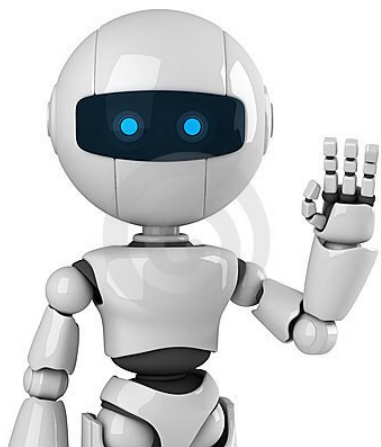


基于ROS的机器人跟随



东北大学 机器人科学与工程学院

张云洲

2018年11月

内容提纲

1. 机器人跟随的概念
2. TurtleBot简介
3. tf坐标变换
4. 基于人体骨架的行人跟随
5. 基于HOG+KCF的行人跟随
6. 实验演示

机器人跟随的概念

- 在跟随机器人的研究中，不同种类的移动机器人配备了不同类型的传感器。一般来说，主要有如下几种传感器方案来实现目标人的检测和跟踪：



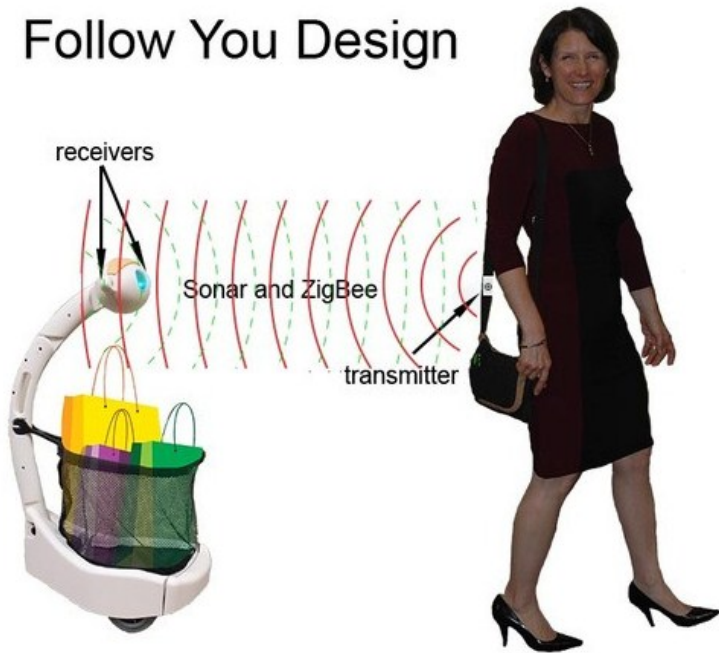
机器人跟随的概念



美国Boston Dynamics研发的BigDog机器人

机器人跟随的概念

Follow You Design



智能搬运机器人Budgee

机器人跟随的概念



Shadow Caddy球童机器人

机器人跟随的概念



RobovieII 超市购物辅助机器人

TurtleBot简介

TurtleBot

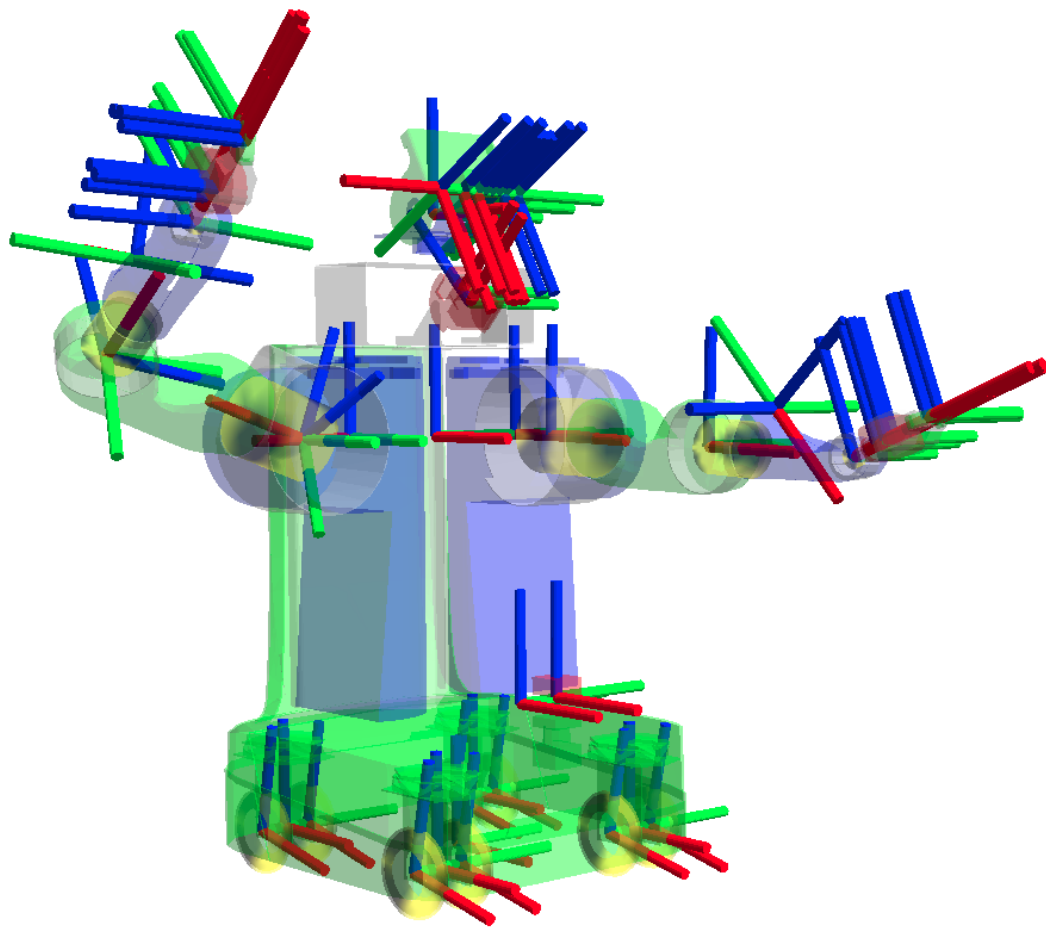
- 由Willow Garage公司推出的一款用于机器人平台。
- 包括Yujin Kobuki移动底座结构和Kinect传感器。
- 左右两轮为主动轮，驱动TurtleBot进行平移和旋转；前后两轮为从动轮，用于保持机身的平衡。
- 配备有里程计和一个单轴陀螺仪，前端还有一个碰撞传感器。



tf坐标变换

Tf: 让用户随时跟踪多个参考系的功能包。

- 使用树型数据结构，根据时间缓冲并维护多个参考系之间的坐标变换关系；
- 可帮助用户在任意时间将点、向量等数据的坐标，在两个参考系中完成坐标变换。



tf坐标变换

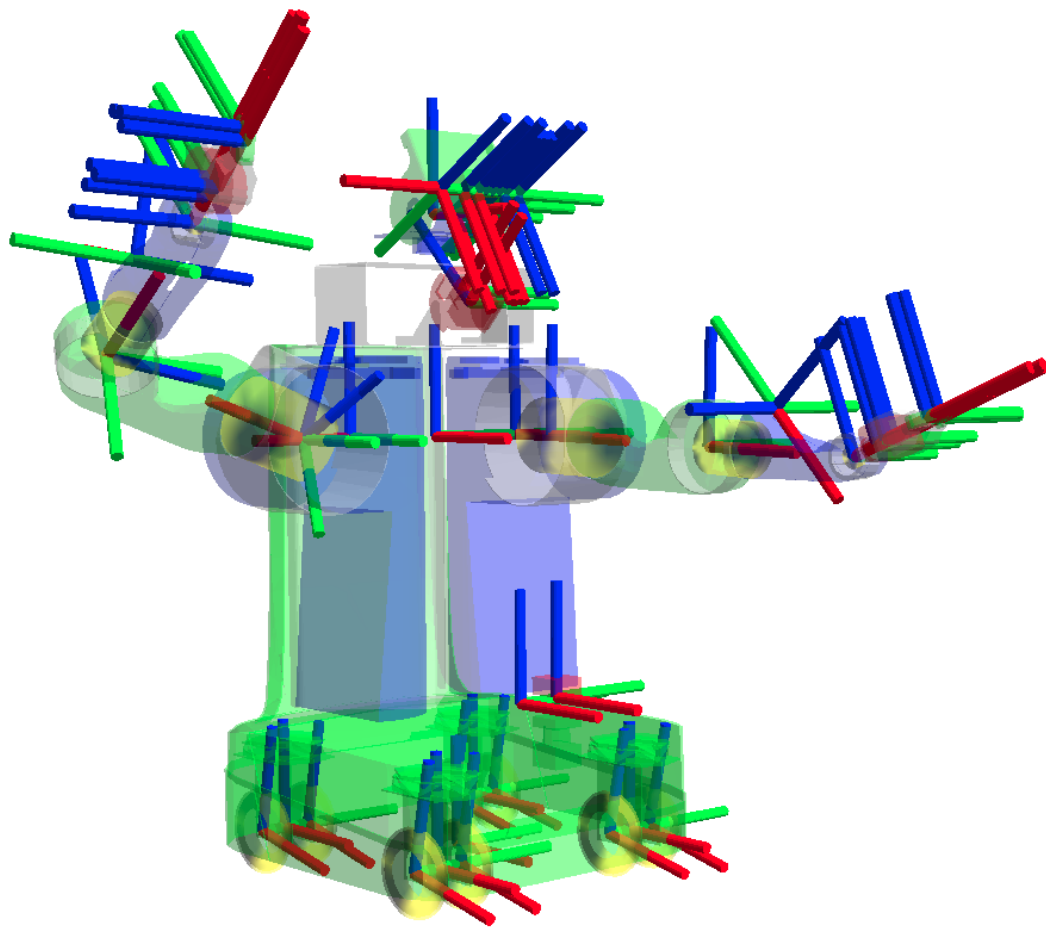
- 使用tf功能包的两个步骤:

(1) 监听tf变换

接收并缓存系统中发布的所有参考系变换，并从中查询所需要的参考系变换。

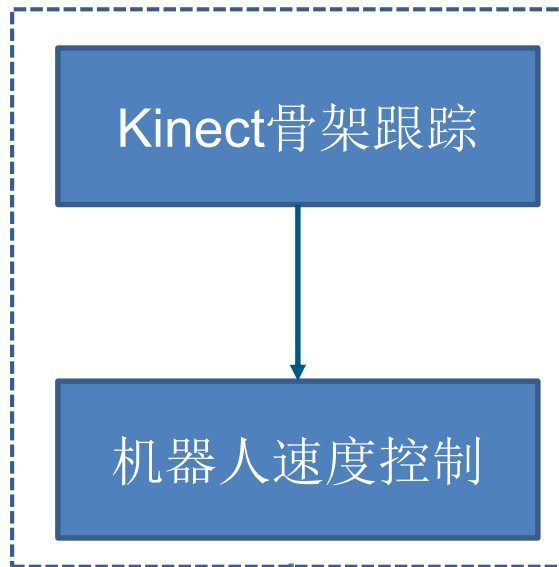
(2) 广播tf变换

向系统中广播参考系之间的坐标变换关系。系统中可能会存在多个对于机器人不同部位的tf变换广播。

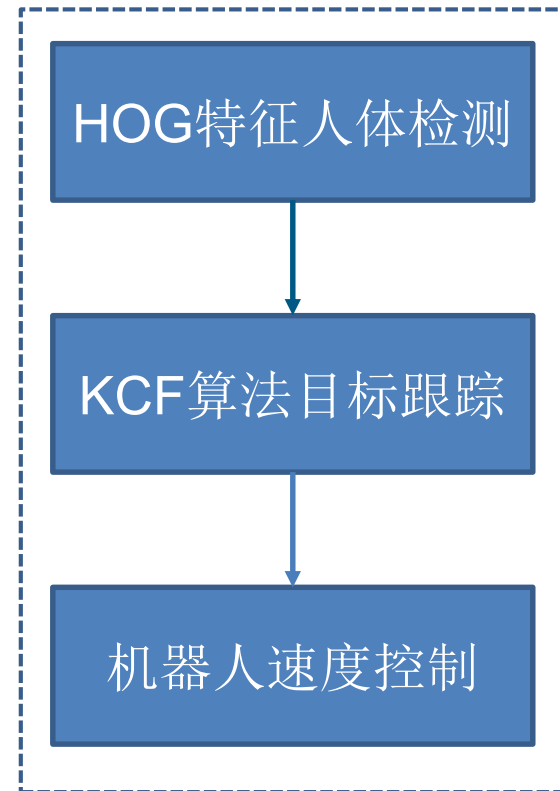


机器人跟随的实现方案

方案一



方案二



一种 3D 体感摄像机，拥有语音识别、人脸识别、动态捕捉、动作识别等功能。Kinect 自左向右分别是红外线投影机、RGB 摄像头、红外深度摄像头，内部包含 4 个麦克风阵列，可以过滤背景噪声以及定位声源；底部仰角马达装置通过控制 Kinect 上下倾斜的角度获取最佳视角。



Kinect V1

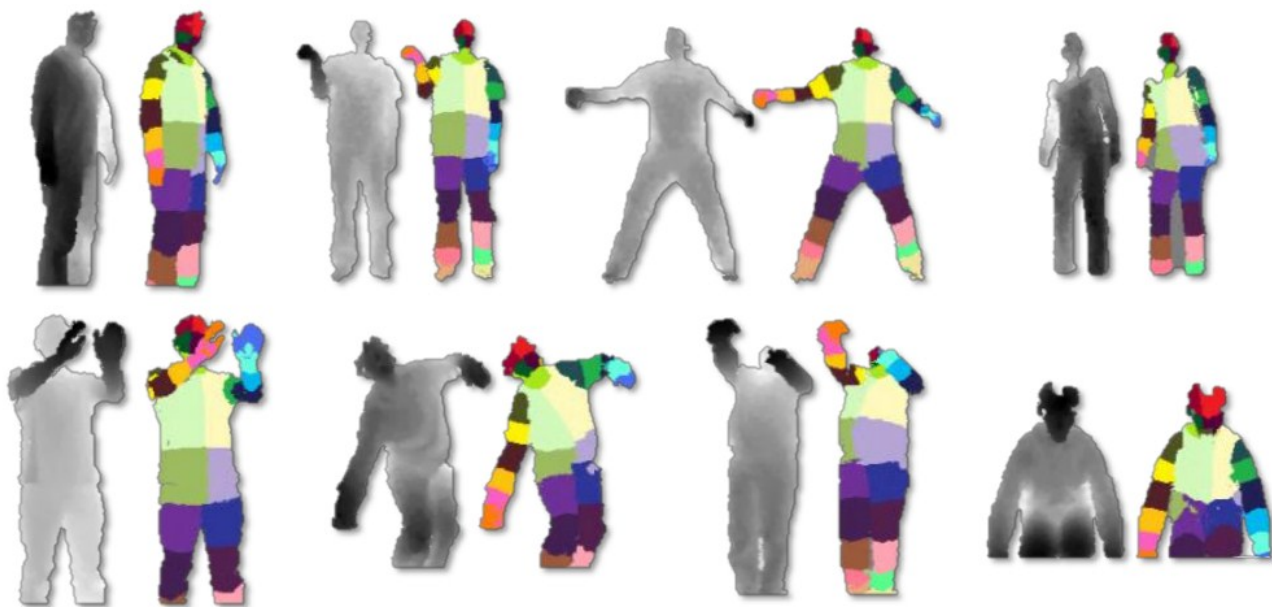
动静分离，识别人体

- 分析接近Kinect的区域，这也是最有可能是“人体”的目标。
- 逐点扫描这些区域深度图像的像素，判断属于人体的哪些部位。这一过程为计算机视觉技术，包括边缘检测、噪声阈值处理、对人体目标特征点的“分类”等环节。
- 将人体从背景环境中区分出来。



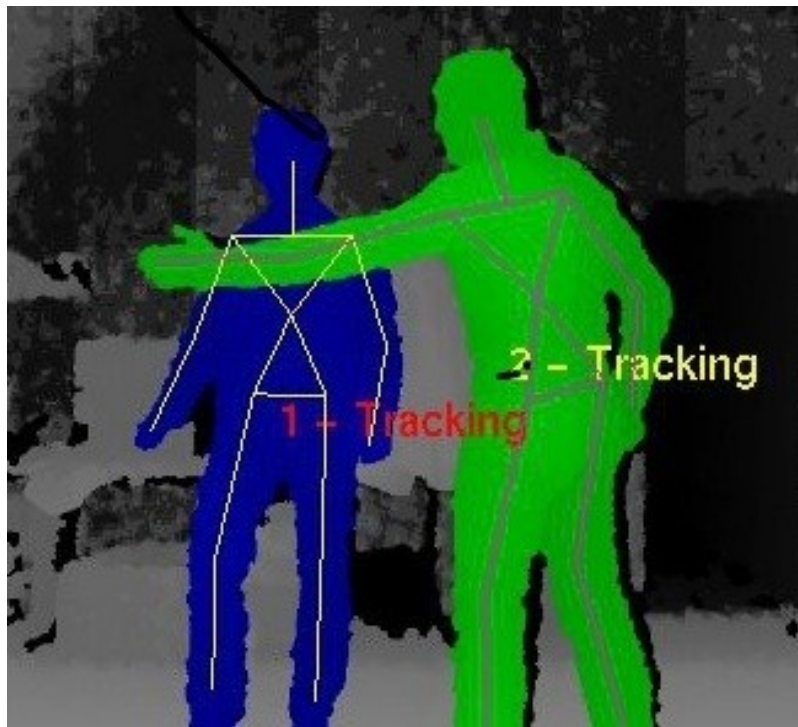
人体部位分类

这一环节的主要工作是从深度图像中将人体各个部位识别出来，比如头部、躯干、四肢、手臂、腿等大块关联的肢体。



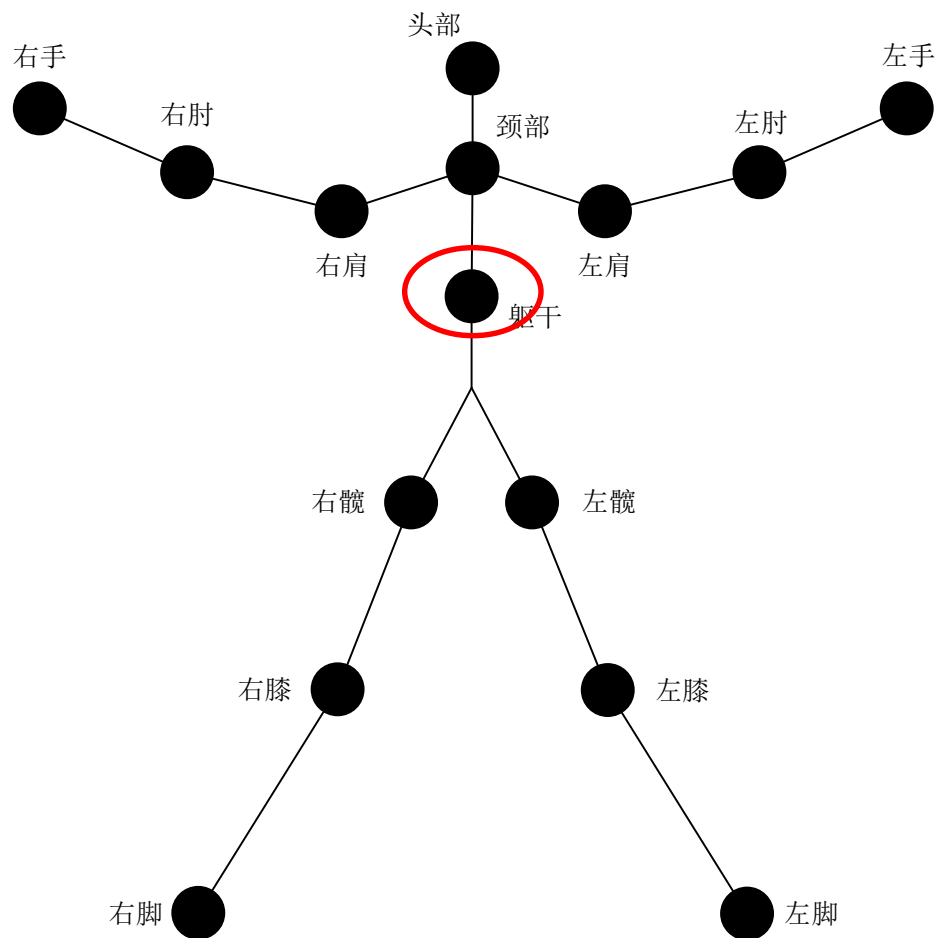
从人体部位识别关节

- 主要基于机器学习进行分类。
- 考虑到人体部位的重叠，分别从多角度进行分析，根据每一个可能的像素来确定关节点。
- 系统根据检测到的关节点生成一幅骨架。



➤ 关节点检测

OpenNI可以检测到15个人体关节点。找到这些关节点后，就完成了对用户的骨骼跟踪，之后就可以提取关节点的相对位置信息来做运动分析。



➤ 基于人体骨架的相对位置信息检测

OpenNI中的人体骨架追踪程序可以获得人体关节的旋转矩阵和平移矩阵，利用平移矩阵的x, y坐标即可得到目标人的位置信息。

```
my@my-ubuntu: ~  
y: 0.373545835861  
z: 0.323212787351  
w: 0.374219496838  
---  
transforms:  
-  
  header:  
    seq: 0  
    stamp:  
      secs: 1495586086  
      nsecs: 652855207  
    frame_id: openni_depth_frame  
    child_frame_id: torso_1  
  transform:  
    translation:  
      x: 0.713596582462  
      y: -0.0493160281057  
      z: -0.0757875485546  
    rotation:  
      x: 0.523761661844  
      y: 0.141869271971  
      z: 0.477094206667  
      w: 0.691323331892  
---
```

1. 安装TurtleBot驱动
2. 安装Kinect驱动
3. 下载OpenNI Tracker
4. 获取人体关节位置信息
5. 控制TurtleBot运动状态

安装TurtleBot驱动

- 参考网页:

<http://wiki.ros.org/TurtleBot/Tutorials/indigo/TurtleBot%20Installation>

- 建议采用Debs Installation
- 如果TurtleBot没有反应, 参考:
- <http://wiki.ros.org/TurtleBot/Tutorials/indigo/Kobuki%20Base>

安装Kinect驱动

- 安装OpenNI
- 安装SensorKinect
- 安装NITE

- 方法参考:

<http://blog.csdn.net/u013453604/article/details/48013959>

- 如果提示No Device Connected, Waiting for Device to be Connected, 可能缺少SensorKinect

安装Kinect驱动

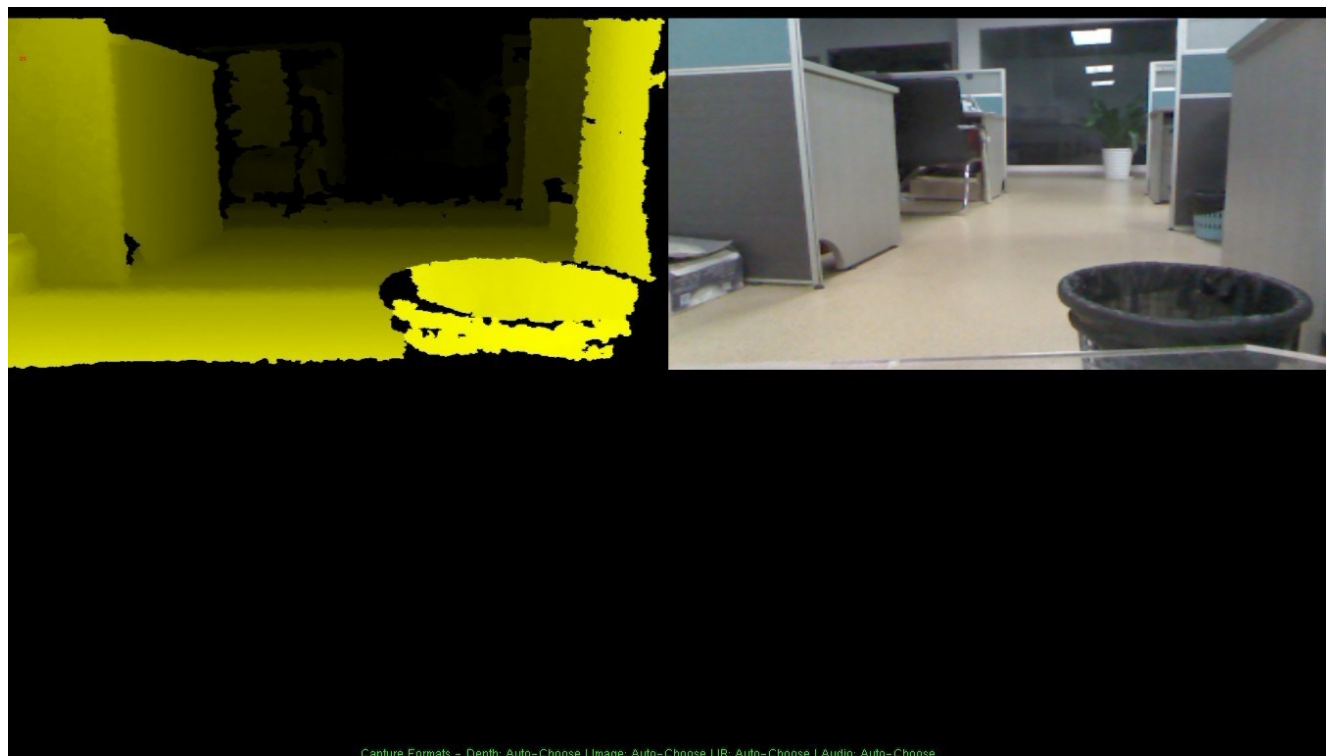
- 需要注意的地方：
- If you're using Virtual Machine launching Ubuntu on it, I don't recommend you continue. I failed too many times in trying to do it on VMs. And finally I succeeded in real machine.

--http://wiki.ros.org/openni_camera

- we now have OpenNI and NITE 2.0, and PrimeSense have been bought by Apple. This tutorial does not work with versions 2 (though 1.5 works just fine), and there is talk of Apple stopping public access to NITE.

--<http://www.20papercups.net/programming/kinect-on-ubuntu-with-openni/>

运行示例程序



```
cd ~/OpenNI-Bin-Dev-Linux-x64-v1.5.7.10/Samples/Bin/x64-Release  
./NiViewer
```

运行示例程序

```
my@my-ubuntu: ~/OpenNI-Bin-Dev-Linux-x64-v1.5.7.10/Samples/Bin/x64-Debug
user 2: head at (-1225.57,238.41,2747.29)
user 1: head at (-148.96,134.92,740.21)
user 2: head at (-1225.52,238.64,2747.23)
user 1: head at (-149.69,132.59,742.40)
user 2: head at (-1225.30,238.50,2746.65)
user 1: head at (-134.08, 69.82,710.29)
user 2: head at (-1225.17,238.47,2746.59)
user 1: head at (-133.99, 68.49,710.32)
user 2: head at (-1224.92,238.97,2746.66)
user 1: head at (-134.70, 66.59,709.60)
user 2: head at (-1224.99,238.83,2746.79)
user 1: head at (-133.59, 65.56,707.33)
user 2: head at (-1225.20,238.58,2746.90)
user 1: head at (-132.29, 63.70,704.93)
user 2: head at (-1225.25,238.39,2747.17)
user 1: head at (-131.90, 62.32,703.54)
user 2: head at (-1225.10,238.68,2747.07)
user 1: head at (-132.17, 62.13,703.30)
user 2: head at (-1224.92,238.92,2746.66)
user 1: head at (-132.17, 63.64,705.97)
user 2: head at (-1224.65,239.00,2746.39)
user 1: head at (-132.05, 64.90,706.70)
user 2: head at (-1224.33,239.08,2746.01)
```

```
cd ~/OpenNI-Bin-Dev-Linux-x64-v1.5.7.10/Samples/Bin/x64-Release
./Sample-NiSimpleSkeleton
```


尽管运行
NiSimpleSkeleton
可以获得人体某
一部位的坐标，
但是没有发布到
/tf中。

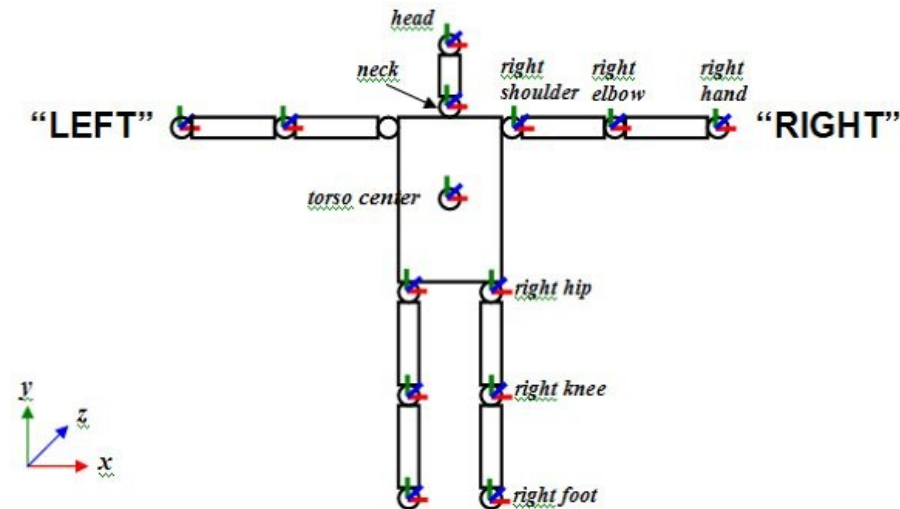
```
my@my-ubuntu: ~  
my@my-ubuntu:~$ rostopic echo /tf  
WARNING: topic [/tf] does not appear to be published yet  
^Cmy@my-ubuntu:~$ rostopic list  
/rosout  
/rosout_agg  
my@my-ubuntu:~$
```

- The OpenNI tracker broadcasts the OpenNI skeleton frames using tf.
 - 下载地址: https://github.com/ros-drivers/opensni_tracker
- 运行方法:
- `roslaunch opensni_tracker opensni_tracker`
 - make sure your Kinect is powered. For example, on TurtleBot, `kinect.launch` needs to be run successfully.

OpenNI Tracker

- The user's pose will be published as a set of transforms (/tf) with the following frame names.

/head	/right_elbow
/neck	/right_hand
/torso	/left_hip
/left_shoulder	/left_knee
/left_elbow	/left_foot
/left_hand	/right_hip
/right_shoulder	/right_knee
	/right_foot



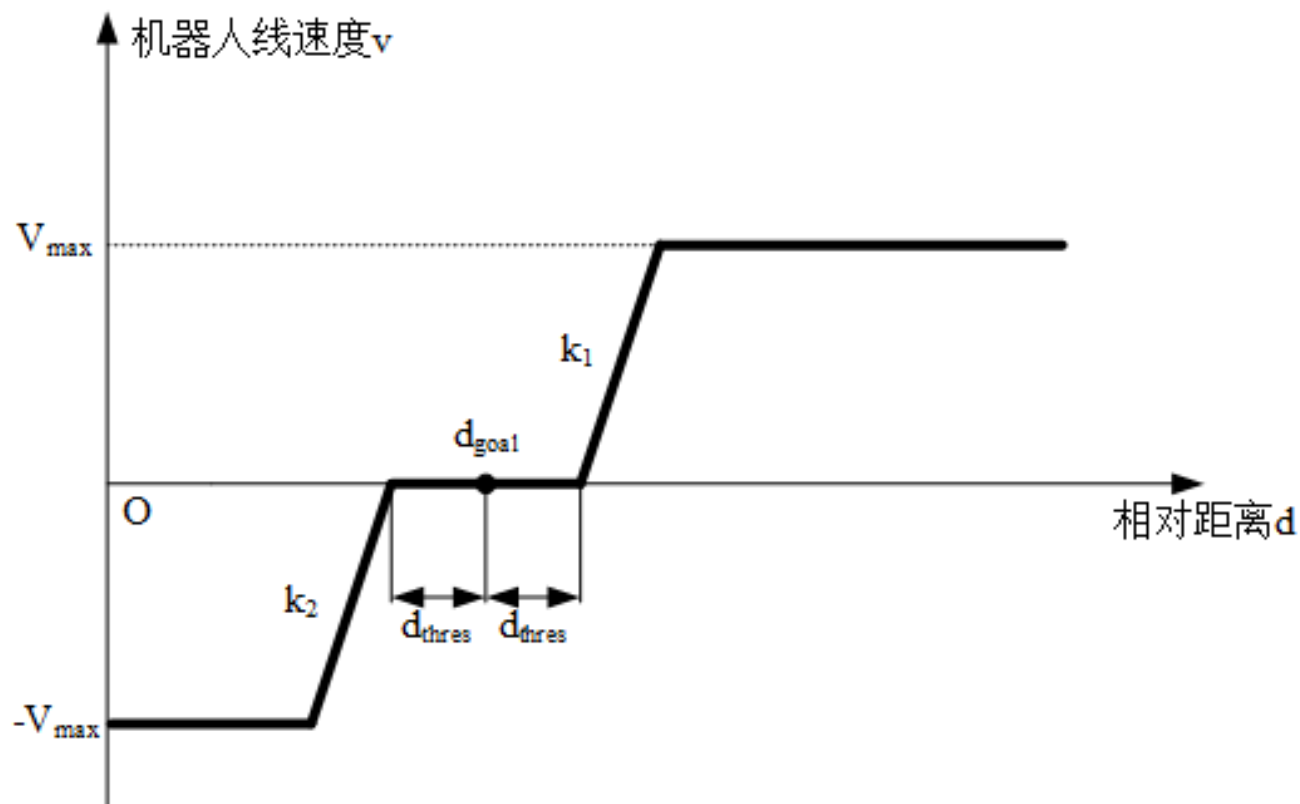
查看/tf下的关节坐标

```
my@my-ubuntu: ~  
y: 0.542531472457  
z: 0.619441282733  
w: 0.387121972884  
---  
transforms:  
-  
  header:  
    seq: 0  
    stamp:  
      secs: 1509015547  
      nsecs: 239743673  
    frame_id: openni_depth_frame  
  child_frame_id: torso_1  
  transform:  
    translation:  
      x: 0.708293280587  
      y: -0.19816114036  
      z: -0.113268760762  
    rotation:  
      x: 0.302714741424  
      y: 0.493269119798  
      z: 0.644753935993  
      w: 0.499341288899  
---
```

rostopic echo /tf

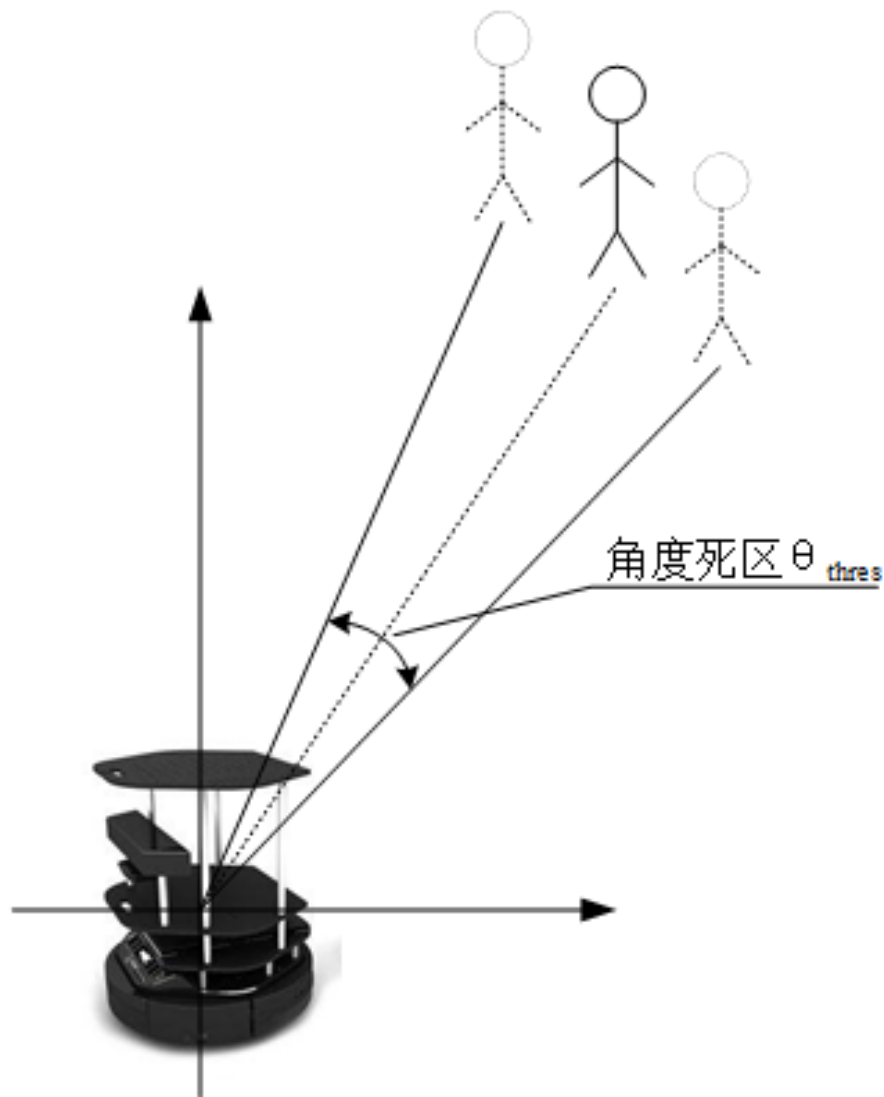
控制TurtleBot运动

- 线速度控制



控制TurtleBot运动

- 角速度控制



1. TurtleBot驱动:

```
roslaunch turtlebot_bringup minimal.launch
```

2. Kinect驱动:

```
roslaunch openni_launch openni.launch
```

3. 启动OpenNI Tracker:

```
roslaunch openni_tracker openni_tracker
```

4. 启动human follower:

```
roslaunch human_follower human_follower
```

➤ 基于HOG特征的人体检测

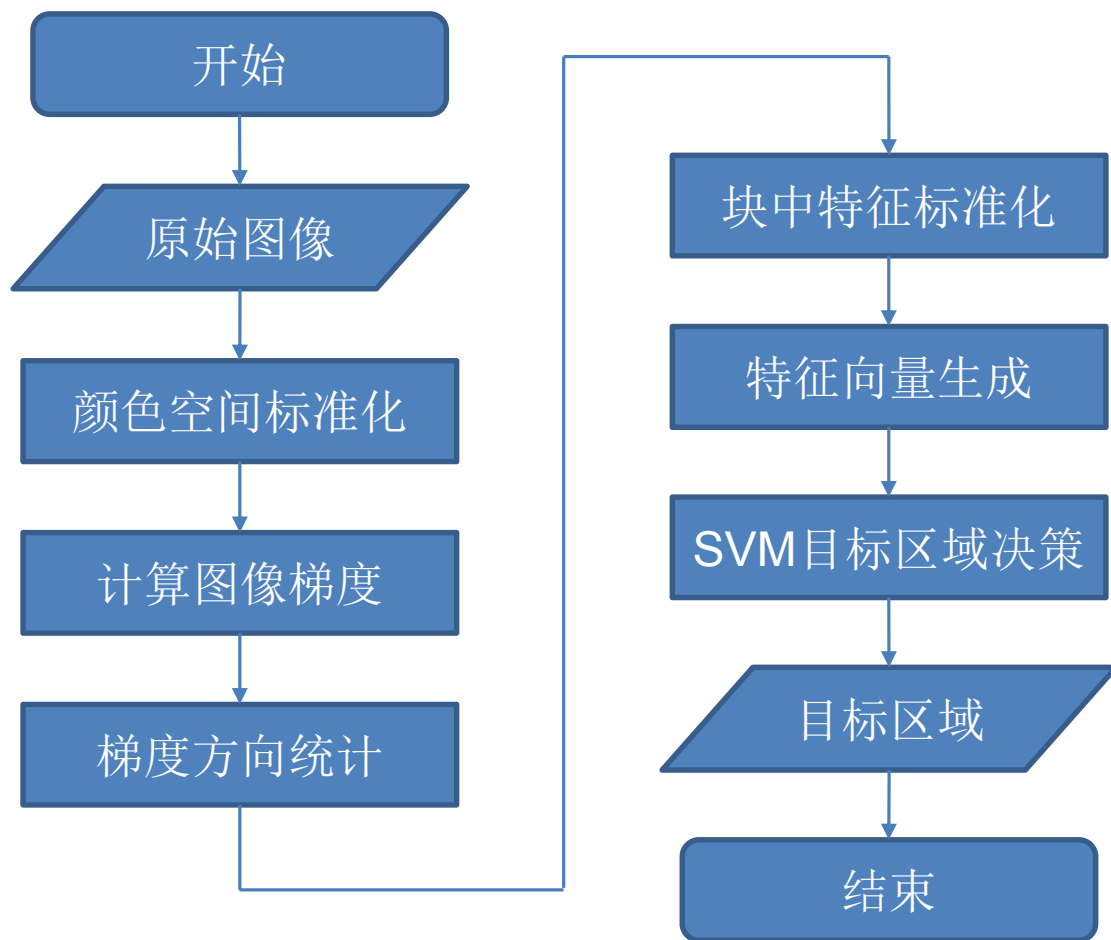
- 作为当前的研究热点，基于HOG特征在目标检测研究中发挥着重要作用：

1. Tian S, Bhattacharya U, Lu S, et al. Multilingual scene character recognition with co-occurrence of **histogram of oriented gradients**[J]. Pattern Recognition, **2016**, 51: 125-134.

2. Wójcikowski M. **Histogram of Oriented Gradients** with Cell Average Brightness for **Human Detection**[J]. Metrology and Measurement Systems, **2016**, 23(1): 27-36.

- HOG是在图像的局部方格单元上操作，对图像中人体的微小姿态变化和光照变化具有一定的抵抗能力。

➤基于HOG特征的人体检测流程图



- 基于KCF算法的目标跟踪
- KCF (Kernelized Correlation Filter) 算法使用了傅里叶变换，避免矩阵求逆操作，保证了算法的实时性。

$$\min_{\mathbf{w}} \sum_i (f(x_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y} \longrightarrow \mathbf{w} = \mathcal{F}^{-1} \left(\frac{\mathcal{F}(X^T \mathbf{y})}{\mathcal{F}(X^T X + \lambda I)} \right)$$

- 使用循环矩阵增加了训练样本，提高了准确率。



+30

+15

基准样本

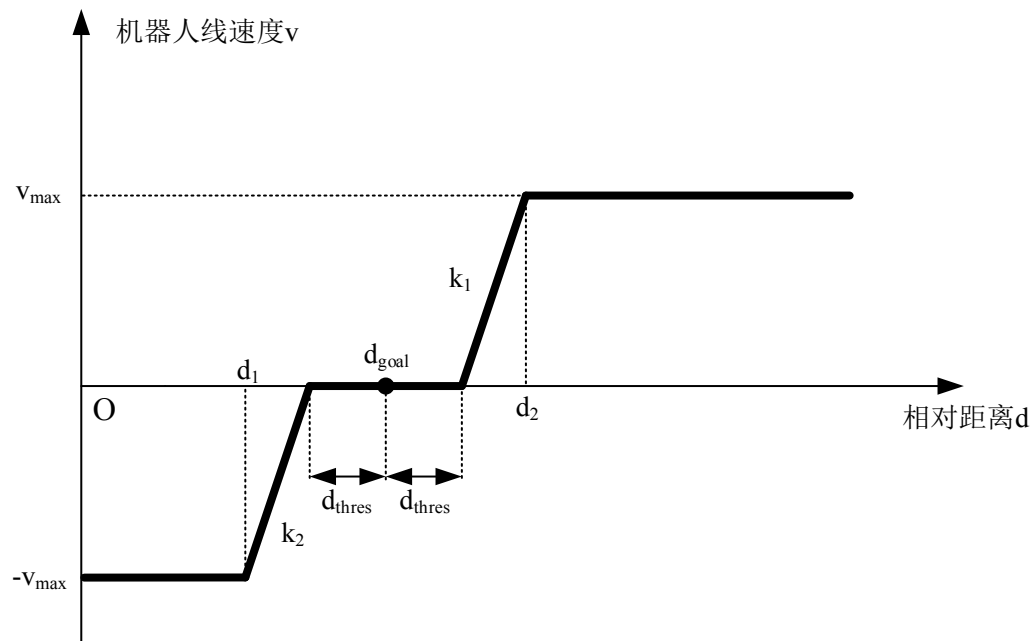
-15

-30

- 相对位置信息检测
- 距离检测
 - 在感兴趣区域中均匀选取若干个点，将这几个点的深度值求和之后取平均值，作为机器人与目标人的距离。
- 角度检测
 - 求出感兴趣区域的中心点横坐标，再求出整幅图像中心点的横坐标，将这两个横坐标的差值作为角度的度量。



➤ 线速度控制策略



$$v = \begin{cases} -v_{\max} & 0 < d \leq d_1 \\ k_2 d - k_2 (d_{\text{goal}} - d_{\text{thres}}) & d_1 < d \leq (d_{\text{goal}} - d_{\text{thres}}) \\ 0 & (d_{\text{goal}} - d_{\text{thres}}) < d \leq (d_{\text{goal}} + d_{\text{thres}}) \\ k_1 d - k_1 (d_{\text{goal}} + d_{\text{thres}}) & (d_{\text{goal}} + d_{\text{thres}}) < d \leq d_2 \\ v_{\max} & d_2 < d \end{cases}$$

➤ 角速度控制策略

$$\omega = \begin{cases} 0 & \theta \in \theta_{thres} \\ -\omega_0 & \theta > 0 \cap \\ \omega_0 & \theta < 0 \cap \end{cases}$$

