

KinectV1 物体识别与机器臂抓取

一 Kinect 常用的三种开发方式

(1) kinect Sensor-->OpenNi-->NITE

openni: OpenNI 的主要目的是要形成一个标准的 API，来搭建视觉和音频传感器与视觉和音频感知中间件通信的桥梁。

kinect Sensor: kinect 的驱动

NITE: The PrimeSense NiTE™ is the most advance and robust 3D computer vision middleware available today. This middleware provides the application with a clear user control API, whether it is hand-based control or a full-body control. The algorithms utilize the depth, color, IR and audio information received from the hardware device, which enable them to perform functions such as hand locating and tracking; a scene analyzer (separation of users from background); accurate user skeleton joint tracking; various gestures recognition; and more.

(2) libfreenect

MAC 下使用 libfreenect2 测试 KinectV2 成功。

(3) Windows 下有官方 SDK 开发包

二 kinect Sensor-->OpenNi-->NITE 安装

(1) 安装 openni:

32 位: Linux-32/OpenNI-Linux-x86-2.2.tar.bz2

64 位: Linux-64/OpenNI-Bin-Dev-Linux-x64-v1.5.7.10.tar.zip

\$ sudo ./install.sh

(2) 安装 Sensor:

32 位: 解压 Linux-32/SensorKinect-master.zip, 进入 SensorKinect-master/Bin/Sensor-Bin-Linux-x86-v5.1.0.25

64 位: Linux-64/SensorKinect093-Bin-Linux-x64-v5.1.2.1.tar.bz2

\$ sudo ./install.sh

(3) 安装 NITE:

32 位: Linux-32/OpenNI-Linux-x86-2.2.tar.bz2

64 位: Linux-64/NITE-Bin-Linux-x64-v1.5.2.23.tar.zip

\$ sudo ./install.sh

(4) 卸载上面的库:

\$ sudo ./install.sh -u

NITE 可以使用: \$ sudo ./uninstall.sh

三 Install ork:

\$ ros-indigo-object-recognition-*

\$ sudo apt-get install ros-indigo-openni-*

\$ sudo apt-get install ros-indigo-freenect-*

四 Install Couch Db

```
$ sudo apt-get install couchdb
```

输入: `$ curl -X GET http://localhost:5984` 可查看信息。

五 Install Web UI

```
$ sudo pip install -U couchapp
```

若不能使用上面的指令需要安装 pip:

```
$ sudo apt-get install python-pip python-dev build-essential
```

```
$ rosrund object_recognition_core push.sh
```

点击: http://localhost:5984/or_web_ui/design/viewer/index.html 可查看模型到信息, 目前为空。

六 获取模型 Capture

预览:

```
$ rosrund object_recognition_capture capture -n 50 --seg_z_min 0.007 -o silk.bag --preview
```

```
$ rosrund object_recognition_capture capture -n 50 --seg_z_min 0.007 -o silk.bag
```

注: `--preview` 表示预览, 通过窗口观察待建模的物体是否在 kinect 识别范围; 需要将待建模物体放在校准图案中间的圆圈中。

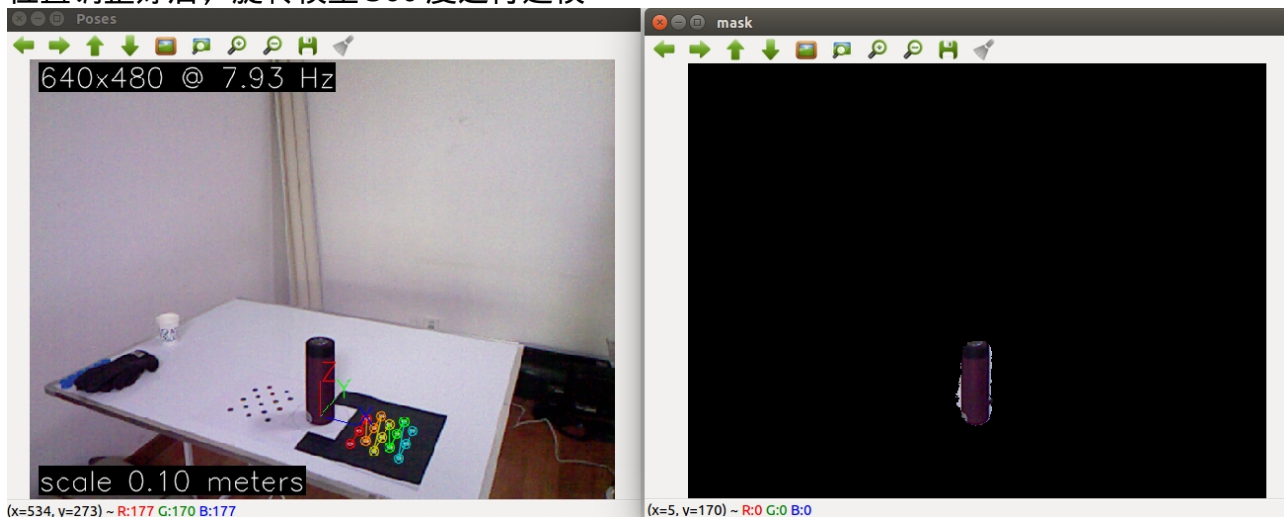
`-n 50` 表示获取多少张图片, 可自己修改, 一般在 40-50 之间, 少了模型建不完整, 多了有重复;

`--seg_z_min 0.007` 表示在平面裁剪的量, 以米为单位;

`-o silk.bag` 表示输出文件名称, 文件放在当前路径;

了解更多参数请 `-h`。

位置调整后, 旋转模型 360 度进行建模



七 upload 模型到数据库

```
$ rosrund object_recognition_capture upload -i silk.bag -n 'bottle' bottle --commit
```

注: `-i silk.bag` 表示输入文件名称, 路径为当前路径;



-n 'bottle' bottle 表示上传模型的名字;
 --commit 表示确认该条指令, 没--commit 的话指令无效;
 了解更多参数请 -h。

八 建模 train

训练一个:需要指明 ID 号

```
$ rosrn object_recognition_reconstruction mesh_object -s ID --visualize --commit
```

ID 在上传后会给出, 也可在 http://localhost:5984/or_web_ui/design/viewer/index.html 中的 object_listing 中查看。

Object Name	Description	Added	ID	Image
bottle		2017-07-07T19:50:41Z	e682109ebe032b6019b5ff0440000bcc	
whh		2017-07-07T20:26:24Z	e682109ebe032b6019b5ff044001fdd1	

训练所有:

```
$ rosrn object_recognition_reconstruction mesh_object --all --visualize --commit
```

九 获取 kinect 转动角度:

因为在 linux 上 kinect 使用的是第三方库 openni, 没有相关获取 kinect 转动角度到相关功能, 所以要在 windows 上装微软的官方 sdk, 先装 KinectSDK 再装 KinectDeveloperToolkit。
 安装完 Sensor Settings 里就可查看 kinect 的转动角度。也可大概估计。

十 修改程序:

在 kinect 可以识别物体后, 在 kinectv1_ros_ws/src/ork/src/ork_transform.cpp 下修改程序:

```
while(ros::ok())
{
    ros::spinOnce();
    for(int i=0;i<object_send.size();++i)
    {
        if(object_send[i].id=="b21783b965e257fbbd0ecd7c6a00036f")
        {
            /*send_info.data[2]=object_send[i].x;
            send_info.data[3]=object_send[i].y;
            send_info.data[4]=object_send[i].z;
            chatter_pub_1.publish(send_info);
            ROS_INFO("bottle \n");*/
            try
            {
                tf::Transform transform;
                transform.setOrigin(tf::Vector3(object_send[i].x,object_send[i].y,object_send[i].z));
                tf::Quaternion q;
                q.setRPY(0, 0, 0);
                transform.setRotation(q);
                br1.sendTransform(tf::StampedTransform(transform, ros::Time::now(), "camera_rgb_optical_frame", "bottle"));
            }
        }
    }
}
```

修改自己建的模型 id, 若抓取物体的位置有偏差, 可在 `transform.setOrigin(tf::Vector3(object_send[i].x,object_send[i].y,object_send[i].z))` 中添加物体 x, y, z 的补偿。也可在接收物体位姿的程序中进行位置补偿(我采用此方式)。若建了新的物体模型可在程序 for 循环中添加 if 条件语句。

`camera_base_tf_publish.cpp` 是为了用来发布 `base_link->camera_link` 的坐标系关系, 以后 kinect 的位置固定了可以将 `base_link->camera_link` 的坐标系关系写入 urdf 文件中。

十一 验证结果:

(1)位置关系:如下图



(2)修改程序:

2.1 修改 kinect 相对于 base_link 的位姿 ork/src/camera_base_tf_publish.cpp

```
transform.setOrigin( tf::Vector3(0.55, -0.70, 0.80) );  
tf::Quaternion q;  
q.setRPY(0, 3.14/8, 3.14/2);  
transform.setRotation(q);  
br.sendTransform(tf::StampedTransform(transform, ros::Time::now(), "base_link", "camera_link"));  
rate.sleep();|
```

位置直接用尺子测量,姿态想象 kinect 的 xyz 轴的正向开始与 base_link 的 -y, -z, x 对应,则到上图所示的 kinect 姿态可通过基于 base_link 坐标系下旋转 z 轴 90 度,再旋转 x 轴 22.5 度变换而得到。

(3)启动步骤:

我使用两台电脑进行测试,一台(arm-computer)运行机器臂控制程序与机器臂通过 usb-can 连接,另一台(kinect-computer)运行 kinect 识别物体程序与 kinect 相连。

Kinect-computer:

1.roslaunch ork ork_demo.launch //加载 kinect1 的参数,运行 object_recognition 程序;

ork_transform(订阅物体识别程序发布的 recognized_object_array 话题,根据物体 ID 发布相应的 TF 变换);camera_base_tf_publish(发布 base_to_camera)
当识别到以训练模型的物体时,则会显示其 id 和位姿等信息:

```
[ INFO] [1499761415.011127051]: bottle

[ INFO] [1499761422.939033659]: i: [0]
[ INFO] [1499761422.939105998]: frame id: [camera_rgb_optical_frame]
[ INFO] [1499761422.939135949]: key: [e682109ebe032b6019b5ff0440000bcc]
[ INFO] [1499761422.939170521]: position x: [0.010371]
[ INFO] [1499761422.939198140]: position y: [0.383392]
[ INFO] [1499761422.939226055]: position z: [1.020445]
[ INFO] [1499761422.939253396]: orientation x: [0.771939]
[ INFO] [1499761422.939280660]: orientation y: [-0.003374]
[ INFO] [1499761422.939308146]: orientation z: [-0.004097]
[ INFO] [1499761422.939336201]: orientation w: [0.635674]

[ INFO] [1499761422.940542041]: bottle

[ INFO] [1499761430.500016976]: i: [0]
[ INFO] [1499761430.500225803]: frame id: [camera_rgb_optical_frame]
[ INFO] [1499761430.500605433]: key: [e682109ebe032b6019b5ff0440000bcc]
[ INFO] [1499761430.500856541]: position x: [0.011187]
[ INFO] [1499761430.501119513]: position y: [0.383424]
[ INFO] [1499761430.501366082]: position z: [1.020316]
[ INFO] [1499761430.501627762]: orientation x: [0.771851]
[ INFO] [1499761430.502000968]: orientation y: [-0.003328]
[ INFO] [1499761430.502251828]: orientation z: [-0.004040]
[ INFO] [1499761430.502499076]: orientation w: [0.635782]
```

2.如果需要可视化识别效果等:roslaunch ork view_ork.launch



arm-computer: <https://github.com/auboROS>

1.roslaunch mra_control mra7a_trajectory_rviz.launch //Server for controlling the mra7a.

2.rosrun mra7a_grasp_bottle_demo grasp_bottle_demo_node //Get the object pose from the TF tree and plan the mra7a to execute the grasp motion.