CS159-Python Programming

项目报告

COURSE PAPER

项目名称: 《再见已是画中人》

团队名称 代码敲不队

队员姓名 池豪 冒雨婷 吴君瑜

授课教师 鲍扬

2020.06.11

目录

第一章、项目背景

第 1 节 选题原因

随着图像处理的技术不断发达，众多图像如表情包、“PS”图片等已经充斥了社交网络，更新了人们审美观念并为社交提供了新型方式。图像处理软件的各种算法和齐全的功能，使用户摇身一变成为“大艺术家”。在自己使用图像处理软件创作图像以及向周围的人展示的过程中，用户收获了无数的乐趣。

因此我们非常希望能够完成一个基于图像处理的有趣的项目，尝试成为一个图像处理软件的开发者，帮助用户实现图形处理的过程。而在讨论和搜集想法的过程中，我们发现了基于。因此，我们希望利用这种 Python 编程软件和所提供的第三方库，实现人像截取，并且通过人机交互界面，将截取下来的人像放置在用户想要的位置，生成一张新的合成图片。

最后经过小组讨论，选定项目题为《再见已是画中人》，效果和介绍详见下一节介绍。

第 2 节 项目介绍

在项目《再见已是画中人》中，程序可以从用户给定的一张含单个人像的图片中捕捉人像，并在 GUI 下通过拖拽、缩放方式将人像覆盖到用户提供的世界名画或风景画素材背景的指定位置，生成合成图片。

要实现上述目标，程序需要有两个板块组成。

首先，第一个板块可以通过算法从用户给定的图片中捕捉人像。捕捉好的人像图片会被保存到事先准备好的路径。

STEP1 效果展示：

原图 捕捉图

其次，第二个板块，通过 PyQt5 包实现，用户可以在 GUI 操作背景下通过拖拽、缩放和还原等方式将人像覆盖到用户提供的世界名画或风景画素材背景的指定位置，生成合成图片。通过运行程序，用户可以使用左键实行拖拽、滚轮实行缩放以及右键实行还原，并按任意键保存图片到事先指定好的路径。

STEP2 效果展示：

背景 人像图片 合成图（大小位置由用户决定）

第 3 节、代码运行须知和教程

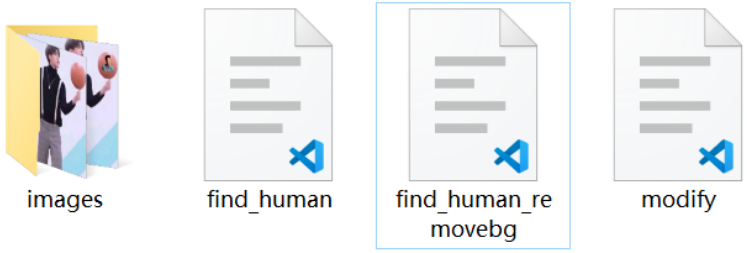
本节作者： 池豪

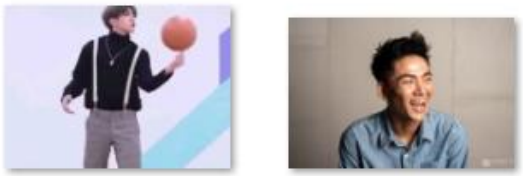
在我们的项目中，因为输出结果、路径更名等多种考虑，我们决定使用三个独立的脚本运行。以确保用户能够运行结果。将代码文件包打开后，会看到一个 images 文件夹和若干脚本代码。其中 images 文件夹用来保存中间产物，放入原始文件以及最后获取输出文件。

STEP1 在运行程序时需要调用 numpy，cv2，tkinter，（removebg，优化备选），PyQt5，PIL，sys

一般情况下 cv2，removebg，PyQt5 是第三方库，需要手动通过 pip install 命令进行安装，在确保上述包正确安装后才可以正确运行程序。

STEP2 把背景图片重命名为 a 并放在 images 文件夹中，把含人像的图片重命名为 b 并放在 images 文件夹中。





注意：默认用户提供图片为 jpg 格式，否则，用户需要打开 find_human.py, modify.py 修改路径方式（寻找注释中的打开图片等类似表述的行，并把该行的 x.jpg 转化为 x.其他格式）。若不修正，图片无法正常打开。一般情况下，都是 jpg 形式的图片，无需修改。

STEP3 在命令行终端将工作路径修改至代码和 images 所包含的文件夹。

STEP4 依次运行 find_human.py，modify.py。

STEP5 在 images 文件夹里面找到输出结果和中间产物。

第二章、人像分离

本章作者：吴君瑜

在“再见已是画中人”这项团队项目中，我负责的是整个环节的第二部分，即是人像背景分离这一部分。在这一部分中，我们运用了基于 opencv 库的 grabcut 算法以及 remove.bg 智能人像分离 api，两种不同的算法来完成。并会比较两种算法的效果。接下来，我将分两个小节介绍它们。

第 1 节、基于 opencv 的 Grabcut 算法

- **简介 Grabcut 算法**：Grabcut 算法是一种基于图论的图像分割办法。输入图像时，它会率先对图像的每一个像素进行标记。接着，它会使用高斯混合模型对前景和背景建模。图中会有 Source_node 和 Sink_node 两个节点，前景与 Source_node 相关，背景与 Sink_node 相关，而像素会连结到哪个节点取决于权值（即是运用高斯混合模型计算出的属于前景或背景的概率）。紧接着，Grabcut 会利用 mincut 算法对图像进行分割，也就是把判定为 Source_node 的像素作为前景裁出，判定为 Sink_node 的像素作为背景裁出，达到人像与背景分离的效果。

- **本次项目所用到的参数：**

img	输入图像
rect	设定前景范围的矩形，格式为（x,y,w,h），分别为左上角坐标和宽度，高度
setMouseCallback()	鼠标响应事件函数
mask	掩模图像，用来区分前景和背景。 cv2.GC_BGD（0）背景 cv2.GC_FGD（1）前景

	cv2.GC_PR_BGD（2）背景 cv2.GC_PR_FGD（3）前景
mode	指示 grabcut 进行哪种模式的操作 GC_INIT_WITH_RECT（=0），用矩形窗初始化 GrabCut； GC_INIT_WITH_MASK（=1），用掩码图像初始化 GrabCut
iterCount	迭代次数

- **代码截图与详解：**

```
import numpy as np
import cv2
from tkinter import messagebox

#定义全局变量
n = 0    #定义鼠标按下的次数
ix = 0   # x,y 坐标的临时存储
iy = 0
rect = (0,0,0,0) #前景区域
#鼠标回调函数
def draw_rectangle(event,x,y,flags,param):
    global n,ix,iy,rect
    if event==cv2.EVENT_LBUTTONDOWN:
        if n==0:###初次保存坐标值
            n+=1
            ix,iy = x,y
        else:         ###第二次显示矩形
            n+=1
            rect = (ix,iy,(x-ix),(y-iy))#前景区域
```

```
#读取图像
img = cv2.imread('./images/b.jpg')##打开图片
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
#选择区域 左上到右下矩形
cv2.namedWindow("img")
cv2.setMouseCallback("img",draw_rectangle)#绑定鼠标
messagebox.showinfo("提示","接下来，请选定所需截下的人物的定位点")
while(n != 2):
    cv2.imshow("img",img)
    cv2.waitKey(2)
```

```
#前景提取
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask2[:,:,np.newaxis]
img = cv2.merge((img, 255*mask2))
#显示图像
cv2.imshow("img",img)####img就是输出的图片
cv2.imwrite('./images/b1.png',img) ###导出图片
cv2.waitKey()
cv2.destroyAllWindows()
```

- **效果展示：**

（左图素材来自网络）

第 2 节、RemoveBG 工具（优化）

- **RemoveBG 简介：**Remove.bg 是基于 Python、Ruby 和深度学习技术开发而来的一款 AI 自动抠图移除背景工具。它可以运用强大的 AI 算法来自动识别前景与背景图，并且将背景图分离，非常简便快捷。

- **代码截图：**

```
from removebg import RemoveBg
####先到removebg注册，pip install removebg
rmbg= RemoveBg("1iF1en4RcZHQUHmpivRi3CTN","error.log")
###前面是api密钥
rmbg.remove_background_from_img_file("C:/Program Files/chengshe/week13/testimg3.jpg")
###输入图片地址，在同文件夹找得到输出的图片
```

- **代码截图：**


可以看到使用 Removebg 法得到的图片边缘有着显著的优化。

第三章、GUI 图片导入和输出

本章作者：冒雨婷

在“再见已是画中人”这项团队项目中，我负责的是整个环节的第一部分与最后一部分，即是插入图片与保存图片这一部分。在这一部分中，我们运用了 PyQt5 中的控件 QPixmap 和 QPainter 与键盘事件。接下来，我将详细介绍他们。

第 1 节、插入图片与背景

- **PyQt5 中的控件 QPixmap**：QPixmap 类用于绘图设备的图像显示，它可以作为一个 QPainterDevice 对象，也可以加载到一个控件中，通常是标签或者按钮，用于在标签或按钮上显示图像

  QPixmap 可以读取的图像文件类型有 BMP，PNG，GIF，JPG 等

- **QPainter：**绘图系统由 QPainter 完成具体的绘制操作，QPainter 类提供了大量高度优化的函数来完成 GUI 所需要的大部分工作。它可以绘制一切想要的图形，从最简单的一条直线到其他任何复杂的图形，例如：点、线、矩形、弧形等。QPainter 也支持一些高级特性，例如**反走样**。

- **本部分中，我们希望以背景图片 b 的大小来设置窗口大小，并且背景大小可以随着窗口的变化进行变化，而插入的图片 a 以其原始形状出现在背景图片 b 之上。**

- **本次项目所用到的参数：**

img	输入图像
rect	设定前景范围的矩形，格式为（x,y,w,h），分别为左上角坐标和宽度，高度
keyPressEvent	键盘事件
resize	主要用于调整图像的大小
geometry	Geometry 属性保存部件相对于其父级对象的位置和大小

- **代码截图与详解：**

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PIL import ImageGrab
import sys
```

```
'''定义主窗口'''
class myWindow(QWidget):

    def __init__(self):

        super(myWindow, self).__init__()
        self.setWindowTitle("图片编辑")  # 设定窗口名称

        self.img1 = QPixmap("./images/a.jpg")  # 导入背景图片a
        self.resize(self.img1.width(), self.img1.height())  # 提取其长宽数据，作为操作窗口的大小

        self.img2 = QPixmap('./images/b1.png')  # 导入人像图片b
        self.scaledimg2 = self.img2  # 初始化人像图片b，为后续移动位置做准备
        self.ori = QPoint(0, 0)  # 初始化人像图片b坐标（0，0）

        self.LeftPressed = bool(False)  # 鼠标左键标志位
        self.isImgLabelArea = bool(True)  # 鼠标进入label图片显示区域
```

```
    '''重载绘图：动态绘图'''
    def paintEvent(self, event):

        painter = QPainter(self)
        pixmap = QPixmap("./images/a.jpg")  # 用背景方法导入背景图片a
        painter.drawPixmap(self.rect(), pixmap)  # 设置背景图片a的大小，使其与窗口的大小相适应

        self.imgPainter = QPainter()
        self.imgPainter.begin(self)
        self.imgPainter.drawPixmap(self.ori, self.scaledimg2)  # 导入人物图像b并显示在初始化的坐标，这里是（0，0）
        self.imgPainter.end()
```

第 2 节、截图当前活动窗口并保存到指定路径

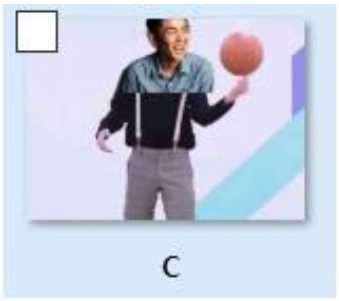
- **本部分中，我们希望在将图片 b 调整到合适位置与大小后，按下键盘上某个键后可以保存效果图到指定文件夹。**

- **代码截图与详解：**

```
48      '''重载键盘事件'''
49      def keyPressEvent(self, QKeyEvent):
50          print("键盘某个按键被按下，执行保存图片功能")
51          print()
52          #temp = self.scaledImg   拿到缩放的图片对象
53          file_path = "./images/c.jpg"  # 合成图片的名字路径
54          #temp.save(file_path)  # 保存到路径对象
55          #print(self.rect())
56          #print(self.geometry())
57          temp_rect = (self.geometry().left(), self.geometry().top(), self.geometry().left()+ self.geometry().width(),
58                       self.geometry().top()+self.geometry().height())
59          src_image = ImageGrab.grab(temp_rect)
60          # src_image = ImageGrab.grab((game_rect[0] + 9, game_rect[1] + 190, game_rect[2] - 9, game_rect[1] + 190 + 450))
61          src_image.save(file_path)
```

- **效果图：**

（导入图片）



（输出结果）

由此可见，截取的人像和风景图片已经出现在了界面上。窗口的大小和界面相互吻合，并且人像覆盖在了风景图片的上方，而经过操作后得到的图片也得到了保存，在指定的文件夹里。

第四章、GUI 图片操作

本章作者：池豪

在“再见已是画中人”这项团队项目中，我负责的是整个环节的 GUI 图片操作部分，即是对图片左键进行拖拽，右键还原，滚轮进行缩放，从而将人像图片放在用户期望的位置。在这一部分中，我们运用了 PyQt5 中的控件 QPixmap 和 QPainter 与鼠标滚轮事件的定义。接下来，我将详细介绍算法和代码实现。

第 1 节、代码原理介绍

通过定义事件，如 mousePressEvent，以及 Qt 里面自带的检验鼠标输入的方法，一旦检测到用户对鼠标进行操作，程序将会提示用户相应的操作被执行。同时程序会更改图片坐标、大小等属性值，并重绘图片。如果用户对操作的结果不满意，可以通过右键还原到初始状态。关于 QPixmap 和 QPainter 的讨论已经足够，接下来将对如何实行各种操作方法展开详细介绍。

基本思想是运用 PyQt5 中的方法来抓取用户的鼠标事件，在保存初始值数据的情况下，记录鼠标移动的偏移量作为图片的偏移量；记录鼠标滚轮的滚动次数作为图片缩放的依据。完成后运用方法重绘图片，同时更新鼠标先前位置

到当前位置，以确保下次拖拽时是在上一次拖拽的基础之上。

需要强调的是，实现拖拽并不是通过拖拽图片对象本身，而是改变图片加入到画板时的坐标，换句话说，其实相当于拖拽坐标系的原点。由于背景图片不和人像图片共用坐标系（详见上一章，QPoint 坐标系仅对人像图片有效），所以最后能达成效果。

本板块用到的方法：

pos 方法	获取鼠标所在位置
width/height 方法	获取图片的宽和高
angleDelta 方法	获取滚轮移动信息
Left/RightButton 方法	获取左右键按下信息
LeftPressed 方法	获取左键持续按下信息
repaint 方法	重绘图片（更新图片）

第 2 节、代码详细解读

```
'''重载鼠标左键按下事件(单击)'''
def mousePressEvent(self, event):
    if event.button() == Qt.LeftButton:  # 左键按下
        print("鼠标左键按下，按住左键并移动鼠标对图片进行拖拽操作")
        print()
        self.LeftPressed = True  # 左键标志位置Ture
        self.presentMousePosition = event.pos()  # 获取鼠标当前位置

'''重载鼠标键松开事件（左右键）'''
def mouseReleaseEvent(self, event):
    if event.button() == Qt.LeftButton:  # 左键释放
        print("鼠标左键松开，拖拽操作完毕")
        print()
        self.LeftPressed = False  # 左键标志位置False

    elif event.button() == Qt.RightButton:  # 右键释放
        print("鼠标右键松开，重置图片到初始位置")
        print()
        self.ori = QPoint(0, 0)  # 坐标返回到初值（0，0）
        self.scaledimg2 = self.img2  # 图片返回到原图状态
        self.repaint()  # 重绘

'''重载鼠标拖动事件（左键）'''
def mouseMoveEvent(self, event):
    if self.LeftPressed:  # 左键持续按下
        self.delta = event.pos() - self.presentMousePosition  # 单次偏移量 =鼠标当前位置-鼠标先前位置
        self.ori += self.delta  # 更新图片坐标，实现拖拽
        self.presentMousePosition = event.pos()  # 更新当前鼠标在窗口上的位置，保障下次拖拽图片时从上一次移动位置开始
        self.repaint()  # 重绘
```

```
    '''重载滚轮滚动事件'''
    def wheelEvent(self, event):
        angle = event.angleDelta() / 8  # 返回QPoint对象，为滚轮转过的数值，单位为1/8度
        angleY = angle.y()  # 竖直滚过的距离
        a1 = self.img2.width() # 提取人物图像b的宽和高，保证缩放时能够等比例缩放
        a2 = self.img2.height()
        if angleY > 0:  # 滚轮上滚
            print("鼠标滚轮上滚，图片放大")
            print()
            self.scaledimg2 = self.img2.scaled(self.scaledimg2.width() + 10,
                                               (self.scaledimg2.width() + 10)/a1*a2)  # 等比缩放，每次宽的长度改变10

            self.repaint()  # 重绘

        else:  # 滚轮下滚（代码基本同上滚，仅把+变成-）
            print("鼠标滚轮下滚，图片缩小")
            print()
            self.scaledimg2 = self.img2.scaled(self.scaledimg2.width() - 10,
                                               (self.scaledimg2.width() - 10)/a1*a2)  # 等比缩放，每次宽的长度改变10

            self.repaint()  # 重绘
```

```
'''主函数'''
if __name__ == "__main__":
    print("""
    在本程序中，将通过以下指令对图片进行操作：\
    鼠标左键对人物图片进行拖拽操作，\n
    用鼠标滚轮对人物图片进行缩放操作，\n
    用鼠标右键重置到初始，\n
    用键盘任意键保存图到指定路径\n
    """)

    app = QtWidgets.QApplication(sys.argv)
    myshow = myWindow()
    myshow.show()  #打开窗口
    sys.exit(app.exec_())
```

最后，运行主函数。提示用户可以进行的操作，并打开窗口。

程序运行效果：

初始位置

拖拽图片

缩放图片（放大）

缩放图片（缩小）

第五章、总结与回顾

第 1 节、局限和改进

局限 1

本程序对输入文件的格式和路径都有一定的要求，如果不能按照程序指定的路径输入文件，会导致程序无法正确导入图片。这让程序的使用难度提升，可移植性下降。而且，如果用户有多张图片要处理，每次都要更改文件名显然是非常不方便的。

解决方法：通过另一个 GUI 窗口让用户导入风景画和人像（例如 word 文档文件打开）。由于 PyQt5 可以读取 png 和 jpg 等多种图片格式，所以不存在读写错误，且可以使用户使用体验提升。

局限 2

图片操作的功能比较单一，只有缩放、拖拽和还原，对于图像编辑软件里的更多功能如旋转，拉伸等操作不能得到较好的还原。如果程序能够进行旋转和拉升操作，世界名画的可观赏性会得到显著提升。

解决方法：通过定义更多的键盘事件和对图片进行操作的算法，可以完成上述要求的操作。

局限 3

我们曾经计划在图片操作完成后进行更加有趣的风格迁移，然而风格迁移对硬件设备的要求较高，生成图片的时间也不少。出于成本、可操作性等多种因素的考虑，我们并没有在最终代码中实现更具观赏性的风格迁移。这让本项目的趣味性受到了一定的限制。

解决方法：无

局限 4

人像分离里面的 GRABCUT 方法是根据图像颜色来区分每一个像素，对于背景和前景颜色相近的图片，它没有办法完全识别。我们曾经考虑过使用交互式的方法，手动弥补缺陷，无奈由于时间的原因，最后并没有实现算法优化。作为替代，我们使用了 Removebg，来代替优化的结果。

（可以看到，背景颜色的相近使得人物抓取效果不佳）

解决方法：采用交互式方法不断迭代、更新，或者采用其他算法。

第 2 节、队员体验和感受

队员 吴君瑜：

本次团队作业加深了我对 Python 的认识，也在一定程度上增强了一些我运用 Python 的能力。虽然由于我的能力不足，未能把我所负责的部分做到最完美，但我仍觉得，这是一次很好的体验。

我所负责的部分是人像分离，人像分离这块，我们采用的是 GRABCUT 这个算法。由于这个算法本身是根据图像颜色来区分每一个像素的，因此，对于背景和前景颜色相近的图片，它没法完全识别。我原先还想通过交互式的方法来手动弥补这个缺陷，奈何时间不太够，而且这类方法可能会大大增加操作的不便利性，最后放弃了。这是一个遗憾吧，没有做到尽善尽美。

最后，我要感谢队长和另一位队员，大家都非常负责积极，能和他们合作完成这次团队作业，我感到非常愉快。

队员 冒雨婷：

本次团队作业中，我主要负责插入图片、背景与保存图片部分，此部分内容不是特别困难，但在某些细节上值得推敲，以获得更好的运行效果。

首先，此次团队作业锻炼了我在完成一个小项目时理解“提问的智慧”与解决问题的能力，不同于平时的 lab，仅有某一明确的目的，这次编程是一个连续的过程，在后续的使用过程中不断地发现某部分缺憾，然后再进行修改完善，边做边改的过程极大地锻炼了我的编程能力。同时，此次作业也让我意识到团队合作的力量，由于个人水平较差，一些问题考虑不全面或 bug 无法独立解决的时候，感谢团队内的其他人给予我的帮助与指导。

虽然这次项目的最终效果有所缺憾，未能完全达到我们预期的效果，但我认为，我们一同探索的过程也是非常有意义的，从这种意义上，我认为，大作业的设计目的也达到了。

队长 池豪：

在学期开始的时候我就想做一个基于图形处理或者是动画的项目，不仅具有很高的观赏价值，也能大幅度提升完成后的成就感。最后在组员的讨论中，我们舍去了已经有很多人研究过的字节动画、傅里叶动画等等项目，转而实行最基本的“PS”图像处理和编辑。

然而，项目的进程并不是那么顺利，由于线上教学的特殊性，沟通的效率会不如线下小组会议，因此没当遇到困难和代码运行失败，我们会在微信群里面互相指出、纠正。

在我负责的板块中，定义鼠标拖拽和缩放是相对有难度的板块，而书写这段代码的过程也加深了我对几何、坐标系以及类方法的理解，作为一次实战经验，对于我今后面向对象的编程起到了很大的帮助。

十分可惜的是，我们小组最初的风格迁移想法由于时间、精力和硬件条件的约束没有办法完成，这使我们的项目在成果上不那么富有趣味。然而，大家齐心协力共同完成的过程这一本身，已经远远超过了结果产出的趣味性。

最后我要感谢两位队员对我的支持和对整个小组的贡献，她们的建议让我对整个小队进程有了更合理的把握，而且她们的执行力、耐心以及责任心都是小队项目完成过程中不可缺失的一环。

全体队员最后感谢本学期的课程指导老师鲍杨老师，无论是在编程学习、还是项目的支持方面，鲍老师都给予了我们小队成员莫大的帮助。