(image_mask):
    image_mask_index = np.argwhere(image_mask > 0)
    miny, minx = np.min(image_mask_index, axis=0)
    maxy, maxx = np.max(image_mask_index, axis=0)
    center_point = ((maxx + minx) // 2, (maxy + miny) // 2)
    return center_point
```

This function gets the center point of the mask, and return the center point.

```
def get_mask_union(mask1, mask2):
    mask = np.min([mask1, mask2], axis=0)
    mask = ((cv2.blur(mask, (3, 3)) == 255) * 255).astype(np.uint8)
    mask = cv2.blur(mask, (5, 5)). astype(np.uint8)
    return mask
```

This function is about some detailed process for fitted mask and comfort view.

```
def switch_face ():
    im1 = cv2.imread(image_face_path)
    im1 = cv2.resize(im1, (600, im1.shape[0] * 600 // im1.shape[1]))
    landmarks1 = get_face_landmarks(im1, detector, predictor)
    im1_size = get_image_size(im1)
    im1_mask = get_face_mask(im1_size, landmarks1)
    cap = cv2.VideoCapture(0)
    while True:
        ret_val , im2 = cap.read()  # camera_image
        landmarks2 = get_face_landmarks(im2, detector, predictor)  # 68_face_landmarks
        im2_size = get_image_size(im2)
        im2_mask = get_face_mask(im2_size, landmarks2)

        affine_im1 = get_affine_image(im1, im2, landmarks1, landmarks2)
        affine_im1_mask = get_affine_image(im1_mask, im2, landmarks1, landmarks2)

        union_mask = get_mask_union(im2_mask, affine_im1_mask)
        point = get_mask_center_point(affine_im1_mask)
        seamless_im = cv2.seamlessClone(affine_im1, im2, mask=union_mask, p=point, flags=cv2.NORMAL
```

This function acts as main function, invoking functions defined before. Firstly, it captures the portrait via the camera. Then, it gets the frame of human faces, masks preparations for replacing face. Finally, it makes the Affine transformation and proceeds Poisson fusion, completing the face swap.

3 Difficulties and conclusions of the project

3.1 Difficulties analysis (technical difficulties and problems encountered)

The first problem is relatively easier to figure out compared to later problems. It turns out that I can't play the audio while it's both the problem of the module uninstalled and the incorrect audio format(I cut the audio from my phone and transfer them to the computer in the format of .m4a. I simply change suffix to .wav which is confirmed inexecutable. So I use a online website to change the format.)

Another problem occurs through the whole lab.(Sign...)It's kind of difficult to understand how the module functions work, especially the functions in cv2 module. So it's hard to make information on the internet clear.

3.2 Solutions:spend more time and find more explanations...

3.3 Test results

(At the end of the report.Strange...I insert here...)

Hat and with hat picture

Feature points with frame around the face and face recognition capture

The face swap figure display

4 Lab Summary

Improvements still needed... There are several dissatisfactions in this lab. First of all, latest knowledge about neural network and openvino toolkit taught in the class is not used. Secondly, position recognition was not finished instantly for unfamiliarity and complexity. What's more, the composing of report type is kind of crude, not detailed enough. In addition, It's kind of troublesome to deal with the Chinese input. Maybe later to fix...

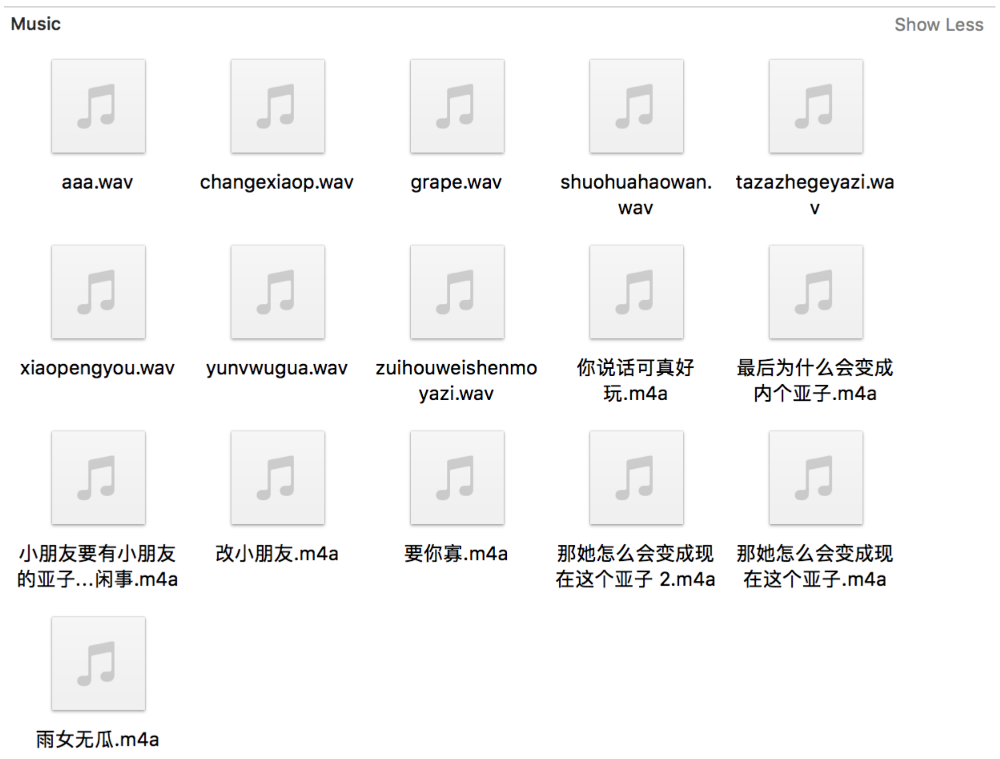


Figure 1: Source figure audio(.m4a format and .wav format)

Last but not least , thanks for the Tea.Xiao, Tea.Chu and Tea.Weiming as they allow me, an external student, to listen to their teaching and provide equipments for me generously.

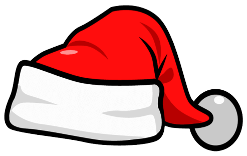


Figure 2: source face

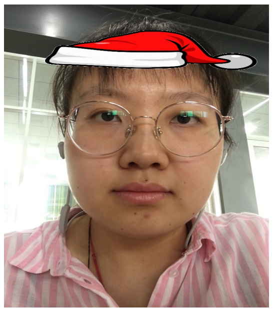


Figure 3: with hat picture

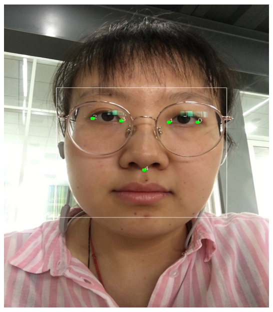


Figure 4: feature points and frame

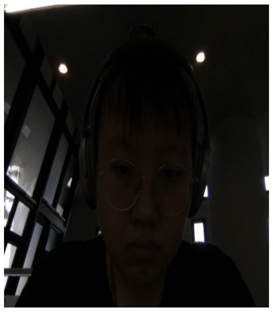


Figure 5: face recognition capture

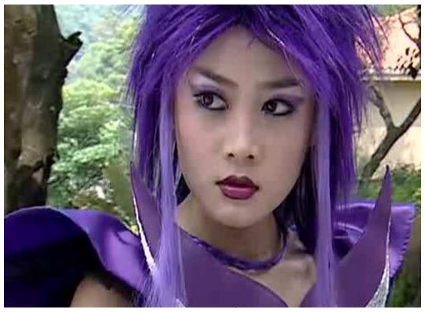


Figure 6: source face

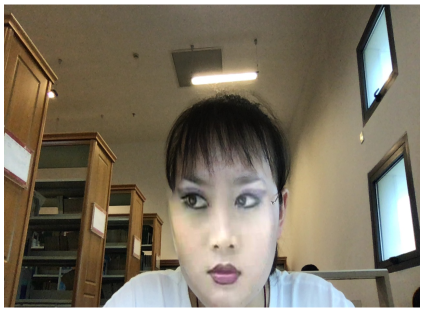


Figure 7: result after swap face