ROS基础 – 实训套件

实验案例 – 教学参考手册

7 语音交互

智能控制原理与技术 – 实训套件

实验案例 – 教学参考手册

安装机器人传感器及机械臂系统环境

ROS基础 – 实训套件

实验案例 – 教学参考手册

6 06 Move\_base与AMCL参数设置

ROS基础 – 实训套件

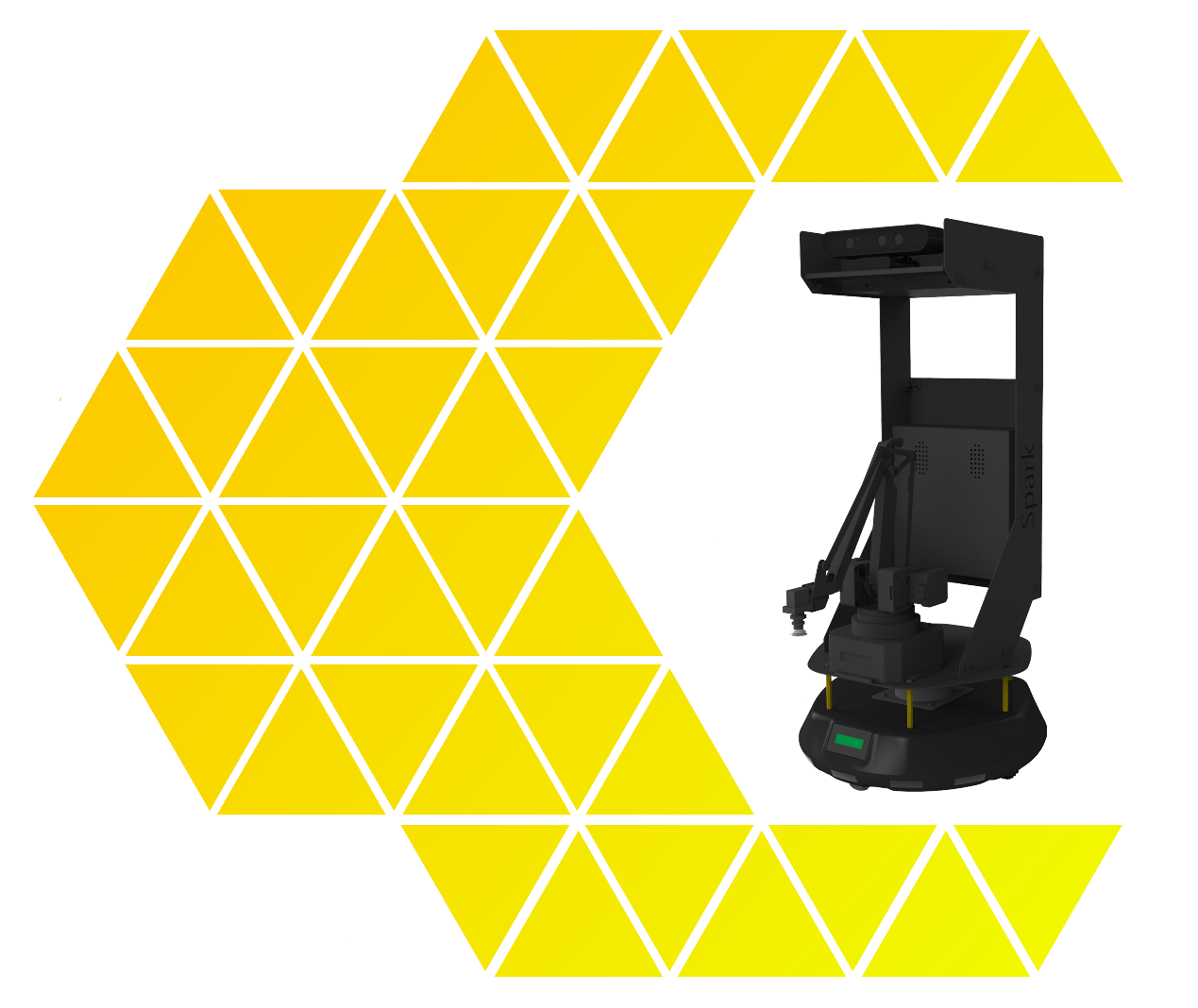
实验案例 – 教学参考手册

5 创建仿真机器人与现实机器人的同步

智能交互技术

实验案例

4 通过语音交互控制spark移动



ROS基础 – 实训套件

实验案例 – 教学参考手册

7 语音交互

**目 录**

[一、实验名称：通过语音交互控制Spark移动 1](#_Toc486434492)

[1、相关技能 1](#_Toc486434493)

[2、相关知识点 1](#_Toc486434494)

[3、实现效果 1](#_Toc486434495)

[4、实验要求 1](#_Toc486434496)

[5、实现思路 2](#_Toc486434497)

[6、验证与测试 5](#_Toc486434498)

[7、参考答案 5](#_Toc486434499)

一、实验名称：通过语音交互控制Spark移动

## 1、相关技能

* 安装深度摄像头驱动
* 安装sphinx Pocket包
* move\_base

## 2、相关知识点

* ASR的使用
* TTS的使用
* 移动底盘通讯
* 语音指令设计

## 3、实现效果

## 4、实验要求

* **本实验要求：**安装深度摄像头驱动，安装sphinx Pocket包，安装move\_base
  1. 安装深度摄像头驱动
  2. 安装sphinx Pocket包，实现语音识别
  3. 安装move\_base，接收语音指令

## 5、实现思路

* 1. 安装深度摄像头驱动

$ cd ~/catkin\_ws/src

$ git clone https://github.com/bosch-ros-pkg/usb\_cam.git

$ cd ~/catkin\_ws

$ catkin\_make

* 1. 安装sphinx Pocket包，实现语音识别

5-2.1安装依赖 ros-kinetic-audio-common、libasound2、libgstreamer0.10、python- gst0.10

|  |
| --- |
| **+ 提示：**  sudo apt-get install ros-kinetic-audio-common  sudo apt-get install libasound2  sudo apt-get install gstreamer0.10-\*  sudo apt-get install python-gst0.10 |

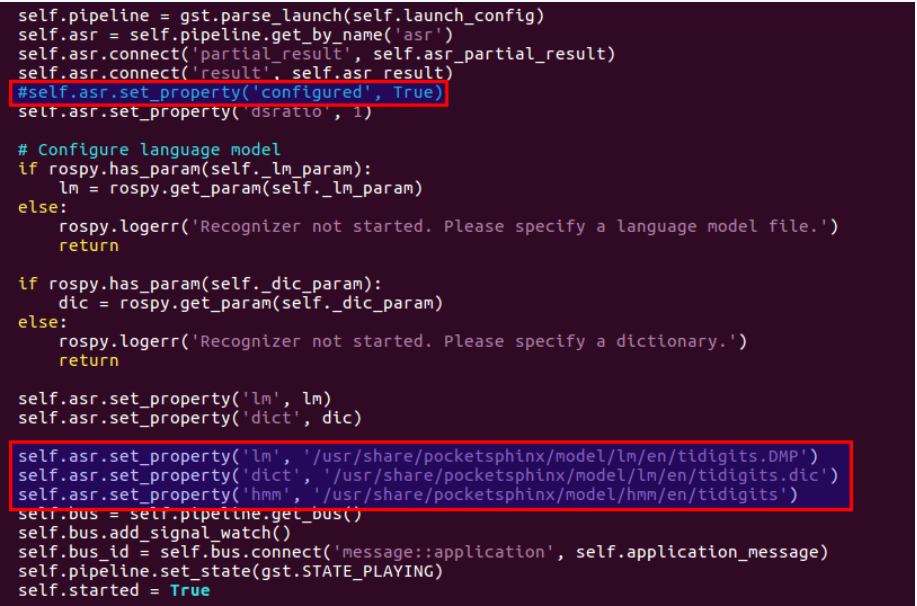
5-2.2安装pocketphinx包 （kinetic版本）

|  |
| --- |
| **+ 提示：**  1）安装 libsphinxbase1  下载地址：[https://packages.debian.org/jessie/libsphinxbase1](https://packages.debian.org/jessie/libsphinxbase1" \t "_blank)  sudo dpkg -i libsphinxbase1\_0.8-6\_amd64.deb  2）安装libpocketsphinx1下载地址：[https://packages.debian.org/jessie/libpocketsphinx1](https://packages.debian.org/jessie/libsphinxbase1" \t "_blank)  sudo dpkg -i libpocketsphinx1\_0.8-5\_amd64.deb  3）安装gstreamer0.10-pocketsphinx  下载地址：[https://packages.debian.org/jessie/gstreamer0.10-pocketsphinx](https://packages.debian.org/jessie/gstreamer0.10-pocketsphinx" \t "_blank)  sudo dpkg -i gstreamer0.10-pocketsphinx\_0.8-5\_amd64.deb  4）安装pocketsphinx  4.1）进入工作空间，git源码  cd ~/catkin\_ws/src git clone <https://github.com/mikeferguson/pocketsphinx>  4.2)下载英文语音包pocketsphinx-hmm-en-tidigits (0.8-5)  下载的网站：[https://packages.debian.org/jessie/pocketsphinx-hmm-en-tidigits](https://packages.debian.org/jessie/pocketsphinx-hmm-en-tidigits" \t "_blank)  sudo dpkg -i pocketsphinx-hmm-en-tidigits\_0.8-5\_all.deb  在包pocketsphinx下面建一个model目录，存放语音模型文件  cd ~/dev/catkin\_ws/src/pocketsphinx  mkdir model  将下载好的语音文件，解压后，将其中的model文件下的所有文件拷贝到~/dev/catkin\_ws/src/pocketsphinx/model下。 |

5.2.3修改recognizer.py（源码见7.0答案）

注释掉self.asr.set\_property('configured', True)

添加lm，dict，hmm支持英语识别（如果是其他语言（如普通话）可以改为别的路径）



5-2.4创建语料库

创建语音单词语句文件一行一句的txt文件

pause speech

continue speech

move forward

move backward

move back

move left

move right

编译生成语音库

通过在线的一个 语言模型（lm）生成

http://www.speech.cs.cmu.edu/tools/lmtool-new.html

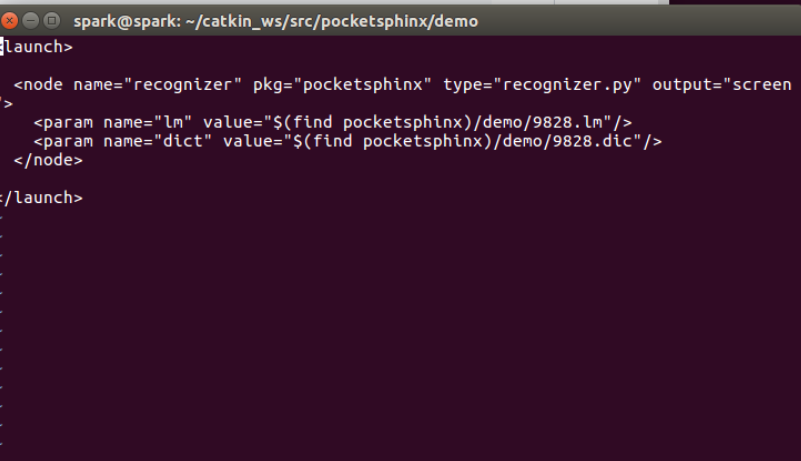
上传语言文件 Upload a sentence corpus file: Browse

在线编译 COMPILE KNOWLEDGE BASE

下载 编译好的文件

使用 .dic 字典文件 音节/音素 字典文件

.lm 语言默默文件 出现的概率



* 1. 安装move\_base，接收语音指令

1.安装

sudo apt-get install ros-kinetic-move-base

2.编写ROS语音控制节点

1)订阅语音识别结果主题

2）发布Twist数据到/cmd\_vel主题

参考 ~/spark/src/spark\_app/spark\_voice/目录下的文件

## 6、验证与测试

1.复制spark-class4到主目录下

2.编译

cd ~/spark-class4

catkin\_make

3.更新环境

source ~/spark-class4/devel/setup.bash

4.运行程序

roslaunch spark\_voice voice\_nav.launch

## 7、参考答案

#!/usr/bin/python

# -\*- coding:utf-8 -\*-

### 修改后的 文件

import roslib

roslib.load\_manifest('pocketsphinx')

import rospy

import pygtk # Python轻松创建具有图形用户界面的程序 播放音乐等

pygtk.require('2.0')

import gtk # GNU Image Manipulation Program (GIMP) Toolkit

import gobject # 亦称Glib对象系统，是一个程序库，它可以帮助我们使用C语言编写面向对象程序

import pygst # 与 pygtk 相关

pygst.require('0.10')

gobject.threads\_init()# 初始化

import gst

from std\_msgs.msg import String

from std\_srvs.srv import \*

import os

import commands

class recognizer(object):

"""GStreamer是一个多媒体框架，它可以允许你轻易地创建、编辑与播放多媒体文件"""

# 初始化系统配置

def \_\_init\_\_(self):

# 创建节点

rospy.init\_node("recognizer")

# 全局参数

self.\_device\_name\_param = "~mic\_name" # 麦克风

self.\_lm\_param = "~lm" # 语言模型 language model

self.\_dic\_param = "~dict" # 语言字典

self.\_hmm\_param = "~hmm" # 识别网络 hiden markov model 隐马尔可夫模型 分中英文模型

# 用 gstreamer launch config 配置 麦克风 一些启动信息

if rospy.has\_param(self.\_device\_name\_param):# 按照指定的麦克风

self.device\_name = rospy.get\_param(self.\_device\_name\_param)# 麦克风名字

self.device\_index = self.pulse\_index\_from\_name(self.device\_name)# 麦克风编号 ID

self.launch\_config = "pulsesrc device=" + str(self.device\_index)# 启动信息

rospy.loginfo("Using: pulsesrc device=%s name=%s", self.device\_index, self.device\_name)

elif rospy.has\_param('~source'):

# common sources: 'alsasrc'

self.launch\_config = rospy.get\_param('~source')

else:

self.launch\_config = 'gconfaudiosrc'

rospy.loginfo("麦克风配置: %s", self.launch\_config) # "Launch config: %s",self.launch\_config

self.launch\_config += " ! audioconvert ! audioresample " \

+ '! vader name=vad auto-threshold=true ' \

+ '! pocketsphinx name=asr ! fakesink'

# 配置ros系统设置

self.started = False

rospy.on\_shutdown(self.shutdown)# 自主关闭

self.pub = rospy.Publisher('~output', String)# 发布 ~output 参数指定的 话题 类型 tring 似乎缺少 指定发布队列大小 tring

rospy.Service("~start", Empty, self.start) # 开始服务

rospy.Service("~stop", Empty, self.stop) # 结束服务

# 检查模型和字典配置

if rospy.has\_param(self.\_lm\_param) and rospy.has\_param(self.\_dic\_param):

self.start\_recognizer()

else:

rospy.logwarn("启动语音识别器必须指定语言模型lm,以及语言字典dic.")

# rospy.logwarn("lm and dic parameters need to be set to start recognizer.")

def start\_recognizer(self):

rospy.loginfo("开始语音识别... ")

# rospy.loginfo("Starting recognizer... ")

self.pipeline = gst.parse\_launch(self.launch\_config)# 解析 麦克风配置

self.asr = self.pipeline.get\_by\_name('asr') # 自动语音识别 模型

self.asr.connect('partial\_result', self.asr\_partial\_result)# 后面的函数

self.asr.connect('result', self.asr\_result)

#self.asr.set\_property('configured', True) # 需要开启配置 hmm模型

self.asr.set\_property('dsratio', 1)

# 配置语言模型

if rospy.has\_param(self.\_lm\_param):

lm = rospy.get\_param(self.\_lm\_param)

else:

rospy.logerr('请配置一个语言模型 lm.')

return

if rospy.has\_param(self.\_dic\_param):

dic = rospy.get\_param(self.\_dic\_param)

else:

rospy.logerr('请配置一个语言字典 dic.')

return

if rospy.has\_param(self.\_hmm\_param):

hmm = rospy.get\_param(self.\_hmm\_param)

else:

rospy.logerr('请配置一个语言识别模型 hmm.')

return

self.asr.set\_property('lm', lm) # 设置asr的语言模型

self.asr.set\_property('dict', dic)# 设置asr的字典

self.asr.set\_property('hmm', hmm) # 设置asr的识别模型

self.bus = self.pipeline.get\_bus()

self.bus.add\_signal\_watch()

self.bus\_id = self.bus.connect('message::application', self.application\_message)

self.pipeline.set\_state(gst.STATE\_PLAYING)

self.started = True

# 解析 麦克风名称 得到 麦克风ID

def pulse\_index\_from\_name(self, name):

output = commands.getstatusoutput("pacmd list-sources | grep -B 1 'name: <" + name + ">' | grep -o -P '(?<=index: )[0-9]\*'")

if len(output) == 2:

return output[1]

else:

raise Exception("Error. pulse index doesn't exist for name: " + name)

# 停止识别器

def stop\_recognizer(self):

if self.started:

self.pipeline.set\_state(gst.STATE\_NULL)

self.pipeline.remove(self.asr)

self.bus.disconnect(self.bus\_id)

self.started = False

# 程序关闭

def shutdown(self):

""" 删除所有的参数，以防影响下次启动"""

for param in [self.\_device\_name\_param, self.\_lm\_param, self.\_dic\_param]:

if rospy.has\_param(param):

rospy.delete\_param(param)

""" 关闭 GTK 进程. """

gtk.main\_quit()

# 开始

def start(self, req):

self.start\_recognizer()

rospy.loginfo("识别器启动")

return EmptyResponse()

# 停止

def stop(self, req):

self.stop\_recognizer()

rospy.loginfo("识别器停止")

return EmptyResponse()

def asr\_partial\_result(self, asr, text, uttid):

"""前线部分结果到主线程. """

struct = gst.Structure('partial\_result')

struct.set\_value('hyp', text)

struct.set\_value('uttid', uttid)

asr.post\_message(gst.message\_new\_application(asr, struct))

def asr\_result(self, asr, text, uttid):

""" 前线结果到主线程 """

struct = gst.Structure('result')

struct.set\_value('hyp', text)

struct.set\_value('uttid', uttid)

asr.post\_message(gst.message\_new\_application(asr, struct))

def application\_message(self, bus, msg):

""" 从总线上接收应用数据. """

msgtype = msg.structure.get\_name()

if msgtype == 'partial\_result':

self.partial\_result(msg.structure['hyp'], msg.structure['uttid'])

if msgtype == 'result':

self.final\_result(msg.structure['hyp'], msg.structure['uttid'])

# 部分结果

def partial\_result(self, hyp, uttid):

""" Delete any previous selection, insert text and select it. """

rospy.logdebug("Partial: " + hyp)

# 最终结果

def final\_result(self, hyp, uttid):

""" Insert the final result. """

msg = String()# 话题消息类型

msg.data = str(hyp)# 识别语音对于成的文字

rospy.loginfo(msg.data)

self.pub.publish(msg)

if \_\_name\_\_ == "\_\_main\_\_":

start = recognizer()

gtk.main()