

2022.04.10周报

纪新龙

本周计划任务

联结算法，制作软件

具体完成情况

联结算法完成，软件外壳在做

下周计划任务

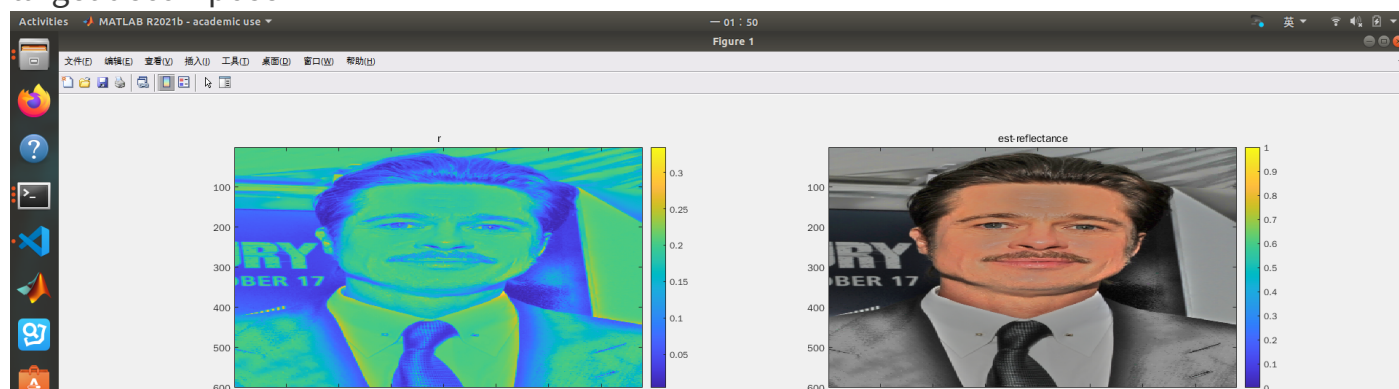
做好软件外壳

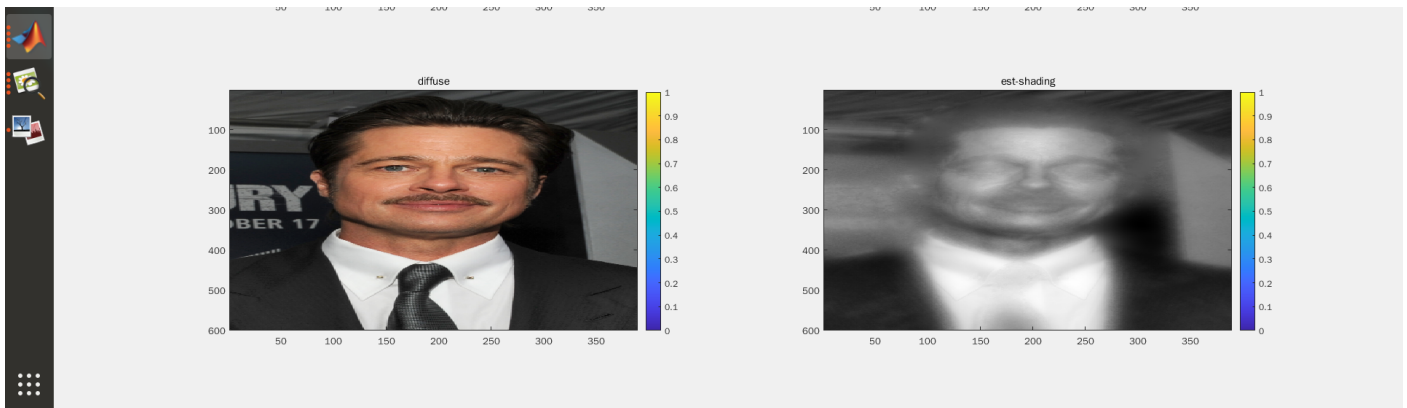
具体完成情况

1. 算法流程示意图

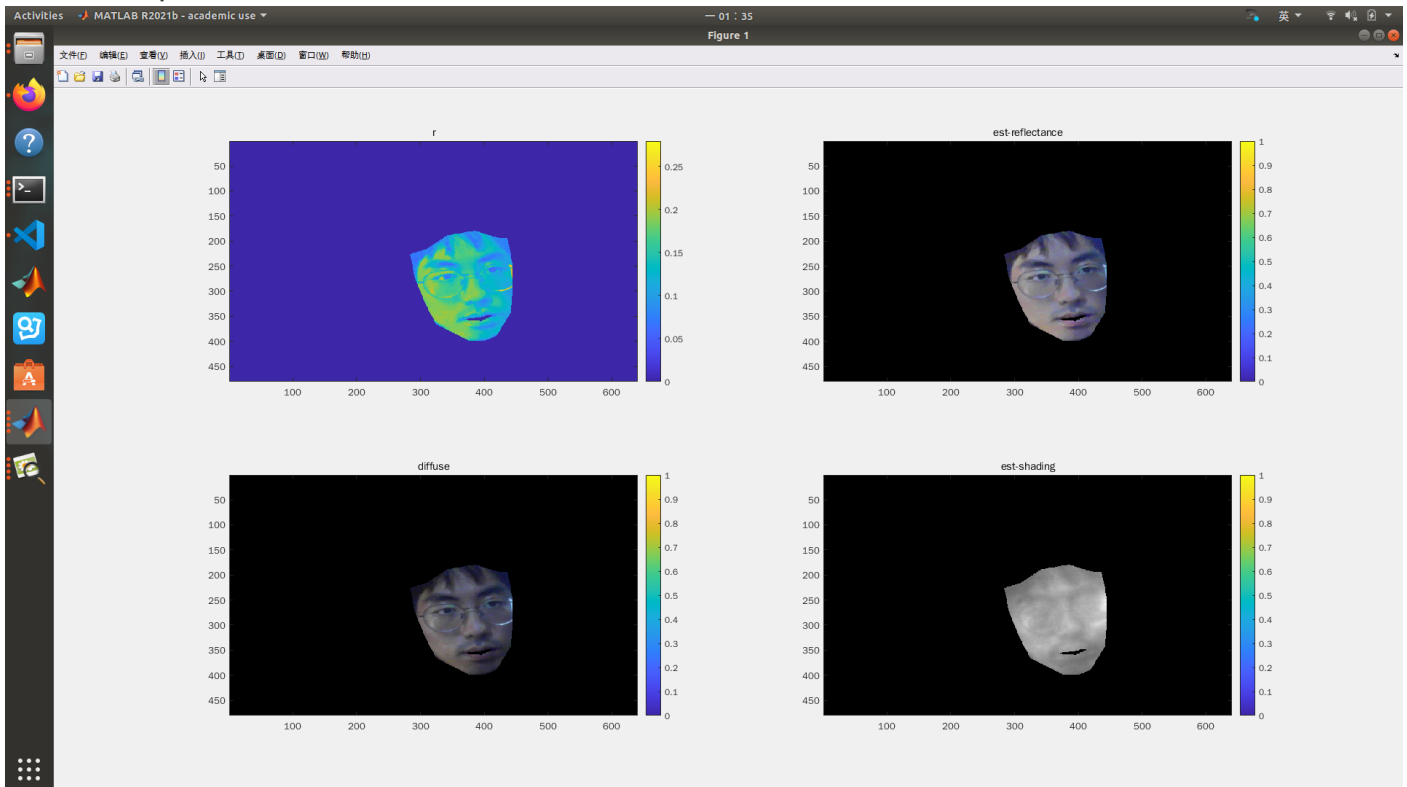
preparations

target decompose





src decompose with mask



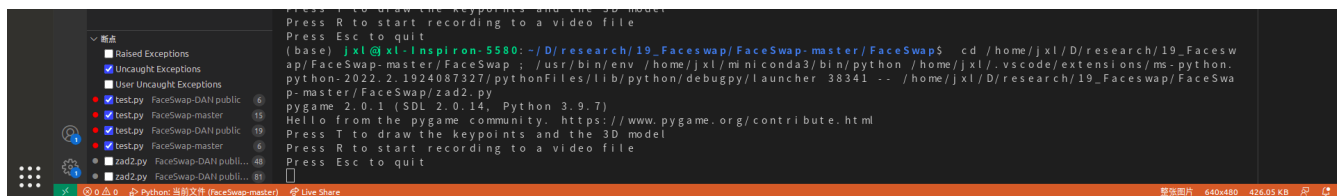
algorithm procedure

1. replace the face in the src with it's reflectance

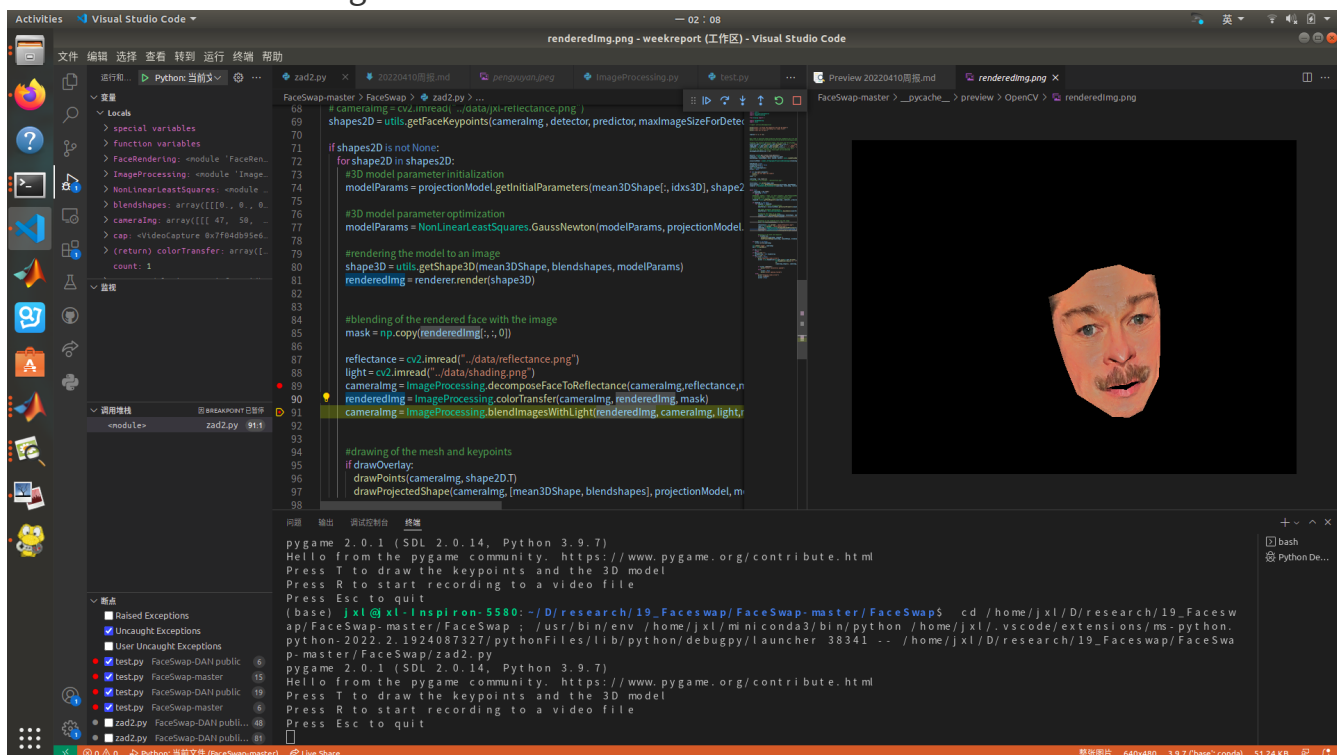
```

FaceSwap-master > FaceSwap > zad2.py > ...
68 # cameralmg = cv2.imread("../data/xx-reflection.png")
69 shapes2D = utils.getFaceKeypoints(cameralmg, detector, predictor, maxImageSizeForDete
70
71 if shapes2D is not None:
72     for shape2D in shapes2D:
73         #3D model parameter initialization
74         modelParams = projectionModel.getInitialParameters(mean3DShape[:, :dxs3D], shape2
75
76         #3D model parameter optimization
77         modelParams = NonLinear.leastSquares.GaussNewton(modelParams, projectionModel
78
79 #rendering the model to an image
80 shape3D = utils.getShape3D(mean3DShape, blendshapes, modelParams)
81 renderedimg = renderer.render(shape3D)
82
83 #blending of the rendered face with the image
84 mask = np.copy(renderedimg[:, :, 0])
85
86 reflectance = cv2.imread("../data/reflectance.png")
87 light = cv2.imread("../data/shading.png")
88
89 cameralmg = ImageProcessing.decomposeFaceToReflectance(cameralmg, reflectance, l
90
91 renderedimg = ImageProcessing.colorTransfer(cameralmg, renderedimg, mask,
92         models.OrthographicProjectionBlendshapes object at 0x7f0544cd9770>
93         > special variables
94         > function variables
95         nBlendshapes: 14
96         nParams: 20
97
98 #drawing of the mesh and keypoints
99 if drawOverlay:
100     drawPoints(cameralmg, shape2D.T)
101     drawProjectedShape(cameralmg, [mean3DShape, blendshapes], projectionModel, m

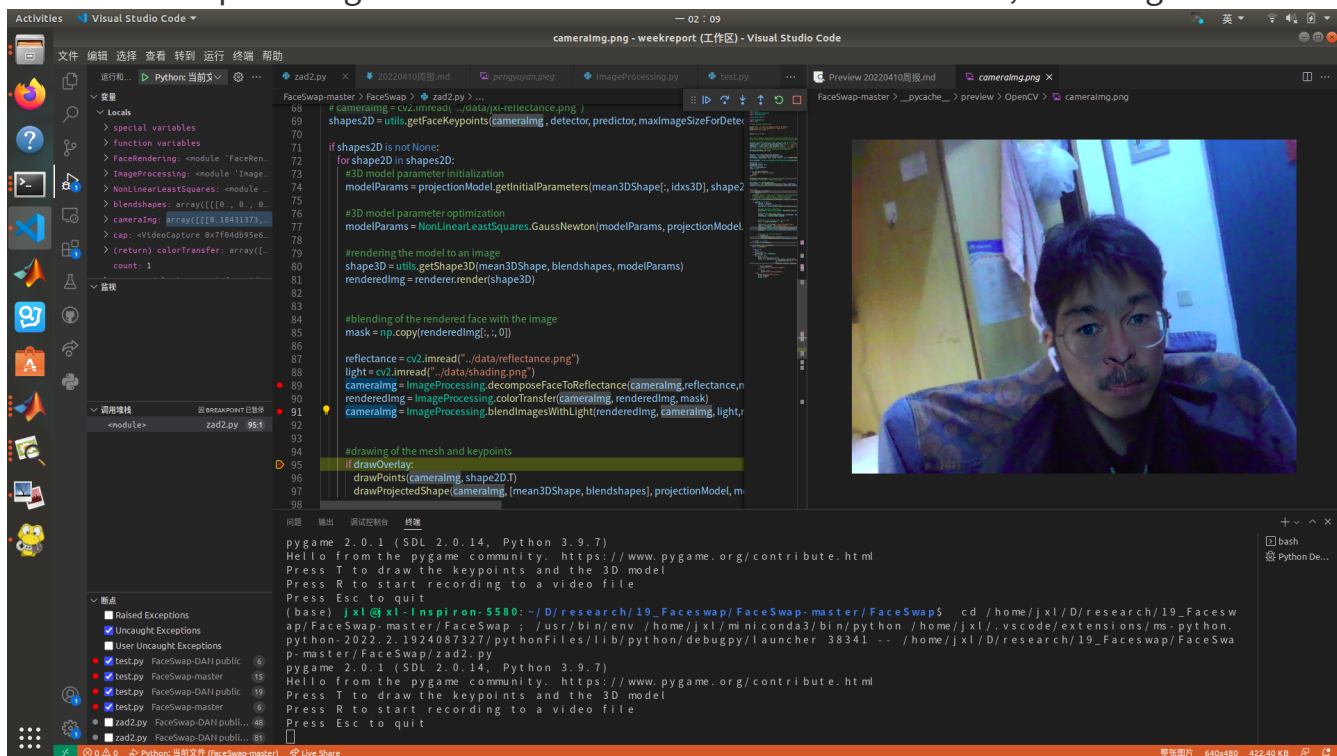
```



2. transfer the color of target face reflectance to src face reflectance



3. blend the morphed target face reflectance WITH src face reflectance, AND relight



result: face identification swapped, with a natural light effect!

2. 视频换脸中出现的问题以及解决方法

2.1 被换脸区域和周围区域像素亮度明显不一样：

原因是本征分解是一个病态的问题，本质是区分出相邻像素之间的差别，而对整个图像所有像素值的绝对大小没有要求。所以可以认为，对分解得到的反射率层、光照层进行整体的尺度放缩后反射率层仍然是之前的反射率层。因此，为了便于反射率层和光照层结果的存储和展示，本征分解的算法模块会在最后将分解到的本征图映射到0~255的空间上存储为结果图片。这种映射成为规范化normalization。

这种规范化可能导致反射率层和光照层相乘之后不再等于原图像，所以在视频中的某些帧中，换脸之后的区域亮于或者暗于周围区域。

我在换脸之前加了一个**重新规范化**的操作，保持反射率层不变，从利用原图和反射率图的商恢复出光照层，这个光照层能保证 $\text{光照层} \times \text{反射率层} = \text{原图}$

2.2 某些帧中，被换脸区域出现奇怪的单色，比如鲜亮的蓝色或者红色：

原因是将double格式的换脸后图像强制类型转化为0~255uint类型时，一些超过255的像素溢出了。我将所有超过255的像素截断为255，避免了这个问题。

2.3 鸡生蛋蛋生鸡问题：

快速本征分解需要一个恰好展示人脸的掩膜，这个掩膜需要展示额头等区域，我是兼用换脸时用的掩膜来做本征分解掩膜的，这样看3DMM操作需要在本征分解之前；但是如果发挥本征分解分离光照对特征点标定的促进作用，需要把本征分解模块放到3DMM之前。

我首先采用两次本征分解的方法，即先对最全图做一个有放缩的快速本征分解，只求能分离掉一些光照干扰，然后在粗糙的反射率图上做换脸的特征点标定、3DMM，这样就有一个针对人脸的掩膜，可以拿这个掩膜对脸部局部区域做精细的本征分解。最后纹理替换、重光照。该方法促进了人脸特征点的标定。

后来我无意中想到另外一种方法，只需要做一次本征分解。即对换脸的遮罩做形态学上的膨胀，使它包含人脸周围一定区域。这个更大的遮罩比换脸的遮罩大一圈，用来做本征分解的遮罩。第*i*帧换脸的遮罩膨胀之后，考虑到帧间人脸的移动不会太大，仍然可以做第*i*+1帧的本征分解遮罩。对第*i*+1帧本征分解之后的图做换脸，利用新产生的换脸遮罩膨胀，更新下一帧要使用的本征分解遮罩。在这个方法的基础上，我调试本征分解的程度，发现分解不充分时特征点标定会失败，分解更充分时特征点标定成功。这说明了本征分解对人脸特征点标定是有意义的，把本征分解模块放到特征点标定之前是对的。同时利用上一帧3DMM膨胀得到的遮罩做这一帧本征分解的遮罩，解决了本征分解遮罩来源问题。

3. 算法流程

- 预处理：对target人脸做本征分解，解出target人脸和标准3D模型对齐的参数。
- 读取src人脸视频第一帧，利用形态学膨胀人脸检测方框得到一个初始的人脸本征分解遮罩

d_Mask_0

i=1, d_Mask = d_Mask_0

- while 有帧可读
 - 读取src人脸视频第*i*帧，利用第*i* - 1帧的保存的d_Mask 对人脸*s*做本征分解，得到反射率层*r*，重新规范化光照层 $l = s/r$
 - 在无光照的*r*上做人脸检测、特征点标定、3D模型的对齐。
 - 得到对齐后的3D模型在二维平面上的投影Mask，此Mask用于脸部纹理替换；对Mask做形态学膨胀，得到新的d_Mask并保存，用于下一帧的本征分解。
 - target人脸像素平均值替换为src人脸像素均值、target脸部纹理替换到src脸部、重光照
- 算法完成