语音控制

通过前面的课程讲解，我们理解了整个实现语音自定义交互的过程。接下来我们讲解的是通过语音交互功能来控制我们的机器人进行移动。而我们的语音控制也是基于前面讲解的自定义交互模式设计的一种功能实现。

整体效果

默认demo通过语音下达前进、后退、左转和右转的指令，可以控制机器人的前进、后退、左转和右转一段时间。
留下的接口还可以添加加速、减速等功能。

实现过程

首先我们得通过唤醒词将机器人唤醒，让他进入唤醒状态。所以我们的前面讲解的语音离线命令词检测和VAD检测得先打开。

```
roslaunch handsfree_voice offline_interactive_ros.launch//语音交互功能节点

rosrun handsfree_voice vad_record.py//vad检测节点
```

然后我们运行

```
rosrun handsfree_voice voice_cmd_vel.py//这个就是语音控制节点
```

根据语音第一节的内容，我们可以先使用仿真环境进行测试，先看一下效果,这里我们也提供了相应的仿真环境。

```
roslaunch handsfree_stage demo_handsfree_room_stage.launch
roslaunch handsfree_stage demo_handsfree_xuda_stage.launch
```

这两个都是仿真环境，在空间不够空旷的情况下或者还没熟练的情况下，可以执行其中任意一个就能进行仿真测试。通过仿真环境，用我们的语音交互的功能去看执行效果,进行驱动功能的调试。

控制核心

就是利用前面的语音交互功能节点和VAD检测节点的配合，来获取我们的交互数据，通过交互数据提取，将ID号的信息通过topic发送到我们的控制节点这边，然后控制节点通过 `/mobile_base/mobile_base_controller/cmd_vel` 发送数据。通过发送线速度和角速度的数据，将线速度和角速度转化成小车车轮的的线速度或者转速度进行控制。

我们是根据英文语音控制的来进行一个中文的语音适配和优化。

驱动控制

```
rospy.on_shutdown(self.cleanup)

        # Set a number of parameters affecting the robot's speed
        self.max_speed = rospy.get_param("~max_speed", 0.4)
        self.min_speed = rospy.get_param("~min_speed", 0.1)
        self.linear_increment = rospy.get_param("~linear_increment", 0.1)

        self.max_angular_speed = rospy.get_param("~max_angular_speed", 0.
8)
        self.min_angular_speed = rospy.get_param("~min_angular_speed", 0.
1)
        self.angular_increment = rospy.get_param("~angular_increment", 0.
1)

        # Initial parameters
        self.speed = rospy.get_param("~start_speed", 0.4)
        self.angular_speed = rospy.get_param("~start_angular_speed", 0.2)

        # We don't have to run the script very fast
        self.rate = rospy.get_param("~rate", 5)
        r = rospy.Rate(self.rate)

        # Time, in seconds, for waiting correct command
        self.initial_wait_time = 5
        self.wait_time = self.initial_wait_time

        # A flag to determine whether or not TIAGo voice control
        self.TIAGo = False

        # Initialize the Twist message we will publish.
        self.cmd_vel = Twist()

        # Publish the Twist message to the cmd_vel topic
        self.cmd_vel_pub = rospy.Publisher('/mobile_base/mobile_base_cont
roller/cmd_vel', Twist, queue_size=5)

        # Subscribe to the /recognizer/output topic to receive voice comm
ands.
        rospy.Subscriber('/recognizer/output', String, self.speech_callba
ck)
```

```
        # A mapping from keywords or phrases to commands
        self.keywords_to_command = {'backward': ['backward', 'go backwar
d', 'move backward', 'back', 'move back'],
                                    'forward': ['forward', 'go forward',
'move forward'],
                                    'turn left': ['turn left', 'left'],
                                    'turn right': ['turn right', 'right'
],
                                    'stop': ['stop', 'halt'],
                                    'faster': ['faster'],
                                    'slower': ['slower'],
                                    'quarter': ['quarter speed', 'quarte
r'],
                                    'half': ['half speed', 'half'],
                                    'full': ['full speed', 'full']}

        rospy.loginfo("Ready to receive voice commands")

        # We have to keep publishing the cmd_vel message if we want the r
obot to keep moving.
        while not rospy.is_shutdown():
            self.cmd_vel_pub.publish(self.cmd_vel)
            r.sleep()
 if command == 'backward':
            self.cmd_vel.linear.x = -self.speed
            self.cmd_vel.angular.z = 0
            rospy.loginfo("Command: " + str(command))
            time.sleep(3)
            rospy.loginfo("Command: " + self.get_command(msg.data))
            self.cmd_vel = Twist()
            self.TIAGo = False
            self.wait_time = self.initial_wait_time
            return

        elif command == 'forward':
            self.cmd_vel.linear.x = self.speed
            print(self.speed)
            self.cmd_vel.angular.z = 0
            rospy.loginfo("Command: " + str(command))
            time.sleep(3)
            rospy.loginfo("Command: " + self.get_command(msg.data))
            self.cmd_vel = Twist()
            self.TIAGo = False
            self.wait_time = self.initial_wait_time
            return
```

语音控制反馈部分

```
  if(todo_id==99999)
```

```
    {
      char* text;
        int x =random(0, 5);
        if(x==1)
      {
          text =(char*)"有什么事情"; //合成文本
      }
        else if(x==2)
      {
          text =(char*)"怎么啦"; //合成文本
      }
        else if(x==3)
      {
          text =(char*)" 来了"; //合成文本
      }
        else
      {
          text =(char*)" 我在"; //合成文本
      }
        printf("收到唤醒指令");
        ret =text_to_speech(text, filename_move, session_begin_params);
        if (MSP_SUCCESS != ret)
        {
          printf("text_to_speech failed, error code: %d.\n", ret);
        }
        printf("合成完毕\n");
        ifwake=1;
        std_msgs::String msg_pub;
        msg_pub.data ="stop";
        pub.publish(msg_pub);
        play_wav((char*)"/home/handsfree/catkin_ws/src/handsfree_voice/res/
tts_sample_move.wav");
        msg_pub.data ="tiago";
        pub.publish(msg_pub);
    }
   if(todo_id==10001&&ifwake==1)
    {
        const char* text ="收到指令，请注意避让"; //合成文本
        ret =text_to_speech(text, filename_move, session_begin_params);
        if (MSP_SUCCESS != ret)
        {
          printf("text_to_speech failed, error code: %d.\n", ret);
        }
        printf("合成完毕\n");
        printf("收到前进指令");
        play_wav((char*)filename_move);
        ifwake=0;
        std_msgs::String msg_pub;
        msg_pub.data ="forward";
        pub.publish(msg_pub);
    }
    if(todo_id==10002&&ifwake==1)
```

```
    {
        const char* text ="收到指令，请注意，倒车"; //合成文本
        ret =text_to_speech(text, filename_move, session_begin_params);
        if (MSP_SUCCESS != ret)
        {
          printf("text_to_speech failed, error code: %d.\n", ret);
        }
        printf("合成完毕\n");
        printf("收到倒车指令");
        play_wav((char*)filename_move);
        ifwake=0;
        std_msgs::String msg_pub;
        msg_pub.data ="backward";
        pub.publish(msg_pub);
    }
    if(todo_id==10003&&ifwake==1)
    {
        const char* text ="收到指令，正在转弯"; //合成文本
        ret =text_to_speech(text, filename_move, session_begin_params);
        if (MSP_SUCCESS != ret)
        {
          printf("text_to_speech failed, error code: %d.\n", ret);
        }
        printf("合成完毕\n");
        printf("收到左转指令");
        play_wav((char*)filename_move);
        ifwake=0;
        std_msgs::String msg_pub;
        msg_pub.data ="left";
        pub.publish(msg_pub);
    }
    if(todo_id==10004&&ifwake==1)
    {
        const char* text ="收到指令，正在转弯"; //合成文本
        ret =text_to_speech(text, filename_move, session_begin_params);
        if (MSP_SUCCESS != ret)
        {
          printf("text_to_speech failed, error code: %d.\n", ret);
        }
        printf("合成完毕\n");
        printf("收到右转指令");
        play_wav((char*)filename_move);
        ifwake=0;
        std_msgs::String msg_pub;
        msg_pub.data ="right";
        pub.publish(msg_pub);
    }
```

在我们的语音控制程序中，里面设置了一个`tiago`标志，只有接收到这个标志，我们接收到的后续的控制数据才算是有效数据。也是一个状态判定。所以我们在接收到唤醒命令的唤醒反馈中添加了一个`tiago`数据，代表后面的发送的数据才是有效的数据。这里我们设计了一个情况，当唤醒时，

让机器人停下现在的任务，这样我们在机器人进行移动的时候，我们通过唤醒可以停止现在的任务。
这里的前进和后退我们添加了一个延时函数，让他每次执行一段时间后，就停止当前的前进和后退动作。避免机器人移动的距离比较远然后语音唤醒功能没办法唤醒成功，导致机器人没办法停止，一直往前前进或者后退。
下面是一个科大讯飞的错误码信息查询，如果在调用科大讯飞的时候报错了相关的错误码，可以通过这个链接查询。
科大讯飞错误码信息查询