Find My Mates

1811615

Yujun Liu

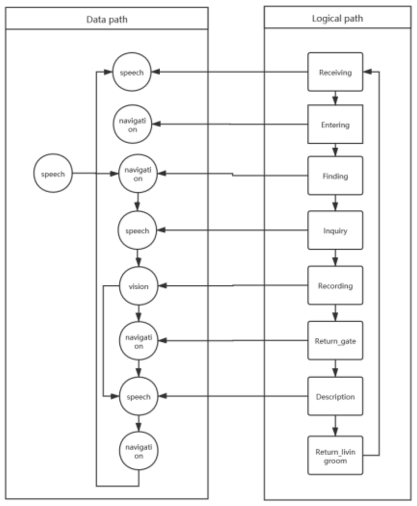
Abstract

In our project, the robot needs to complete the task of finding my mates. There are three modules: navigation, vision and voice. The voice part has three subtasks: having basic conversations with the host, chatting with the guests and assisting the completion of taking photos. Speech recognition and speech synthesis are the basic technologies, which are realized by Xunfei API. When taking photos for the guest, the robot calculates the relative position of the face center and the robot vision center to judge whether the guest is in the center of the vision.

Keywords: find my mates; speech recognition; speech synthesis; taking photos

1 Overall introduction

The background of the project is the host only knows the names of the guests who participate in the banquet, so a robot is sent to collect the information of the guests, and then reports to the host. The flow chart for this project is as follows:



After receiving the guest’s name from the host, the robot will enter the designated space and look for the guest. When robot finds the target, it will chat with the guest and take a picture for him/her. Then, robot will record the guest’s age, clothing, location and so on. After that, the robot will return to the host and report the description and location of the guest. Finally, the robot will go to the starting point and prepare finding next guest.

Our project mainly includes three modules: navigation, vision and voice. This paper will introduce the voice part.

The basic technologies for this part are speech recognition and speech synthesis. Speech recognition is mainly used to accept operator’s commands and comprehend guest’s questions. And speech synthesis is mainly used to answer the questions of the guests, remind the guest to wave hand, assist the completion of taking photos, and report the guest’s information to the operator.

2 Function

In voice part, there are three functions: having basic conversations with the host, chatting with the guests and assisting the completion of taking photos.

2.1 having basic conversations with the host

2.1.1 Introduction to the nodes

fmm_speech.py: Get the name of the guest from the host and report the description and position of the guest to the host.

speech_recognition.cpp: Speech recognition.

soundplay_node.py: Speech synthesis.

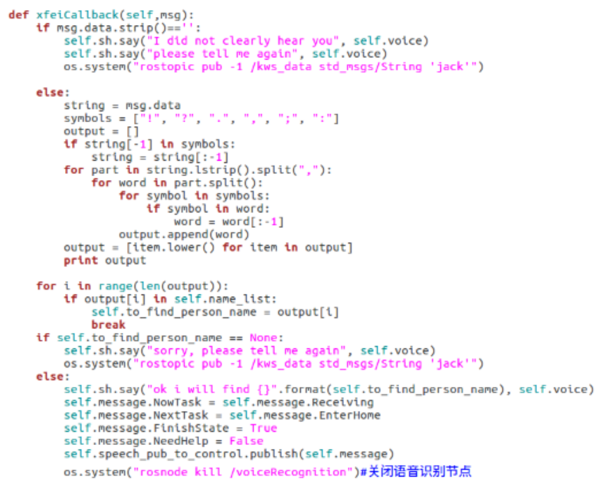
2.1.2 Detailed analysis for fmm_speech.py

There are six parts in this node:

Part 1: Import package.

Part 2: The definitions of publishers and subscribers and initialization of parameters, including to_find_person_name, age, gender and so on.

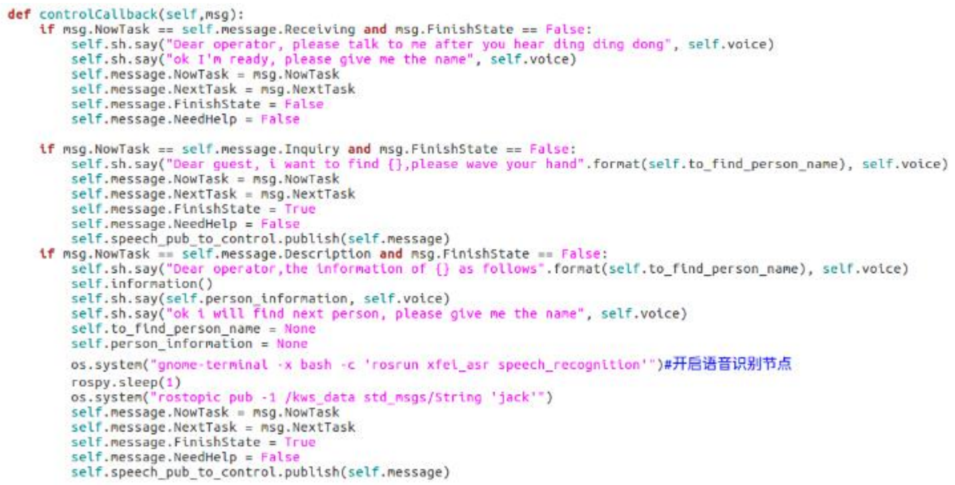
Part 3: The node subscribes to the topic /xunfei_to_control, which is published from the speech_recognition node. The message type is string and the content is the recognition result of the guest's name. Then, the robot can get the name of the guest. The callback function is xfeiCallback. The code is as follows:



```python
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
```

Speech recognition is realized by the speech_ recognition node.

Part 4: The node subscribes to the topic /find_my_mate/control, which is published from the fmm_control node. And the callback function is controlback. There are three cases in this part. The code is as follows:



```python
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: The value of NowTask is receiving. The robot will say "Please give me the name".

Case 2: The value of NowTask is inquiry. The robot will say "Dear guest, I want to find XXX. Please wave your hand".

Case 3: The value of NowTask is description. The robot will say "Dear host, the information of XXX is as follows." Then, the robot will report the description to the host, which is assigned in the function information.

```
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
```

Speech synthesis is realized by the soundplay_node node.

Part 5: The node subscribes to the topic /image/people_feature. The message type is description and the content is the feature of the guest. The callback function is visDesCallback, which is used to assign the feature to self. The code is as follows:

```
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: The node subscribes to the topic /image/people_position. The message type is position and the content is the position of the guest. The callback function is visPosCallback, which is used to assign the position to self. The code is as follows:

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

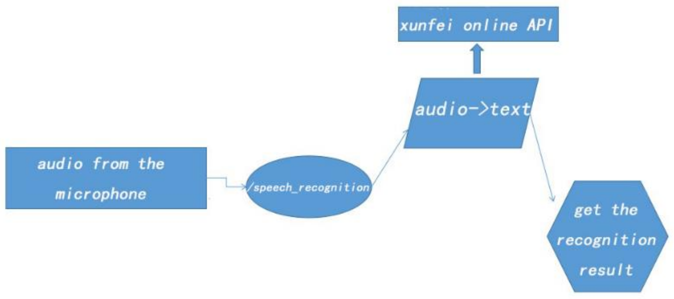
2.1.3 Detailed analysis for speech_recognition.cpp

When using the iat_record node in xfei_asr package, we need to choose whether to upload the user words and recognizing the audio from microphone or from an audio file. So, I modified this node and directly set to not upload the user words, recognize from the microphone.

In addition, I also made another change. Before speech recognition, an audio cue will be played to remind the host that the robot is ready to receive the name.

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

When using this node, the process is as follows:



First, get audio from the microphone. Then, use the xunfei online API for recognition. If voice activity detection (VAD) features are found or 10 seconds have passed, the result will be returned in the form of text.

2.2 chat with the guest

2.2.1 Introduction to the nodes

en_iat_publish.cpp: Speech recognition.

en_voice_assistance.cpp: Speech synthesis

2.2.2 flow

Step 1: Get the question of the guest from the microphone.

Step 2: Use the en_iat_publish node to recognize the audio. And it will publish a topic /xfwords. The message type is string and the content is the recognition result of the audio.

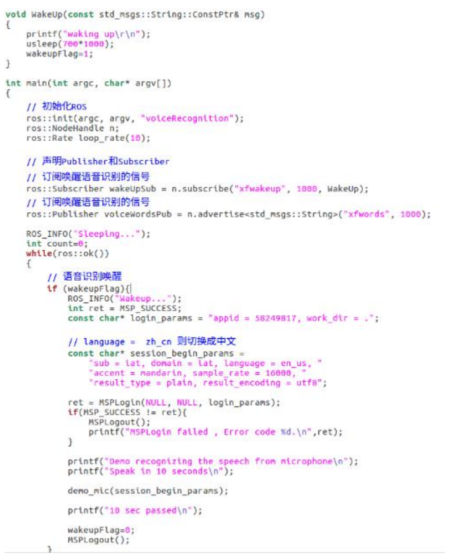
Step 3: The en_voice_assistance node will subscribe it. The robot will search for appropriate answers in the dialogue library based on the keywords in the recognition result. And then, convert the text into audio.

Step 4: Play the audio. At this point, the function of chatting with guest is completed.

2.2.3 Detailed analysis for en_iat_publish.cpp

This node is similar to the speech_recognition node. Neither of them uses user words. If the robot needs to have more complex conversations with the guest, we can add uncommon words into the user words to upload. Because the robot will recognize these words first, and the matching rate of these words is very high.

The difference between the two nodes is that the en_iat_publish node don't play the audio cue. But every time the robot performs speech recognition, the topic /xfwakeup needs to be sent to wake up the function.

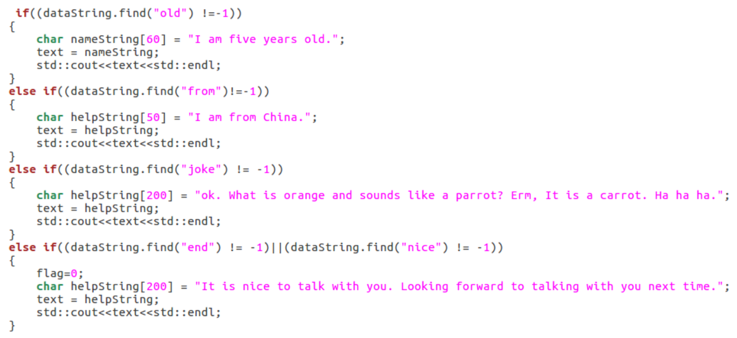


```
void WakeUp(const std_msgs::String::ConstPtr& msg)
{
    printf("waking up\r\n");
    usleep(700*1000);
    wakeupFlag=1;
}

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 Detailed analysis for en_voice_assistance.cpp

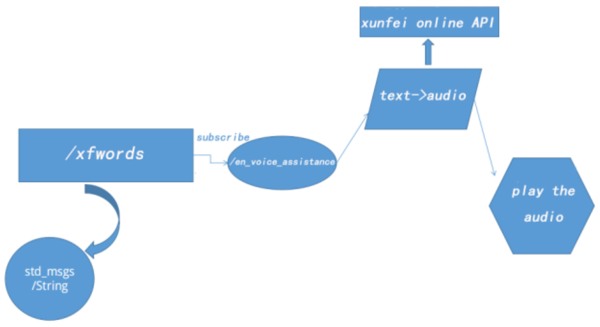
When using the the tts_subscribe_speak node in xfei_asr package, it will directly synthesize the content of message. I modified this node and added a dialogue library. The robot will search for appropriate answers in the dialogue library based on the keywords in the recognition result. And then, synthesis the answer and play the audio.

There is part of the dialogue library.



```
 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;
}
```

When using this node, the process is as follows:



First, subscribe to the topic /xfwords. The message type is string and the content is the recognition result. And then, by using the xunfei online API, the answer is formed into an audio. Finally, play the audio.

2.3 taking photos

2.3.1 Introduction to the nodes

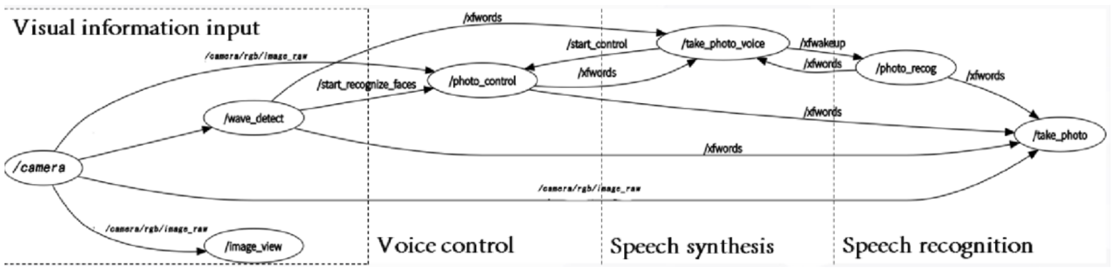
photo_control.py: Judge the position of the guest. If not, guide the guest to move.

take_photo_voice.cpp: Ask if the guests want to take a picture. And synthesize the mobile commands in the photo_control node. This node is similar to en_voice_assistance node. The only difference is the dialogue library. No more details here.

photo_recog.cpp: Judge the guest's answer. This node is same as en_iat_publish node. No more details here.

take_photo.py: Save the picture and complete the function of taking photos.

2.3.2 flow



Step 1: If the photo_control node subscribes to the topic /start_recognize_faces, the robot will begin to detect faces. If a face is detected, it will publish a topic /xfwords. The message type is string and the content is “Now, I find XXX”.

Step 2: The take_photo_voice node will subscribe it. The robot will ask if the guest wants to take a picture. And then, it will publish the topic /xfwakeup.

Step 3: The robot needs to judge the answer of the guest by recognition. So, the photo_recog node will subscribe to this topic and publish another topic /xfwords. The message type is string and the content is recognition result of the guest’s answer.

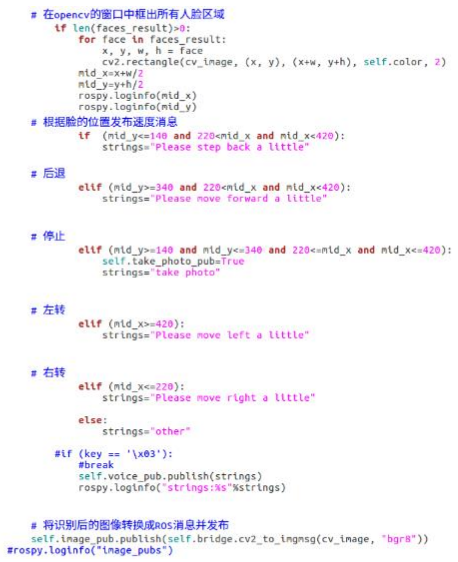
Step 4: The take_photo_voice node will subscribe it. If the guest agrees, the node will publish the /start_control topic.

Step 5: The photo_control node will subscribe to this topic. Then the robot will give suggestions according to the position of the guest, until the guest is in the center of the robot vision, it will say "take photo". Speech synthesis will be realized by the take_photo_voive node. The process is similar to the previous, and will not be repeated here.

Step 6: If the content of the message in the topic /xfwords is “take photo”, the take_photo node will save the current field of view as a picture. At this point, the function of taking photos is completed.

2.3.3 Detailed analysis for photo_control.py

As for how to judge whether the customer is in the center of the vision, the robot realizes this task by calculating the relative position of the face center and the robot vision center. If not, it will issue a movement command to the guest, including “ step back”,” move forward”,” move left” and ” move right”.

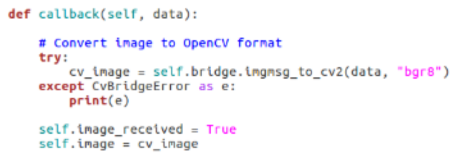


```
    # 在opencv的窗口中框出所有人脸区域
        if len(faces_result)>0:
            for face in faces_result:
                x, y, w, h = face
                cv2.rectangle(cv_image, (x, y), (x+w, y+h), self.color, 2)
            mid_x=x+w/2
            mid_y=y+h/2
            rospy.loginfo(mid_x)
            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")
```

2.3.4 Detailed analysis for take_photo.py

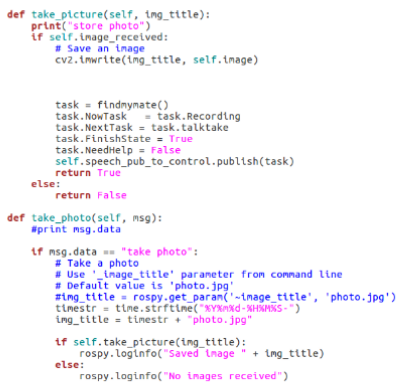
There are two parts in this node.

Part 1: Convert the image to opencv format.



```
def callback(self, data):

    # Convert image to OpenCV format
    try:
        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
    except CvBridgeError as e:
        print(e)

    self.image_received = True
    self.image = cv_image
```

Part 2: If the content of the message in the topic /xfwords is “take photo”, the take_photo node will save the current field of view as a picture. And set the value of the NowTask to recording, the value of the NextTask to talktake.



```
def take_picture(self, img_title):
    print("store photo")
    if self.image_received:
        # Save an image
        cv2.imwrite(img_title, self.image)


        task = findmymate()
        task.NowTask   = task.Recording
        task.NextTask = task.talktake
        task.FinishState = True
        task.NeedHelp = False
        self.speech_pub_to_control.publish(task)
        return True
    else:
        return False

def take_photo(self, msg):
    #print msg.data

    if msg.data == "take photo":
        # Take a photo
        # Use '_image_title' parameter from command line
        # Default value is 'photo.jpg'
        #img_title = rospy.get_param('~image_title', 'photo.jpg')
        timestr = time.strftime("%Y%m%d-%H%M%S-")
        img_title = timestr + "photo.jpg"

        if self.take_picture(img_title):
            rospy.loginfo("Saved image " + img_title)
        else:
            rospy.loginfo("No images received")
```

3 future work

Some improvements are necessary to make the robot perform better

3.1 problem

When environment is complex, the result of speech recognition is not accurate enough. To achieve a high recognition rate, a lot of conditions need to be met: speaking closely, standard pronunciation, quiet environment, etc.

3.2 improvement

3.2.1 recognition technology of voiceprint features

Design a machine learning or deep learning algorithm that can recognize each person's voiceprint features, classify different traits in each person's voice. Through this, we can complete the decomposition of multiple voices and reorganize each person's words. In addition, different languages, such as English and Japanese, can also be classified and identified.

3.2.2 microphone array technology

Adjust the arrangement of microphones and use sound source localization algorithms based on microphone arrays, such as beamforming-based methods, and time-of-arrival delay (TDOA) methods.