辅助瞄准项目报告

小组成员：

曹皓 10225101418

周恒立 10225101410

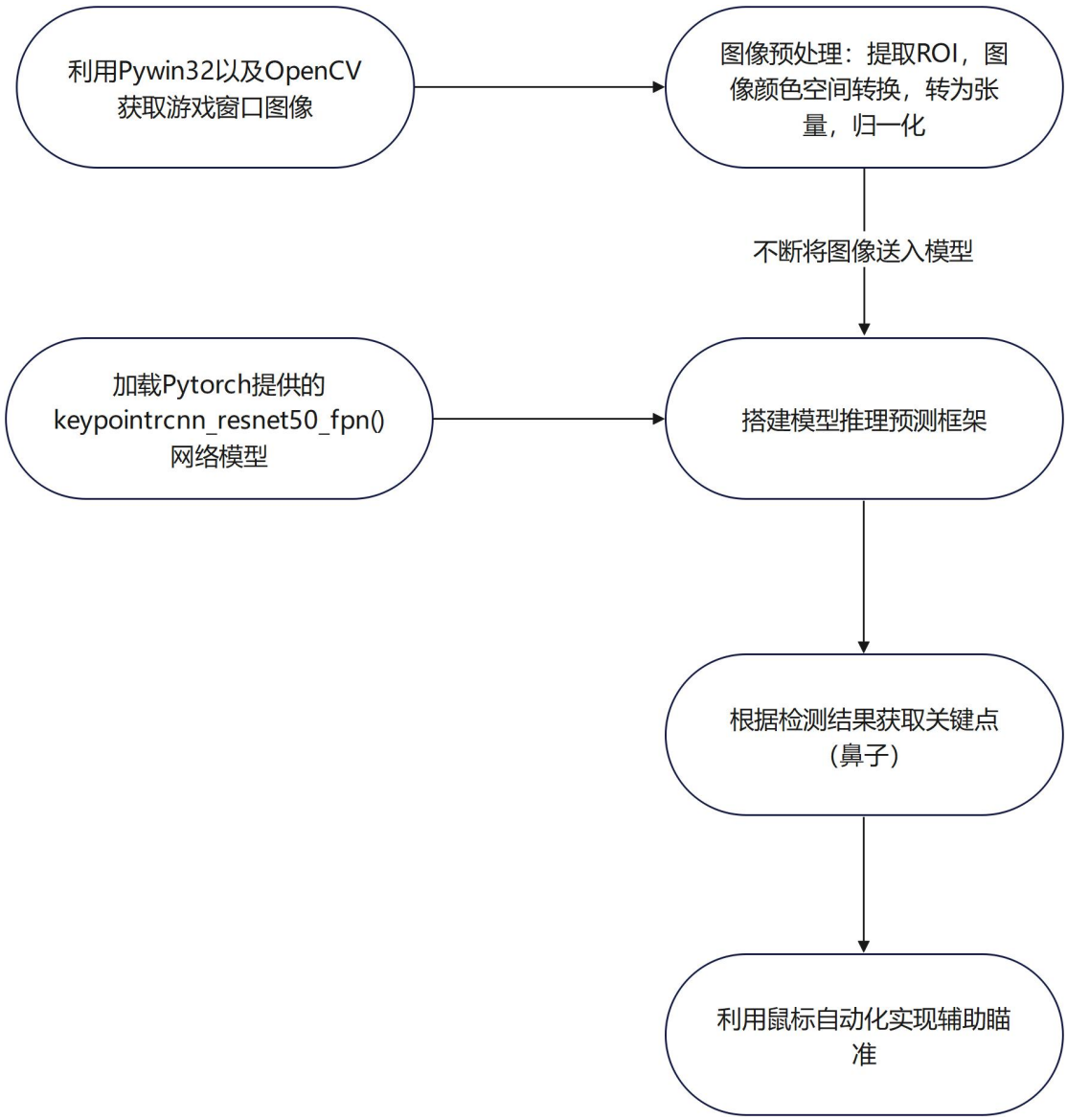
陈炯祥 10225101517

一、摘要

随着科技的飞速发展，人工智能与计算机视觉技术在各行各业中展现出前所未有的应用潜力，其中在游戏领域的应用尤为引人注目，激发了无数游戏爱好者对提升游戏技能、探索智能化辅助工具的热情。在这一背景下，利用 AI 技术来提升游戏体验，如实现自动瞄准、策略分析等功能，成为了技术探索的新方向。本项目采用 OpenCV 进行图像预处理和特征提取并结合 Pytorch 框架下提供的 keypointrcnn_resnet50_fpn()网络模型，利用目标骨骼关键点检测算法实现 FPS 类游戏的辅助瞄准。

二、实现思路

流程图如下：



三、运行要求

1. 具备空闲独立显卡，使用 cpu 的处理速率无法实际使用。
2. 安装所需库包
3. 在 Pytorch 官网下载最新版本的 CUDA
4. 从 main.py 开始运行

四、具体实现代码

1. 用户交互界面[1]

在 main 函数中，利用 python 标准 gui 库 tkinter，创建和用户交互的界面。通过 getButtonInfo 函数遍历所有窗口并获取所有窗口句柄及标题。为每个窗口创建一个按钮，传递窗口标题给 lambda 函数 runhelper。

```
def main():
    user_interface = Tk()  # 界面开头
    user_interface.title("辅助瞄准")
    user_interface.geometry("420x640")
    user_interface.resizable(0, 0)
    Label(user_interface, text='以下是你打开的窗口，请选择要使用的
窗口', font=(None, 15), fg='#66CCFF', bg="black").pack()
    ht=Region_Detector.getButtonInfo()

    for wnd in ht.items():
        if wnd[1]!= "":
            Button(user_interface, text=wnd[1], width=39,
height=1, command=lambda wnd=wnd:runhelper(wnd[1]),
activeforeground="red", activebackground="black").pack()

    mainloop()  # 界面结尾

def getButtonInfo():
    hwnd_title = {}

    def getAllWindowHandle(hwnd, use):
        if win32gui.IsWindow(hwnd) and
win32gui.IsWindowEnabled(hwnd) and
win32gui.IsWindowVisible(hwnd):
            hwnd_title.update({hwnd:
win32gui.GetWindowText(hwnd)})
        win32gui.EnumWindows(getAllWindowHandle, None)
        return hwnd_title
```

2. 获取游戏窗口截图[2]

利用 Pywin32 调用 window API 实现截图操作

```
    def getWindowsRGB(self,hwnd):
        """游戏窗口截图"""
        hwndDC = win32gui.GetWindowDC(hwnd)
        # 根据窗口句柄获取窗口的设备上下文 DC（Device Context）
        mfcDC = win32ui.CreateDCFromHandle(hwndDC)
        # 根据窗口的 DC 获取 mfcDC
        saveDC = mfcDC.CreateCompatibleDC()
        # mfcDC 创建可兼容的 DC
        saveBitMap = win32ui.CreateBitmap()
        # 创建 bitmap 准备保存图片
        rctA = win32gui.GetWindowRect(hwnd)
        Screen_w = rctA[2] - rctA[0]  # 游戏界面宽度
        Screen_h = rctA[3] - rctA[1]  # 游戏界面高度
        # 获取图片大小
        # 截取从左上角（0，0）长宽为（w，h）的图片
        saveBitMap.CreateCompatibleBitmap(mfcDC, Screen_w,
Screen_h)
        # 为 bitmap 开辟空间
        saveDC.SelectObject(saveBitMap)
        # 高度 saveDC，将截图保存到 saveBitmap 中
        saveDC.BitBlt((0, 0), (Screen_w, Screen_h), mfcDC, (0, 0),
win32con.SRCCOPY)
        signedIntsArray = saveBitMap.GetBitmapBits(True)
        img = np.frombuffer(signedIntsArray, dtype="uint8")
        img.shape = (Screen_h, Screen_w, 4)
        # bit 图转 mat 图
        win32gui.DeleteObject(saveBitMap.GetHandle())
        mfcDC.DeleteDC()
        saveDC.DeleteDC()
        # 释放内存
        return img, (Screen_w,Screen_h)  # 转为 RGBA 图返回
```

3. 图像预处理：截取 ROI、图像颜色空间转换、图像张量化、图像归一化

截取屏幕中央准星周围的区域作为 ROI

```
    def getRegion(self,img,Screen):
        Screen_w,Screen_h = Screen[0],Screen[1]-self.win_more
        Screen_cx = Screen_w // 2  # 游戏界面中心 x
        Screen_cy = Screen_h // 2  # 游戏界面中心 y
        Screen_c = [Screen_cx, Screen_cy]  # 游戏界面中心坐标
        x0 = Screen_w // 3    # 游戏界面检测框左上角 x0
        y0 = Screen_h // 3    # 游戏界面检测框左上角 y0
```

```
        x1 = 2*x0   # 游戏界面检测框右下角 x1
        y1 = 2*y0   # 游戏界面检测框右下角 y1
        # 选取 roi = img[y0,y1,x0,x1]窗口
        roi = img[int(y0):int(y1),int(x0):int(x1)]
        return
roi,Screen_c,[x0,y0+self.win_more,x1,y1+self.win_more]
```

利用 openCV 的 cvtColor 函数将图像从 RGBA 格式转为 RGB 格式

```
    image = cv2.cvtColor(image, cv2.COLOR_RGBA2RGB)
```

利用 torchvision 库中的 transforms 模块对图像进行张量化和归一化

```
import torchvision.transforms as transforms
    transform_d = transforms.Compose([transforms.ToTensor()])
    image_t = transform_d(image)    ## 对图像进行变换
```

4. 加载 pytorch 提供的 keypointrcnn_resnet50_fpn()网络模型，搭建模型推理预测框架[3]

加载模型完成配置

```
from torchvision.models.detection import
KeypointRCNN_ResNet50_FPN_Weights
# 加载 pytorch 提供的 keypointrcnn_resnet50_fpn()网络模型，可以对 17
个人体关键点进行检测。
#如果有可用的 GPU，则使用 GPU，否则使用 CPU。
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
#pytorch 模型的 python 接口
model =
torchvision.models.detection.keypointrcnn_resnet50_fpn(weights
=KeypointRCNN_ResNet50_FPN_Weights.DEFAULT)
model.to(device)
model.eval()
```

模型检测图像的标签和 17 个人体关键点的标签

```
OBJECT_LIST = [
    '__BACKGROUND__', 'person', 'bicycle', 'car', 'motorcycle',
    'airplane', 'bus', 'train', 'trunk', 'boat', 'traffic light',
    'fire hydrant', 'N/A', 'stop sign', 'parking meter', 'bench',
    'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant',
    'bear', 'zebra', 'giraffe', 'N/A', 'backpack', 'umbrella',
'N/A',
    'N/A', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis',
'snowboard',
    'sports ball', 'kite', 'baseball bat', 'baseball glove',
'skateboard',
    'surfboard', 'tennis racket', 'bottle', 'N/A', 'wine glass',
```

```
    'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
    'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog',
'pizza',
    'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
'N/A',
    'dining table', 'N/A', 'N/A', 'toilet', 'N/A', 'tv', 'laptop',
    'mouse', 'remote', 'keyboard', 'cell phone', 'microwave',
'oven',
    'toaster', 'toaster', 'sink', 'refrigerator', 'N/A', 'book',
'clock',
    'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush'
]
KEYPOINT_LIST = ['nose', 'left_eye', 'right_eye', 'left_ear',
                             'right_ear', 'left_shoulder',
'right_shoulder', 'left_elbow',
                             'right_elbow', 'left_wrist',
'right_wrist', 'left_hip', 'right_hip',
                             'left_knee', 'right_knee',
'left_ankle', 'right_ankle']
```

5. 将图像送入模型进行检测[3]

保留置信度大于 0.9 的结果，并利用 openCV 绘制检测框

```
def Capture_Point(image, confidence=0.9):
    image = cv2.cvtColor(image, cv2.COLOR_RGBA2RGB)
    # 准备需要检测的图像
    transform_d = transforms.Compose([transforms.ToTensor()])
    image_t = transform_d(image)    ## 对图像进行变换
    pred = model([image_t.to(device)])         ## 将模型作用到图像
上
    # 检测出目标的类别和得分
    pred_class = [model_elem.OBJECT_LIST[ii] for ii in
list(pred[0]['labels'].cpu().numpy())]
    pred_score = list(pred[0]['scores'].detach().cpu().numpy())
    # 检测出目标的边界框
    pred_boxes = [[ii[0], ii[1], ii[2], ii[3]] for ii in
list(pred[0]['boxes'].detach().cpu().numpy())]
    ## 只保留识别的概率大约 confidence 的结果。
    pred_index = [pred_score.index(x) for x in pred_score if x >
confidence]
    for index in pred_index:
        box = pred_boxes[index]
        box = [int(i) for i in box]
```

```
        cv2.rectangle(image,(int(box[0]),int(box[1])),(int(box[
2]),int(box[3])),(0,255,255))
        texts = pred_class[index] + ":" +
str(np.round(pred_score[index], 2))
        cv2.putText(image, texts,(box[0], box[1]),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 255), 2)

    pred_keypoint = pred[0]["keypoints"]
    # 检测到实例的关键点
    pred_keypoint =
pred_keypoint[pred_index].detach().cpu().numpy()
    # 对实例数量索引
    my_result = {}
    for index in range(pred_keypoint.shape[0]):
        # 对每个实例的关键点索引
        keypoints = pred_keypoint[index]
        for ii in range(keypoints.shape[0]): ##ii 为第几个坐标点
            x = int(keypoints[ii, 0]) #x 坐标
            y = int(keypoints[ii, 1]) #y 坐标
            visi = keypoints[ii, 2] #置信度
            if visi > 0.:
                cv2.circle(image, (int(x),int(y)), 1,
(0,0,255),4)
                texts = str(ii+1)
                cv2.putText(image,texts, (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 255), 2)
                my_result[texts] = (int(x), int(y))
    return image,my_result
```

6. 绘制 ROI 检测区域

```
def show_capture(win_name,roi_image):
    cv2.namedWindow(win_name, cv2.WINDOW_NORMAL)
    cv2.resizeWindow(win_name, 640, 360)
    cv2.imshow(win_name, roi_image)
    cv2.moveWindow(win_name, 0,0)
    cv2.getWindowImageRect(win_name)
```

7. 实现鼠标自动化[4]

```
import ctypes

class MouseInput(ctypes.Structure):
```

```
    _fields_ = [("dx", ctypes.c_long),  # 水平方向的绝对位置/相对移动量(像素), dwFlags 中包含 MOUSEEVENTF_ABSOLUTE 标识就是绝对移动, 否则是相对移动
                ("dy", ctypes.c_long),  # 垂直方向的绝对位置/相对移动量(像素)
                ("mouseData", ctypes.c_ulong),  # 某些事件的额外参数, 如: MOUSEEVENTF_WHEEL(中键滚动), 可填正负值, 一个滚动单位是 120(像素?); 还有 MOUSEEVENTF_XDOWN/MOUSEEVENTF_XUP
                ("dwFlags", ctypes.c_ulong),  # 事件标识集, 可以是移动或点击事件的合理组合, 即可以一个命令实现移动且点击
                ("time", ctypes.c_ulong),  # 事件发生的时间戳, 可以指定发生的时间 传入 0 则使用系统提供的时间戳
                ("dwExtraInfo", ctypes.POINTER(ctypes.c_ulong))]  # 应用可通过 GetMessageExtraInfo 来接收通过此参数传递的额外消息

class Inner(ctypes.Union):
    _fields_ = [("ki", KeyboardInput),
                ("mi", MouseInput),
                ("hi", HardwareInput)]

class Input(ctypes.Structure):
    _fields_ = [("type", ctypes.c_ulong),  # 输入事件类型
                ("ii", Inner)]

"""
MouseInput
"""
MOUSEEVENTF_MOVE = 0x0001  # 移动
MOUSEEVENTF_LEFTDOWN = 0x0002  # 左键按下
MOUSEEVENTF_LEFTUP = 0x0004  # 左键释放
MOUSEEVENTF_RIGHTDOWN = 0x0008
MOUSEEVENTF_RIGHTUP = 0x0010
MOUSEEVENTF_MIDDLEDOWN = 0x0020
MOUSEEVENTF_MIDDLEUP = 0x0040
MOUSEEVENTF_ABSOLUTE = 0x8000  # 鼠标移动事件, 如果设置此标记就是绝对移动, 否则就是相对移动.
WHEEL_DELTA = 120  # 鼠标滚轮滚动一个单位的最小值
"""
Input
"""
INPUT_MOUSE = 0  # 鼠标输入事件
INPUT_KEYBOARD = 1  # 键盘输入事件
```

```
INPUT_HARDWARE = 2  # 硬件输入事件
"""
SendInput
cInputs: pInputs 数组的个数，即 SendInput 可以一下发好几个鼠标事件/键盘事件，这些事件存储在一个连续的数组空间里
pInputs: 输入事件的实例的指针
cbSize: 每一个 INPUT 事件的结构体空间，鼠标事件和键盘事件应该不能同时放到 pInputs 数组中，因为他们的 size 不同
该函数返回成功插入键盘或鼠标输入流的事件数。如果函数返回零，则输入已被另一个线程阻塞。要获取扩展错误信息，请调用 GetLastError.
"""
def SendInput(*inputs):  # 接收任意个参数，将其打包成为元组形参，双*是打包成为字典形参
    nInputs = len(inputs)
    pointer = Input * nInputs
    pInputs = pointer(*inputs)
    cbSize = ctypes.sizeof(Input)
    return ctypes.windll.user32.SendInput(nInputs, pInputs, cbSize)

class Mouse:
    @staticmethod
    def leftClick():
        return SendInput(Input(INPUT_MOUSE, Inner(mi=MouseInput(0, 0, 0, MOUSEEVENTF_LEFTDOWN | MOUSEEVENTF_LEFTUP, 0, None))))

    @staticmethod
    def move(x, y, absolute=False):
        if not absolute:
            return SendInput(Input(INPUT_MOUSE, Inner(mi=MouseInput(x, y, 0, MOUSEEVENTF_MOVE, 0, None))))
        else:
            # 绝对值移动时，屏幕宽高的取值范围都是[0, 65535]，需要自行换算一下
            # 获取显示分辨率(非物理分辨率)
            w, h = ctypes.windll.user32.GetSystemMetrics(0), ctypes.windll.user32.GetSystemMetrics(1)
            rx, ry = int(x * 65535 / w), int(y * 65535 / h)
            return SendInput(Input(INPUT_MOUSE, Inner(mi=MouseInput(rx, ry, 0, MOUSEEVENTF_MOVE | MOUSEEVENTF_ABSOLUTE, 0, None))))
```

8. 获取关键点（鼻子）并利用鼠标自动化完成辅助瞄准

```
def runhelper(wn):
    win_name = wn
    print("现在捕获的窗口是: " ,end="")
    print(win_name)
    win_size = [1920, 1080]  #窗口大小
    win_index = [0,0]  #窗口放置位置
    win_more = 0   #窗口多余部分，即窗口标题
    RD = Region_Detector(win_name, win_more) #创建窗口处理对象
    hwdn = RD.getWindowHandle()    #获取窗口句柄
    RD.setWindows(hwdn, win_size, win_index)    #设置窗口参数

    while True:
        MouseX, MouseY = pyautogui.position()
        start_time = time.time()    #开始时间
        img, Screen = RD.getWindowsRGB(hwdn)   #获取窗口截图
        img_roi, Screen_c,[x0,y0,x1,y1] = RD.getRegion(img,
Screen)   #获取窗口截图 ROI 图片
        roi_image,my_result = Capture_Point(img_roi,
confidence=0.9) #ROI 图像进行推理，返回图像与 ROI 图像中位置坐标
        print("捕获结果: ", my_result)
        fps = int(1/(time.time() - start_time)) #   计算推理帧率
        print("处理帧数:{}".format(fps))
        show_capture(win_name, roi_image)  # OpenCV 窗口显示结果图
像
        if cv2.waitKey(1) & 0xFF == ord("q"):#退出快捷键
            break
        if my_result == {}:
            continue
        x= int(my_result["1"][0])
        y = int(my_result["1"][1])
        currentMouseX = (640+x-MouseX)*1.1
        currentMouseY = (360+y-MouseY)*1.1
        sendinput.Mouse.move(int(currentMouseX),int(currentMous
eY))
        time.sleep(0.001)
        sendinput.Mouse.leftClick()
        sendinput.Mouse.leftClick()
        sendinput.Mouse.leftClick()
        sendinput.Mouse.leftClick()
    cv2.destroyAllWindows()
```

五、功能展示

1. 用户交互界面

2. ROI 检测窗口和检测结果

3. 辅助瞄准演示视频

辅助瞄准演示视频.mp4

参考文献

[1] Python----pywin32 如何获取窗口句柄_pywin32 获取窗口句柄-CSDN 博客
[2] pywin32 调用 Windows API 实现截图操作_pywin32 截图-CSDN 博客
[3] Pytorch 快速入门系列---（十九）Pytorch 实现 R-CNN 系列目标检测网络_pytorch r-cnn-CSDN 博客
[4] 【使用 Python 编写游戏辅助工具】第三篇：鼠标连击器的实现_python 游戏辅助原理-CSDN 博客