自定义交互设计和使用

实现流程

整体效果

VAD实时判断周围的能量，根据能量的门限判断是否识别为语音，如果识别为语音，将其保存下来然后通过离线关键词检测，判断是否有唤醒关键词，有唤醒关键词再反馈给唤醒状态机，否则保持休眠状态。当唤醒状态机处于唤醒状态时。这时候下达的指令如果是有效的指令，则机器人执行相应的指令同时唤醒状态机返回休眠模式。
根据我们上一个文档，我们可以知道这是我们的语音交互的整体流程，这里我们讲解一下怎么设计一个属于自己，拥有自己的唤醒词，有自己的语音交互动作的语音交互功能的开发。

语音交互的基础，命令词的设置

修改科大讯飞的语法文件call.bnf

```
#BNF+IAT 1.0 UTF-8;
!grammar control;

!slot <move>;
!slot <time>;
!slot <wake>;

!start <callstart>;
<callstart>:<control>;
<control>:<wake>|<move>;


<wake>:小道!id(99999)|小道小道!id(99999);//（此处可以修改唤醒词）

<move>:往前走!id(10001)|前进!id(10001)|向前进!id(10001)|后退!id(10002)|向后退!id(10002)|退后!id(10002)|向左转!id(10003)|左转!id(10003)|右转!id(10004)|向右转!id(10004)|退出语音模式!id(10005)|关机!id(10005);//（此处可以修改命令词）
```

我们的语音交互的实现是通过VAD算法实时检测周围的声音环境，这里我们开了一个线程，这个线程利用vad算法在实时检测声音，根据检测声音的情况，判断是否达到阈值当达到一定的阈值的时

候，我们就会将会记录并保存这一段触发阈值的声音为一个wav文件。
通过修改语法文件，我们可以通过科大讯飞的离线语音命令词识别功能对我们前面保存的wav文件进行命令词的识别。这样就可以实现了一个实时的语音命令词识别的功能。所以我们在语法文件中修改对应的唤醒词和命令词，就可以通过不同的语音命令去唤醒或下达命令，可以实现自定义的语音唤醒和语音控制。这里是我们默认的语音功能，下面有相应的教程讲解会提供如何自定义添加一个功能。

下面是我们的数据处理的方法

```
static int16_t get_order(char *_xml_result){
  if(_xml_result == NULL){
    printf("no");
  }
//get confidence
  char *str_con_first1 = strstr(_xml_result,"<confidence>");
  //printf("\n%s",str_con_first1);
  char *str_con_second1 = strstr(str_con_first1,"</confidence");
  //printf("\n%s",str_con_second1);
   char *str_con_first2 = strstr(str_con_second1,"<confidence>");
  //printf("\n%s",str_con_first1);
  char *str_con_second2 = strstr(str_con_first2,"</confidence");
  //printf("\n%s",str_con_second1);
  char str_confidence2[10] = {};
  strncpy(str_confidence2, str_con_first2+12, str_con_second2 - str_con_f
irst2-12);
  //printf("\n%s\n",str_confidence2);
  char *str_con_first = strstr(_xml_result,"<object>");
  char *str_con_second = strstr(str_con_first,"</object");
  char str_confidence[10] = {};
  char str_order[5] = {};
  strncpy(str_confidence, str_con_first+25, str_con_second - str_con_firs
t-45);
  memcpy(str_order,str_confidence,5);
  //printf("\n%s\n",str_order);
  std_msgs::String msg_pub;
  if(atoi(str_confidence2) <40)
   {
    msg_pub.data="00001";
    pub.publish(msg_pub);
   }
   else
   {
  msg_pub.data =str_order;
  pub.publish(msg_pub);
   }
```

因为科大讯飞提供的demo，利用引擎反馈回来的数据就是就是下面给出的例子所示：

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?><nlp>
```

```
    <version>1.1</version>
    <rawtext>打电话给丁伟</rawtext>
    <confidence>94</confidence>
    <engine>local</engine>
    <result>
      <focus>dialpre|contact</focus>
      <confidence>81|100</confidence>
      <object>
        <dialpre id="10001">打电话给</dialpre>
        <contact id="65535">丁伟</contact>
      </object>
    </result>
  </nlp>
```

这组数据包含了反馈回来的结果，我们主要关注的是相应的置信度的参数 ，我们设定的ID号和相应的内容。这里我们将他们提取出来，就是通过在这里面查找相应的字符元素，然后获取相应数据的位置信息，然后将需要对应数据的信息提取出来存在一个数组里，然后在转换成`string_msg`型的ROS话题数据，通过ROS话题去调度相应的反馈。可以将上面的注释给取消掉，然后看注释理解一下这个提取的方法。
这里我们主要提取出来的数据就是ID号和置信度的数据，通过ID号知道当前的命令是什么，通过置信度检查当前的命令有多少可能性是我们提出的命令。这里可以进行一个命令置信度低的一个筛查，将低置信度的命令给忽略掉。我们这里设定的阈值是40，这个值可以根据需要自行更改。当阈值过低的话，很容易触发错误指令。

因为我们使用的是VAD算法，实时检测声音，如果有杂音的话，他也会实时检测到，这里我们通过设置几个标志位，来表示不同的状态，利用标志位做一个简单的状态机。通过状态机来控制机器人，判断目前是否满足执行相应的动作的条件 。

这里给出的demo是将功能和唤醒分为两个模块来实现我们的控制。这里可以基于这种模式进行结构更加多样化的控制 ，当然那样需要的设计的逻辑，需要考虑的方面也更加的多，这里我们只给了一个简单的demo。

首先我们来看一下我们的语音唤醒的设计。

```
    int ret;
    time( &rawtime );
    info = localtime( &rawtime );
    int todo_id = atoi(msg->data.c_str());
    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);
    }
```

这里我们设计了一个随机函数，虽然他是一个伪随机，当随机种子固定后，出现的结果也是固定的，但是这个设计能够提升我们的一个交互性，我们的机器人的语音反馈不在是固定的只能用一个反馈语句进行交互，而是可以在几个反馈语句中随机选择一个进行反馈。

接下来就是讲解功能的添加

例如我们加一个时间交互。
然后在`<control>:<wake>|<move>;`处添加一个`|<time>`;
即
`<control>:<wake>|<move>|<time>;`

添加一个`<time>`:时间!id(10006)|现在几点!id(10006)|现在!id(10006);

这时候如果在对话的时候，使用了时间的命令词，则会反馈下面的id(10006);

我们在语音合成tts_offline_sample.cpp中可以修改接收到的命令为相应id时，应该做什么处理。
比如

```
    if(todo_id==10006&&ifwake==1)
    {
        strftime(buffer,80,"现在时间是:%Y年%m月%e日，%H点%M分%S秒", info);//以年
月日_时分秒的形式表示当前时间
```

```
        const char* text =buffer; //合成文本
        ret =text_to_speech(text, filename, session_begin_params);
        if (MSP_SUCCESS != ret)
        {
          printf("text_to_speech failed, error code: %d.\n", ret);
        }
        printf("收到获取时间指令");
        printf("合成完毕\n");
        play_wav((char*)"/home/handsfree/talking/talking/catkin_ws/src/offl
inelistener/config/tts_sample.wav");
        ifwake=0;
        std_msgs::String msg_pub;
        msg_pub.data ="stop";
        pub.publish(msg_pub);

    }
```

`if(todo_id==10006&&ifwake==1)`此处的判断的是在接受到命令的同时是否处于唤醒状态。如果满足这两个条件执行后面的命令。这里就是读取当前计算机设备的时间信息，然后通过科大讯飞的语音合成功能将这个时间信息按我们设定的语句转换为wav文件，然后将语音通过外放播报出来，实现人机语音交互。

这里的关机（预设ID:10005）命令也是非常重要的！！

因为我们的vad算法是通过打开线程通过循环函数去控制，而ROS也相当于开了一个线程，所以我们如果不进行一个处理的话，会导致进程卡死，这里我们就设置了这个关机指令，通过关机指令来结束我们其他的线程。包括后续要加入的语音巡逻功能也是如此，只要添加了线程类或者在一个循环里的ROS代码，都可以通过这条指令根据相应的关机话题去处理。比如：

```
def roscallback(data):
    if int(data.data)==10005:
            global start
            start= 0
            print start
    rospy.loginfo(data.data)
//这里就是设定一个全局的变量，跳出循环函数，这里我们的vad结束后语音就没办法接受数据了，
所以我们这里一跳出循环，程序就结束了，或者将这个线程杀死了。
pub = rospy.Publisher('vad_listener_hear', String, queue_size=10)
rospy.init_node('vad_listener', anonymous=True)
sub = rospy.Subscriber("order_todo", String, roscallback)
rate = rospy.Rate(10) # 10hz
global start
while start==1:
  #下面是循环执行的任务
```

通过修改这些参数配置，可以调用我们机器人上有的能够提供ROS功能包的功能相结合，实现语音来控制其他功能的实现，这里我们提供了`语音控制`和`语音导航`和`语音巡逻`的demo。这些会在后面的课程进行单独的讲解。

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