寻找客人

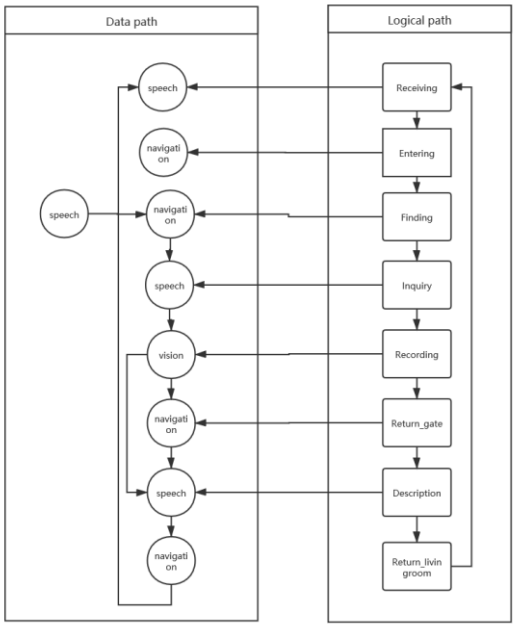
1811615 刘昱君

摘要：本项目中，机器人需要完成的任务是寻找客人，分为导航，视觉和语音三个模块。语音模块具有三个子任务：与主人以及客人进行语音交互，在语音的辅助下为客人拍照。通过讯飞在线 API 实现了语音识别和语音合成这两个基础技术。当为客人拍照时，机器人会通过计算人脸中心和机器人视野中心的相对位置判断客人的位置是否合适。

关键词：寻找客人；语音识别；语音合成；拍照

1 整体框架

该项目的背景为：主人举办了一场宴会，但只知道参加宴会的客人的姓名，因此，需要使用机器人获取客人的特征和位置。项目流程图如下：



机器人从主人处获取到客人的名字后，会进入特定的空间寻找客人。当机器人找到目标后，将与客人聊天，为其拍照，记录客人的年龄、性别、位置等信息。然后，机器人将返回主人处汇报客人的特征和位置。最终，返回起点，准备寻找下一位客人。

本项目主要包括三个模块：导航，视觉和语音。本文主要介绍语音部分。这部分的基本技术是语音识别和语音合成。语音识别主要用于接受主人的命令，理解客人的问题；语音合成主要用于回答客人的问题，提醒客人挥手，协助完成拍照功能，向主人汇报客人的信息。

2 功能实现

语音部分具有三个功能：与主人以及客人进行语音交互，在语音的辅助下为客人拍照。

2.1 语音交互（主人）

2.1.1 节点简介

fmm_speech.py：从主人处获取客人的名字，以及向主人汇报客人的特征和位置。

speech_recognition.cpp：语音识别。

soundplay_node.py：语音合成。

2.1.2 fmm_speech.py 节点详解

此节点包含六个部分：

Part 1：导入包。

Part 2：定义发布器和订阅器，以及 to_find_person_name, age, gender 等参数的初始化。

Part 3：如果节点订阅到 speech_recognition 节点发布的主题/xunfei_to_control，机器人将获取到客人的姓名。消息类型为字符串，内容为对客人姓名的识别结果，回调函数 xfeiCallback 的代码如下：

```
def xfeiCallback(self,msg):
    if msg.data.strip()=='':
        self.sh.say("I did not clearly hear you", self.voice)
        self.sh.say("please tell me again", self.voice)
        os.system("rostopic pub -1 /kws_data std_msgs/String 'jack'")

    else:
        string = msg.data
        symbols = ["!", "?", ".", ",", ";", ":"]
        output = []
        if string[-1] in symbols:
            string = string[:-1]
        for part in string.lstrip().split(","):
            for word in part.split():
                for symbol in symbols:
                    if symbol in word:
                        word = word[:-1]
                output.append(word)
        output = [item.lower() for item in output]
        print output

    for i in range(len(output)):
        if output[i] in self.name_list:
            self.to_find_person_name = output[i]
            break
    if self.to_find_person_name == None:
        self.sh.say("sorry, please tell me again", self.voice)
        os.system("rostopic pub -1 /kws_data std_msgs/String 'jack'")
    else:
        self.sh.say("ok i will find {}".format(self.to_find_person_name), self.voice)
        self.message.NowTask = self.message.Receiving
        self.message.NextTask = self.message.EnterHome
        self.message.FinishState = True
        self.message.NeedHelp = False
        self.speech_pub_to_control.publish(self.message)
        os.system("rosnode kill /voiceRecognition")#关闭语音识别节点
```

语音识别是通过 speech_recognition 节点实现的。

Part 4：如果节点订阅到 fmm_control 节点发布的主题/find_my_mate/control，回调函数 controlback 中分三种情况，代码如下：

```
def controlCallback(self,msg):
    if msg.NowTask == self.message.Receiving and msg.FinishState == False:
        self.sh.say("Dear operator, please talk to me after you hear ding ding dong", self.voice)
        self.sh.say("ok I'm ready, please give me the name", self.voice)
        self.message.NowTask = msg.NowTask
        self.message.NextTask = msg.NextTask
        self.message.FinishState = False
        self.message.NeedHelp = False

    if msg.NowTask == self.message.Inquiry and msg.FinishState == False:
        self.sh.say("Dear guest, i want to find {},please wave your hand".format(self.to_find_person_name), self.voice)
        self.message.NowTask = msg.NowTask
        self.message.NextTask = msg.NextTask
        self.message.FinishState = True
        self.message.NeedHelp = False
        self.speech_pub_to_control.publish(self.message)
    if msg.NowTask == self.message.Description and msg.FinishState == False:
        self.sh.say("Dear operator,the information of {} as follows".format(self.to_find_person_name), self.voice)
        self.information()
        self.sh.say(self.person_information, self.voice)
        self.sh.say("ok i will find next person, please give me the name", self.voice)
        self.to_find_person_name = None
        self.person_information = None
        os.system("gnome-terminal -x bash -c 'rosrun xfei_asr speech_recognition'")#开启语音识别节点
        rospy.sleep(1)
        os.system("rostopic pub -1 /kws_data std_msgs/String 'jack'")
        self.message.NowTask = msg.NowTask
        self.message.NextTask = msg.NextTask
        self.message.FinishState = True
        self.message.NeedHelp = False
        self.speech_pub_to_control.publish(self.message)
```

Case 1：参数 NowTask 的值为 receiving。机器人会说“Please give me the name”。

Case 2：参数 NowTask 的值为 inquiry。机器人会说“Dear guest, I want to find XXX. Please wave your hand”。

Case 3：参数 NowTask 的值为 description。机器人会说“Dear host, the information of XXX is as follows.”。然后，机器人会将客人的特征和位置汇报给主人，这些信息是在函数

information 中被赋值的，代码如下：

```
def information(self):
    if self.gender == "male":
        self.person_information = "the guest is a " + self.gender + ", "
        if self.age != "unknown":
            self.person_information = self.person_information + "he is about {} years old, ".format(self.age)
        if self.skin_color != "unknown":
            self.person_information = self.person_information + "his skin color is " + self.skin_color + ", "
        if self.hair_style != "unknown":
            self.person_information = self.person_information + "his hair style is " + self. hair_style + ", "
        if self.glasses != "unknown":
            if self.glasses != "none":
                self.glasses = "glasses"
            self.person_information = self.person_information + "he is wearing " + self.glasses + ", "
        if self.clothes_color != "unknown":
            if self.clothes != "unknown":
                self.person_information = self.person_information + "he is wearing a " + self.clothes_color + " " + self.clothes + ", "
            else:
                self.person_information = self.person_information + "he dresses up in " + self.clothes_color + ", "
        if self.pose == "sit":
            self.person_information = self.person_information + "he is sitting on the " + self.position
        if self.pose == "stand":
            self.person_information = self. person_information + "he is standing around the " + self.position

    if self.gender == "female":
        self.person_information = "the guest is a " + self.gender + ", "
        if self.age != "unknown":
            self.person_information = self.person_information + "she is about {} years old, ".format(self.age)
        if self.skin_color != "unknown":
            self.person_information = self.person_information + "her skin color is " + self.skin_color + ", "
        if self.hair_style != "unknown":
            self.person_information = self.person_information + "her hair style is " + self. hair_style + ", "
        if self.glasses != "unknown":
            self.person_information = self.person_information + "she is wearing " + self.glasses + ", "
        if self.clothes_color != "unknown":
            if self.clothes != "unknown":
                self.person_information = self.person_information + "she is wearing a " + self.clothes_color + " " + self.clothes + ", "
            else:
                self.person_information = self.person_information + "she dresses up in " + self.clothes_color + " "
        if self.pose == "sit":
            self.person_information = self.person_information + "she is sitting on the " + self.position
        if self.pose == "stand":
            self.person_information = self. person_information + "she is standing around the " + self.position
```

语音合成是通过 soundplay_node 节点实现的。

Part 5：如果节点订阅到主题/image/people_feature，消息类型是 description，内容是客人的特征，回调函数为 visDesCallback，用于将特征赋值给 self，代码如下：

```
def visDesCallback(self,msg):
    if self.flag==1:
        self.age = msg.age
        self.gender = msg.gender
        self.skin_color = msg.skin_color
        self.hair_style = msg.hair_style
        self.glasses = msg.glasses
        self.clothes_color = msg.clothes_color
        self.clothes = msg.clothes
        self.pose = msg.pose
        self.flag=0
```

Part 6：如果节点订阅到主题/image/people_position，消息类型是 position，内容是客人的位置。回调函数为 visPosCallback，用于将位置赋值给 self，代码如下：

```
def visPosCallback(self,msg):
    self.position = msg.data
```

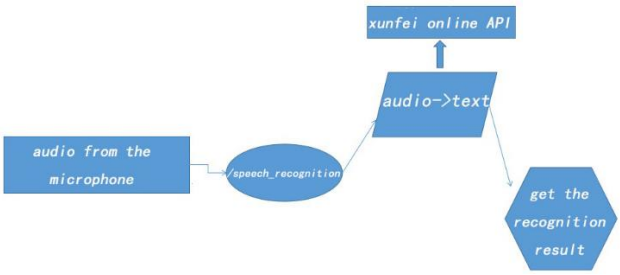
2.1.3 Speech_recognition.cpp 节点详解

当使用 xfei_asr 包中的 iat_record 节点进行语音识别时，需要选择是否上传用户词典，以及是从麦克风获取音频还是识别本地音频。因此，我修改了此节点，直接设置为不上传用户词典，从麦克风获取音频。

另外，我还做了另一个设置。在进行语音识别之前，机器人将发出提示音频以提醒主人，它已准备好接收命令。

```
sound_client.playWave("/home/qian/catkin_ws/src/xfei_asr/sounds/question_start_signal.wav");
```

使用此节点时，流程如下：



从麦克风处获取音频后，使用讯飞在线 API 进行语音识别。如果检测到 VAD 特征或已

过 10 秒钟，则将识别结果以文本形式返回。

2.2 语音交互（客人）

2.2.1 节点简介

en_iat_publish.cpp：语音识别。

en_voice_assistance.cpp：语音合成。

2.2.2 流程

Step 1：通过麦克风获取音频，音频内容为客人的问题。

Step 2：使用 en_iat_publish 节点进行语音识别。发布主题/xfwords，消息类型为字符串，内容为音频的识别结果。

Step 3：en_voice_assistance 节点将订阅此主题。通过提取识别结果中的关键字在对话库中搜索恰当的回答。然后，将回答转换为音频。

Step 4：播放音频。至此，与客人聊天的功能完成。

2.2.3 en_iat_publish.cpp 节点详解

该节点类似于 speech_recognition 节点，都没有使用用户词典。但如果机器人需要与客人进行更复杂的对话，可以将不常见的词添加到用户词典中，并选择上传用户词典。程序会优先识别这些单词，匹配率非常高。

两个节点之间的区别在于 en_iat_publish 节点不播放提示音频。但是，每当机器人执行语音识别功能之前，都需要主题/ xfwakeup 进行唤醒。

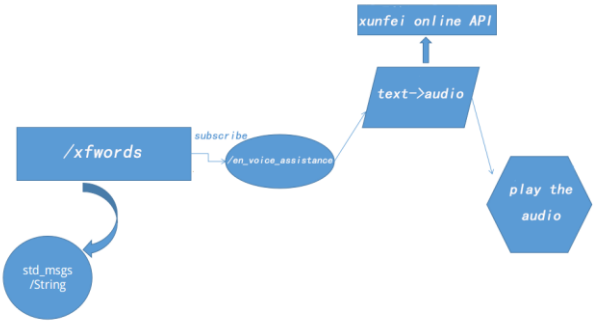
```
void WakeUp(const std_msgs::String::ConstPtr& msg)
{
    printf("waking up\r\n");
    usleep(700*1000);
    wakeupFlag=1;
}

int main(int argc, char* argv[])
{
    // 初始化ROS
    ros::init(argc, argv, "voiceRecognition");
    ros::NodeHandle n;
    ros::Rate loop_rate(10);

    // 声明Publisher和Subscriber
    // 订阅唤醒语音识别的信号
    ros::Subscriber wakeUpSub = n.subscribe("xfwakeup", 1000, WakeUp);
    // 订阅唤醒语音识别的信号
    ros::Publisher voiceWordsPub = n.advertise<std_msgs::String>("xfwords", 1000);

    ROS_INFO("Sleeping...");
    int count=0;
    while(ros::ok())
    {
        // 语音识别唤醒
        if (wakeupFlag){
            ROS_INFO("Wakeup...");
            int ret = MSP_SUCCESS;
            const char* login_params = "appid = 58249817, work_dir = .";

            // language =  zh_cn 则切换成中文
            const char* session_begin_params =
                "sub = iat, domain = iat, language = en_us, "
                "accent = mandarin, sample_rate = 16000, "
                "result_type = plain, result_encoding = utf8";

            ret = MSPLogin(NULL, NULL, login_params);
            if(MSP_SUCCESS != ret){
                MSPLogout();
                printf("MSPLogin failed , Error code %d.\n",ret);
            }

            printf("Demo recognizing the speech from microphone\n");
            printf("Speak in 10 seconds\n");

            demo_mic(session_begin_params);

            printf("10 sec passed\n");

            wakeupFlag=0;
            MSPLogout();
        }
```

2.2.4 en_voice_assistance.cpp 节点详解

当使用 xfei_asr 包中的 tts_subscribe_speak 节点进行语音合成时，会直接合成消息的内容。我修改了此节点，添加了一个对话库。机器人将根据识别结果中的关键字在对话库中搜索恰当的回答。然后，将回答进行语音合成并播放音频。这是对话库的一部分：

```
if((dataString.find("old") !=-1))
{
    char nameString[60] = "I am five years old.";
    text = nameString;
    std::cout<<text<<std::endl;
}
else if((dataString.find("from")!=-1))
{
    char helpString[50] = "I am from China.";
    text = helpString;
    std::cout<<text<<std::endl;
}
else if((dataString.find("joke") != -1))
{
    char helpString[200] = "ok. What is orange and sounds like a parrot? Erm, It is a carrot. Ha ha ha.";
    text = helpString;
    std::cout<<text<<std::endl;
}
else if((dataString.find("end") != -1)||(dataString.find("nice") != -1))
{
    flag=0;
    char helpString[200] = "It is nice to talk with you. Looking forward to talking with you next time.";
    text = helpString;
    std::cout<<text<<std::endl;
}
```

使用此节点时，流程如下：



首先，订阅主题/ xfwords，消息类型为字符串，内容为对客人问题的识别结果。然后，通过讯飞在线 API，将回答转换成音频。最后，播放音频。

2. 3 拍照

2. 3. 1 节点简介

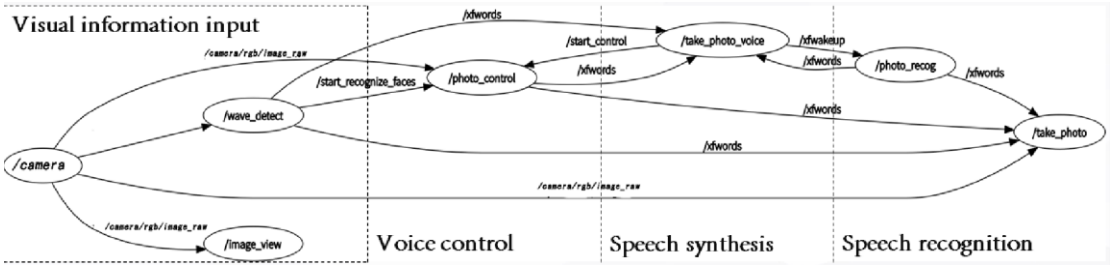
photo_control.py：判断客人是否位于视野中央，如果没有，指导客人进行移动。

take_photo_voice.cpp：询问客人是否要拍照，并将 photo_control 节点中的移动命令进行语音合成。该节点与 en_voice_assistance 节点类似，只是对话库不同，此处不再赘述。

photo_recog.cpp：使用语音识别来判断客人的答案。该节点与 en_iat_publish 节点相同，此处不再赘述。

take_photo.py：保存图片，完成拍照功能。

2. 3. 2 流程



Step 1：photo_control 节点订阅主题/start_recognize_faces 后，机器人开始进行人脸检测。如果检测到人脸，它将发布主题/xfwords，消息类型为字符串，内容为“Now, I find XXX.”。

Step 2：take_photo_voice 节点订阅到此主题后，询问客人是否要拍照，并发布主题/xfwakeup。

Step 3：机器人需要通过语音识别来判断客人的回答。因此，photo_recog 节点将订阅该主题并发布另一个主题/xfwords。消息类型为字符串，内容为对客人回答的识别结果。

Step 4：take_photo_voice 节点订阅/xfwords 主题。如果客人同意，该节点将发布主题/start_control。

Step 5：photo_control 节点将订阅该主题。机器人将根据客人的位置给出建议，指导客人移动到视野中央。如果客人已经到达合适的位置，机器人会说“take photo”。语音合成将由 take_photo_voive 节点实现。该过程与之前的过程类似，此处不再赘述。

Step 6：如果/xfwords 主题中消息的内容为“take photo”，则 take_photo 节点会将当前视野保存为一张图片。至此，拍照功能完成。

2.3.3 photo_control.py 节点详解

机器人通过计算人脸中心和机器人视野中心的相对位置来判断客人的位置是否合适。如果客人没有位于机器人视野中心，机器人将指导客人进行移动，包括“ step back”, “ move forward”, “ move left”, “ move right”。

```
    # 在opencv的窗口中框出所有人脸区域
        if len(faces_result)>0:
            for face in faces_result:
                x, y, w, h = face
                cv2.rectangle(cv_image, (x, y), (x+w, y+h), self.color, 2)
            mid_x=x+w/2
            mid_y=y+h/2
            rospy.loginfo(mid_x)
            rospy.loginfo(mid_y)
        # 根据脸的位置发布速度消息
            if  (mid_y<=140 and 220<mid_x and mid_x<420):
                strings="Please step back a little"

        # 后退
            elif (mid_y>=340 and 220<mid_x and mid_x<420):
                strings="Please move forward a little"


        # 停止
            elif (mid_y>=140 and mid_y<=340 and 220<=mid_x and mid_x<=420):
                self.take_photo_pub=True
                strings="take photo"


        # 左转
            elif (mid_x>=420):
                strings="Please move left a little"


        # 右转
            elif (mid_x<=220):
                strings="Please move right a little"

            else:
                strings="other"

        #if (key == '\x03'):
            #break
            self.voice_pub.publish(strings)
            rospy.loginfo("strings:%s"%strings)


        # 将识别后的图像转换成ROS消息并发布
        self.image_pub.publish(self.bridge.cv2_to_imgmsg(cv_image, "bgr8"))
    #rospy.loginfo("image_pubs")
```

3 展望

为了使机器人展现出更好的性能，还需要进行一些改进。

3.1 问题

机器人在面对复杂环境时，语音识别的鲁棒性还不够好。要想达到较高的识别率，需要满足很多的条件：说话要靠近，发音要标准，环境要安静，不能持续对话，不能打断等等。

3.2 改进

3.2.1 声纹特征识别技术

设计一种可以识别每个人的声纹特征的机器学习或深度学习算法。算法可以将每个人声音里的不同特质进行分类以便完成多个声音的分解，并重组每个人的话。此外，也可以对不同的语言，如英语、日语等，进行分类识别。

3.2.2 麦克风阵列技术

通过麦克风的排列以及基于麦克风阵列的声源定位算法，如基于波束形成的方法、基于声达时延差（TDOA）的方法，提高语音识别的准确率。