第 6 课 人体跟踪

通过机器学习训练的人体姿态模型对人体进行识别，根据识别到的内容控制舵机转动，使被识别的人体一种处于机器人回传画面的中心。

1.实验流程

首先，导入 MediaPipe 的人体姿态估计模型，并通过订阅话题消息，获取摄像头的实时画面。

随后，基于搭建的模型，检测画面内的人体躯干关键点，进而获取人体中心点坐标。

最后，根据人体中心点与画面中心点的坐标，更新 PID 控制器，控制机器人随着人体的移动而移动。

程序源码位于：

/home/jetauto_ws/src/jetauto_example/scripts/body_control/include/body_track.py

```
class BodyControlNode:
    def __init__(self, name):
        rospy.init_node(name, anonymous=True)
        self.name = name
        self.drawing = mp.solutions.drawing_utils
        self.body_detector = mp_pose.Pose(
            static_image_mode=False,
            model_complexity=0,
            min_tracking_confidence=0.5,
            min_detection_confidence=0.5)

        self.pid_d = pid.PID(0.03, 0, 0)
        #self.pid_d = pid.PID(0, 0, 0)

        self.pid_angular = pid.PID(0.0025, 0, 0.0005)
        #self.pid_angular = pid.PID(0, 0, 0)

        self.go_speed, self.turn_speed = 0.007, 0.04
        self.linear_x, self.angular = 0, 0

        self.fps = fps.FPS()  # fps计算器

        self.turn_left = False
        self.turn_right = False
        self.go_forward = False
        self.back = False
        self.next_frame = True
        self.depth_frame = None

        camera = rospy.get_param('/depth_camera/camera_name', 'astra_cam')
        self.image_sub = rospy.Subscriber('/%s/rgb/image_raw'%camera, Image, self.image_callback, queue_size=1)
        self.depth_image_sub = rospy.Subscriber('/%s/depth/image_raw'%camera, Image, self.depth_image_callback, queue_size=1)
        self.mecanum_pub = rospy.Publisher('/jetauto_controller/cmd_vel', Twist, queue_size=1)
        rospy.sleep(0.2)
        self.mecanum_pub.publish(Twist())
```

2.实验步骤

注意：输入指令时需要严格区分大小写，且可使用“Tab”键补齐关键词。

1) 启动 JetAuto 机器人，将其连接至远程控制软件 NoMachine。关于远程桌面工具的安装与连接可参考“**第 7 章 开发环境搭建->1.远程桌面工具的安装与连接**”。

2) 双击系统桌面的图标  ，打开命令行终端。

3) 输入指令“**sudo systemctl stop start_app_node.service**”，并按下回车，关闭 APP 控制服务。

```
jetauto@jetauto-desktop:~$ sudo systemctl stop start_app_node.service
```

4) 输入指令“**roslaunch jetauto_example body_track.launch**”，并按下回车，开启玩法。

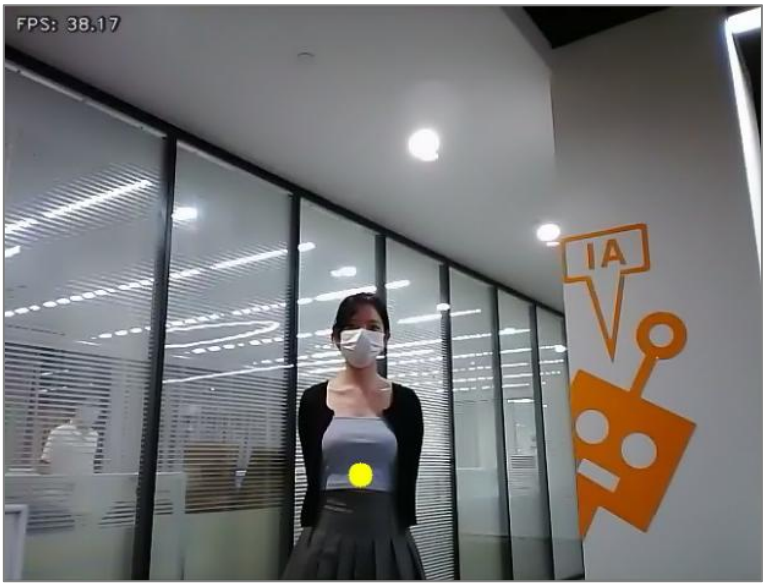
```
jetauto@jetauto-desktop:~$ roslaunch jetauto_example body_track.launch
```

如需关闭程序，点击程序对应的终端窗口，使用快捷键“**Ctrl+C**”，即可关闭程序。并输入指令“**sudo systemctl start start_app_node.service**”打开 APP 自启服务。

```
jetauto@jetauto-desktop:~$ sudo systemctl start start_app_node.service
jetauto@jetauto-desktop:~$
```

3.功能实现

启动玩法后，站立在摄像头的视野范围内。当检测到人体，回传画面内会标识出人体中心点。人体移动，JetAuto 机器人会之进行移动。两者间距始终保持为 300cm。



4.程序分析

4.1 基础配置

◆ **搭建人体姿态估计模型**

导入 MediaPipe 的人体姿态估计模型，并直接引用官方配置。

```
31        self.drawing = mp.solutions.drawing_utils
32        self.body_detector = mp_pose.Pose(
33            static_image_mode=False,
34            model_complexity=0,
35            min_tracking_confidence=0.5,
36            min_detection_confidence=0.5)
```

第一个参数“**static_image_mode**”是输入图像的处理模式。默认值为“**False**”，表示将输入图像视为视频流，即检测完第一张图像后，仅对随后的图像进行地标跟踪。直至追踪失败，才会再次对图像进行检测。这种检测模式有助于减少计算量与延迟。

当值为“**True**”，程序会对所有输入图像都进行检测，这种模式适用于检测批量静态的、互不关联的图像。

第二个参数“**model_complexity**”是模型复杂度。一般情况下，模型越复杂，识别精度越高，但识别速度越慢。

第三个参数“**min_tracking_confidence**”是各帧之间跟踪置信度阈值，取值范围是 0.0-1.0，默认值是 0.5。若各帧之间的关键点符合率高于此阈值，则视为跟踪成功。

第四个参数“**min_detection_confidence**”是检测置信值阈值，取值范围是 0.0-1.0，默认值是 0.5。若姿态检测概率高于此阈值，则视为检测成功。

- **订阅/发布话题**

通过订阅摄像头相关的话题消息，获取摄像头的实时画面。

```
56      camera = rospy.get_param('/depth_camera/camera_name', 'astra_cam')
57      self.image_sub = rospy.Subscriber('/%s/rgb/image_raw'%camera, Image,
        self.image_callback, queue_size=1)
58      self.depth_image_sub = rospy.Subscriber('/%s/depth/image_raw'%camera,
        Image, self.depth_image_callback, queue_size=1)
```

通过发布移动控制相关的话题消息，实现对 JetAuto 机器人的移动控制。

```
59      self.mecanum_pub = rospy.Publisher('/jetauto_controller/cmd_vel', Twist,
        queue_size=1)
```

4.2 人体检测

- **获取人体躯干关键点**

基于先前搭建的人体姿态估计模型，检测摄像头图像内的人体躯干关键点，并获取各点坐标。

```
89          results = self.body_detector.process(image)
```

- **人体中心点**

根据各个关键点求得人体中心点坐标。

```
95          center = get_body_center(h, w, results.pose_landmarks.landmark)
```

通过调用 cv2 库的 circle()函数，在回传画面内绘制人体中心点。

```
96          cv2.circle(image, tuple(center), 10, (255, 255, 0), -1)
```

第一个参数“**image**”是输入图像；

第二个参数“**tuple(center)**”是圆心坐标；

第三个参数“**10**”是圆半径；

第四个参数“**(255, 255, 0)**”是边线颜色；

第五个参数“**-1**”是边线粗细，取值“-1”意为填充整个圆形，即绘制实心圆。

4.3 人体追踪

◆ **维持机体与人体间距**

根据中心点坐标，求解人体与机器人的间距。

```
            roi_h, roi_w = 5, 5
            w_1 = center[0] - roi_w
            w_2 = center[0] + roi_w
            if w_1 < 0:
                w_1 = 0
            if w_2 > w:
                w_2 = w
            h_1 = center[1] - roi_h
            h_2 = center[1] + roi_h
            if h_1 < 0:
                h_1 = 0
            if h_2 > h:
                h_2 = h

            cv2.rectangle(image, (w_1, h_1), (w_2, h_2), (255, 255, 0), 2)
            roi = self.depth_frame[h_1:h_2, w_1:w_2]
            distances = roi[np.logical_and(roi>0, roi<40000)]
```

获得人体与机体的间距后，通过更新 PID 控制器，控制机器人移动，令两者间距保持为 300cm。

```
            if distance > 600 or distance < 100:
                distance = 300
            self.pid_d.SetPoint = 300
            if abs(distance - 300) < 30:
                distance = 300
            self.pid_d.update(distance)  #更新pid
            tmp = self.go_speed - self.pid_d.output
            self.linear_x = tmp
            if tmp > 0.2:
                self.linear_x = 0.2
            if tmp < -0.2:
                self.linear_x = -0.2
            if abs(tmp) < 0.008:
                self.linear_x = 0
```

◆ **机体跟随人体移动**

根据人体中心点与画面中心点的坐标差更新 PID 控制器，控制机器人移动，令人体处于画面中心，以达到人体跟踪效果。

```
137             self.pid_angular.SetPoint = w/2
138             if abs(center[0] - w/2) < 20:
139                 center[0] = w/2
140             self.pid_angular.update(center[0])  #更新pid
141             tmp = self.turn_speed + self.pid_angular.output
142             self.angular = tmp
143             if tmp > 2:
144                 self.angular = 2
145             if tmp < -2:
146                 self.angular = -2
147             if abs(tmp) < 0.05:
148                 self.angular = 0
149             #if self.angular == 0:
150             twist.linear.x = self.linear_x
151             #else:
152             twist.angular.z = self.angular
```

```
160             self.mecanum_pub.publish(twist)
```