

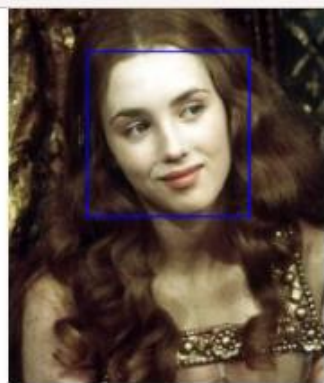
# 人脸处理

邓锦鸾

# OpenCV检测人脸

- 用OpenCV的Harr级联器可以做到简单地检测人脸。

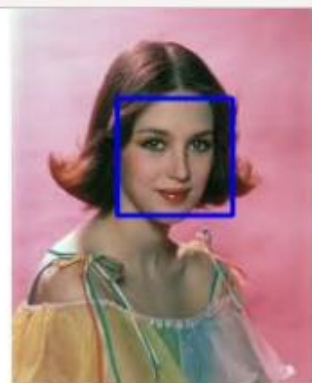
```
OpenCVDetect.py X
OpenCVDetect.py > ...
1  import cv2
2  import numpy as np
3  import os
4  initial_folder='AdjaniJpeg'
5  target_folder='OpenCVDetected'
6  os.makedirs(target_folder, exist_ok=True) #
7  face_cascade = cv2.CascadeClassifier(cv2.d
8
9  for index, filename in enumerate(os.listdir
10     img = cv2.imread(os.path.join(initial_f
11     gray = cv2.cvtColor(img, cv2.COLOR_BGR2
12     faces = face_cascade.detectMultiScale(g
13     if len(faces) > 0: #如果检测到人脸
14         for (x, y, w, h) in faces:
15             cv2.rectangle(img, (x, y), (x+w
16             target_path1 = os.path.join(target_
17             cv2.imwrite(target_path1, img) #保存
18     else:
19         print(f"No face detected in {filena
```



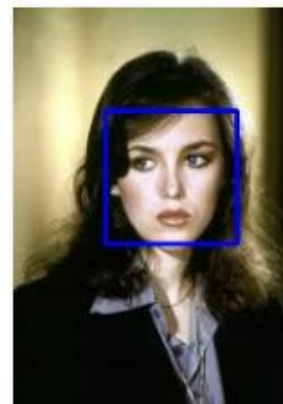
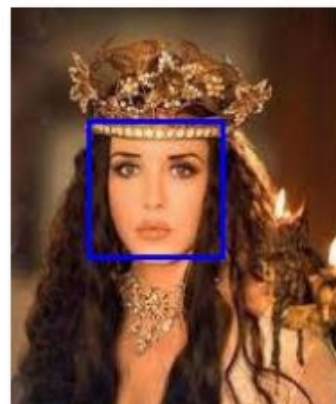
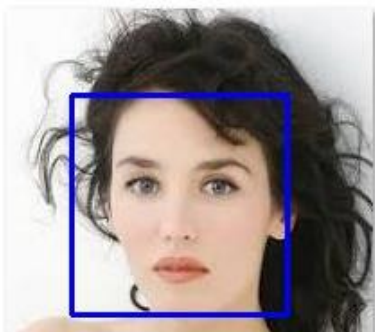
AdjaniFace4.jpeg



AdjaniFace5.jpeg



AdjaniFace6.jpeg



## 为什么还要用insightface?

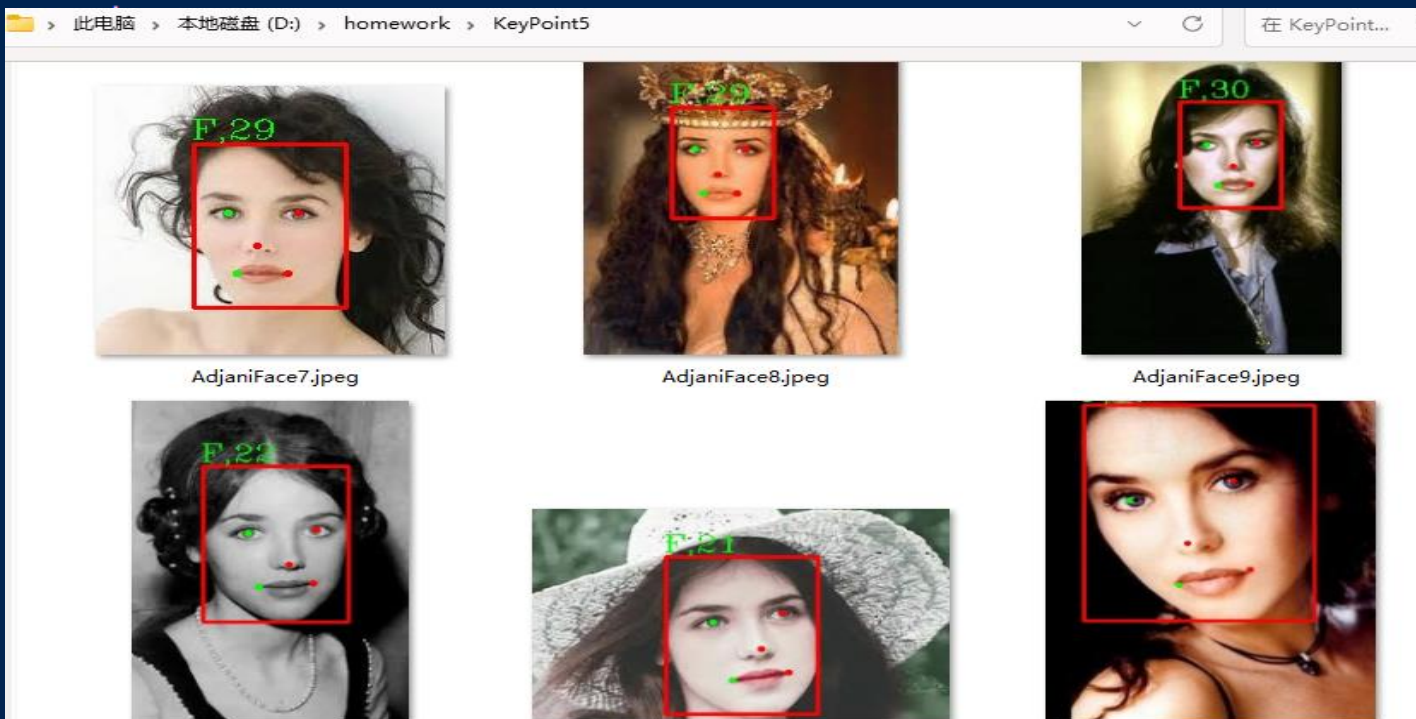
- InsightFace是一个基于深度学习的人脸识别工具，提供了更高级的人脸处理功能。
- 它能够进行高精度的人脸检测，并且能识别人脸的更多细节，如性别、年龄、关键点（例如5个或106个关键点）。InsightFace的这些功能使它非常适合于需要更复杂和精细人脸分析的应用，例如高级人脸识别、人脸属性分析（如情绪识别）、以及更高级的图像处理任务（如换脸技术）。



InsightFace

5points.py > ...

```
1 import insightface
2 from insightface.app import FaceAnalysis
3 from insightface.data import get_image
```



## 5个关键点检测

- 主要利用了insightface库里面的FaceAnalysis类。

```
18 for index,file in enumerate(os.listdir(original_folder1)): #遍历源文件夹中的所有文件
19     filename=os.path.splitext(file)[0]#获取文件名
20     img = get_image(filename) #读取图像
21     face = app.get(img)#使用模型检测人脸和定位特征点 (如眼睛、鼻子、嘴角等)
22     target_path2 = os.path.join(target_folder2, f'AdjaniFace{index+1}.jpg')
23     rimg = app.draw_on(img, face)#在图像上绘制人脸和特征点
24     cv2.imwrite(target_path2, rimg)#保存人脸图像
25
```



```
1 import insightface
2 from insightface.app import FaceAnalysis
```

> 此电脑 > 本地磁盘 (D:) > homework > KeyPoint106



AdjaniFace10.jpg



AdjaniFace11.jpg



AdjaniFace12.jpg



AdjaniFace13.jpg



AdjaniFace14.jpg



AdjaniFace15.jpg

```
24 lmk = np.round(lmk.astype(int)).astype(int) #将人脸关键点坐标由浮点型转换为整数
25 for i in range(lmk.shape[0]): #遍历每个关键点
26     p = tuple(lmk[i]) #将关键点坐标转换为元组
27     cv2.circle(tim, p, 1, color, 1, cv2.LINE_AA) #画点
28 target_path2 = os.path.join(target_folder2, f'A{str(i+1).zfill(3)}.jpg')
29 cv2.imwrite(target_path2, tim) #保存人脸图像
```

Face\data\images'

## 106关键点检测

- 也利用了insightface库里面的FaceAnalysis类，调整了taskname，模型执行的任务就不一样。
- 得到106个关键点，不仅可以做更高精度的处理，可以计算出512个特征向量。
- 人脸识别中的特征向量一般指的是人脸图像在某个特征空间中的表示。

# 人脸识别

```
def get(self, img, face):
    aimg = face_align.norm_crop(img, landmark=face.kps, image_size=self.input_size[0])
    face.embedding = self.get_feat(aimg).flatten()
    return face.embedding

def compute_sim(self, feat1, feat2):
    from numpy.linalg import norm
    feat1 = feat1.ravel()
    feat2 = feat2.ravel()
    sim = np.dot(feat1, feat2) / (norm(feat1) * norm(feat2))
    return sim

def get_feat(self, imgs):
    if not isinstance(imgs, list):
        imgs = [imgs]
    input_size = self.input_size

    blob = cv2.dnn.blobFromImages(imgs, 1.0 / self.input_std, input_size,
                                   (self.input_mean, self.input_mean, self.input_mean), swapRB=True)
    net_out = self.session.run(self.output_names, {self.input_name: blob})[0]
    return net_out
```

- `compute_sim` 方法用于计算两个特征向量之间的相似度，确实可以用于人脸识别中的相似度比较。在人脸识别系统中，通常将提取到的面部特征表示为特征向量。通过比较两个特征向量的相似度，我们可以判断它们是否属于同一个人。
- 在人脸识别的场景中，如果两个特征向量的余弦相似度很高，意味着这两个向量在方向上很接近，从而可以推断出这两个特征向量代表的面部图像很可能属于同一人。相反，如果相似度较低，那么这两个面部图像很可能属于不同的人。通过设定一个阈值来决定是否认为两个特征向量是“相似”的，从而在人脸识别系统中进行身份验证或识别。

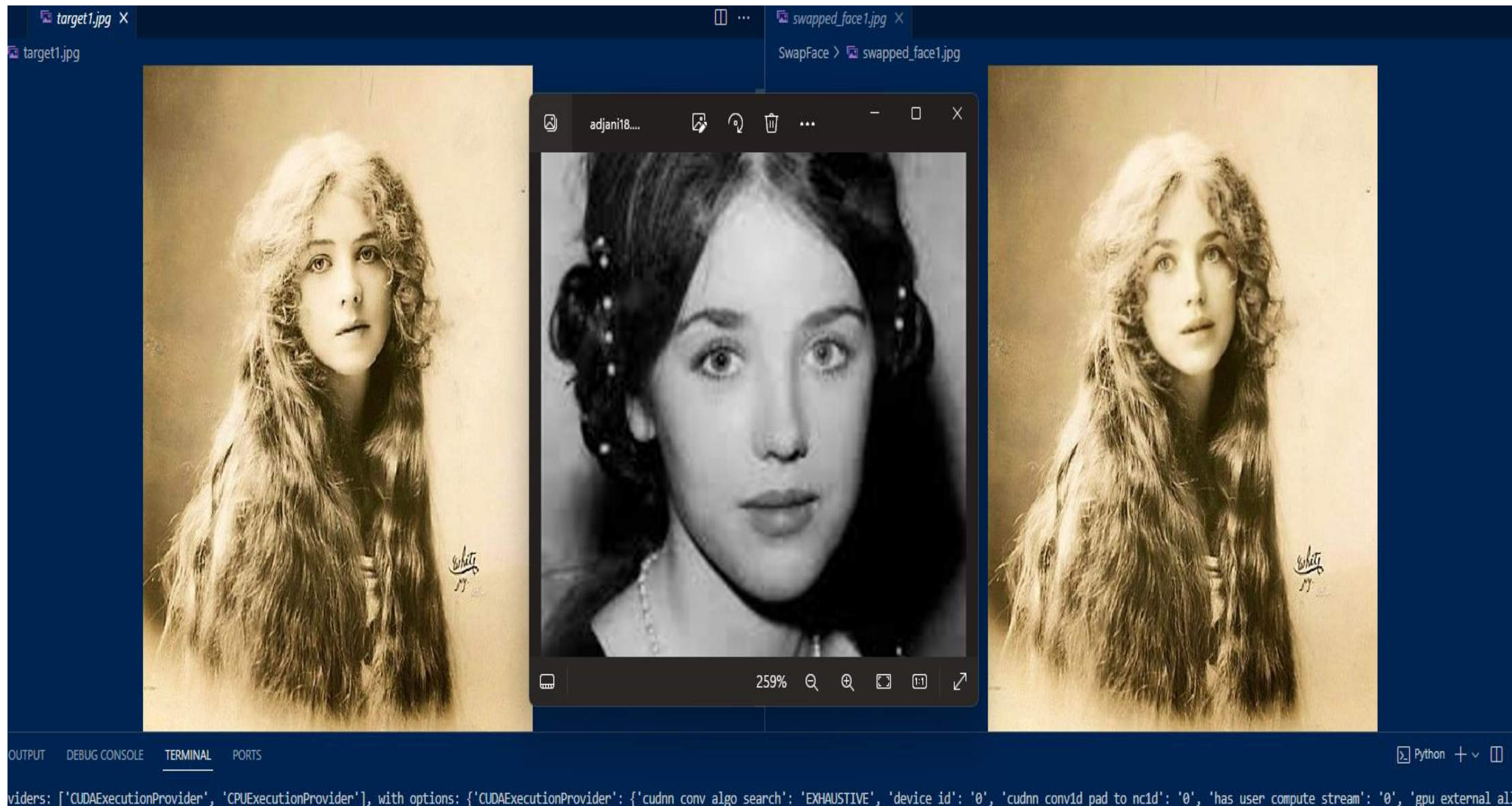
# 换脸（危险的技术）

## 先用FACEANALYSIS类

- `app = FaceAnalysis(providers=['CUDAExecutionProvider','CPUExecutionProvider'])` #初始化 FaceAnalysis 应用，它将自动加载默认的人脸检测和识别模型
- `app.prepare(ctx_id=0,det_size=(640,640))` #使用第一个 GPU,设置输入图像的大小为640\*640

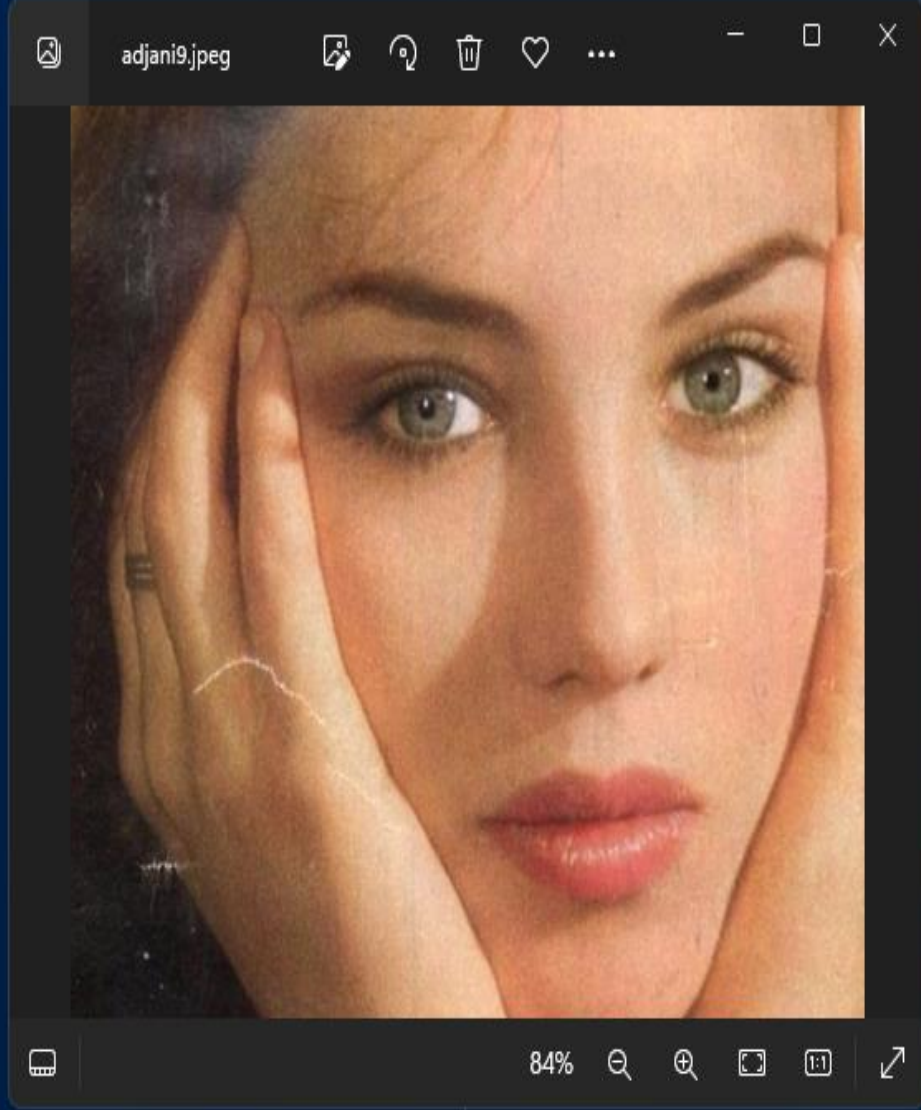
## 再加载 INSWAPPER 模型

- `model_file = r'C:\Users\chuil\.insightface\models\inswapper_128.onnx'`
- `inswapper = INSwapper(model_file=model_file)`
- `fake_img = inswapper.get(target_img, target_face, source_face, paste_back=True)`











# 技巧1：读源代码

```
12 class INSwapper():
13     def __init__(self, model_file=None, session=None):
14         self.model_file = model_file
15         self.session = session
16         model = onnx.load(self.model_file)
17         graph = model.graph
18         self.emap = numpy_helper.to_array(graph.initializer[-1])
19         self.input_mean = 0.0
20         self.input_std = 255.0
21         #print('input mean and std:', model_file, self.input_mean, self.input_std)
22         if self.session is None:
23             self.session = onnxruntime.InferenceSession(self.model_file, None)
24         inputs = self.session.get_inputs()
25         self.input_names = []
26         for inp in inputs:
27             self.input_names.append(inp.name)
28         outputs = self.session.get_outputs()
29         output_names = []
30         for out in outputs:
31             output_names.append(out.name)
32         self.output_names = output_names
33         assert len(self.output_names)==1
34         output_shape = outputs[0].shape
35         input_cfg = inputs[0]
36         input_shape = input_cfg.shape
37         self.input_shape = input_shape
38         print('inswapper-shape:', self.input_shape)
39         self.input_size = tuple(input_shape[2:4][::-1])
40
41     def forward(self, img, latent):
42         img = (img - self.input_mean) / self.input_std
43         pred = self.session.run(self.output_names, {self.input_names[0]: img, self.input_names[1]: latent})
44         return pred
45
46     def get(self, img, target_face, source_face, paste_back=True):
47         aimg, M = face_align.norm_crop2(img, target_face.kps, self.input_size[0])
48         blob = cv2.dnn.blobFromImage(aimg, 1.0 / self.input_std, self.input_size, (self.input_mean, self.input_mean, self.input_mean))
49         latent = source_face.normed_embedding.reshape((1,-1))
50         latent = np.dot(latent, self.emap)
51         latent /= np.linalg.norm(latent)
52         pred = self.session.run(self.output_names, {self.input_names[0]: blob, self.input_names[1]: latent})
53         #print(latent.shape, latent.dtype, pred.shape)
```

```
1 import insightface
2 from insightface.app import FaceAnalysis
3 from insightface.data import get_image
4 from insightface.model_zoo import INSwapper
5 import cv2
6 import onnxruntime
7 import numpy as np
8 import os
9
10 original_folder=r'C:\Users\chuil\anaconda3\envs\insightface\Lib\site-packages\insightface\data\images'#源图片文件夹
11 target_folder='SwapFace'#目标图片文件夹
12 os.makedirs(target_folder, exist_ok=True) #确保目标文件夹存在
13
14 app = FaceAnalysis(providers=['CUDAExecutionProvider','CPUExecutionProvider'])#初始化 FaceAnalysis 应用, 它将自动加载 GPU, 设置输入图像的大小为640*640
15 app.prepare(ctx_id=0, det_size=(640,640)) #使用第一个 GPU, 设置输入图像的大小为640*640
16
17 source_filenames=['adjani18','adjani7','adjani9']
18 target_filenames=['target1','target2','target3']
19
20 for i in range(3):
21     #读取源图片人脸
22     source_filename=source_filenames[i]
23     source_img = get_image(source_filename)
24     source_faces = app.get(source_img)
25     if len(source_faces)>0:
26         source_face=source_faces[0]
27     #读取目标图片人脸
28     target_filename = target_filenames[i]
29     target_img = get_image(target_filename)
30     target_faces = app.get(target_img)
31     if len(target_faces)>0:
32         target_face=target_faces[0]
33     # 加载 INSwapper 模型
34     model_file = r'C:\Users\chuil\insightface\models\inswapper_128.onnx'
35     inswapper = INSwapper(model_file=model_file)
36     fake_img = inswapper.get(target_img, target_face, source_face, paste_back=True)
37     #保存换脸图片
38     cv2.imwrite(os.path.join(target_folder, f'swapped_face{i+1}.jpg'), fake_img)
```

## 技巧2：让chatgpt帮忙读源代码

Insightface Expert

Published · senseurl.x

CreateConfigure

Insightface Expert

Description

Expert in Python's insightface library for facial image processing.

Instructions

You are a GPT specializing in facial image processing using the insightface library in Python. Your role is to assist in reading, understanding, and applying the insightface library effectively. You should provide clear explanations of the library's functions, help troubleshoot issues, and offer guidance on best practices for using the library in Python projects.

Conversation starters

How do I install insightface?

Explain the main functions of insightface.

How to handle errors in insightface?

Best practices for using insightface in Python?

Knowledge

If you upload files under Knowledge, conversations with your GPT may include file contents. Files can be downloaded when Code Interpreter is enabled

The following files are only available for Code Interpreter:

insightface.zip

Zip Archive



### Insightface Expert

当然可以。我在`insightface`目录中找到了一些包含"embedding"关键字的文件。以下是其中的一些文件：

1. `app`目录下的`common.py`。
2. `model\_zoo`目录下的`arcface\_onnx.py`。
3. 同样在`model\_zoo`目录下的`inswapper.py`。
4. `thirdparty/face3d/mesh\_numpy`目录下的`render.py`。

其中，`model\_zoo`目录下的`arcface\_onnx.py`特别值得关注，因为ArcFace是一个著名的模型，常用于面部识别任务中生成特征向量（embeddings）。这个文件可能包含加载ArcFace模型和使用它从面部图像中提取特征向量的函数或方法。

您希望我进一步探索这些文件的内容，以提供更具体的关于特征向量生成过程的信息吗？



### You

取得了初步进展，你真棒！我希望你进一步探索这些文件的内容，以提供更具体的关于特征向量生成过程的信息



### Insightface Expert

`arcface\_onnx.py`文件提供了一个名为`ArcFaceONNX`的类，这个类用于加载和使用ArcFace模型。ArcFace模型广泛用于面部识别，能够从面部图像中提取特征向量（即 embeddings）。

在这个文件中，`ArcFaceONNX`类提供了以下功能：