人脸识别控制推箱子项目报告

小组队名：人生苦短小队
小组成员：胡济捷 519120990031
铃木俊吉 519120990004

一．项目介绍：

本项目结合了面部识别技术与推箱子的小游戏，用人脸控制游戏小人的上下左右来玩推箱子小游戏。

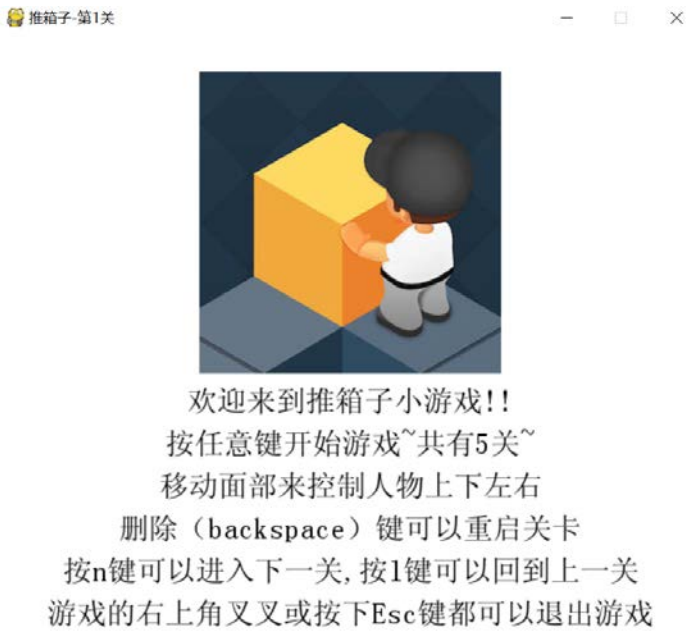
推箱子是一个非常古老的来自日本的游戏，这个游戏早已家喻户晓，而单纯玩推箱子小游戏也已不能够满足用户的游戏体验，因此我们想到结合面部来玩这个游戏。用面部来控制游戏中人物的移动可以极大增添这个游戏的趣味性与新颖性。

项目的主要思想是利用第三方类库 opencv 达到面部识别技术，再利用第三方类库 pygame 来实现推箱子的游戏，最后将两者进行结合。我们项目代码的主要流程也分为这三个部分。

二．游戏玩法：

启动程序后，会出现游戏说明页，按下任意键后可以开启摄像头并开始游戏。

在游戏中，移动面部来控制人物上下左右的移动。游戏共有 5 个关卡，由易到难。按下删除键可以重启这一关卡，按下 n 键可以跳转下一关，按下 l 键可以返回上一关。



三．所用到的第三方类库：

1. OpenCV：

OpenCV (Open Source Computer Vision Library)是一个开源计算机视觉和机器学习软件的第三方类库。OpenCV 旨在为计算机视觉应用程序提供通用的基础结构。该库拥有 2500 多种优化算法。我们项目中主要运用的是人脸识别的模块。主要的原理与代码参考以下官网介绍。

https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

2. Pygame

pygame 是跨平台 Python 模块，专为电子游戏设计，包含图像、声音。Pygame 具有高度的可移植性，几乎可以在所有平台和操作系统上运行。我们项目利用 pygame 的代码创建了推箱子小游戏。代码参考自以下 pygame 官网。

https://www.pygame.org/docs/

四．项目代码：

首先要先导入所用到的库（提前需要 pip install opencv-python,pygame）

Opencv 用于面部识别；pygame 用于创建游戏；

sys 用于终止程序；os 用于调整窗口弹出位置

```
import cv2 #用于面部识别
import pygame,sys,os
from pygame.locals import *
import copy
```

接下来项目代码的实现主要分为三个流程：

1. 建立面部识别，并定义上下左右移动
2. 建立推箱子游戏大框架，添加细节
3. 结合上述两个模块

以下叙述会分为这三块进行陈述。

（一）面部识别：（胡济捷）

首先调用摄像头，并导入分类器。分类器的下载地址：

https://github.com/opencv/opencv/tree/master/data/haarcascades

此处的路径运用了绝对路径，防止出现找不到文件的错误。

```
#调用摄像头
cap = cv2.VideoCapture(0)
classfier = cv2.CascadeClassifier("./haarcascades/haarcascade_frontalface_default.xml")
```

再利用 opencv 的程序实现面部识别。如果检测到人脸，则利用 cv2.rectangle 将人脸框出，并获取所框矩阵的坐标。

为了能够更好的检测人脸的移动，这里定义了 center 点作为检测对象，并建立了一个 face_center 的列表，将所有获取到的人脸点的坐标放入列表中。

```
if len(faceRects) > 0:  # 大于0则检测到人脸
    global eyesRects
    for faceRect in faceRects:
        x, y, w, h = faceRect
        cv2.rectangle(frame, (x-10, y-10), (x + w + 10, y + h + 10), (0, 255, 0), 2) #框出人脸
    center=((x+w/2),(y+h/2))
    # cv2.circle(frame, (int(x+w/2),int(y+h/2)), 1, (0,0,255), 4)
    # get_face.append((x,y))
    face_center.append(center)  #center点比左上角的点变化更稳定
```

利用储存在 face_center 里的面部坐标，我们分别将列表内相邻的 x 值（左右的变化）和相邻的 y 值（上下的变化）做减法，来获取每一帧的变化值，并累计，在列表内达到 35 个值（大概 1-2 秒）时做一次判断。如果累计值达到一定额度，则可以判别为向某一方向移

动。经过多次的试验，我们设定的左右移动判断值为 23，上下移动的判断值为 17。同时，为了防止列表过大，在进行一次判断后，会清空列表。

```
335     if len(faceRects)>0:
336         if len(face_center)==35:
337             movement='' #清空之前的movement
338             movesum_lr=0  #清空之前积累的数据
339             movesum_ud=0
340             for a in range(len(face_center)-1):
341                 move_lr = face_center[a][0]-face_center[a+1][0] #左右的move是比对list里相邻两项的x值
342                 move_ud = face_center[a][1]-face_center[a+1][1] #上下的move是比对list里相邻亮相的y值
343                 movesum_lr += move_lr
344                 movesum_ud += move_ud
345             if abs(movesum_lr) > 23 and abs(movesum_ud) > 17:
346                 movement = 'not detected'
347             elif abs(movesum_lr) < 23 and abs(movesum_ud) < 17:
348                 movement = 'no movement'
349 
350             else:
351                 if movesum_lr >= 23:
352                     movement = 'right'
353                 if movesum_lr <=-23:
354                     movement = 'left'
355                 if movesum_ud >= 17:
356                     movement = 'up'
357                 if movesum_ud <=-17:
358                     movement = 'down'
```

此外，为了使用户的游戏体验更好，我们翻转了摄像窗口，使得用户能更好的进行左右移动，还利用 cv2.putText 将 movement 添加到了窗口上。

```
328     frame=cv2.flip(frame,1)  #镜像翻转
329 
330     frame = cv2.putText(frame, movement, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (255, 255, 255), 2) #
331 
332     cv2.imshow('frame', frame) #显示窗口
333 
```

最后，我们在程序中试图解决多个人脸问题。但是发现面部识别的准确度比较低，有时候窗帘和空调口也会被识别为人脸，造成一人情况下也出现多个人脸框的情况，无法直接解决此问题，而尝试了加入眼睛识别，也出现了眼睛识别的不稳定性造成识别中止的情况出现。最后通过互联网搜索，我们发现在检测的参数中，minNeighbor 这个参数可以增加识别的精准性。

“Minimum number (minus 1) of neighbor rectangles that makes up an object.

All the groups of a smaller number of rectangles than min neighbors-1 are rejected. If min neighbors is 0, the function does not any grouping at all and returns all the detected candidate rectangles, which may be useful if the user wants to apply a customized grouping procedure.”

以上是搜索到的 minNeighbor 参数的解释，当设置这个参数为 0 时，会返回所有检测到的矩形，经过亲身实验，发现=0 时在脸部返还了大量的矩形，而其他地方比如手、图形会返还个别（1-4 个）矩形，而=3 时，会只返回矩形>2 的物体的矩形，此时还会出现一些误差，因此我们将这个参数设到 15，此时空调接口、窗帘、风景画这些在参数为 3 时被识别为人脸的物品都不再被识别。在这个基础上，我们设置了 len(faceRects)>=2 时，会显示”too many faces.”以此解决多个人脸的问题。

```
315     faceRects = classfier.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=15, minSize=(40, 40))
316 
```

（二）**游戏部分：**（铃木俊吉）

我们利用字典来存储了游戏关卡（共 5 关），为了增加游戏体验和消除脸部上下识别不稳定带来的体验，我们更改了关卡，将关卡设计成多左右移动的关卡，难度也有所降低。其中每一个数字都对应推箱子游戏中的元素。

```
#字典形式的关卡 15*15的
#0: 无填充 1: 墙 2: 地板 3: 指定位置（goal） 4: 箱子 5: 完成目标的箱子 6: 人物
```

我们自定义了类 Sur 来集合游戏的主程序。在 Class Sur 的类里，我们定义了游戏的各个部分（初始化、记录目标位置、渲染、标记、自动进入下一关、移动等）。

```
94    class Sur():  #定义一个类
95        def __init__(self,css):  #初始化
```

比如我们定义了 fun 来记录游戏中每一关卡箱子需要推到的目标位置在字典中的位置，便于“进入下一关”时匹配箱子的位置。

```
def fun(self): #记录goal的位置 (y,x)
    lists=[]
    for y in range(len(self.cs)): #字典的某一行
        for x in range(len(self.cs[y])): #字典一行里的某一个数字
            if self.cs[y][x]==3 or self.cs[y][x]==5:
                lists.append((y,x))
    return lists
```

我们“定义了进入下一关”的程序，如果所有的箱子都在目标位置，则进入下一关。如果通过了所有的关卡，我们设置了跳出一个“恭喜通关”的通关界面。

```
def exfun(self,root): #进入下一关
    for y in range(len(self.cs)):
        for x in range(len(self.cs[y])):
            for i in self.bj:
                if  i[0]==y and i[1]==x and self.cs[y][x]!=5:
                    return  #没有箱子就返回
    self.gk+=1
```

定义移动规则，cs[y][x]代表字典中第 y 行第 x 个的位置，移动的前提就是移动到达的位置不能是墙，其次再分为三种情况人物旁是箱子、变色箱子以及没有障碍物。每一种移动情况所带来元素的行列变动有所不同。

移动规则分上下左右，移动到左边位置人物坐标变为[x-1]，右边[x+1],上边[y-1]，下边[y+1]。

```
def move(self,num):
    if num==4: #左
        for y in range(len(self.cs)):
            for x in range(len(self.cs[y])): #cs[y][x]是指关卡字典里第y行第x个
                if self.cs[y][x]==6 and self.cs[y][x-1]!=1: #如果是人 且 旁边不是墙 就移动
                    if self.cs[y][x-1]==4 and self.cs[y][x-2]!=1 and self.cs[y][x-2]!=4 and self.cs[y][x-2]!=5: #旁边有箱子，如果箱子旁边不是墙
                        self.cs[y][x]=2 #移动：原先的位置变成地板
                        self.cs[y][x-1]=6 #角色往左移动了
                        self.cs[y][x-2]=4 #箱子
                        return
                    elif self.cs[y][x-1]==5 and self.cs[y][x-2]!=1 and self.cs[y][x-2]!=4 and self.cs[y][x-2]!=5: #旁边是变色的箱子 如上
                        self.cs[y][x]=2
                        self.cs[y][x-1]=6
                        self.cs[y][x-2]=4
                        return
                    elif self.cs[y][x-1]==2 or self.cs[y][x-1]==3: #旁边没有障碍物
                        self.cs[y][x-1]=6
                        self.cs[y][x]=2
                        return
```

除了基础的游戏定义之外，我们增加了一些按键：n 键表示下一关，l 键上一关，删除键重启此关卡，带来更好的游戏体验。

```
385                 elif event.key==K_BACKSPACE: #backspace
386                     sur.css=copy.deepcopy(css)
387                     gk=sur.gk
388                     sur.name=f"cs{gk}"
389                     sur.cs=sur.css[sur.name]
390                 elif event.key==K_n: #n键下一关
391                     if sur.gk<5:
392                         sur.gk=sur.gk+1
393                         sur.title=f"推箱子-第{sur.gk}关"
394                         sur.name=f"cs{sur.gk}"
395                         sur.cs=sur.css[sur.name]
396                         sur.bj=sur.fun()
397                         pygame.display.set_caption(sur.title)
398                 elif event.key==K_l:#l键上一关
399                     if sur.gk>1:
400                         sur.gk=sur.gk-1
401                         sur.title=f"推箱子-第{sur.gk}关"
402                         sur.name=f"cs{sur.gk}"
403                         sur.cs=sur.css[sur.name]
404                         sur.bj=sur.fun()
405                         pygame.display.set_caption(sur.title)
```

我们增加了游戏开始的说明界面，向用户说明游戏的一些按键，并设定了按任意键开始。

```
121     def startScreen(self): #加载开始页面
122         titleRect = self.m7.get_rect()
123         topCoord = 30
124         titleRect.top = topCoord
125         titleRect.centerx = 300
126         topCoord += titleRect.height
127         font = pygame.font.Font('simsun.ttf', 25)
128 
129         instructionText = ['欢迎来到推箱子小游戏!!',
130                            '按任意键开始游戏~共有5关~',
131                            '移动面部来控制人物上下左右',
132                            '删除（backspace）键可以重启关卡',
133                            '按n键可以进入下一关,按l键可以回到上一关',
134                            '游戏的右上角叉叉或按下Esc键都可以退出游戏']
135         root.fill((255,255,255))
136         root.blit(self.m7, titleRect)
```

还增加了音乐的播放。

```
305 pygame.mixer.init()# 初始化
306 track = pygame.mixer.music.load(file)# 加载音乐文件
307 pygame.mixer.music.play()# 开始播放音乐流
308 
```

（三）面部识别与游戏的结合：（胡济捷）

为了让面部的移动可以反应在游戏人物的移动，我们将识别按键移动改为了根据面部的 movement 来移动。

原程序：

```
230                 elif event.type==KEYDOWN:
231                     if event.key==276:#左
232                         sur.move(4)
233                     elif event.key==273:#上
234                         sur.move(8)
235                     elif event.key==275:#右
236                         sur.move(6)
237                     elif event.key==274:#下
238                         sur.move(2)
```

更改后：

```
368                 if movement == 'left':#左
369                     sur.move(4)
370                 elif movement == 'up':#上
371                     sur.move(8)
372                 elif movement == 'right':#右
373                     sur.move(6)
374                 elif movement == 'down':#下
375                     sur.move(2)
```

为了使两个模块能同时运行，我们取消了游戏的 def main，而是把面部识别和游戏的主程序放在了同一个”While True:”下。这样程序运行时，可以同时打开摄像头和游戏程序，并且能不断循环更新摄像捕捉以及游戏画面。

```
310 while True:
311     ret, frame = cap.read()
312 
313     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #转换灰度
314 
315     faceRects = classfier.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=3, minSize=(40, 40))
```

```
403 
404 
405         root.fill((128, 128, 128)) #填充背景色
406         sur.xrfun(root) #渲染窗口
407         sur.bjxrfun() #运行标记渲染
408         sur.exfun(root) #下一关
409         pygame.display.update() #更新画面
410         clock.tick(50)
411 
```

此外，为了让两个窗口弹出时不叠在一起，添加了游戏窗口弹出位置的优化。

```
95          x = 600
96          y = 100
97          os.environ['SDL_VIDEO_WINDOW_POS'] = "%d,%d" % (x,y)  #调整窗口弹出的位置
```

五．展望：

1. 上下识别的准确度还有待提高

比起左右的识别，上下的识别准确度更不稳定。由于人脸上下移动时幅度较小，而在平时移动产生的上下的震动又会比较大，所以上下的鉴别会带来一定的不稳定性，降低了准确性。为了消除这一不稳定性对用户体验的破坏，我们在关卡设计中，设计了多为左右移动的关卡。

2. 移动必须一步一动，身体回倾也会被检测为移动，有待改进

在移动之后，用户可能会身体回倾，但是此时的移动也会被机器识别为移动，程序无法判断用户真实的移动和身体自然回倾所造成的移动的区别。

六．反思：

在这个项目里，我们实现了将面部识别与小游戏的结合，完成了利用人脸控制游戏人物推箱子的程序项目。为普通的推箱子游戏添加了新颖感和趣味性。

在做程序的过程中，碰到了各种奇怪的问题，比如找不到分类器的文件而报错、没有把游戏的主程序放在 while true 下导致游戏在动但是面部识别卡住了、按下 n 键关卡出现错误等等。在出现问题的时候，主要是利用互联网搜索以及官方文件里的代码介绍来解决问题，还有一点感触就是细节真的很重要，一个变量的位置错误或是没有清空，就会导致整个程序的运行发生一些很奇怪的偏差。最后也很可惜有一些问题还是没能得到很好的解决。

在做程序的过程中，加强了自己去学习一些新代码的能力，增长了自己解决问题的能力，也对课堂上学习的知识有了更牢固的认识，收获匪浅。虽然 debug 的过程非常痛苦，但是最后看到自己写出来的代码顺利运行的时候，获得了巨大的喜悦与满足感。

人生苦短，要珍惜每一个瞬间好好努力。

最后，感谢鲍杨老师这一学期的悉心教导！！

参考网页：

https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

https://www.pygame.org/docs/

https://docs.opencv.org/master/d1/de5/classcv_1_1CascadeClassifier.html#aaf8181cb63968136476ec4204ffca498

http://blog.sina.com.cn/s/blog_9fcb9cbb01012b5b.html