## CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Tutorials</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Tips &amp; FAQs</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Install Astra camera</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Install Softkinetic Camera</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>Install RealSense camera</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>Install Xtion2</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>Install Ensenso</td>
<td>29</td>
</tr>
<tr>
<td>9</td>
<td>Install Structure core Camera</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>Install ELP USB 4K camera</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>Install Insta360 Air</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>Install Chainer/PyTorch with GPU Support</td>
<td>37</td>
</tr>
<tr>
<td>13</td>
<td>Install PCL from source</td>
<td>41</td>
</tr>
<tr>
<td>14</td>
<td>Deep Learning with Your Own Image Dataset</td>
<td>43</td>
</tr>
<tr>
<td>15</td>
<td>Contributions</td>
<td>67</td>
</tr>
<tr>
<td>16</td>
<td>checkerboard_detector</td>
<td>69</td>
</tr>
<tr>
<td>17</td>
<td>imagesift</td>
<td>79</td>
</tr>
<tr>
<td>18</td>
<td>jsk_perception</td>
<td>83</td>
</tr>
<tr>
<td>19</td>
<td>jsk_pcl_ros</td>
<td>283</td>
</tr>
<tr>
<td>20</td>
<td>jsk_pcl_ros_utils</td>
<td>497</td>
</tr>
<tr>
<td>21</td>
<td>jsk_recognition_msgs</td>
<td>565</td>
</tr>
<tr>
<td>22</td>
<td>jsk_recognition_utils</td>
<td>571</td>
</tr>
</tbody>
</table>
jsk_recognition is a stack for the perception packages which are used in JSK lab.
The code is open source, and available on github.
1.1 Nodelet based designe

Programs in jsk_recognition is written as nodelet to overcome communication overhead and they will not be activated until output topics is subscribed by other nodes.

The nodelets in jsk_recognition inherit jsk_topic_tools::ConnectionBasedNodelet. The superclass provides functionality “Do not subscribe input topics if no output is required”.

1.2 Representation of subset of image and pointcloud

jsk_recognition uses mask image, point indices and so on to represent subset of image and pointcloud.

- **Mask Image** sensor_msgs/Image
  
  Mask image is a black and white image (encoded as MONO8) to represent subset of image.

- **Label Image** sensor_msgs/Image
  
  Label image is a image (encoded as TYPE_32FC1) to represent subsets of image.

- **ROI** sensor_msgs/CameraInfo
  
  ROI is an rectangular space of image. roi fields of sensor_msgs/CameraInfo is used to represent ROI.

- **Rect** geometry_msgs/PolygonStamped
  
  Rect is a 2-D polygonal region of image. geometry_msgs/PolygonStamped is originally designed for 3-D polygon, but in this case, it is used as 2-D polygon.

- **Point Indices** pcl_msgs/PointIndices
  
  Point indices represent subset of pointcloud.

- **Cluster Point Indices** jsk_recognition_msgs/ClusterPointIndices
  
  Cluster point indices is a set of point indices.
Tutorials for nodes/nodelets in jsk_recognition package.

### 2.1 Publish static image topic

#### 2.1.1 Introduction

You may need image topics to try recognition nodes/nodelets in jsk_perception, and this tutorial shows how to create image topic easily.

#### 2.1.2 Step by step

For this purpose, you can use `image_publisher.py`.

Terminal 1:

```
$ roscore
```

Terminal 2:

```
$ rosrun jsk_perception image_publisher.py
  _file_name:=$(rospack find jsk_perception)/sample/kiva_pod_image_color.jpg
  _rate:=30
```

```
_file_name:=... and _rate:=... is remapping of rosparam and you can get the value with:
```

```
$ rosparam get /image_publisher/file_name
/home/wkentaro/Projects/jsk/src/jsk-ros-pkg/jsk_recognition/jsk_perception/sample/
  →kiva_pod_image_color.jpg
```

```
$ rosparam get /image_publisher/rate
30
```

The image is published to topic `/image_publisher/output`, so you can see it by:

```
$ rosrun image_view image_view image:=/image_publisher/output
```

```
image:=... is remaping of topic, and the image_view will subscribe /image_publisher/output in this case.
```

The result is as shown below:
2.1.3 A single command

You can run upper programs in a single command with writing roslaunch file:

```
$ # emacs tutorial_image_publisher.launch
$ vim tutorial_image_publisher.launch
```

Or you can download the file from:

```
tutorial_image_publisher.launch
```

```xml
<launch>
  <node name="image_publisher"
    pkg="jsk_perception" type="image_publisher.py">
    <!--
    <param name="file_name" value="$(find jsk_perception)/sample/kiva_pod_image_color.jpg" type="str" />
    <param name="rate" value="30" type="int" />
    Below also works.
    -->
    <rosparam subst_value="true">
      file_name: $(find jsk_perception)/sample/kiva_pod_image_color.jpg
      rate: 30
    </rosparam>
  </node>
  <node name="image_view"
    pkg="image_view" type="image_view">
    <!--
    <remap from="image" to="/image_publisher/output" />
    Below works because image_publisher and image_view is in the same namespace.
    -->
  </node>
</launch>
```
You can launch the roslaunch file by:

```
$ roslaunch ./tutorial_image_publisher.launch
```

## 2.2 Find object with color filtering

### 2.2.1 Introduction

In this tutorial, we show a sample of finding object by color.

### 2.2.2 Step by step

First, you need image topic, so please run `tutorial_image_publisher.launch` in previous tutorial, and below `rostopic list` output is expected:

```
$ rostopic list
/image_publisher/output
/image_publisher/output/camera_info
/image_publisher/parameter_descriptions
/image_publisher/parameter_updates
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/rosout
/rosout_agg
```

For color filtering, you can use `hsv_color_filter`:

```
$ rosrun opencv_apps hsv_color_filter image:=/image_publisher/output __use_camera_info:=false __name:=hsv_color_filter
$ rosrun image_view image_view image:=/hsv_color_filter/image
```

You can reconfigure rosparam using `rqt_reconfigure`.

```
rosrun rqt_reconfigure rqt_reconfigure
```

You can set parameters as below:
The result is like below:
2.2.3 A single command

You can run upper programs in a single command with writing file:

```
$ vim tutorial_color_filtering.launch
```

Or you can download the file from:

```
tutorial_color_filtering.launch
```

```xml
<launch>
  <node name="hsv_color_filter"
    pkg="opencv_apps" type="hsv_color_filter">
    <remap from="image" to="image_publisher/output" />
    <rosparam>
      use_camera_info: false
      h_limit_max: 360
      h_limit_min: 340
      s_limit_max: 220
      s_limit_min: 170
      v_limit_max: 130
      v_limit_min: 80
    </rosparam>
  </node>

  <node name="image_view_color_filtering"
    pkg="image_view" type="image_view">
    <remap from="image" to="hsv_color_filter/image" />
  </node>

  <node name="rqt_reconfigure"
    pkg="rqt_reconfigure" type="rqt_reconfigure"
    args="hsv_color_filter"/>

</launch>
```
You can launch the roslaunch file by:

```
$ roslaunch ./tutorial_color_filtering.launch
```

## 2.3 Get masked image

### 2.3.1 Introduction

In previous tutorial, we got object mask image of red box, and the next step is masking the color image using it. We shows how to apply mask image to color image in this tutorial.

### 2.3.2 Step by step

First, you need image topic, so please run `tutorial_image_publisher.launch` in previous tutorials: Publish static image topic, Find object with color filtering, and below rostopic list output is expected:

```
$ rostopic list
```

For this purpose, we can use `apply_mask_image`. The advance from previous tutorials is it subscribes 2 topics: color image and mask for processing.

```
$ rosrun jsk_perception apply_mask_image
   _clip:=false _approximate_sync:=false _queue_size:=100
   ~input:=image_publisher/output ~input/mask:=hsv_color_filter/image
```

_clip:=false is for disabling cropping of the region. _approximate_sync:=false and _queue_size:=... is for synchronization method of timestamp, that is needed if you subscribe multiple topics. In this case, the both topic has the exactly the same stamp and it can be synchronized without approximation. You can see the timestamp by below:

```
$ rostopic echo /image_publisher/image/header/stamp
$ rostopic echo /hsv_color_filter/image/header/stamp
```

You can see the timestamp by below:
The result will be like below:

![Result Image]

### 2.3.3 Get better results

To get better results, you need some processing to the mask, and below launch file includes the program:

Or you can download the file from:

```
tutorial_get_masked_image.launch
```

```xml
<launch>
  <node name="apply_mask_image"
    pkg="jsk_perception" type="apply_mask_image">
    <remap from="~input" to="image_publisher/output" />
    <remap from="~input/mask" to="hsv_color_filter/image" />
    <rosparam>
      clip: false
    </rosparam>
  </node>

  <node name="image_view_apply_mask"
    pkg="image_view" type="image_view">
    <remap from="image" to="apply_mask_image/output" />
  </node>

  <!-- !!! ADVANCED !!! -->

  <node name="bounding_object_mask_image"
    pkg="jsk_perception" type="bounding_object_mask_image">
    <remap from="~input" to="hsv_color_filter/image" />
  </node>
</launch>
```

(continues on next page)
<node name="apply_mask_image_better" pkg="jsk_perception" type="apply_mask_image">
  <remap from="~input" to="image_publisher/output" />
  <remap from="~input/mask" to="bounding_object_mask_image/output" />
  <rosparam>
    clip: false
  </rosparam>
</node>

<node name="image_view_apply_mask_better" pkg="image_view" type="image_view">
  <remap from="image" to="apply_mask_image_better/output" />
</node>

</launch>

You can launch the roslaunch file by:

```
$ roslaunch ./tutorial_get_masked_image.launch
```

The result will be like below:

![Image result](image.png)

What’s next?

You can see nodes/nodelets listed in jsk_perception and jsk_pcl_ros.
3.1 Cannot compile jsk_recognition because of “memory allocation error”

`jsk_pcl_ros` and `jsk_perception` requires much memory to be compiled because of PCL. On average, each cpp file requires 2.5GB memory to compile.

If your machine does not have enough memory, please use smaller number of CPUs to compile

```
catkin build -p 1 -j 1
```

3.2 How to install PCL from source?

Refer to script used on Travis:

```
#!/usr/bin/env bash

# Install PCL 1.8

if [ ! -e /usr/local/include/pcl-1.8/pcl/pcl_base.h ]; then
  cd /tmp
  version="1.8.0rc2"
  url="https://github.com/PointCloudLibrary/pcl/archive/pcl-$version.tar.gz"
  fname=pcl-$version.tar.gz
  wget $url -O $fname
  tar zxf $fname
  cd pcl-$version
  mkdir build
  cd build

  # pcl::CropBox does not work properly in kinetic
  # with PCL_ENABLE_SSE:BOOL=TRUE flag
  # https://github.com/PointCloudLibrary/pcl/pull/1917
  if [ $(lsb_release -c -s) = "xenial" ]; then
    cmake -DCMAKE_BUILD_TYPE=Release -DPCL_ENABLE_SSE:BOOL=FALSE ..
  else
    cmake -DCMAKE_BUILD_TYPE=Release ..
  fi

  make
```

(continues on next page)
3.3 How to install OpenCV from source?

Refer to script used on Travis:

```bash
#!/usr/bin/env bash

# Install OpenCV 3
sudo -H apt-get install -y -q -qq ros-$ROS_DISTRO-opencv3

# Setup dependencies to rebuild from source
rosinstall_generator --tar --rosdistro $ROS_DISTRO \
cv_bridge \
image_geometry \
image_transport \
image_view2 \
jsk_data \
jsk_topic_tools \
opencv_apps \
>> /tmp/$$.rosinstall

cd ~/$REPOSITORY_NAME/src
wstool merge /tmp/$$.rosinstall
wstool up -j4
```
3.4 How to install OpenNI2?

By Apt

```
sudo apt-get install libopenni2-dev
```

From Source

```
sudo apt-get install g++
sudo apt-get install python
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libudev-dev
sudo apt-get install openjdk-6-jdk
sudo apt-get install freeglut3-dev
sudo apt-get install graphviz
git clone https://github.com/occipital/openni2
cd openni2
make
export PATH=$(/bin/$(pwd)/Bin/x64-Release):$PATH
export LD_LIBRARY_PATH=$(/bin/$(pwd)/Bin/x64-Release):$LD_LIBRARY_PATH
```

```
# need to build openni2_camera from source
cd ~/catkin_ws/src
git clone https://github.com/ros-drivers/openni2_camera.git
catkin build -iv
```

3.5 How to generate doc on local machine

```
roscd jsk_recognition
cd ../doc
source setup.sh
make html
```

Then you can see doc with _build/html/index.html


4.1 Spec

<table>
<thead>
<tr>
<th>Spec/Device</th>
<th>Astra</th>
<th>Astra Mini</th>
<th>Astra S</th>
<th>Astra Mini S</th>
<th>Astra Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized Range</td>
<td>0.6 - 5.0m</td>
<td>0.6 - 5.0m</td>
<td>0.4 - 2.0m</td>
<td>0.35 - 1.0m</td>
<td>0.6 - 5.0m</td>
</tr>
<tr>
<td>RGB Image Size</td>
<td>HD (10FPS) VGA (30FPS) QVGA (30FPS)</td>
<td>HD (10FPS) VGA (30FPS) QVGA (30FPS)</td>
<td>HD (10FPS) VGA (30FPS) QVGA (30FPS)</td>
<td>HD (10FPS) VGA (30FPS) QVGA (30FPS)</td>
<td>HD (30FPS) VGA (30FPS) QVGA (30FPS)</td>
</tr>
<tr>
<td>Depth Image Size</td>
<td>VGA (30FPS) QVGA (30FPS)</td>
<td>VGA (30FPS) QVGA (30FPS)</td>
<td>VGA (30FPS) QVGA (30FPS)</td>
<td>VGA (30FPS) QVGA (30FPS)</td>
<td>VGA (30FPS) QVGA (30FPS)</td>
</tr>
<tr>
<td>Field of View</td>
<td>60° horiz 49.5° vert 73° diagonal</td>
<td>60° horiz 49.5° vert 73° diagonal</td>
<td>60° horiz 49.5° vert 73° diagonal</td>
<td>60° horiz 49.5° vert 73° diagonal</td>
<td></td>
</tr>
</tbody>
</table>

4.2 Install SDK && Try Sample

```bash
cd ~/Downloads
# this can be broken
# wget "https://www.dropbox.com/sh/p2uowlt3swdrfno/AACfEbv7ejIU-4FHy4Fy102Wa?dl=1" -O Tools_SDK_OpenNI.zip
sudo pip install gdown
gdown "https://drive.google.com/uc?id=0B9P1L--7Wd2vSktr2XFy2E20WXm" -O Tools_SDK_OpenNI.zip
mkdir ~/Downloads/Tools_SDK_OpenNI
```

(continues on next page)
unzip ~/.Downloads/Tools_SDK_OpenNI.zip

```
cd 2-Linux
tar zxvf OpenNI-Linux-x64-2.2-0118.tgz
cd OpenNI-Linux-x64-2.2
```
sudo ./install.sh

cd ~/.Downloads/Tools_SDK_OpenNI/2-Linux/OpenNI-Linux-x64-2.2/Samples/Bin

```
./SimpleViewer  # This should open a viewer for depth image
```

### 4.3 Use Astra camera with openni2_camera ROS package

```
sudo apt-get install ros-$ROS_DISTRO-openni2-camera ros-$ROS_DISTRO-openni2-launch

cd ~/.Downloads/Tools_SDK_OpenNI/2-Linux/OpenNI-Linux-x64-2.2/Samples/Bin
```
sudo cp libOpenNI2.so /usr/lib/libOpenNI2.so
suo cp OpenNI2/Drivers/* /usr/lib/OpenNI2/Drivers/

Then, edit /usr/lib/pkgconfig/libopenni2.pc to be like below:

```
prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include/openni2

Name: OpenNI2
Description: A general purpose driver for all OpenNI cameras.
Version: 2.2.0.3
Cflags: -I${includedir}
```

```
cd <your catkin workspace>/src
# if you do not initialize wstools
# wstool init .
   --git --git -v indigo-devel -y -u
```

```
cd ros-drivers/openni2_camera
source /opt/ros/$ROS_DISTRO/setup.bash
catkin bt
```

```
source <your catkin workspace>/devel/setup.bash
roslaunch openni2_launch openni2.launch
```

Chapter 4. Install Astra camera
4.4 Topics per Devices

Below are tested with `roslaunch openni2_launch openni2.launch`.

<table>
<thead>
<tr>
<th>Topics/Device</th>
<th>Astra</th>
<th>Astra S</th>
<th>Astra Pro</th>
<th>Astra Mini S</th>
</tr>
</thead>
<tbody>
<tr>
<td>/camera/rgb/image_raw</td>
<td>o</td>
<td>o</td>
<td>?</td>
<td>o</td>
</tr>
<tr>
<td>/camera/depth/image_raw</td>
<td>o</td>
<td>o</td>
<td>?</td>
<td>o</td>
</tr>
<tr>
<td>/camera/depth/points</td>
<td>o</td>
<td>o</td>
<td>?</td>
<td>o</td>
</tr>
<tr>
<td>/camera/depth_registered/image_raw</td>
<td>o</td>
<td>o</td>
<td>?</td>
<td>o</td>
</tr>
<tr>
<td>/camera/depth_registered/points</td>
<td>x</td>
<td>o</td>
<td>?</td>
<td>o</td>
</tr>
</tbody>
</table>
5.1 Devices

This instruction supports below cameras:

- Softkinetic
- DepthSense DS325

5.2 Attention

- Field of view of RGB Image and Depth Image is different, it is not as kinect2.

<table>
<thead>
<tr>
<th>RGB Image</th>
<th>Depth Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>63.2°x49.3°x75.2°</td>
<td>74°x58°x87°</td>
</tr>
</tbody>
</table>

- Resolution of RGB Image and Depth Image is different, it is not as kinect2.

<table>
<thead>
<tr>
<th>RGB Image</th>
<th>Depth Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGA or HD</td>
<td>QVGA</td>
</tr>
</tbody>
</table>

If you want more information, please read datasheet. (DS325 DataSheet)

5.3 Installation

1. Install DepthSenseSDK. You need to login to download it. (Middleware -> Download)

2. Install ROS package:

   cd ~/ros/indigo/src
   wstool set ipa320/softkinetic https://github.com/ipa320/softkinetic.git --git --vindigo_dev -y --update
   cd softkinetic/softkinetic && catkin bt -iv

3. Connect device and run:

   roslaunch softkinetic_camera softkinetic_camera_demo.launch
5.4 Known Issues

5.4.1 Error: No space left on device

```
[ INFO] [1432824767.747549238]: Number of Devices found: 1
Argument Exception: argument "node": context already has control over the node
terminate called after throwing an instance of 'DepthSense::StreamingException'
  what(): VIDIOC_STREAMON failed (No space left on device)
```

If you have above error, you can try below:

```
vim /etc/modprobe.d/blacklist.conf
# add 'blacklist snd_usb_audio' at the bottom and then reboot
```

References

- https://github.com/ipa320/softkinetic/issues/48
CHAPTER SIX

INSTALL REALSENSE CAMERA

• SDK
  – Data Sheet of RealSense Products

You need to install Intel® RealSense™ SDK 2.0 (librealsense2) and ROS Wrapper for Intel® RealSense™ Devices (realsense-ros).

And there are 3 ways to install librealsense.

• Method 1: Install deb packages of librealsense2 from Intel Repository and build realsense-ros from source.
• Method 2: Install deb packages of librealsense2 from ROS Repository and install deb packages of realsense-ros from ROS Repository.
• Method 3: Build librealsense2 and realsense-ros from source.

Basically method 1 is most stable. Please consider which case you use depend on your use case.

If you want to use 14.04 or older ubuntu versions, Intel® RealSense™ SDK 2.0 is not released as deb packages. So you need to build them from source. In addition, librealsense works stably on 4.4.xx kernels. So you may need to upgrade your ubuntu kernel Please see this page for more details about manual installation.

And if you want to use legacy devices like (F200, R200, LR200 and ZR300), please use old librealsense.

6.1 Method 1: Install librealsense2 from Intel repo and build realsense-ros from source

Intel® RealSense™ SDK 2.0 is released as deb packages. So you can install it with apt command. Please see this page and this page for more details.

```bash
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE || sudo apt-key adv --keyserver hkp:// keyserver.ubuntu.com:80 --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE
sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo $(lsb_release -cs) main" -u
```

(continues on next page)
sudo apt update
sudo apt-get install librealsense2-dkms librealsense2-utils librealsense2-dev
→librealsense2-dbg

and please build realsense-ros from source

mkdir -p ~/catkin_ws/src
cd catkin_ws/
catkin init
cd src/
git clone https://github.com/IntelRealSense/realsense-ros.git
catkin build

If you install librealsense2 from Intel repo, make sure realsense-ros and librealsense are not installed from ROS repository. This will happen if you run rosdep install with some package have dependencies for them. Please run rosdep with --skip-keys=librealsense2.

### 6.2 Method 2: Install *librealsense2* and *realsense-ros* from ROS Repository

librealsense2 and realsense-ros are also released as debian packages from ROS repository. Please see this page for more details.

```
sudo apt install ros-$ROS_DISTRO-librealsense2 ros-$ROS_DISTRO-realsense2-camera ros-
→$ROS_DISTRO-realsense2-description
```

And these packages lack a udev file for realsense devices. So you need to install it manually. Please see this issue for more details about this issue.

*wget https://github.com/IntelRealSense/librealsense/raw/master/config/99-realsense-
→libusb.rules
sudo cp 99-realsense-libusb.rules /etc/udev/rules.d/
*

### 6.3 Installation of librealsense for ubuntu 14.04 or older (Old documentation)

librealsense 2.0 or above is not distributed with debian package for ubuntu 14.04. so you have to build librealsense from source. In addition, librealsense works stably on 4.4.xx kernels. So you need to upgrade your ubuntu kernel Please see this page for more details about manual installation.

```
sudo apt-get install ros-indigo-realsense-camera

cd ~
git clone https://github.com/IntelRealSense/librealsense.git
cd librealsense
git checkout v0.9.2

sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules && udevadm trigger
```

(continues on next page)
# Requirement Installation
# If you already installed, you can skip here.

## gcc-4.9 and g++-4.9
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install gcc-4.9 g++-4.9
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 60 --slave /usr/bin/g++ g++ /usr/bin/g++-4.9

## openssl
sudo apt-get install libssl-dev

# uvcvideo patch Installation
./scripts/patch-uvcvideo-4.4.sh v4.4-wily
# this script sometimes causes error below with Ubuntu 14.04
# cp: will not overwrite just-created './.config' with '/usr/src/linux-headers-4.4.4-040404-generic/.config'
# If you got this error, see https://github.com/IntelRealSense/librealsense/issues/146
# or see https://github.com/IntelRealSense/librealsense/issues/70
# my solution is https://gist.github.com/knorth55/8e76494a694a287a8cf00b54c38e29ad
sudo modprobe uvcvideo
# if you get error below, patch script is not successful.
# modprobe: ERROR: could not insert 'uvcvideo'

And then, please build old realsense-ros from source.

```
mkdir -p ~/catkin_ws/src
cd catkin_ws/src
git clone https://github.com/intel-ros/realsense.git
git checkout 1.5.0
cd ../..
rosdep install --skip-keys=librealsense --ignore-src --from-path -i src -y -r
catkin build
```

## 6.4. Sample Launch

You can launch a realsense driver for D400 Series and see images or point cloud from a device.

```
roslaunch realsense2_camera demo_pointcloud.launch
```

If you use L515, please run below commands (be careful that L515 RGB does not support 4:3 images such as 640x480)

```
roslaunch realsense2_camera rs_rgbd.launch color_width:=1280 color_height:=720 depth_width:=1024 depth_height:=768
rviz -d $(find realsense2_camera)/rviz/pointcloud.rviz
```

If you use T265, you can launch a demo launch with

```
```
roslaunch realsense2_camera demo_t265.launch

If you have both of T265 and D400 Series, you can launch it concurrently and see result of visual odometry with point cloud (please align both camera).

roslaunch realsense2_camera rs_d400_and_t265.launch
rviz -d $(find realsense2_camera)/rviz/t265.rviz

# please add point cloud visualization plugin

For legacy devices

source ~/catkin_ws/devel/setup.bash
# for SR300
roslaunch realsense_camera sr300_nodelet_rgbd.launch
# for R200
roslaunch realsense_camera r200_nodelet_rgbd.launch

# another terminal
rosrun rviz rviz

### 6.5 Video

- SR300
- R200
7.1 Spec

<table>
<thead>
<tr>
<th>Spec/Device</th>
<th>Xtion2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized Range</td>
<td>0.8 - 3.5m</td>
</tr>
<tr>
<td>RGB Image Size</td>
<td>5MP (2592x1944)</td>
</tr>
<tr>
<td>Depth Image Size</td>
<td>VGA (30FPS)</td>
</tr>
<tr>
<td></td>
<td>QVGA (30FPS)</td>
</tr>
<tr>
<td>Field of View</td>
<td>74° horiz</td>
</tr>
<tr>
<td></td>
<td>52° vert</td>
</tr>
<tr>
<td></td>
<td>90° diagonal</td>
</tr>
</tbody>
</table>

7.2 Install SDK && Try Sample

```bash
# Download SDK.zip from https://www.asus.com/3D-Sensor/Xtion-2/HelpDesk_Download/
cd ~/Downloads
tar xf ASUS-Linux-x64-OpenNI2.2.tar.gz
cd ASUS-Linux-x64-OpenNI2.2/
sudo apt-get install ros-$ROS_DISTRO-openni2-launch ros-$ROS_DISTRO-openni2-camera
"libopenni2-dev"
sudo ./install.sh
# you need to copy libSenDuck.so by manual
sudo cp ASUS/Xtion2/lib/libSenDuck.so /usr/lib/OpenNI2/Drivers/
cd Samples/Bin
./SimpleViewer  # This should open a viewer for depth image
```
7.3 Use Xtion2 camera with openni2_camera ROS package

```
cd ~/ros/$ROS_DISTRO/src
git --git --v 0.2.8 -y -u
source /opt/ros/$ROS_DISTRO/setup.bash
catkin b

source ~/ros/$ROS_DISTRO/devel/setup.bash
roslaunch openni2_launch openni2.launch
```
8.1 Install SDK & Demo

```bash
cd ~/Downloads

# install codemeter
wget -O codemeter_6.40.2402.501_amd64.deb https://download.ensenso.com/s/ensensosdk/
download?files=codemeter_6.40.2402.501_amd64.deb
sudo dpkg -i codemeter_6.40.2402.501_amd64.deb

# install ensenso sdk
wget -O ensenso-sdk-2.0.147-x64.deb https://download.ensenso.com/s/ensensosdk/
download?files=ensenso-sdk-2.0.147-x64.deb
sudo dpkg -i ensenso-sdk-2.0.147-x64.deb

# install ueye_driver
wget -O uEye_4.81.1_Linux_64.tgz https://download.ensenso.com/s/idsdrivers/download?
download?files=uEye_4.81.1_Linux_64.tgz
tar czf uEye_4.81.1_Linux_64.tgz
sudo bash ./ueyesdk-setup-4.81.01-eth-amd64.gz.run
sudo bash ./ueyesdk-setup-4.81.01-usb-amd64.gz.run
sudo service ueyeethdrc start
sudo service ueyeusbdrv start

# demo
nxView
```

8.2 Use Ensenso Camera with ensenso ROS Package

```bash
mkdir catkin_ws/src -p
cd catkin_ws/src
git clone https://github.com/crigroup/ensenso.git
cd ..
catkin build
roslaunch ensenso viewer.launch serial:=CAMERA_SERIAL rqt:=false

(continues on next page)```
rosrun tf static_transform_publisher 0 0 0 0 0 0 base camera_optical_frame
rviz
9.1 Install SDK & Try Sample

- Join the Developer Program and Download SDK (Cross-Platform)
- https://developer.structure.io/sdk

```
# Download SDK to ~/Downloads
cd ~/Downloads
unzip StructureSDK-CrossPlatform-0.7.2.zip
cd StructureSDK-CrossPlatform-0.7.2-ROS

# Create udev rules for non root users
chmod +x ./DriverAndFirmware/Linux/Install-CoreDriver-Udev-Linux.sh
sudo ./DriverAndFirmware/Linux/Install-CoreDriver-Udev-Linux.sh

# Update firmware of the sensor
chmod +x ./DriverAndFirmware/linux/CoreFirmwareUpdater-0.9.7-Linux-x86_64
sudo ./DriverAndFirmware/linux/CoreFirmwareUpdater-0.9.7-Linux-x86_64

# Build the SDK
chmod +x ./Scripts/build.sh
./Scripts/build.sh

# Run the examples
./Builds/linux-release-x86_64/Samples/CorePlayground/CorePlayground
```

9.2 Install ROS Driver beta

```
# Move SDK Repository to catkin workspace
cd ~/Downloads
mv StructureSDK-CrossPlatform-0.7.2-ROS\ Driver\ beta/ ~/catkin_ws/src/
cd catkin_ws/src/StructureSDK-CrossPlatform-0.7.2-ROS\ Driver\ beta/ROS

# changed permission
chmod +x ros1_driver/cfg/SCParams.cfg

# source to_ros1.sh script and build the package
```

(continues on next page)
chmod +x to_ros1.sh
source to_ros1.sh
catkin build

# launch
source ~/catkin_ws/devel/setup.bash
roslaunch structure_core_ros_driver sc.launch
INSTALL ELP USB 4K CAMERA

- ELP USB 4K camera: http://www.webcamerusb.com
- libuvc_camera: http://wiki.ros.org/libuvc_camera

10.1 Install Udev

```bash
roscd jsk_perception
sudo cp udev/99-elp-usb-4k.rules /etc/udev/rules.d/
sudo service udev restart
```
10.2 Sample

roslaunch jsk_perception sample_elp_usb_4k.launch
INSTALL INSTA360 AIR

- Insta360 Air: https://www.insta360.com/jp/product/insta360-air
- libuvc_camera: http://wiki.ros.org/libuvc_camera
11.1 Install Udev

```
roscd jsk_perception
sudo cp udev/99-inst360-air.rules /etc/udev/rules.d/
sudo service udev restart
```

11.2 Sample

```
roslaunch jsk_perception sample_insta360_air.launch
# For kinetic or lower
roslaunch jsk_perception sample_insta360_air.launch use_usb_cam:=true
```
INSTALL CHAINER/PyTorch WITH GPU SUPPORT

This documentation describes how to install Chainer/PyTorch with GPU support.

12.1 Requirements

  You can check whether your PC has a GPU by `lspci | grep -i nvidia`.

12.2 Version Compatibilities for 18.04

(Recommended) Use CUDA 9.1 from Official ubuntu repository (https://packages.ubuntu.com/bionic/nvidia-cuda-dev)

- Chainer
  - chainer == 6.7.0 (last version supporting python2. See https://github.com/chainer/chainer/releases/tag/v6.7.0)
  - cupy-cuda91 == 6.7.0 (chainer v6.7.0 requires cupy/cudnn for hardware acceleration support https://docs.chainer.org/en/v6.7.0/install.html)

- PyTorch
  - pytorch == 1.1.0 (Latest pytorch version supporting CUDA 9.1 https://download.pytorch.org/whl/cu90/torch_stable.html)
  - CUDA >= 9.0 (Minimum required version for PyTorch 1.1.0 https://pytorch.org/get-started/previous-versions/#v110)

(Experimental) Use CUDA 10.2 from Nvidia Developer’s site (https://developer.nvidia.com/cuda-10.2-download-archive)

- Chainer
  - chainer == 6.7.0 (last version supporting python2. See https://github.com/chainer/chainer/releases/tag/v6.7.0)
  - cupy >= v6.7.0 < 7.0.0 (chainer v6.7.0 requires cupy/cudnn for hardware acceleration support https://docs.chainer.org/en/v6.7.0/install.html)
  - cuDNN < 8 (cupy 6.7.0 requires cuDNN v5000= and <=v7999)
  - CUDA 10.2 (cuDNN v7.6.5 requires CUDA 10.2 https://developer.nvidia.com/rdp/cudnn-archive)
• PyTorch
  – pytorch >= 1.4.0
  – CUDA >= 9.2 (Minimum required version for PyTorch https://pytorch.org/get-started/previous-versions/#v140)

12.3 Install CUDA

  ```bash
  # If you’d like to use CUDA8.0 on Ubuntu 14.04. wget https://developer.nvidia.com/compute/cuda/8.0/Prod2/local_installers/cuda-repo-ubuntu1404-8-0-local-ga2_8.0.61-1_amd64-deb
  mv cuda-repo-ubuntu1404-8-0-local-ga2_8.0.61-1_amd64-deb cuda-repo-ubuntu1404-8-0-local-ga2_8.0.61-1_amd64-deb sudo dpkg -i cuda-repo-ubuntu1404-8-0-local-ga2_8.0.61-1_amd64.deb sudo apt-get update sudo apt-get install cuda
  ```
  • Add below to your ~/.bashrc:
    ```bash
    # setup cuda & cudnn export LD_LIBRARY_PATH=/usr/local/lib:/usr/lib:$LD_LIBRARY_PATH export LIBRARY_PATH=/usr/local/lib:/usr/lib:$LIBRARY_PATH export CPATH=/usr/include:$CPATH export CFLAGS=-I/usr/include export LDFLAGS=-L/usr/local/lib -L/usr/lib
    if [ -e /usr/local/cuda ]; then
      export CUDA_PATH=/usr/local/cuda
      export PATH=$CUDA_PATH/bin:$PATH
      export CPATH=$CUDA_PATH/include:$CPATH
      export LD_LIBRARY_PATH=$CUDA_PATH/lib64:$CUDA_PATH/lib:$LD_LIBRARY_PATH
      export CFLAGS=-I$CUDA_PATH/include
      export LDFLAGS=-L$CUDA_PATH/lib64 -L$CUDA_PATH/lib
    fi
    ```
    ```bash
    # If you’d like to use CUDA9.2 on Ubuntu 16.04. # Choose the green buttons on the web page like x86_64 -> Ubuntu -> version -> deb (network). # Excute 1-3 and then, change step 4 as follows: sudo apt install cuda-9-2
    ```
  • Ubuntu 18.04 : You can use CUDA 9.1 by deafult
    ```bash
    sudo apt install nvidia-cuda-toolkit
    ```
  • (Experimental) Ubuntu 18.04 : CUDA 10.2 is the latest version which supports jsk_perception. Download deb file from https://developer.nvidia.com/cuda-downloads?target_os=Linux:
    ```bash
    # If you’d like to use CUDA10.2 on Ubuntu 18.04. # goto https://developer.nvidia.com/cuda-10.2-download-archive # Choose the green buttons on the web page like x86_64 -> Ubuntu -> version -> deb (network). # Excute all steps, but change the last step as follows: sudo apt install cuda-10-2
    ```
  – If you install CUDA from nvidia, Make sure to uninstall CUDA tools from packages.ubuntu.com
    ```bash
    `bash sudo apt remove nvidia-cuda-toolkit
    sudo apt remove nvidia-cuda-dev`
    ```
– Also set environment variables to ~/.bashrc

```bash
# set PATH for cuda 10.0 installation if
if [ -d "/usr/local/cuda-10.2/bin/" ]; then
export PATH=/usr/local/cuda-10.2/bin$PATH:+${PATH}
export LD_LIBRARY_PATH=/usr/local/cuda-10.2/lib64$LD_LIBRARY_PATH:+${LD_LIBRARY_PATH}
export CFLAGS=-I/usr/local/cuda-10.2/include
```

• After rebooting, you can see the memory usage of your GPU by `nvidia-smi`

### 12.4 Install CUDNN

• If you install `pip install cupy-cuda91`, you do not need to install CUDNN manually. (c.f. https://github.com/jsk-ros-pkg/jsk_visualization/issues/809). Thus, default 18.04 user can use CUDA 9.1 and `cupy-cuda91==6.7.0` for `chainer==6.7.0` and you can SKIP this section.

Installing CUDNN manually only requires for experimental user who install CUDA 10.2 manually.

• You need to login at https://developer.nvidia.com/cudnn

• Go to cuDNN Download and choose version

• Download deb files of cuDNN Runtime Library and cuDNN Developer Library

```bash
# If you’d like to install cuDNN for CUDA9.2 on Ubuntu 16.04 # Download cuDNN v7.3.1 Runtime Library for Ubuntu16.04 (Deb) sudo dpkg -i libcudnn7_7.3.1.20-1-cuda9.2_amd64.deb # Download cuDNN v7.3.1 Developer Library for Ubuntu16.04 (Deb) sudo dpkg -i libcudnn7-dev_7.3.1.20-1+cuda9.2_amd64.deb # Download cuDNN v7.6.5 Developer Library for Ubuntu18.04 (Deb) sudo dpkg -i libcudnn7_7.6.5.32-1+cuda10.2_amd64.deb sudo dpkg -i libcudnn7-dev_7.6.5.32-1+cuda10.2_amd64.deb
```

### 12.5 Install Chainer

```bash
sudo pip install chainer==6.7.0
```

### 12.6 Install CuPy

• (Default) Chainer 6.7.0 requires CuPy 6.7.0 and if you have CUDA 9.1, you can use CuPy pre-compiled binary package.

  – Pre-compiled Install CuPy for CUDA 9.1

```bash
sudo pip install cupy-cuda91==6.7.0
```

• (Experimental) If you have newer CUDA version. You need to install CuPy with source distribution. This requires CUDNN before you run `pip install cupy`.

  – Source Install CuPy for CUDA 10.2

```bash
```
sudo pip install -vvv cupy --no-cache-dir

12.7 Install PyTorch

- 18.04 provides CUDA 9.1 by default. To install PyTorch compatible with this version, download following wheel from https://download.pytorch.org/whl/cu90/torch_stable.html, and install manually.

```bash
sudo pip install torch-1.1.0-cp27-cp27mu-linux_x86_64.whl sudo pip install torchvision-0.3.0-cp27-cp27mu-manylinux1_x86_64.whl
```

- (Experimental) If you manually install CUDA 10.2 manually, you can use latest PyTorch.

```bash
sudo pip install torch==1.4.0
```

- See https://github.com/jsk-ros-pkg/jsk_recognition/pull/2601#issuecomment-876948260 for more info.

12.8 Try Chainer Samples

You can try to run samples to check if the installation succeeded:

```bash
roslaunch jsk_perception sample_fcn_object_segmentation.launch gpu:=0
roslaunch jsk_perception sample_people_pose_estimation_2d.launch GPU:=0
roslaunch jsk_perception sample_regional_feature_based_object_recognition.launch
˓
```

12.9 Try PyTorch Samples

You can try to run samples to check if the installation succeeded:

```bash
roslaunch jsk_perception sample_hand_pose_estimation_2d.launch gpu:=0
```

12.10 Trouble Shooting

- After installing CUDA and rebooting, `nvidia-smi` returns `command not found`
If your PC uses dual boot, please check BIOS setting and secure boot is disabled.

- When installing `jsk_perception`, `rosdump install --from-paths --ignore-src -y -r src` fails due to pip version:

Please make sure you have pip >= 9.0.1. If not, please try `sudo python -m pip install pip==9.0.1`, for example. Please do not execute `pip install -U pip`. (2018.11.20)
The following settings can be applied at the same time by combining cmake options.

### 13.1 Install PCL from source without SSE

This is sometimes needed for AttentionClipper.

```bash
cd
wget https://github.com/PointCloudLibrary/pcl/archive/pcl-1.8.1.tar.gz
tar xvzf pcl-1.8.1.tar.gz
cd pcl-pcl-1.8.1
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release -DPCL_ENABLE_SSE:BOOL=FALSE
make -j 4
sudo make install
```

And rebuild all PCL related packages from source

### 13.2 Install PCL from source with CUDA

This is needed for Kinfu.

```bash
cd
wget https://github.com/PointCloudLibrary/pcl/archive/pcl-1.8.1.tar.gz
tar xvzf pcl-1.8.1.tar.gz
cd pcl-pcl-1.8.1
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release -DWITH_CUDA:BOOL=ON
make -j 4
sudo make install
```

And rebuild all PCL related packages from source
DEEP LEARNING WITH YOUR OWN IMAGE DATASET

This page shows the overview of how to start deep learning with your own image dataset.

14.1 1. Collect raw images

Collect available images from Web or take pictures by yourself.

14.2 2. Annotate images

Label each image or each region in the images.

14.2.1 Annotate images with labelme

This documentation describes how to annotate images with an annotation tool named labelme.

Install labelme

```
pip install labelme
```

For more information, please see https://github.com/wkentaro/labelme

How to use labelme

Split your data

We recommend you to make `train`, `val` and `test` directory, and split your image files into them.
Prepare label list

List up all labels and save them into a file (e.g. labels.txt).
Note that “ignore” label must be described as __ignore__ and “background” label as _background_, and they must be placed in the first and second line respectively in labels.txt.
All labels must not have “-”, because it is used to distinct instances.

Do annotation

1. First, open file with GUI.

```
labelme [--labels labels.txt] [directory | file]
```

2. Click “Create Polygons” and draw polygons.

3. Choose the class of the object from “Label List”.

   If you would like to create dataset for instance segmentation, please remember to name the polygon `<class name>-<instance id>`.
4. You can edit polygon by clicking “Edit Polygons”

5. When you have finished annotating all objects listed in “Label List” in the image, click “Save” to save .json file.
Create dataset

After annotation, you can create dataset from raw images, labels and JSON files.

- For semantic segmentation, see https://github.com/wkentaro/labelme/tree/master/examples/semantic_segmentation
- For instance segmentation, see https://github.com/wkentaro/labelme/tree/master/examples/instance_segmentation

Store dataset

At last, place your dataset where you want.

For example, $HOME/.ros/data/jsk_perception/learning_datasets/awesome_dataset.

```bash
awesome_dataset
|-- test
 |  |-- JPEGImages
 |  |-- SegmentationClass
 |  |-- SegmentationClassPNG
 |  |-- SegmentationClassVisualization
 |  |-- SegmentationObject
 |  |-- SegmentationObjectPNG
 |  |-- SegmentationObjectVisualization
 |     `-- class_names.txt
|-- train
 |  |-- JPEGImages
 |  |-- SegmentationClass
 |  |-- SegmentationClassPNG
 |  |-- SegmentationClassVisualization
```

(continues on next page)
14.3 3. Train neural network

Choose one network model according to your task.

There are some training scripts in jsk_perception package.

14.3.1 Train FCN (Fully Convolutional Network)

This page shows how to train FCN with your own dataset.

FCN is a neural network model used for semantic segmentation.

Any size of image can be applied to this network as long as your GPU has enough memory.

Available Dataset Class

SemanticSegmentationDataset (imported from jsk_recognition_utils.datasets)

This class assumes the following directory structure for each split.

```
path_to_awesome_dataset/
|-- JPEGImages
  |-- foo.jpg
  |-- bar.jpg
  `-- etc.
|-- SegmentationClass
  |-- foo.npy
  |-- bar.npy
  `-- etc.
|-- class_names.txt
  `-- etc.
```

Arguments

- `--train_dataset_dir` (string, default: \$(rospack find jsk_perception)/learning_datasets/kitchen_dataset/train)
  - Directory name which contains dataset for training.
- `--val_dataset_dir` (string, default: \$(rospack find jsk_perception)/learning_datasets/kitchen_dataset/test)
  - Directory name which contains dataset for validation.
- `--model_name` (string, default: fcn32s)
  - Model name. Currently, fcn32s, fcn16s, fcn8s and fcn8s_at_once are supported.
- `--gpu` (int, default: 0)
  - GPU id. -1 means CPU mode, but we recommend to use GPU for much faster computing.
• **--batch_size** (int, default: 1)
  Number of images used simultaneously in each iteration.
  You should decrease this number when you face memory allocation error.

• **--max_epoch** (int, default: 100)
  Stop trigger for training.

• **--lr** (float, default: 1e-10)
  Learning rate.
  Perhaps you should decrease this number when you face NaN value as loss.

• **--weight_decay** (float, default: 0.0001)
  Weight decay.

• **--out_dir** (string, default: `${ROS_HOME}/learning_logs/<timestamp>`) Output directory name.

• **--progressbar_update_interval** (float, default: 10)
  Interval for updating progress bar shown while training.
  The unit is [iteration].

• **--print_interval** (float, default: 100)

• **--print_interval_type** (string, default: iteration)
  Interval for printing information like current epoch, elapsed time, loss, etc. on terminal.
  Note that `XXX_interval_type` can be chosen from {'epoch', 'iteration'}.

• **--log_interval** (float, default: 10)

• **--log_interval_type** (string, default: iteration)
  Interval for logging information to `<out_dir>/log.json`.

• **--plot_interval** (float, default: 5)

• **--plot_interval_type** (string, default: epoch)
  Interval for plotting loss to `<out_dir>/loss_plot.png`.

• **--eval_interval** (float, default: 10)

• **--eval_interval_type** (string, default: epoch)
  Interval for running evaluation.

• **--save_interval** (float, default: 10)

• **--save_interval_type** (string, default: epoch)
  Interval for saving snapshot of model to `<out_dir>/model_snapshot.npz`.
  Trainer listens to loss for validation dataset according to this interval, and saves snapshot when the loss value becomes minimum.
Output

All these files will be automatically generated under `<out_dir>`.

- `log.json`
- `loss_plot.png`
- `model_name.yaml`
- `model_snapshot.npz`
- `network_architecture.dot`
- `params.yaml`
- `target_names.yaml`

Usage

```
rosrun jsk_perception train_fcn.py [ARGS]
```

14.3.2 Train Mask-RCNN

This page shows how to train Mask-RCNN with your own dataset.

Mask-RCNN is a neural network model used for instance segmentation.

Any size of image can be applied to this network as long as your GPU has enough memory.

Available Dataset Class

`InstanceSegmentationDataset` (imported from `jsk_recognition_utils.datasets`)

This class assumes the following directory structure for each split.

```
path_to_awesome_dataset/
|-- JPEGImages
 | |-- foo.jpg
 | |-- bar.jpg
 | `-- etc.
|-- SegmentationClass
 | |-- foo.npy
 | |-- bar.npy
 | `-- etc.
|-- SegmentationObject
 | |-- foo.npy
 | |-- bar.npy
 | `-- etc.
|-- class_names.txt
 `-- etc.
```
Arguments

- `--train_dataset_dir` (string, default: `${rospack find jsk_perception)/learning_datasets/kitchen_dataset/train}`
  Directory name which contains dataset for training.

- `--val_dataset_dir` (string, default: `${rospack find jsk_perception)/learning_datasets/kitchen_dataset/test}`
  Directory name which contains dataset for validation.

- `--model_name` (string, default: `resnet50`)
  Model name. Currently, `vgg16`, `resnet50`, and `resnet101` are supported.

- `--gpu` (int, default: 0)
  GPU id. -1 means CPU mode, but we recommend to use GPU for much faster computing.

- `--batch_size` (int, default: 1)
  Number of images used simultaneously in each iteration.
  You should decrease this number when you face memory allocation error.

- `--max_epoch` (int, default: 100)
  Stop trigger for training.

- `--lr` (float, default: 0.00125)
  Learning rate.
  Perhaps you should decrease this number when you face NaN value as loss.

- `--weight_decay` (float, default: 0.0001)
  Weight decay.

- `--out_dir` (string, default: `${ROS_HOME}/learning_logs/<timestamp>`) Output directory name.

- `--progressbar_update_interval` (float, default: 10)
  Interval for updating progress bar shown while training.
  The unit is [iteration].

- `--print_interval` (float, default: 100)
  Interval for printing information like current epoch, elapsed time, loss, etc. on terminal.
  Note that `XXX_interval_type` can be chosen from `{epoch, iteration}`.

- `--log_interval` (float, default: 10)
  Interval for logging information to `<out_dir>/log.json`.

- `--plot_interval` (float, default: 5)
  Interval for plotting Mean Average Precision to `<out_dir>/accuracy_plot.png`, and loss to `<out_dir>/loss_plot.png`.

- `--eval_interval` (float, default: 10)
• **--eval_interval_type** (string, default: *epoch*)
  Interval for running evaluation.

• **--save_interval** (float, default: *10*)

• **--save_interval_type** (string, default: *epoch*)
  Interval for saving snapshot of model to `<out_dir>/model_snapshot.npz`.
  Trainer listens to loss for validation dataset according to this interval, and saves snapshot when the loss value becomes minimum.

### Output

All these files will be automatically generated under `<out_dir>`.

- **accuracy_plot.png**
- **fg_class_names.yaml**
- **log.json**
- **loss_plot.png**
- **model_snapshot.npz**
- **network_architecture.dot**
- **params.yaml**

### Usage

```
rosrun jsk_perception train_mask_rcnn.py [ARGS]
```

### Sample Output

There are some pre-trained mask rcnn model on jsk_perception. Getting trained data by build `jsk_perception` or run script `install_trained_data`

`73b2 kitchen model` is some of the typical example of pre-trained mask rcnn model on `jsk_perception`. The results of mask rcnn using `73b2 kitchen model` are as follows.

Click the below image to see more mask rcnn results.
Sample usage with pre-trained model

You can use 73b2 kitchen model with jsk_perception node.

```
roslaunch jsk_perception sample_mask_rcnn_73b2_kitchen.launch
```

The result is displayed on the screen as a `/mask_rcnn_73b2_kitchen/output/viz` topic like below image.

![Sample Mask R-CNN Result](image)

14.3.3 Train SSD

This page shows how to train SSD with your own dataset.

SSD is a neural network model used for object detection.

Available Dataset Class

`DetectionDataset` (imported from `jsk_recognition_utils.datasets`)  

This class assumes the following directory structure for each split.

```
path_to_awesome_dataset/
|-- JPEGImages
 |  |-- foo.jpg
 |  |-- bar.jpg
 |  `-- etc.
```

(continues on next page)
BboxDetectionDataset (imported from jsk_recognition_utils.datasets)
This class assumes the following directory structure for each split. This can be generated using labelme

```
path_to_awesome_dataset/
|-- JPEGImages
 | |-- foo.jpg
 | |-- bar.jpg
 | `-- etc.
|-- Annotations
 | |-- foo.xml
 | |-- bar.xml
 | `-- etc.
|-- AnnotationsVisualization (not necessary)
 | |-- foo.jpg
 | |-- bar.jpg
 | `-- etc.
`-- class_names.txt
```

Arguments

- `--train-dataset-dir` (string, default: 
  $(rospack find jsk_perception)/learning_datasets/kitchen_dataset/train)
- `--val-dataset-dir` (string, default: 
  $(rospack find jsk_perception)/learning_datasets/kitchen_dataset/test)

Directory name which contains dataset for training and validation respectively.

- `--dataset-type` (string, default: instance)
  Choose dataset type. If you use DetectionDataset structure, choose instance. If you use BboxDetectionDataset, choose bbox.
- `--model-name` (string, default: ssd512)
  Model name. Currently, ssd300 and ssd512 are supported.
- `--gpu` (int, default: 0)
  GPU id. -1 means CPU mode, but we recommend to use GPU for much faster computing.
- `--batch-size` (int, default: 8)
  Number of images used simultaneously in each iteration.

You should decrease this number when you face memory allocation error.

14.3.  3. Train neural network
• --max-epoch (int, default: 100)
  Stop trigger for training.
• --out-dir (string, default: ${ROS_HOME}/learning_logs/<timestamp>)
  Output directory name.

Output

All these files will be automatically generated under <out_dir>.

• log.json
• model_snapshot.npz

Usage

rosrun jsk_perception train_ssd.py [ARGS]

Sample usage with pre-trained model

There are some pre-trained SSD model on jsk_perception. Getting trained data by build jsk_perception or run script install_trained_data

You can use 73b2 kitchen model with jsk_perception node.

roslaunch jsk_perception sample_ssd_object_detector_73b2_kitchen.launch
Sample Output

73b2 kitchen model is some of the typical example of pre-trained SSD model on jsk_perception. The results of SSD using 73b2 kitchen model are as follows.

14.4 4. Infer with trained network

Now you can use your trained network to infer things as a ROS node.
14.4.1 fcn_object_segmentation.py

What is this?

Segment object in pixel wise with Fully Convolutional Networks.

**Subscribing Topic**

**Default**

- ~input(sensor_msgs/Image)
  
  Raw image.

**Optional**

- ~input/mask(sensor_msgs/Image)
  
  Mask whose black region must be the background label: 0. This topic is subscribed only when param ~use_mask is true.

**Publishing Topic**

- ~output(sensor_msgs/Image)
  
  Label image each object in param ~target_names is segmented.

- ~output/proba_image(sensor_msgs/Image)
  
  Probability image of each object. The encoding is 32FCX, where X is the length of ~target_names.
Parameters

Default

• ~gpu (Int, Default: -1)
  GPU id. -1 represents CPU mode.

• ~target_names (List of String, Required)
  Target names for classification.

• ~backend (String, Default: chainer)
  Neural network framework. Currently chainer and torch are supported.

• ~model_name (String, Required)
  Currently fcn8s, fcn8s_at_once, fcn16s or fcn32s is only supported. See models in https://github.com/wkentaro/fcn/tree/master/fcn/models.

• ~model_file (String, Required)
  Saved npz or h5 file for trained model.

• ~use_mask (Bool, default: False)
  If True, ~input/mask is subscribed and ignore black region in the mask image.

• ~bg_label (Int, default: 0)
  Label value for background. This is used with rosparam ~proba_threshold

• ~proba_threshold (Float, default: 0.0)
  Threshold for labeling pixels as uncertain, and the uncertain region will be labeled as background with rosparam ~bg_label.

Optional

• ~queue_size (Int, default: 10)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics. This is used only when param ~use_mask is true.

• ~approximate_sync (Bool, default: False)
  Whether to use approximate for input topics. This is used only when param ~use_mask is true.

• ~slop (Float, default: 0.1)
  How many seconds you allow about the difference of timestamp. This is used only when param ~use_mask and ~approximate_sync are true.
Sample

```
roslaunch jsk_perception sample_fcn_object_segmentation.launch
```

Sample with mask image

```
roslaunch jsk_perception sample_fcn_object_segmentation.launch use_mask:=true
```
14.4.2 mask_rcnn_instance_segmentation.py

What is this?

Predict object instance masks and labels.

**Subscribing Topic**

- `~input(sensor_msgs/Image)`
  Raw image.

**Publishing Topic**

- `~output/cluster_indices(jsk_recognition_msgs/ClusterPointIndices)`
  Image indices of each instance.
- `~output/labels(jsk_recognition_msgs/LabelArray)`
  Class labels of each instance.
- `~output/label_cls(sensor_msgs/Image)`
  Label image color-coded by class.
- `~output/label_ins(sensor_msgs/Image)`
  Label image color-coded by instance.
- `~output/viz(sensor_msgs/Image)`
  Visualized image of recognition result.
- `~output/class(jsk_recognition_msgs/ClassificationResult)`
  Class information of detected objects.
- `~output/rects(jsk_recognition_msgs/RectArray)`
  Rectangles of detected objects.
Parameters

- `~gpu` (Int, default: 0)
  GPU id.
- `~model_name` (String, `mask_rcnn_resnet50`)
  Model name. `mask_rcnn_resnet50`, `mask_rcnn_fpn_resnet50` and `mask_rcnn_fpn_resnet101` are supported.
- `~score_thresh` (Float, default: 0.7)
  Score threshold of detections.
- `~fg_class_names` (List of String, None)
  Foreground class names that is used to identify number of class. It is also used for the name field of `~output/labels`. When `pretrained_model: coco` is set, default COCO pretrained_model and fg_class_names will be loaded. When `pretrained_model: voc` is set, default VOC pretrained_model and fg_class_names will be loaded.
- `~pretrained_model` (String, required)
  Pretrained model path. When `pretrained_model: coco` is set, default COCO pretrained_model and fg_class_names will be loaded. When `pretrained_model: voc` is set, default VOC pretrained_model and fg_class_names will be loaded.
- `~classifier_name` (String, default: `rospy.get_name()`)
  Name of this classifier
- `~anchor_scales` (List, default: `[4, 8, 16, 32]`)
  Anchor scales parameter for Mask RCNN.
- `~min_size` (Int, default: 600)
  Min size parameter of the input image for Mask RCNN.
- `~max_size` (Int, default: 1000)
  Max size parameter of the input image for Mask RCNN.

Sample

```bash
roslaunch jsk_perception sample_mask_rcnn_instance_segmentation.launch
~gpu:=0 COCO:=true  # COCO dataset (~80 classes)
roslaunch jsk_perception sample_mask_rcnn_instance_segmentation.launch
~gpu:=0 COCO:=false  # VOC dataset (~20 classes)
```
Launch

```
roslaunch jsk_perception mask_rcnn_instance_segmentation.launch gpu:=0 COCO:=true INPUT_IMAGE:=/camera/color/image_raw
```

### 14.4.3 ssd_object_detector

SSD (Single Shot Multibox Detector) object detector node

#### Usage

**Predicting**

```
roslaunch jsk_perception sample_ssd_object_detector.launch gpu:=0
```

You can see camera image with results in `image_view`
• Publishing Topics:
  - `~output/image (sensor_msgs/Image)`
    Output image with bounding boxes
  - `~output/labels (jsk_recognition_msgs/LabelArray)`
    Class labels of each instance
  - `~output/class (jsk_recognition_msgs/ClassificationResult)`
    Class information of detected objects
  - `~output/rect (jsk_recognition_msgs/RectArray)`
    Rectangles of detected objects
  - `~output/cluster_indices (jsk_recognition_msgs/ClusterPointIndices)`
    Image indices of each instance

• Subscribing Topics:
  - `~input (sensor_msgs/Image)`
    Input image

• Parameters:
  - `~classifier_name (String, default: rospy.get_name())`
    Name of this classifier
- ~gpu (Int, default: -1)
  Index of gpu used for prediction. Set -1 for using CPU.
- ~model (String, default: ssd300)
  Name of model structure. ssd300 and ssd512 are supported.
- ~model_path (String, default: voc0712)
  Name of pretrained model or path to trained model file.
- ~label_names (String or String[], default: [])
  Path to label yaml file or label names. If this parameter is not specified, use label of VOC dataset.
- ~nms_thresh (Double, default: 0.45)
  Threshold for non maximum suppression
- ~score_thresh (Double, default: 0.60)
  Threshold for confidence score
- ~profiling (Bool, default: False)
  Time profiling

Training dataset

1. Prepare dataset
   Each dataset must have image files in one directory and one file which contains label names.

   ```
   # label_names.yml
   - Apple
   - Bread
   - Candy
   - Drink
   - Egg
   ```

   ```
   ./dataset
   |- label_names.yml
   |- image0001.jpg
   |- image0002.jpg
   |- image0003.jpg
   |- image0004.jpg
   ... 
   ```

2. Install annotation tool

   ```
   git clone https://github.com/yuyu2172/image-labelling-tool
cd image-labelling-tool
pip install -e .
   ```

3. Annotate dataset

   Launch annotation tool that is installed in the previous section.

   ```
   python ./image-labelling-tool/flask_app.py --image_dir dataset/ --label_names dataset/label_names.yml --file_ext jpg
   ```

14.4. 4. Infer with trained network
If annotation tool is launched successfully, you can see the message as follows:

```
Loaded 90 images
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Open `http://127.0.0.1:5000/` with a web browser and annotate the dataset. After annotating images, `json` files should be generated in the same directory as images.

```
jskuser@machine:~/dataset$ ls
image0000.jpg  image0051__labels.json  image0102__labels.json
   ↓image0154.jpg
image0000__labels.json  image0052.jpg  image0103.jpg
   ↓image0154__labels.json
image0001.jpg  image0052__labels.json  image0103__labels.json
   ↓image0155.jpg
image0001__labels.json  image0053.jpg  image0104.jpg
   ↓image0155__labels.json
image0002.jpg  image0053__labels.json  image0104__labels.json
   ↓image0156.jpg
image0002__labels.json  image0054.jpg  image0105.jpg
   ↓image0156__labels.json
image0003.jpg  image0054__labels.json  image0105__labels.json
   ↓image0157.jpg
image0003__labels.json  image0055.jpg  image0106.jpg
   ↓image0157__labels.json
...`
```

4. Train from the annotated dataset

```
cd dataset/
roslaunch jsk_perception ssd_train_dataset.py --gpu 0 dataset/label_names.
```

When training process starts successfully, progress are shown on the terminal as follows:

```
Loaded 27 labels
train: 112, test: 28
epoch  iteration  lr  main/loss  main/loss/loc  main/loss/conf  validation/main/loc
        0       10  0.0001  9.95250  1.16335  8.78914
        0       20  0.0001  6.11782  0.73005  5.38776
        1       30  0.0001  4.26923  0.51191  3.75732
        1       40  0.0001  4.01094  0.51191  3.48108
        2       50  0.0001  3.16160  0.37409  2.78751
        2       60  0.0001  2.80446  0.33136  2.47310
        3       70  0.0001  2.57667  0.34800  2.22867
        3       80  0.0001  2.53918  0.35070  2.24248
        4       90  0.0001  2.31911  0.33070  1.98841
        4      100  0.0001  2.25847  0.28572  1.97274
        4      110  0.0001  2.09784  0.26564  1.83220
        5      120  0.0001  2.25548  0.30922  1.94626
        5      130  0.0001  2.22602  0.31158  1.91443
        6      140  0.0001  2.14396  0.32211  1.81185
        6      150  0.0001  1.95564  0.27404  1.68159
        7      160  0.0001  1.79976  0.27300  1.52675
```

Weight files are generated on every 200 iterations by default. (e.g. `result/model_iter_400`
as a weight file after 400 iterations) You can monitor loss and stop training at anytime and resume with option --resume.

A generated weight file can be used as model file for ssd_object_detector.py by setting -model_path parameter as path to the weight file.

Pretrained Models

Hand Detection

You can use pretrained hand detection models.

```
roslaunch jsk_perception sample_ssd_hand_detector.launch gpu:=0
```

Reference

- ChainerCV SSD code
- Image Labelling tools

Author

Yuki Furuta <furushchev@jsk.imi.i.u-tokyo.ac.jp>
Yusuke Niitani <niitani@jsk.imi.i.u-tokyo.ac.jp>
jsk_recognition welcomes your contributions. We maintain this project on Github.

15.1 Issues
If you find a problem, go to Issues page and report it.

15.2 Pull request
If you want to fix a bug or add new feature, go to Pull requests page and make PR.

15.3 Contribution guideline

• Letter size of arguments in launch file
  – Small letter arguments:
    • Users can change them on command line.
    • Example: roslaunch jsk_perception sample_ssd_object_detector.launch gpu:=0
  – Capital letter arguments:
    • Users are not recommended to change them on command line.
    • Users should change their values in a launch file with <include> tag.
ROS nodes for checkerboard detection and pose estimation.

### 16.1 checkerboard_calibration

![Image of checkerboard calibration process](image-url)
16.1.1 What is this?

A ROS node for calibrating camera intrinsic parameter by using checker board. You can find marker pattern in jsk-ros-pkg/calibboard_sheet. This node only supports chess board pattern.

You can click mouse middle button at either image window and then get calibration result as ROS_INFO. Also, you can clear current calibration data by clicking mouse right button at either window.

Please consider using camera_calibration package.

16.1.2 Subscribing Topics

- image (sensor_msgs/Image)
  Input image for calibration. The image assumed to be image_raw, which means it is not rectified.
- Image (sensor_msgs/Image)
  Deprecated.

16.1.3 Publishing Topics

None.

16.1.4 Parameters

- display (Int, default: 1)
  Set 1 to show viewer (not image_view but OpenCV window).
- grid_size_x (Int, required)
- grid_size_y (Int, required)
  The number of grids along x and y axis.
- rect_size_x (Float, required)
- rect_size_y (Float, required)
  Size of checkerboard in meters.

16.1.5 Sample

roslaunch checkerboard_detector sample_checkerboard_calibration.launch
16.2 checkerboard_detector

checkerboard_detector is an executable to detect checker board. You can find marker pattern in jsk-ros-pkg/calibboard_sheet.

16.2.1 Subscribing Topics

- `image` (sensor_msgs/Image)
- `camera_info` (sensor_msgs/CameraInfo)

Input image and camera info. Intrinsic camera parameter is acquired from camera_info.

- `Image` (sensor_msgs/Image)
- `CameraInfo` (sensor_msgs/CameraInfo)

These topics are deprecated.
16.2.2 Publishing Topics

- **ObjectDetection** (posedetection_msgs/ObjectDetection)
- **objectdetection_pose** (geometry_msgs/PoseStamped)
  
  Pose of checkerboard in posedetection_msgs/ObjectDetection and geometry_msgs/PoseStamped.
- **corner_point** (geometry_msgs/PointStamped)
  
  Corner points.
- **polygons** (jsk_recognition_msgs/PolygonArray)
  
  Publish checkerboard as jsk_recognition_msgs/PolygonArray. It is useful to visualize in rviz.
- **debug_image** (sensor_msgs/Image)
  
  Debug image showing detected checkerboard.

16.2.3 Parameters

- **display** (Int, default: 0)
  
  Set 1 to enable debug view (not image_view but OpenCV window). Note that debug_image is published even when this parameter is set to 0.
- **board_type** (String, default: chess)
  
  Type of marker. chess, circle, circles, acircle and acircles are supported. circle and circles are the same, and also acircle and acircles are.
- **rect%d_size_x** (Float, required)
- **rect%d_size_y** (Float, required)
  
  Size of checkerboard in meters, where %d means index of checker board starting from 0.
- **grid%d_size_x** (Int, required)
- **grid%d_size_y** (Int, required)
  
  The number of grids along x and y axis.
- **type%d** (String, default: checker%dx%d)
  
  Name of checkerboard written to objects/type field in ObjectDetection topic.
- **maxboard** (Int, default: -1)
  
  Maximum number of checkerboard. -1 means infinity, so this node will detect as much as possible.
- **use_P** (Bool, default: false)
  
  By default, use camera matrix (K) and unrectified image (image_raw). If you use rectified image (image_rect), use_P should be true.
- **invert_color** (Bool, default: false)
  
  Invert white and black before searching cross points or circles.
- **message_throttle** (Int, default: 1)
  
  Finding checkerboards every message_throttle images.
• `frame_id(String, default: "")`

Frame ID written to `ObjectDetection` topic header. If empty string is specified, `frame_id` of `image` topic will be used.

• `queue_size(Int, default: 1)`

• `publish_queue_size(Int, default: 1)`

Size of queue of subscriber is `queue_size`, publisher is `publish_queue_size`.

• `axis_size(Float, default: 0.05)`

• `circle_size(Int, default: 6)`

For setting displayed marker size. Set `circle_size` in [pixel]. Set `axis_size` in [m].

• `verbose(Int, default: 1)`

Output information about input image, number of detected checker board and elapsed time as `ROS_INFO` if this parameter is greater than 0.

### 16.2.4 Sample

```
roslaunch checkerboard_detector sample_checkerboard_detector.launch
```

### 16.2.5 Trouble Shooting

• Q. Estimated checker board pose is not correct

  A. First check debug image and all the detected corner points correctly superimposed on camera view.

    – If the detected corner points is **not correct**, you need to modify checker board grid size (`grid_size` parameters).

    – If the detected corner points is **correct**, confirm checker board size (`rect_size` parameters) and intrinsic camera parameter is calibrated well.

• Q. How many number of grids better?

  A. I strongly recommend to choose oddxeven or evenxodd. Because if you choose oddxodd or evenxeven, detector will have two potential poses.
16.3 objectdetection_tf_publisher

16.3.1 What is this?

Subscribe posedetection_msgs/ObjectDetection message and broadcast TF.

16.3.2 Subscribing Topics

- ObjectDetection(posedetection_msgs/ObjectDetection)
  Input object pose.

16.3.3 Publishing Topics

- /tf(tf2_msgs/TFMessage)
  Object transform.
16.3.4 Parameters

- `~use_simple_tf`(Bool, default: False)
  Whether to use tf.TransformBroadcaster or not. If false, this node will use `dynamic_tf_publisher`

Parameters below will be enabled only when `~use_simple_tf` is False.

- `~checker_board_params/header_frame`(String, required)
  Parent frame ID.

- `~frame_id`(String, default: object)
  Child frame ID.

- `~checker_board_params/position_x`(Float, default: 0.0)
- `~checker_board_params/position_y`(Float, default: 0.0)
- `~checker_board_params/position_z`(Float, default: 0.0)
- `~checker_board_params/orientation_x`(Float, default: 0.0)
- `~checker_board_params/orientation_y`(Float, default: 0.0)
- `~checker_board_params/orientation_z`(Float, default: 0.0)
- `~checker_board_params/orientation_w`(Float, default: 0.0)
  Object pose.

16.3.5 Sample

```bash
roslaunch checkerboard_detector sample_objectdetection_tf_publisher.launch
```
16.4  objectdetection_transform_echo

16.4.1 What is this?

Subscribe posedetection_msgs/ObjectDetection messages and get transform between them.
See also: objectdetection_tf_publisher.py

16.4.2 Subscribing Topics

- detection1(posedetection_msgs/ObjectDetection)
- detection2(posedetection_msgs/ObjectDetection)

Input object pose. This node will calculate pose from detection2 to detection1 at frame_id frame.

16.4.3 Publishing Topics

- pose(geometry_msgs/PoseStamped)

Transform between input poses.
16.4.4 Parameters

- `frame_id` (String, default: "")
  Frame ID of pose topic.

16.4.5 Sample

```
roslaunch checkerboard_detector sample_objectdetection_transform_echo.launch
```
ROS nodes to compute local features of 2-D images. It supports SIFT, SURF, STAR, BRISK.

### 17.1 Extract SIFT and visualize keypoints.

Sample is available at `imagesift/sample/sift_keypoints.py`

```python
# load image
>>> from scipy.misc import lena
>>> img = lena()  # gray-scale image

>>> import imagesift

# extract sift keypoints
>>> frames, desc = imagesift.get_sift_keypoints(img)

# draw keypoints on the image
>>> out = imagesift.draw_sift_frames(img, frames)

# view image with opencv2 window
>>> import cv2
>>> cv2.imshow('sift image', out)
>>> cv2.waitKey(0)
```

You will get output like:
17.2 ImageSift

Extract sift features from input image.

17.2.1 Subscribing Topic

- /image (sensor_msgs/Image)
  Input image. This triggers output Feature0D.
- /camera_info (sensor_msgs/CameraInfo)
  Input camera_info.

17.2.2 Publishing Topic

- /Feature0D (posedetection_msgs/Feature0D)
  This appears with input image.
- /ImageFeature0D (posedetection_msgs/ImageFeature0D)
  This appears with both inputs image and camera_info.
17.2.3 Parameters

- `~image_transport` (String, default: raw)
  
  Set compressed or theora to subscribe compressed images

17.2.4 Run

You can run executable like below:

```bash
$ rosrun imagesift imagesift
```

To subscribe compressed image, run executable like below:

```bash
$ rosrun imagesift imagesift _image_transport:=compressed
```

17.2.5 Sample

```bash
$ roslaunch imagesift imagesift_sample.launch
```
ROS nodes and nodelets for 2-D image perception.

18.1 Bag of Features for Object Recognition

18.1.1 Tools

- scripts/create_sift_dataset.py
  - extract SIFT descriptor features from images.
  - the input data path format should be like below:

```plaintext
dataset
  +-- champion_copper_plus_spark_plug
      |    +-- img0000.jpg
```

(Demo of recognizing oreo snack from novel image input.)
• scripts/create_bof_dataset.py
  – extract BoF from descriptor features.
  – extract BoF Histogram from descriptor features.
• scripts/sklearn_classifier_trainer.py
  – train classifier in scikit-learn with specified dataset and classifier model.

18.1.2 Example

```bash
$ roscd jsk_perception/data
# download sample data
$ sudo pip install gdown
$ gdown "https://drive.google.com/uc?id=0B9P1L--7Wd2vNm9zMTJWOGxobkU&export=download"
→ -O 20150428_collected_images.tgz
# create descriptors dataset
$ tar zxf 20150428_collected_images.tgz
$ rosrun jsk_perception create_sift_dataset.py 20150428_collected_images
# extract Bag of Features & its histogram
$ rosrun jsk_perception create_bof_dataset.py extract_bof 20150428_collected_images_sift_feature.pkl.gz
$ rosrun jsk_perception create_bof_dataset.py extract_bof_histogram 20150428_collected_images_sift_feature.pkl.gz
   `rospack find jsk_perception`/trained_data/apc2015_sample_bof.pkl.gz
   -O `rospack find jsk_perception`/trained_data/apc2015_sample_bof_hist.pkl.gz
# train classifier
$ rosrun jsk_perception sklearn_classifier_trainer.py
   `rospack find jsk_perception`/trained_data/apc2015_sample_bof_hist.pkl.gz
   -O `rospack find jsk_perception`/trained_data/apc2015_sample_clf.pkl.gz
# run for novel image
$ roslaunch jsk_perception sample_bof_object_recognition.launch
# check the result
$ rostopic echo /sklearn_classifier/output
data: oreo_mega_stuf
...
18.2 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.2. 73b2 kitchen dataset results
18.3 Train FCN Depth Prediction

This page shows how to train FCN Depth Prediction with your own dataset.

FCN Depth Prediction is a neural network model used for depth prediction with semantic segmentation.

Any size of image can be applied to this network as long as your GPU has enough memory.

18.3.1 Available Dataset Class

DepthPredictionDataset (imported from jsk_recognition_utils.datasets)

This class assumes the following directory structure.

```
path_to_awesome_dataset
|-- fool
| |-- train
|   |-- bar1
|     |-- depth.npz
|     |-- depth_gt.npz
|     |-- image.png
|     |-- label.png
```

(continues on next page)
18.3.2 Arguments

- **--dataset_dir** (string, default: $(rospack find jsk_perception)/learning_datasets/human_size_mirror_dataset)
  Directory name which contains dataset for training and validation.
- **--model** (string, default: FCN8sDepthPredictionConcatFirst)
  Model name. Currently, FCN8sDepthPredictionConcatFirst is supported.
- **--gpu** (int, default: 0)
  GPU id. -1 means CPU mode, but we recommend to use GPU for much faster computing.
- **--batch_size** (int, default: 1)
  Number of images used simultaneously in each iteration.
  You should decrease this number when you face memory allocation error.
- **--epoch** (int, default: 100)
  Stop trigger for training.
- **--out** (string, default: ${ROS_HOME}/learning_logs/<timestamp>)
  Output directory name.

18.3.3 Output

All these files will be automatically generated under <out_dir>.

- batch_size.txt
- dataset.txt
- log.json
- loss_plot.png
- model.txt
- model_snapshot.npz
- network_architecture.dot

18.3.4 Usage

```bash
rosrun jsk_perception train_fcn_depth_prediction.py [ARGS]
```
18.4 AddMaskImage

Add two mask image into one mask image.

18.4.1 Subscribing Topic

• ~input/src1(sensor_msgs/Image)
• ~input/src2(sensor_msgs/Image)
  Input mask images.

18.4.2 Publishing Topic

• ~output(sensor_msgs/Image)
  Added mask image.
18.4.3 Parameters

- ~approximate_sync (Bool, default: false)
  Approximately synchronize ~input/src1 and ~input/src2 if it's true.

18.4.4 Sample

```bash
roslaunch jsk_perception sample_add_mask_image.launch
```

18.5 alexnet_object_recognition.py

18.5.1 What is this?

Recognize object with Alex net by resizing input image to 227 x 227. This node requires pretrained chainer model. For training Alex net, please refer to chainer imagenet example

18.5.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image.
18.5.3 Publishing Topic

- ~output (jsk_recognition_msgs/ClassificationResult)
  Classification result of input image.
- ~debug/net_input (sensor_msgs/Image)
  Resized image to 227 x 227.

18.5.4 Parameters

- ~gpu (Int, Default: -1)
  GPU id. -1 represents CPU mode.
- ~target_names (List of String, Required)
  Target names for classification.
- ~model_name (String, Required)
  Currently alex or alex_batch_normalization is only supported. See models in $(rospack find jsk_recognition_utils)/python/jsk_recognition_utils/chainermodels.
- ~model_file (String, Required)
  Trained model file.
- use_mask (Bool, Default: False)
  If true, topic ~input/mask is enabled.
- ~approximate_sync (Bool, Default: False)
  Use approximate synchronization policy.
- ~queue_size (Int, Default: 10)
  Queue size for synchronization.
- ~slop (Float, Default: 0.1)
  Slop for approximate sync.

18.5.5 Example

roslaunch jsk_perception sample_alexnet_object_recognition.launch
18.6 apply_context_to_label_probability

18.6.1 What is this?

Apply context as label candidates to label probability image. The probabilities of labels other than the candidates are set as 0.

18.6.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Label probability image.
- ~input/candidates (jsk_recognition_msgs/LabelArray)
  Array of candidates label to update ~candidates.

18.6.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Label probability image after the context is applied.
- ~output/label (sensor_msgs/Image)
  Label image after the context is applied.
18.6.4 Advertising Service

- ~update_candidates (jsk_recognition_msgs/SetLabels)
  Update ~candidates like you do so with dynamic_reconfigure. (List of string is not supported in current dynamic_reconfigure)

18.6.5 Parameters

- ~candidates (List of String, default: [])
  If no candidates are provided, this node does nothing and just transports the input msg.
- ~candidates_fixed (List of String, default: [])
  Candidates that never changed by services or topics.

18.6.6 Sample

```bash
roslaunch jsk_perception sample_apply_context_to_label_probability.launch
```

18.7 ApplyMaskImage

Apply mask image to original image and visualize it. It’s a utility to visualize mask image.
18.7.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Original image.
- ~input/mask (sensor_msgs/Image)
- ~input/camera_info (sensor_msgs/CameraInfo, optional)
  Mask image.

18.7.2 Publishing Topic

- ~output (sensor_msgs/Image)
  Masked image. The image is clipped by bounding box of mask image and filtered by the mask. The region not specified by mask image is filled by 0.
- ~output/mask (sensor_msgs/Image)
  Clipped mask image. The image is clipped by bounding box of mask image.
- ~output/camera_info (sensor_msgs/CameraInfo)
  camra_info whose roi is bounding box of mask image.

18.7.3 Parameters

- ~approximate_sync (Bool, default: false)
  Approximately synchronize inputs if it’s true.
- ~negative (Bool, default: false)
  Flip the max region of mask image or not.
- ~clip (Bool, default: true)
  Clip the max region of mask image or not.
- ~negative/before_clip (Bool, default: true)
  This decides order of negative and clip. If true, negative process is before clipping.
- ~queue_size (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- ~cval (Int, default: 0)
  Used to fill image before masking with input image and mask.
- ~mask_black_to_transparent (Bool, default: false)
  Change black region of mask image to transparent and publish RGBA8 image as ~output if its’ true.
- ~use_rectified_image (Bool, default: true)
  Set camera_info.roi.do_rectify of the output to true if it’s true.
18.7.4 Sample

```bash
roslaunch jsk_perception sample_apply_mask_image.launch
```

18.8 aws_auto_checkin_app.py

18.8.1 What is this?

Face recognition using Amazon Rekognition, see `SearchFacesByImage` and Auto Check-In App for more info.

18.8.2 Subscribing Topic

- `image/compressed(sensor_msgs/CompressedImage)`
  
  Input compressed image. (only when `~image_transport` is compressed.)
  
  This topic name is resolved from `image`.

- `image(sensor_msgs/Image)`
  
  Input raw image. (only when `~image_transport` is `raw`.)

- `face_roi(opencv_apps/FaceArrayStamped)`
  
  Rectangles on the face of input image. Use ROI value.

  ```
  msg.faces[].face.x : X coordinates of the center of the face image in the `~image`
  msg.faces[].face.y : Y coordinates of the center of the face image in the `~image`
  msg.faces[].face.width : Width of the face image
  msg.faces[].face.height : Height of the face image
  ```
18.8.3 Publishing Topic

- **face_name** (opencv_apps/FaceArrayStamped)
  Publish recognized face name as well as face image. The face.{x,y,width,height} corresponds to input face_roi, that means x, y is the center of face rectangle.
- **~output/rects** (jsk_recognition_msgs/RectArray)
  Rectangles of matched faces.
- **~output/class** (jsk_recognition_msgs/ClassificationResult)
  Detected face class labels and probabilities.

18.8.4 Parameters

- **~use_window** (Bool, default: False)
  Show input image on the window, if it is true.
- **~approximate_sync** (Bool, default: True)
  Approximately synchronize inputs if it’s true.
- **~queue_size** (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- **~slop** (Float, default: 1.0)
  Maximum allowed time for approximate synchronization in [sec].
  This parameter is enabled only when ~approximate_sync is True.
- **~classifier_name** (String, default: rospy.get_name())
  Classifier name written to classifier field of ~output/class.
- **~image_transport** (String, default: compressed)
  Set raw to subscribe raw image.
- **~env_path** (String, default: env.json)
  Json file for environment variables.

  See DynamoDB for how to obtain DynamoDB Table name and Rekognition for CollectionId.

```json
{
  "Region": "%%REGION%%",
  "DynamodbTable": "%%DYNAMODB_TABLE%%",
  "CollectionId": "%%COLLECTION_ID%%",
  "FaceAreaThreshold": 1e4,
  "FaceSimilarityThreshold": 90,
  "MaxFaces": 1
}
```

- **~aws_credentials_path** (String, Default: aws.json)
  Json file for aws access key and secret key to run AWS Rekognition. See Set up an AWS account and create an IAM user for how to obtain keys.
```json
{
  "aws_access_key_id" : "####################",
  "aws_secret_access_key" : "********************"
}
```

### 18.8.5 Sample

```bash
roslaunch jsk_perception sample_aws_auto_checkin_app.launch use_aws_face_detection:=true
```

If `use_aws_face_detection` is true, AWS face detection API is used. Otherwise OpenCV face detection is used. AWS face detection is more accurate.

For JSK user, Download `env.json` file and `aws.json` from Gdrive and put these under `/tmp` directory to run sample code.

To add new people to face database, add face image file to Amazon S3, `auto-check-in-gapp-register... buckets`

For more info about managing the face collection, see Searching faces in a collection.
18.9 aws_detect_faces.py

18.9.1 What is this?

Detect faces within an images using Amazon Rekognition. Please refer to DetectFaces.
18.9.2 Subscribing Topic

- `image/compressed(sensor_msgs/CompressedImage)`  
  Input image. This topic name is resolved from `image`.

18.9.3 Publishing Topic

- `~faces(opencv_apps/FaceArrayStamped)`  
  Detected face positions, facial landmarks, emotions, presence of beard, sunglasses, and so on.
- `~poses(geometry_msgs/PoseArray)`  
  Pose of the face as determined by its pitch, roll, and yaw. The pose is published in Quaternion, to reconstruct Roll, Pitch, Yaw, use following codes.

```python
def cb(msg):
    for pose in msg.poses:
        angles = euler_from_quaternion([pose.orientation.x, pose.orientation.y, pose.orientation.z, pose.orientation.w])
        print("roll: {}, pitch: {}, yaw: {}",format(angles[0]*180/math.pi, angles[1]*180/math.pi, angles[2]*180/math.pi))

if __name__ == '__main__':
    rospy.init_node('pose_example')
    rospy.Subscriber('/aws_detect_faces/poses', PoseArray, cb)
    rospy.spin()
```

- `~attributes(jsk_recognition_msgs/ClassificationResult)`  
  Attributes of the face, such as emotions, presence of beard, sunglasses, and so on, with confidence.
- `~landmarks(jsk_recognition_msgs/PeoplePoseArray)`  
  The location of landmarks on the face.
- `~output(sensor_msgs/Image)`  
  Visualization image of detected faces.

18.9.4 Parameters

- `~aws_credentials_path(String, Default: aws.json)`  
  Json file for aws access key and secret key to run AWS Rekognition. see Set up an AWS account and create an IAM user for how to obtain keys.

```json
{
    "region" : "xxxxxxxxxxxxxxxxxxxxxxxxx",
    "aws_access_key_id" : "xxxxxxxxxxxxxxxxxxxx",
    "aws_secret_access_key" : "xxxxxxxxxxxxxxxxxxxx"
}
```
• `attributes` (Bool, Default: ALL)
  Facial attributes you want to be returned. Set ALL to return all attributes or DEFAULT to return `BoundingBox, Confidence, Pose, Quality, and Landmarks`. See https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/doc_source/API_DetectFaces.md#request-parameters more info.

• `use_window` (Bool, Default: False)
  Set true to display a window including detected face and attributes.

• `buff_size` (Int, Default: 614400)
  Buffer size for image subscription. The default buff_size of rospy.Subscriber is 65536, which does not process latest incoming image even if queue_size=1. See https://answers.ros.org/question/220502/image-subscriber-lag-despite-queue-1/ for more detail.

• `always_subscribe` (Bool, Default: True)
  Set true to process even if not one subscribing.

18.9.5 Example

```
roslaunch jsk_perception sample_aws_detect_faces.launch
```

Optional Arguments:
- `attributes` (default "ALL"): set either ALL or DEFAULT for returning attributes
- `use_usb_cam` (default "false"): set true to use USB camera image as input
- `use_window` (default "true"): set false if you do not want to display window

For JSK user, Download `aws.json` file from Gdrive and put this under `/tmp` directory to run sample code.

18.10 BackgroundSubstraction

18.10.1 What is this?

Extract moving foreground from static background.

18.10.2 Subscribing Topic

- `image` (sensor_msgs/Image)
  Input image.
18.10.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Output mask image regarded as moving foreground.

18.10.4 Sample

```bash
roslaunch jsk_perception sample_background_subtraction.launch
```

18.11 Bing

18.11.1 What is this?

Compute objectness in a image frame, and publishes object proposals.

**Note:** This node is supported only when OpenCV3 is installed.

18.11.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Raw image.

18.11.3 Publishing Topic

- ~output (jsk_recognition_msgs/RectArray)
  Object proposals.
- ~output/objectness (sensor_msgs/Image, encoding: 32FC1)
  Float image which describes objectness.
18.11.4 Parameters

- ~score_threshold (Double, default: 0.0)
  Score threshold for objectness values. Max value is 1.0.
- ~max_num (Int, default: -1.0)
  Maximum number of returning bounding boxes. -1.0 means returning all bounding boxes.

18.11.5 Sample

```
roslaunch jsk_perception bing.test
```

18.11.6 Reference

@inproceedings{BingObj2014,
  title={BING: Binarized Normed Gradients for Objectness Estimation at 300fps},
  author={Ming-Ming Cheng and Ziming Zhang and Wen-Yan Lin and Philip H. S. Torr},
  booktitle={IEEE CVPR},
  year={2014},
}

18.12 binpack_rect_array.py

Pack `jsk_recognition_msgs/RectArray` to one image by solving 2-D binpack problem.
18.12.1 Subscribing Topics

• `~input (sensor_msgs/Image)`
  Input image

• `~input/rect_array (jsk_recognition_msgs/RectArray)`
  Input array of bounding box regions

18.12.2 Publishing Topics

• `~output (sensor_msgs/Image)`
  Output image

18.12.3 Parameters

• `~approximate_sync (default: false)`
  Synchronize `~input` and `~input/rect_array` with approximate time sync.

18.12.4 Sample

```
roslaunch jsk_perception sample_binpack_rect_array.launch
```

18.13 BlackHat

See MorphologicalOperator
18.14 BlobDetector

Detect blob of binary image and output label of it. The implementation is based on Imura’s implementation.

18.14.1 Subscribing Topic

- `~input` (sensor_msgs/Image)
  
  Input image. It should be single channel.

18.14.2 Publishing Topic

- `~output` (sensor_msgs/Image)
  
  Label image. Each pixel value means the label which the pixel should belong to and 0 means the pixel is masked (black in `~input` image)

18.14.3 Parameter

- `~min_size` (Integer, default: 10)
  
  Minimum size of blob
18.14.4 Sample

```bash
roslaunch jsk_perception sample_blob_detector.launch
```

18.15 bof_histogram_extractor.py

### 18.15.1 What is this?

Extract Bag-of-Features histogram.

For other information about object recognition using Bag-of-Features, see [here](#).

### 18.15.2 Subscribing Topic

- `~input (posedetection_msgs/Feature0D)`
  
  Feature data.

- `~input/label (sensor_msgs/Image)`
  
  Label image with which features will be extracted.
18.15.3 Publishing Topic

- `~output (jsk_recognition_msgs/VectorArray)`
  Extracted feature data.

18.15.4 Parameters

- `~bof_data (String, required)`
  Path to Bag-of-Features data.
  The file contains `jsk_recognition_utils.feature.BagOfFeatures` object, and it should be pickled and gzipped.
- `~queue_size (Int, default: 10)`
  Maximum number of messages which subscriber keeps in the queue.
- `~approximate_sync (Bool, default: False)`
  Whether to use ApproximateSync for input topics.
- `~slop (Float, default: 0.1)`
  Number of seconds you allow about the difference of timestamp.
  This parameter is enabled only when `~approximate_sync` is True.

18.15.5 Sample

```
roslaunch jsk_perception sample_bof_object_recognition.launch
```

18.16 BoundingBoxToRect

Convert `jsk_recognition_msgs/BoundingBoxArray` to `jsk_recognition_msgs/RectArray`.
18.16.1 Subscribing Topics

- ~input (jsk_recognition_msgs/BoundingBoxArray)
  Input bounding boxes.
- ~input/info (sensor_msgs/CameraInfo)
  CameraInfo to project bounding boxes.
- ~internal (jsk_recognition_msgs/BoundingBoxArrayWithCameraInfo)
  Internal topic to synchronize timestamp of ~input and ~input/info.

18.16.2 Publishing Topics

- ~output (jsk_recognition_msgs/RectArray)
  Projected 2-D bounding box.
- ~internal (jsk_recognition_msgs/BoundingBoxArrayWithCameraInfo)

18.16.3 Parameters

- ~approximate_sync (Bool, default: false)
  Approximately synchronize ~input/src1 and ~input/src2 if it's true.
- ~queue_size (Int, default: 100) How many messages you allow about the subscriber to keep in the queue.

18.16.4 Example
18.17 BoundingObjectMaskImage

18.17.1 What is this?

Publish mask that is bounding object mask of the input mask.

18.17.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Input mask image.

18.17.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Bounding object mask image.

18.17.4 Parameters

None.
18.17.5 Sample

```bash
catkin_make run
roslaunch jsk_perception sample_bounding_object_mask_image.launch
```

18.18 BoundingRectMaskImage

18.18.1 What is this?

Publish mask that is bounding rect mask of the input mask.

18.18.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input mask image.

18.18.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  Bounding rect mask image.
18.18.4 Parameters

None.

18.18.5 Sample

```bash
roslaunch jsk_perception sample_bounding_rect_mask_image.launch
```

18.19 classification_node.py

![Image of a person with text]

human: 55.66% glass: 16.98% cup: 16.71% kettle: 10.63%

The ROS node for Classification with CLIP.
18.19.1 System Configuration

This node requires to work with the Docker Container for inference. Please build the container at first following Setup instruction.

Prerequisite

This node requires NVIDIA GPU and more than 4GB GRAM to work properly. You have to install nvidia-container-toolkit for using GPU with docker. Please follow official instruction.

Build the docker image

You have to build the docker image of OFA

```
roscd jsk_perception/docker
make
```
18.19.2 Subscribing topic

• ~image(sensor_msgs/Image)
  Input image

18.19.3 Publishing topic

• ~result(jsk_recognition_msgs/ClassificationResult)
  Classification result
• ~result/image(sensor_msgs/Image)
  Images used for inference
• ~visualize(std_msgs/String)
  Classification result to visualize

18.19.4 Action topic

• ~inference_server/goal(jsk_recognition_msgs/ClassificationTaskActionGoal)
  Classification request with custom categories and image
• ~inference_server/result(jsk_recognition_msgs/ClassificationTaskActionResult)
  Classification result of ~inference_server/goal

18.19.5 Parameters

• ~host(String, default: localhost)
  The host name or IP of inference container
• ~port(Integer, default: 8080)
  The HTTP port of inference container

18.19.6 Dynamic Reconfigure Parameters

• ~queries(string, default: human;kettle;cup;glass)
  Default categories used for subscribing image topic.

Run inference container on another host or another terminal

In the remote GPU machine,

```
cd jsk_recognition/jsk_perception/docker
./run_jsk_vil_api clip --port (Your vacant port)
```

In the ROS machine,

```
roslaunch jsk_perception classification.launch port:=(Your inference container port)
  --host:=(Your inference container host) CLASSIFICATION_INPUT_IMAGE:=(Your image topic)
  --name) gui:=true
```

(continues on next page)
Run both inference container and ros node in single host

roslaunch jsk_perception classification.launch run_api:=true CLASSIFICATION_INPUT_IMAGE:=(Your image topic name) gui:=true

18.20 Closing

See MorphologicalOperator

18.21 ColorHistogramLabelMatch

Compute similar region of image to specified histogram based on superpixels image.

18.21.1 Input Topic

- ~input/histogram(jsk_recognition_msgs/ColorHistogram)
  Reference histogram.
- ~input(sensor_msgs/Image)
  Input image. This image should be bgr8 or rgb8 image.
- ~input/label(sensor_msgs/Image)
  Label of ~input image. Label image should be int32 image.
- ~input/mask(sensor_msgs/Image)
  Mask image of ~input image. Only masked region is taken into account.
18.21.2 Publishing Topic

- `~output/extracted_region(sensor_msgs/Image)`
  Result of correlation computation as mask image.
- `~output/coefficient_image(sensor_msgs/Image)`
  Result of correlation computation as float image.
- `~debug(sensor_msgs/Image)`
  Debug image

18.21.3 Parameters

- `~coefficient_method(Int, Default: 0)`
  Method to compute coefficient.
  Choose from `correlancy (0), chi_squared (1), intersect (2), bhattacharyya (3), EMD_Manhattan (4) and EMD_Euclid (5).`
  This parameter can be changed by dynamic_reconfigure.
- `~max_value(Int, Default: 255)`
- `~min_value(Int, Default: 0)`
  Maximum and minimum index of histogram
  These parameters can be changed by dynamic_reconfigure.
- `~masked_coefficient(Float, Default: 0.0)`
  Value to fill masked region
  This parameter can be changed by dynamic_reconfigure.
- `~threshold_method(Int, Default: 0)`
  Method to binalize coefficient image.
  Choose from `smaller_than (0), greater_than (1), otsu (2) and otsu_inv (3).`
  This parameter can be changed by dynamic_reconfigure.
- `~coef_threshold(Float, Default: 0.8)`
  Threshold used in binalization.
  This parameter can be changed by dynamic_reconfigure.
- `~use_mask(Default: false)`
  Do not use mask image if this parameter is false.
18.21.4 Sample

```bash
roslaunch jsk_perception sample_color_histogram_label_match.launch
```

18.22 ColorizeFloatImage

Colorize float image with heatmap gradation.

18.22.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Float(32FC[1-4]) image.

18.22.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  RGB8 image with heatmap gradation.
18.22.3 Parameters

• ~channel (int, default: 0)
  Choose channel to colorize. If value is default, first channel is selected.

18.22.4 Sample

```bash
roslaunch jsk_perception sample_color_float_image.launch
```
18.23 ColorizeLabels
Put colors on the subscribed topic of labeling result

### 18.23.1 Subscribing Topic

- `~input (sensor_msgs/Image)`

  Input datatype is `sensor_msgs/Image` but not for normal color image but for labeling result container.

### 18.23.2 Publishing Topic

- `output (sensor_msgs/Image)`

### 18.23.3 Parameters

None

### 18.23.4 Samples

**rosrun**

Launch color image publisher anything you like.

```bash
rosrun usb_cam usb_cam_node
```

Run the labeling node anything you like.

```bash
rosrun jsk_perception slic_super_pixels image:=/usb_cam/image_raw
```

Run this colorize_labels node with proper remapping.

```bash
rosrun jsk_perception colorize_labels ~input:=/slic_super_pixels/output
```
18.24 ConcaveHullMaskImage

18.24.1 What is this?

Publish mask that is computed concave hull from the input mask.

18.24.2 Subscribing Topic

• ~input (sensor_msgs/Image)
  Input mask image.

18.24.3 Publishing Topic

• ~output (sensor_msgs/Image)
  Concave hull mask image.

18.24.4 Parameters

• min_area (Float, default: 0)
  Minimum area of concave.
• max_area (Float, default: Image’s height * width)
  Maximum area of concave.
18.24.5 Sample

```
roslaunch jsk_perception sample_concave_hull_mask_image.launch
```

18.25 ConsensusTracking

18.25.1 What is this?

Publishing a ROI mask of the tracking object.

18.25.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Raw image.
- `~input/polygon (geometry_msgs/PolygonStamped)`
  Initial window of object to track.

18.25.3 Publishing Topic

- `~output/mask (sensor_msgs/Image)`
  A ROI mask of the tracking object.
- `~debug/image (sensor_msgs/Image)`
  Debug image which has visualized keypoints and rectangle of the tracking object in frame.

18.25.4 Parameters

- `~queue_size (Int, default: 10)`
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- `~approximate_sync (Bool, default: False)`
  Whether to use approximate for input topics.

18.25.5 Sample

```
roslaunch jsk_perception sample_consensus_tracking.launch
```
18.26 ContourFinder

18.26.1 @Deprecated

contour_finder in jsk_perception is deprecated.
Please use opencv_apps's find_contours(http://wiki.ros.org/opencv_apps).

18.27 ConvexHullMaskImage

18.27.1 What is this?

Publish mask that is computed convex hull from the input mask.

18.27.2 Subscribing Topic

- ~/input(sensor_msgs/Image)
  Input mask image.
18.27.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Convex hull mask image.

18.27.4 Parameters

None.

18.27.5 Sample

```bash
roslaunch jsk_perception sample_convex_hull_mask_image.launch
```

18.28 craft_node.py

18.28.1 What is this?

Detect text region from image. This is a ROS node for CRAFT: Character-Region Awareness For Text detection.

In order to use this feature, you need to install pytorch (pytorch >= 1.9.0 is recommended). However, python2 user is not supposed to be able to install that version of torch. For python2 users, download the appropriate wheel file in your environment from https://download.pytorch.org/whl/cu90/torch_stable.html (melodic) and install with pip as follow:

```bash
pip install --user torch-1.1.0-cp27-cp27mu-linux_x86_64.whl
pip install --user torchvision-0.3.0-cp27-cp27mu-manylinux1_x86_64.whl
```
18.28.2 Subscribing Topic

- \texttt{~input (sensormsgs/Image)}
  Raw image.

18.28.3 Publishing Topic

- \texttt{~output/cluster_indices (jsk_recognition_msgs/ClusterPointIndices)}
  Image indices of each detected text region.
- \texttt{~output/rects (jsk_recognition_msgs/RectArray)}
  Rectangles of detected texts.
- \texttt{~output/polygons (jsk_recognition_msgs/PolygonArray)}
  Polygons of detected texts. Unlike \texttt{RectArray}, this is output as a list of points on a 2D plane.

18.28.4 Parameters

- \texttt{~gpu (Int, default: 0)}
  GPU id. -1 represents CPU mode.
- \texttt{~model_path (String, required)}
  Pretrained model path.
- \texttt{~refine_model_path (String, required)}
  Pretrained link refiner model path.
- \texttt{~text_threshold (Double, default: 0.70)}
  Text confidence threshold.
- \texttt{~text_low_bound_score (Double, default: 0.40)}
  Text low bound score.
- \texttt{~link_threshold (Double, default: 0.1)}
  Link confidence threshold.
- \texttt{~mag_ratio (Double, default: 1.5)}
  Image magnification ratio.
- \texttt{~max_image_size (Int, default: 1280)}
  Max image size for inference.
18.28.5 Sample

```
roslaunch jsk_perception sample_craft_node.launch
```

18.29 deep_sort_tracker_node.py

18.29.1 What is this?

Tracking objects using target bounding box.

18.29.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Raw image.
- `~input/rects (jsk_recognition_msgs/RectArray)`
  Rectangles on an input image.
- `~input/class (jsk_recognition_msgs/ClassificationResult)`
  Class labels for each rectangles.

18.29.3 Publishing Topic

- `~output/labels (jsk_recognition_msgs/LabelArray)`
  Tracking id labels of each tracked objects.
- `~output/viz (sensor_msgs/Image)`
  Visualized image of recognition result.

18.29.4 Parameters

- `~gpu (Int, default: 0)`
  GPU id.
- `~target_labels (List of String, default: None)`
  If this param is specified, tracking only specified label class. If this value is None, use all label.
- `~pretrained_model (String, required)`
  Pretrained model path.
- `~approximate_sync (Bool, default: False)`
  Whether to use approximate for input topics.
- `~queue_size (Int, default: 100)`
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
• \texttt{\textasciitilde{slop}} (Float, default: 0.1)

How many seconds you allow about the difference of timestamp when you specify \texttt{\textasciitilde{approximate\_sync}}.

### 18.29.5 Sample

```bash
roslaunch jsk_perception sample_deep_sort_tracker.launch gpu:=0
```

### 18.30 depth\_image\_filter.py

#### 18.30.1 What is this?

Create a mask image from the distance of a depth image.

#### 18.30.2 Subscribing Topic

- \texttt{\textasciitilde{input}} (sensor\_msgs\_Image)

  Input depth image.

#### 18.30.3 Publishing Topic

- \texttt{\textasciitilde{output/mask}} (sensor\_msgs\_Image)

  Mask image.

#### 18.30.4 Parameters

- \texttt{\textasciitilde{threshold}} (Float, Default: 0.0)

  Mask threshold. Normally, pixels with a depth value farther than this value will be masked.

- \texttt{\textasciitilde{negative}} (Bool, Default: False)

  Extract mask or the negative.
18.30.5 Example

```bash
roslaunch jsk_perception sample_depth_image_filter.launch
```

18.31 DilateMaskImage

See MorphologicalOperator

18.32 draw_classification_result.py

18.32.1 What is this?

Publish an image topic on which `jsk_recognition_msgs/ClassificationResult` is visualized.

18.32.2 Subscribing Topic

- `~input (jsk_recognition_msgs/ClassificationResult)`
  Classification result.
- `~input/image (sensor_msgs/Image)`
  Image on which the result is visualized.

18.32.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  Image on which classification result is drew.
18.32.4 Example

```
roslaunch jsk_perception sample_draw_classification_result.launch
```

18.33 draw_rects

18.33.1 What is this?

Draw rectangles (and their classes if available) on image
If you want to draw non-ascii characters, use `draw_rects.py`. The sample program described below (`sample_draw_rects_for_non_ascii_labels.launch`) needs to build `jsk_recognition_utils` in order to install Japanese fonts.

### 18.33.2 Subscribing Topic

- **~input (sensor_msgs/Image)**
  Raw image.

- **~input/rects (jsk_recognition_msgs/RectArray)**
  Rectangles on an input image.

- **~input/class (jsk_recognition_msgs/ClassificationResult)**
  Class labels for each rectangles.
18.33.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  Image on which rectangles and classes are drawn.

18.33.4 Parameters

- `~approximate_sync (Bool, default: False)`
  Whether to use approximate for input topics.
- `~queue_size (Int, default: 100)`
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- `~use_classification_result (Bool, default: False)`
  Use `~input/class` as class labels if enabled.
- `~show_proba (Bool, default: True)`
  Show probability for each class labels if enabled. This option is valid when `~use_classification_result` is True.
- `~rect_boldness (Int, default: 2)`
  Boldness for each rectangles.
- `~font_path (String, default: "")`
  Font path. This value is only valid in `draw_rects.py`. Specify the font with the characters you want to display.
- `~label_size (Double, default: 1.0)`
  Text size for each class labels.
- `~label_boldness (Double, default: 2.0)`
  Boldness for each characters of class label texts.
- `~label_font (Enum[Int], default: FONT_HERSHEY_SIMPLEX)`
  Font for class labels.
- `~label_margin_factor (Double, default: 1.1)`
  Margin factor for class label background rectangle
- `~resolution_factor (Double, default: 2.0)`
  Factor for resolution of output image. When this option is set as 1.0, an output image has the same resolution as an input image.
- `~interpolation_method (Enum[Int], default: INTER_LANCZOS4)`
  Method for interpolation on input image resizing.
18.33.5 Example

```bash
roslaunch jsk_perception sample_draw_rects.launch
```

The sample to display Japanese is as follows.

```bash
roslaunch jsk_perception sample_draw_rects_for_non_ascii_labels.launch
```

18.34 DualFisheyeToPanorama

Generate panorama image by stitching dual-fisheye camera images. This is ROS wrapper of [drNoob13/fisheyeStitcher](https://github.com/drNoob13/fisheyeStitcher)

18.34.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Input dual-fisheye image.
18.34.2 Publishing Topic

- ~output (sensor_msgs/Image)
  Output panorama image.
  Output topic is published by image_transport plugins, so ~output/compressed topic is also available.

18.34.3 Parameters

- ~image_transport (String, default: raw)
  Image transport type for ~input topic.
  You can choose compressed and so on to subscribe ~input/compressed compressed image.

- ~light_compen (Bool, default: false)
  Light Fall-off Compensation. The outer edge of the fisheye image has low brightness.

- ~refine_align (Bool, default: false)
  Adjusts any discontinuities caused by objects with varying depth in the stitching boundaries. Note that this process takes a large CPU usage.

- ~fovd (Double, default: 195.0)
  Field of view of fisheye camera [degree].

- ~save_unwarped (Bool, default: false)
  Save unwarped (rectified) fisheye images under ~/.ros. These images are useful to update panorama parameter.

- ~mls_map_path (String, default: “”)
  Path to .yml.gz file which contains MLS grids information. How to generate this file is explained below.

- ~blend_image_height (Int, default: 1920)
  Image size to be used for panorama blending.

- ~blend_image_width (Int, default: 1920)
  Image size to be used for panorama blending.

- ~blend_param_p_wid (Int, default: 55)
  Parameters used for panorama blending.

- ~blend_param_p_x1 (Int, default: 75)
  Parameters used for panorama blending.

- ~blend_param_p_x2 (Int, default: 1775)
  Parameters used for panorama blending.

- ~blend_param_row_start (Int, default: 590)
  Parameters used for panorama blending.

- ~blend_param_row_end (Int, default: 1320)
  Parameters used for panorama blending.
• ~output_image_height (Int, default: 2000)
  Output image size
• ~output_image_height (Int, default: 2000)
  Output image size

### 18.34.4 Sample

Before running the sample, please `catkin build jsk_perception`.

```bash
catkin build jsk_perception
rosrun jsk_perception install_sample_data.py
rosbag play $(rospack find jsk_perception)/sample/data/insta360_air.bag --loop --clock
roslaunch jsk_perception sample_dual_fisheye_to_panorama.launch
```

Panorama blending will run with high resolution parameters by default. (blending with 3840x1920, output is 4000x2000) This configuration requires lots of CPU power. If you want to run with low resolution parameters:

```bash
roslaunch jsk_perception sample_dual_fisheye_to_panorama.launch resolution_mode:=low
```

### 18.34.5 Update panorama parameter

This section describes how to update the panorama parameter for generating a panorama image. Note that different cameras basically require different parameters, even if they are of the same type. However, if you’re lucky, you can use the parameters of another camera, so you can try that first.

Here is an example using insta360.

1. Unwarp and save the left and right fisheye images. The images are `$HOME/.ros/l_img_crop.jpg` and `$HOME/.ros/r_img_crop.jpg`

```bash
roslaunch jsk_perception sample_insta360_air.launch use_usb_cam:=true save_unwarped:=true
```

1. Annotate the corresponding points of the left and right images. We use `labelme` as a GUI tool. When annotating, it is easier to start with `l_img_crop.jpg`, which has a narrow angle of view than `r_img_crop.jpg`. `$HOME/.ros/l_img_crop.json` and `$HOME/.ros/r_img_crop.json` should be outputted.

```bash
sudo pip install labelme==4.5.7
cd $HOME
labelme $HOME/.ros/l_img_crop.jpg
labelme $HOME/.ros/r_img_crop.jpg
```
1. Save the correspondence points to a matlab file.

   `rosrun jsk_perception create_mls_correspondence.py`

1. Run `mls_rigid_example2.m` to generate the MLS grids file. Matlab can be installed from the University of Tokyo license. You also need to install the image processing toolbox and the parallel computing toolbox. Please check the official wiki.
1. Convert the MLS grid file to a yaml file so that OpenCV can read it.

```bash
rosrun jsk_perception mls_matlab2opencv.py
```

1. Compress the yaml file to gz and place it under jsk_perception.

```bash
gzip ~/.ros/fisheye_stitcher_grid_xd_yd_3840x1920.yml
mv ~/.ros/fisheye_stitcher_grid_xd_yd_3840x1920.yml.gz $(rospack find jsk_perception)/
→ config/fisheye_stitcher_grid_xd_yd_3840x1920.yml.gz
```

### 18.35 EdgeDetector

#### 18.35.1 @Deprecated

`edge_detector` in `jsk_perception` is deprecated.

Please use `opencv_apps`’s `edge_detector` (http://wiki.ros.org/opencv_apps).

### 18.36 ErodeMaskImage

See `MorphologicalOperator`.

### 18.37 extract_image_channel.py

#### 18.37.1 What is this?

**Example**

Extraction of Target Object Probability Image in Object Segmentation

Extract a specified channel from input image.
18.37.2 Subscribing Topic

- ~input(sensor_msgs/Image)
  
  Input image.

18.37.3 Publishing Topic

- ~output(sensor_msgs/Image)
  
  Image which is the extracted channel from input image.

18.37.4 Parameters

- ~channel (Int, default: -1)
  
  The channel which is extracted from the input image. If negative value channel is passed the extraction is skipped.

18.37.5 Sample

```bash
roslaunch jsk_perception sample_extract_image_channel.launch
```
18.38 face_pose_estimation.py

18.38.1 What is this?

Estimate people face pose / gender from RGB Image and PeoplePoseArray. Please refer to Hyperface.

18.38.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Input image.

- `~input/pose_2d (jsk_recognition_msgs/PeoplePoseArray)`
  
  Input people pose in 2D input image.

- `~input/pose (jsk_recognition_msgs/PeoplePoseArray)`
  
  Input people pose in 3D.
18.38.3 Publishing Topic

- `~output/pose(geometry_msgs/PoseArray)`
  Estimated face poses for each person.
- `~output/gender(jsk_recognition_msgs/ClassificationResult)`
  Estimated gender with confidence for each person.
- `~output/rects(jsk_recognition_msgs/RectArray)`
  Face rectangles in 2D input image used for cropping.

These 3 topics are need to be synchronized.

18.38.4 Parameters

- `~gpu (Int, Default: -1)`
  GPU id. -1 represents CPU mode.
- `~classifier_name (String, Default: rospy.get_name())`
  Name of this classifier.
- `~approximate_sync (Bool, Default: False)`
  Use approximate synchronization policy.
- `~queue_size (Int, Default: 100)`
  Queue size for synchronization.
- `~slop (Double, Default: 0.1)`
  Slop for approximate sync.
- `~face_padding (Double, Default: 0.0)`
  Padding size factor for face rectangles.
- `~face_threshold (Double, Default: 0.5)`
  Threshold for confidence of detected faces.

18.38.5 Example

```
roslaunch jsk_perception sample_face_pose_estimation.launch gpu:=0
```

18.39 fast_rcnn.py

18.39.1 What is this?

Publish an image with object bounding boxes, scores and labels.
### 18.39.2 Subscribing Topic

- **~input (sensor_msgs/Image)**
  Raw image.

- **~input/rect_array (jsk_recognition_msgs/RectArray)**
  Object location proposals.

### 18.39.3 Publishing Topic

- **~output/class (jsk_recognition_msgs/ClassificationResult)**
  Detected object class labels and probabilities.

- **~output/rect_array (jsk_recognition_msgs/RectArray)**
  Rects of detected objects.

### 18.39.4 Parameters

- **~model (String, required)**
  Network model name. (vgg_cnn_m_1024 or vgg16) vgg_cnn_m_1024 is small network and requires ~2GB GPU memory. vgg16 is large network and requires ~5GB GPU memory.

- **~gpu (Int, default: -1)**
  GPU ID. Negative value means CPU mode.

- **~classifier_name (String, default: rospy.get_name())**
  Classifier name written to classifier field of ~output/class.

- **~approximate_sync (Bool, default: False)**
  Whether to use approximate for input topics.

- **~queue_size (Int, default: 10)**
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.

- **~slop (Float, default: 0.1)**
  How many seconds you allow about the difference of timestamp when you specify ~approximate_sync.
18.39.5 Example

```
roslaunch jsk_perception sample_fast_rcnn.launch
```

18.40 fcn_depth_prediction.py

18.40.1 What is this?

Predict depth of transparent object in pixel-wise with Fully Convolutional Networks.
18.40.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Raw RGB image.
- ~input/depth (sensor_msgs/Image)
  Raw depth image.

18.40.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Output depth image. The value of each pixel is equal to ~output/depth_pred_raw in the region labeled as transparent, otherwise equal to ~input/depth. The image encoding is 32FC1.
- ~output/label (sensor_msgs/Image)
  Output label image. Each object is segmented according to param ~target_names.
- ~output/proba_image (sensor_msgs/Image)
  Probability image of each object predicted according to param ~target_names. If the number of classes including background is XX, then the image encoding is 32FCXX.
- ~output/depth_pred_raw (sensor_msgs/Image)
  Predicted whole depth image. This is used for generating ~output. The image encoding is 32FC1.

18.40.4 Parameters

- ~backend (String, Default: chainer)
  Framework for neural networks. Currently, only chainer is supported.
- ~gpu (Int, Default: -1)
  GPU id. -1 represents CPU mode.
- ~target_names (List of String, Required)
  Target names for classification.
- ~model_name (String, Required)
  Currently, fcn8s_depth_prediction and fcn8s_depth_prediction_concat_first are supported.
- ~model_file (String, Required)
  Saved .npz file for trained model.
- ~bg_label (Int, default: 0)
  Label value for background. This is used with rosparam ~proba_threshold
- ~proba_threshold (Float, default: 0.0)
  Threshold for labeling pixels as uncertain, and the uncertain region will be labeled as background with rosparam ~bg_label.
• `~queue_size` (Int, default: 10)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two input topics.

• `~approximate_sync` (Bool, default: False)
  Whether to use approximate for input topics.

• `~slop` (Float, default: 0.1)
  How many seconds you allow about the difference of timestamp. This is used only when param `~approximate_sync` is true.

18.40.5 Sample

```bash
roslaunch jsk_perception sample_fcn_depth_prediction.launch
```

18.41 FilterMaskImageWithSize

Filter mask image with the size of white region relative to image size and reference mask size.

18.41.1 Subscribing Topic

• `~input` (sensor_msgs/Image)
  Input mask.

• `~input/reference` (sensor_msgs/Image)
  Reference mask. Only subscribed if `~use_reference` is true.

18.41.2 Publishing Topic

• `~output` (sensor_msgs/Image)
  Filtered mask.

18.41.3 Parameters

• `~use_reference` (Bool, default: false)
  If true, `~input/reference` is subscribed, and rospams: `~min_relative_size` and `~max_relative_size` are enabled.

• `~min_size, ~max_size` (Float, default: 0,1)
  Size threshold of white region of input mask relative to image size.

• `~min_relative_size, ~max_relative_size` (Float, default: 0,1)
  Size threshold relative to the reference mask’s white region. Enabled with `~use_reference:=true`.
• \texttt{approximate\_sync} (Bool, default: false)
  Approximately synchronize topics if it is true.
• \texttt{queue\_size} (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.

18.41.4 Sample

```
roslaunch jsk_perception sample\_filter\_mask\_image\_with\_size\_launch
```

18.42 fisheye\_ray.py

18.42.1 What is this?

Subscribe a 2D point of an fisheye image and publish orientation of the point in 3D space.
18.42.2 Subscribing Topic

- `image` (sensor_msgs/Image)
  Input fisheye image.
- `clicked_point` (geometry_msgs/PointStamped)
  Index of the pixel of `image`.
  Only x and y field are used for computing orientation, and z field is ignored.
- `camera_info` (sensor_msgs/CameraInfo)
  Not used now.

18.42.3 Publishing Topic

- `~output` (geometry_msgs/PoseStamped)
  Orientation of `clicked_point` in 3D space.
- `~output_point` (geometry_msgs/PointStamped)
  3D point 1m distant from `~frame_id` origin, which expresses the orientation

18.42.4 Parameters

- `~frame_id` (String, default: fisheye)
  Frame ID of published messages.

18.42.5 Sample

```
roslaunch jsk_perception sample_fisheye_ray.launch
```
18.43 FisheyeToPanorama

This nodelet will publish Rectified or Panoramized Fisheye Image.
We recommend you to set scale factor as small as possible to reduce calculation.
This was tested with Prosilica GC2450C and NM30 lens.
Below pictures show image rectification system.
K = 341.0 (NM30 with Prosilica2450GC) ($\varphi =$ rad)
K = 5.7 (NM30 with Prosilica2450GC) ($\varphi =$ degree)
18.43.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image.

18.43.2 Publishing Topic

- ~output (sensor_msgs/Image)
  Rectified or Panoramized Image
- ~output_bilinear (sensor_msgs/Image)
  When ~use_panorama and ~simple_panorama are true, publish Panoramized Image

18.43.3 Parameters

- ~use_panorama (Bool, default: false)
  If true => publish Panorama View Image
  If false => publish Rectified View Image
- ~simple_panorama (Bool, default: false)
  This is effective only when ~use_panorama is true
  If true => show Simple Panorama View
  If false => show Calculated Panorama View
- ~degree (Float, default: 60.0)
  Field of view of output image.
  This parameter can be changed by `dynamic_reconfigure`.
- ~scale (Float, default: 0.2)
  Output image size.
  This parameter can be changed by `dynamic_reconfigure`.
- ~upside_down (Bool, default: false)
  Turn the image upside down.
  This parameter can be changed by `dynamic_reconfigure`.
- ~offset_degree (Float, default: 180.0)
  Offset degree for Panorama View.
  This parameter can be changed by `dynamic_reconfigure`. 
18.43.4 Sample

```
roslaunch jsk_perception sample_fisheye_to_panorama.launch
```

18.44 FlowVelocityThresholding

Apply thresholding to optical flow and creates mask image.

18.44.1 Subscribing Topic

- `~input/flows` (opencv_apps/FlowArrayStamped): Optical flow.
- `~input/camera_info` (sensor_msgs/CameraInfo): Camera info to get image size. Subscribed if `~use_camera_info` is true.

18.44.2 Publishing Topic


18.44.3 Parameters

- `~approximate_sync` (Bool, default: false): Approximately synchronize inputs if it’s true.
- `~queue_size` (Int, default: 100): How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- `~use_camera_info` (Bool, default: true): If true, output mask size is got from `~input/camera_info`.
18.44.4 Sample

```
roslaunch jsk_perception sample_flow_velocity_thresholding.launch
```

18.45 GaussianBlur

18.45.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Input image.

18.45.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Output transformed image.
18.45.3 Parameters

- `~kernel_size` (Integer, default: 11)
  Kernel size for blur transform. It should be odd number.
- `~sigma_x`, `~sigma_y` (Double, default: 10)
  Gaussian kernel standard deviation in X or Y direction.

18.45.4 Sample

```bash
roslaunch jsk_perception sample_gaussian_blur.launch
```

18.46 GrabCut

Input original image and seed of foreground/background to obtain foreground and background image by applying GrabCut algorithm.

![Sample image showing foreground and background seed masks and output images.](image_url)

18.46.1 Subscribing Topic

- `~input` (sensor_msgs/Image)
  Input image from which foreground/background will be extracted.
- `~input/foreground` (sensor_msgs/Image)
- `~input/background` (sensor_msgs/Image)
  Seed mask image defining foreground/background.
18.46.2 Publishing Topic

- ~output/foreground(sensor_msgs/Image)
- ~output/background(sensor_msgs/Image)
  Foreground/background image extracted from ~input.
- ~output/foreground_mask(sensor_msgs/Image)
- ~output/background_mask(sensor_msgs/Image)
  Foreground/background mask image extracted from ~input.

18.46.3 Parameters

- ~seed_pixel_policy(Int, default: 0)
  If 0, regard pixels in seed mask as absolute foreground/background.
  If 1, regard pixels in seed mask as probable foreground/background.
  This parameter can be changed by dynamic_reconfigure.

18.46.4 Sample

```bash
roslaunch jsk_perception sample_grabcut.launch
```
18.47 GridLabel

Generate labels of grid.

18.47.1 Subscribing Topic

• ~/input (sensor_msgs/CameraInfo or sensor_msgs/Image)

  Input is sensor_msgs/CameraInfo or sensor_msgs/Image. If use_camera_info is true, sensor_msgs/CameraInfo will be used. If use_camera_info is false, sensor_msgs/Image will be used.

18.47.2 Publishing Topic

• ~/output (sensor_msgs/Image (CV_32SC1))

  Output labels as image. Encoding is CV_32SC1.
18.47.3 Parameters

- ~label_size (Integer, default: 32)
  label size
- ~use_camera_info (Boolean, default: false)
  if this parameter is true, it uses sensor_msgs/CameraInfo for ~input.

18.47.4 Sample

roslaunch jsk_perception sample_grid_label.launch

18.48 hand_pose_estimation_2d.py

18.48.1 What is this?

Estimate hand pose in 2d. Please refer to original paper.
In order to use this feature, you need to install pytorch (pytorch >= 1.4.0 is recommended).

18.48.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image.
- ~input/depth (sensor_msgs/Image)
  Input depth image.
- ~input/info (sensor_msgs/CameraInfo)
  Input camera info.
18.48.3 Publishing Topic

- ~output/vis(sensor_msgs/Image)
  Visualization image of detected hand poses.
- ~output/pose(jsk_recognition_msgs/HandPoseArray)
  If with_depth is true, publish 3D joint position.
  If with_depth is false, publish 2D joint position in image.
- ~output/pose_2d(jsk_recognition_msgs/HandPoseArray)
  If with_depth is true, publish 2D joint position.

18.48.4 Parameters

- ~gpu(Int, Default: -1)
  GPU id. -1 represents CPU mode.
- ~thre1(Float, Default: 0.3)
  Threshold of hand bounding box heatmap value.
- ~thre2(Float, Default: 0.2)
  Threshold of hand keypoint heatmap value.
- ~thre3(Int, Default: 5)
  Threshold of undetected keypoints quantity.
- ~visualize(Boolean, Default: True)
  If ~visualize is true, draw estimated hand keypoints.
- ~model_file(String, Required)
  Trained SRHandNet model file.
- ~with_depth(Boolean, Default: False)
  If true, subscribe ~input/depth and ~input/info.
- ~sync_camera_info(Boolean, Default: False)
  Synchronize ~input/info if enabled, otherwise the last received camera info message is used.
- ~approximate_sync(Boolean, Default: True)
  Use approximate synchronization policy.
- ~queue_size(Int, Default: 10)
  Queue size for synchronization.
- ~slop(Float, Default: 0.1)
  Slop for approximate sync.
18.48.5 Example

For 2d hand pose detection.

```bash
roslaunch jsk_perception sample_hand_pose_estimation_2d.launch gpu:=0
```

For 3d hand pose detection.

```bash
roslaunch jsk_perception sample_hand_pose_estimation_3d.launch gpu:=0
```

18.48.6 Reference

```{article}
Wang:2019:SRH,
  doi = {10.1109/TIP.2019.2955280},
  title = {SRHandNet: Real-time 2D Hand Pose Estimation with Simultaneous Region Localization},
  journal = {IEEE Transactions on Image Processing},
  author = {Yangang Wang, Baowen Zhang and Cong Peng},
  number = 1,
  month = Oct.,
  volume = 29,
  year = 2019,
  pages = {2977 - 2986},
}
```

18.49 HoughCircles

18.49.1 @Deprecated

hough_circles in jsk_perception is deprecated.
Please use opencv_apps’s hough_circles(http://wiki.ros.org/opencv_apps).

18.50 HoughLines

18.50.1 @Deprecated

hough_lines in jsk_perception is deprecated.
Please use opencv_apps’s hough_lines(http://wiki.ros.org/opencv_apps).
18.51 HSVDecomposer

Split the original color image into 3 greyscale images of each color (H,S,V) strength.

### 18.51.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Divide input camera image into 3 images (H,S,V) according to its color info.

### 18.51.2 Publishing Topic

- `~output/hue (sensor_msgs/Image)`
- `~output/saturation (sensor_msgs/Image)`
- `~output/value (sensor_msgs/Image)`
18.51.3 Parameters

none

18.51.4 Samples

```bash
roslaunch jsk_perception sample_hsv_decomposer.launch
```

18.52 human_mesh_recovery.py

18.52.1 What is this?

Ros Wrapper of Human Mesh Recovery. See: End-to-end Recovery of Human Shape and Pose
18.52.2 Subscribing Topic

- `~input(sensor_msgs/Image)`
  Input image.
- `~input/pose(jsk_recognition_msgs/PeoplePoseArray)`
  Input 2d people pose array. This is used only when param `~with_people_pose` is true.

18.52.3 Publishing Topic

- `~output/pose(jsk_recognition_msgs/PeoplePoseArray)`
  Estimated 3d people pose array.

18.52.4 Parameters

- `~gpu(Int, Default: -1)`
  GPU id. -1 represents CPU mode.
- `~num_stage(Int, Default: 3)`
  Number of stage size (forwarding iteration count).
- `~smpl_model_file(String, Required)`
  Trained SMPL model file path.
- `~resnet_v2_50_model_file(String, Required)`
  Trained ResNet_V2_50 model file path.
- `~encoder_model_file(String, Required)`
  Trained Encoder model file path.
- `~with_people_pose(Boolean, Default: False)`
  If this param is true, subscribe `~input/poes`.
- `~approximate_sync(Boolean, Default: True)`
  Use approximate synchronization policy. This is used only when param `~with_people_pose` is true.
- `~queue_size(Int, Default: 10)`
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics. This is used only when param `~with_people_pose` is true.
- `~slop(Float, Default: 0.1)`
  Slop for approximate sync. This is used only when param `~with_people_pose` and `~approximate_sync` are true.
18.52.5 Example

```bash
rosrun jsk_perception sample_human_mesh_recovery.launch gpu:=0
```

18.53 `image_cluster_indices_decomposer.py`

18.53.1 What is this?

Publish an image topic to which cluster of image indices is colorized.

18.53.2 Subscribing Topic

- `~input(sensor_msgs/Image)`
  
  Raw image.
  
- `~input/cluster_indices(jsk_recognition_msgs/ClusterPointIndices)`
  
  Cluster of indices to visualize.
18.53.3 Publishing Topic

• ~output (sensor_msgs/Image)
  Visualized image with colorized region which is assigned with the indices.

18.53.4 Parameters

• ~approximate_sync (Bool, default: False)
  Whether to use approximate for input topics.

• ~queue_size (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.

• ~slop (Float, default: 0.1)
  How many seconds you allow about the difference of timestamp when you specify ~approximate_sync.

18.53.5 Sample

```bash
rosrun jsk_perception install_test_data.py
roslaunch sample_image_cluster_indices_decomposer.launch
```
18.54  image_publisher.py

Publish image from loaded file.

18.54.1 Publishing Topics

- ~output (sensor_msgs/Image)
- ~output/compressed (sensor_msgs/CompressedImage)
  Compressed Image.
- ~output/compressedDepth (sensor_msgs/CompressedImage)
  Compressed Depth Image. This is valid when encoding is 32FC1.
- ~output/camera_info (sensor_msgs/CameraInfo)

18.54.2 Parameters

- ~file_name (str default: image.png)
  full path to the file to be loaded
- ~publish_info (bool default: True)
  publish ~output/camera_info if true
- ~encoding (str default: bgr8)
- ~frame_id (str default: camera)
- ~rate (Float default: 1.0)
- ~fovx (Float default: None)
• `~fovy` (Float default: None)
  
  If `~publish_info` is True and `~fovx[degree]` and `~fovy[degree]` are specified, calculate camera intrinsic parameter and publish it as a values of `~output/camera_info`.

### 18.54.3 Sample

```
roslaunch jsk_perception sample_image_publisher.launch
```

### 18.55 image_time_diff.py

Publish difference between current input image and stored one.

Image is stored when start msgs is subscribed.
18.55.1 Subscribing Topics

- `~input/hue(sensor_msgs/Image)`: Input images.
- `~input/saturation(sensor_msgs/Image)`: Input images.
- `~start(std_msgs/Header)`: Store input image and start comparing.
- `~stop(std_msgs/Header)`: Release stored image and stop comparing.

18.55.2 Publishing Topics

- `~output/diff(jsk_recognition_msgs/ImageDifferenceValue)`: Publish an image difference.
- `~output/diff_image(sensor_msgs/Image)`: Publish a diff image.

18.55.3 Parameters

- `rate(FLOAT, default: 10)`: Comparing rate [Hz].
- `~saturation_threshold(int, default: 0)`: Threshold of saturation to get diff of hue.
  This parameter can be changed by dynamic_reconfigure.

18.55.4 Sample

```
roslaunch jsk_perception sample_image_time_diff.launch
```

18.56 image_to_label.py

18.56.1 What is this?

Publish a label image converted from raw image.
18.56.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Raw image.

18.56.3 Publishing Topic

- ~output (sensor_msgs/Image, encoding: 32SC1)
  Label image.

18.57 kalman-filtered-objectdetection-marker.l

18.57.1 What is this?

Apply kalman filter to posedetection_msgs/ObjectDetection message and show marker.
18.57.2 Subscribing Topic

- `input/ObjectDetection` (posedetection_msgs/ObjectDetection)
  Result of object detection.
- `input/image` (sensor_msgs/Image)
  Trigger topic to publish dummy pose to `attention_clipper/input/pose`.
- `input/switch` (std_msgs/Bool)
  Whether to subscribe `input/ObjectDetection`.

18.57.3 Publishing Topics

- `object_detection_marker_array` (visualization_msgs/MarkerArray)
  Marker of raw pose.
- `filtered` (visualization_msgs/MarkerArray)
  Marker of filtered pose.
- `outlier` (visualization_msgs/MarkerArray)
  Marker of outlier.
- `attention_clipper/input/pose` (geometry_msgs/PoseStamped)
  Pose of detected object.
- `posewithcovstamped` (geometry_msgs/PoseWithCovarianceStamped)
- `simplecov` (geometry_msgs/PoseWithCovarianceStamped)
  Pose of detected object with covariance.

18.57.4 Advertising Service

- `targetobj` (posedetection_msgs/TargetObj)
  Service API to return object pose.

18.57.5 Parameters

- `~object_width` (Float, default: 0)
  Width/height of object.
- `~object_height` (Float, default: 0)
- `~relative_pose` (String, default: 0 0 0 0 0 0 1)
  Relative pose.
- `~detection_topic` (String, default: /kinect_head/rgb/ObjectDetection)
  This parameter is not used for now.
- `~marker_life` (Float, default: 300)
  Life time of marker in seconds.
• ~base_frame_id (String, default: /base_footprint)
  Frame ID of base.
• ~target_type (String, default: "")
  Target type.

18.57.6 Sample

roslaunch jsk_perception sample_kalman_filtered_objectdetection_marker.launch

18.58 Kmeans

Apply Kmeans clustering to the input image.
18.58.1 Subscribing Topic

- ~input(sensor_msgs/Image)
  Input image.

18.58.2 Publishing Topic

- ~output(sensor_msgs/Image)
  Output transformed image.

18.58.3 Parameters

- ~n_clusters(Integer, default: 10)
  Number of clusters in Kmeans clustering.

18.59 LabDecomposer

Decompose BGR/RGB image into separate planes in CIE-Lab color space.
18.59.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Input image.

18.59.2 Publishing Topic

- `~output/l (sensor_msgs/Image)`
- `~output/a (sensor_msgs/Image)`
- `~output/b (sensor_msgs/Image)`

  L*, a and b separated planes. Each image has CV_8U encoding.

18.59.3 Sample

```
roslaunch jsk_perception sample_lab_decomposer.launch
```

18.60 LabelArrayToMaskImage

18.60.1 What is this?

Convert a label image to mask image.
18.60.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input label image.

18.60.3 Publishing Topic

Default Topics

- `~output (sensor_msgs/Image)`
  Output mask image.

18.60.4 Parameters

- `~label_values (List[int], default: [])`
  List of label values. The specified label value will be converted to a mask image.

18.60.5 Sample

```bash
roslaunch jsk_perception sample_label_array_to_mask_image.launch
```

18.61 label_image_classifier.py

18.61.1 What is this?

Classify from label image.
Return largest segmented label.
18.61.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input label image.

18.61.3 Publishing Topic

- `~output (jsk_recognition_msgs/ClassificationResult)`
  Classification result of input image.

18.61.4 Parameters

- `~ignore_labels (List of Int, Default: [])`
  Ignoring labels.

18.61.5 Example

```
roslaunch jsk_perception sample_label_image_classifier.launch
```

18.62 label_image_decomposer.py

18.62.1 What is this?

Publish an image topic to which tile mask applied and tile image with decomposed with label.
18.62.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Raw image.
- ~input/label (sensor_msgs/Image)
  Label image to decompose the input raw image with.

18.62.3 Publishing Topic

Default Topics

- ~output (sensor_msgs/Image)
  Label image mask applied image.
- ~output/label_viz (sensor_msgs/Image)
  Applied label to raw image to visualize. It computes the average pixel value of each label and draw with edge line. If parameter ~label_names is specified, the label names are also drawn as a legend.

Optional Topics

- ~output/tile (sensor_msgs/Image)
  Tile image listing decomposed images. This is published when ~publish_tile is True. See Parameters
- ~output/fg_mask, ~output/bg_mask (sensor_msgs/Image)
  Mask image for foreground and background according to the value of label. Currently, it assumes 0 value is background label. This is published when ~publish_mask is True. See Parameters

18.62.4 Parameters

- ~approximate_sync (Bool, default: False)
  Whether to use approximate for input topics.
- ~publish_tile (Bool, default: False)
  Whether to publish tile image with decomposed images.
- ~publish_mask (Bool, default: False)
  Whether to publish fore/background mask images.
- ~queue_size (Int, default: 10)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- ~bg_label (Int, default: 0)
  Background label which is not colorized.
- ~slop (Float, default: 0.1)
  How many seconds you allow about the difference of timestamp when you specify ~approximate_sync.
- ~label_names (List of String, default: [])
  The name of labels, and if this parameter is specified, the ~output/label_viz image is drawn with label name legend.
• ~only_label (Bool, default: False)
  If True, ~input is not subscribed and ~input/label is only used to visualize the input label.
• ~alpha (Float, default: 0.3)
  Ratio of label color when overlaying label image on input rgb image: label_color * alpha + rgb2gray(rgb) * (1 - alpha). Note that this parameter is not used if ~only_label is True.

18.62.5 Sample

roslaunch jsk_perception sample_label_image_decomposer.launch

18.63 LabelToMaskImage

18.63.1 What is this?

Example: Object Segmentation

Convert label image to mask image with specified label value with rosparam.
18.63.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  Label image.

18.63.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Mask image.

18.63.4 Parameters

- ~label_value (Int, default: 0)
  Label value whose region will be converted to mask image.

18.63.5 Sample

```bash
roslaunch jsk_perception sample_label_to_mask_image.launch
```
18.64 MaskImageGenerator

Simply generate a mask image according to ~input image and dynamic reconfigure parameters.

18.64.1 Subscribing Topic

- `~input(sensor_msgs/Image)`

  Input image and it’s used to know original width and height.
18.64.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Output mask image.

18.64.3 Parameters

- `~offset_x` (Int, default: 0)
- `~offset_y` (Int, default: 0)
- `~width` (Int, default: 256)
- `~height` (Int, default: 256) Coordinates of top left point and size of mask image.

18.64.4 Sample

```
roslaunch jsk_perception sample_mask_image_generator.launch
```

18.65 mask_image_to_label.py

18.65.1 What is this?

Publish an label image converted by mask image.

18.65.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Mask image.
18.65.3 Publishing Topic

- ~output (sensor_msgs/Image, encoding: 32SC1)
  Label image.

18.65.4 Sample

```
roslaunch jsk_perception sample_mask_image_to_label.launch
```

18.66 MaskImageToRect

Convert a mask image into jsk_recognition_msgs/RectArray.

18.66.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input mask image.

18.66.2 Publishing Topic

- ~output (jsk_recognition_msgs/RectArray)
  An array of rectangles contains rectangles. They are bounding rectangles of the input mask.
18.66.3 Parameters

- `~rect_type` (Int, default: 0)
  
  Specify output rect types.

  0: Outputs a rect that encloses the entire mask.

  1: Outputs bounding rectangles containing each contour that is output by applying OpenCV’s `findContours` with `CV_RETR_EXTERNAL` to the mask image.
18.66.4 Sample

```bash
roslaunch jsk_perception sample_mask_image_to_rect.launch
```

18.67 MaskImageToROI

Convert a mask image into camera info with roi.

18.67.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input mask image.
- `~input/camera_info (sensor_msgs/CameraInfo)`
  Original camera info.

18.67.2 Publishing Topic

- `~output (sensor_msgs/CameraInfo)`
  Camera info with ROI field filled.
18.67.3 Sample

roslaunch jsk_perception sample_mask_image_to_roi.launch

18.68 matchtemplate.py

18.68.1 What is this?

Run image template matching and publish the position of matched image.

18.68.2 Subscribing Topic

- reference (sensor_msgs/Image)
  Input reference image.
- search (sensor_msgs/Image)
  Input image to detect if part of which matches the reference.
- set_reference_point (geometry_msgs/PointStamped)
  Center point of reference window in reference image.
- set_search_rect (jsk_recognition_msgs/Rect)
  Searched window rectangle in search image.
  Please set this to top left point, width and height.
18.68.3 Publishing Topic

- `current_template` (sensor_msgs/Image)
  Template image cropped from reference.
- `result` (geometry_msgs/TransformStamped)
  Pixel index of the center point of matched rectangle.
- `debug_image` (sensor_msgs/Image)
  Visualization of matching result.

This topic is published only when `~show_debug_image` is set to true.

18.68.4 Parameters

- `~current_template_id` (String, default: '')
  Not used now.
- `~match_method` (Int, default: 1)
  Choose from TM_SQDIFF (0), TM_SQDIFF_NORMED (1), TM_CCORR (2), TM_CCORR_NORMED (3), TM_CCOEFF (4) and TM_CCOEFF_NORMED (5).
- `~template_color_space` (String, default: mono8)
  Choose from mono8, bgr8, bgra8, hsv8 and hsva8.
- `~show_debug_image` (Bool, default: True)
  Publish debug_image if true.
- `~auto_search_area` (Bool, default: True)
  Enable feedback of searching status if true.
- `~default_template_width` (Int, default: 32)
  Allowed value: 0 <= width <= 128.
- `~default_template_height` (Int, default: 32)
  Allowed value: 0 <= height <= 128.
- `~default_search_width` (Int, default: 64)
  Allowed value: 0 <= width <= 256.
- `~default_search_height` (Int, default: 64)
  Allowed value: 0 <= height <= 256.
18.68.5 Sample

```
roslaunch jsk_perception sample_matchtemplate.launch
```

18.69 MorphologicalGradient

See MorphologicalOperator

18.70 MorphologicalOperator (DilateMaskImage, Closing etc.)
Apply morphological transformations.

### 18.70.1 Nodes

- `dilate_mask_image`
- `erode_mask_image`
- `opening`
- `closing`
- `morphological_gradient`
- `top_hat`
- `black_hat`
18.70.2 Nodelets

- jsk_perception/DilateMaskImage
- jsk_perception/ErodeMaskImage
- jsk_perception/Opening
- jsk_perception/Closing
- jsk_perception/MorphologicalGradient
- jsk_perception/TopHat
- jsk_perception/BlackHat

18.70.3 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image

18.70.4 Publishing Topic

- ~output (sensor_msgs/Image)
  Output transformed image.

18.70.5 Parameters

- ~method (0, 1 or 2, default: 0)
  Method to transform image.
  0 means rectangular box model, 1 means cross model and 2 means ellipse.
- ~size (Integer, default: 1)
  Kernel size to transform.
- ~iterations (Integer, default: 1)
  Iterations of transforming image.

18.70.6 Sample

roslaunch jsk_perception sample_morphological_operator.launch
18.71 MultiplyMaskImage

Multiply (bitwise) two mask image into one mask image.

18.71.1 Subscribing Topic

- ~input/src1 (sensor_msgs/Image)
- ~input/src2 (sensor_msgs/Image)
  Input mask images.

18.71.2 Publishing Topic

- ~output (sensor_msgs/Image)
  Added mask image.
18.71.3 Parameters

- `~approximate_sync` (Bool, default: false)
  Approximately synchronize `~input/src1` and `~input/src2` if it's true.
- `~queue_size` (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.

18.71.4 Sample

```
roslaunch jsk_perception sample_multiply_mask_image.launch
```

18.72 non_maximum_suppression.py

18.72.1 What is this?

Regard highly overlapped rectangles in an image as the same object and remove extra rectangles.

18.72.2 Subscribing Topic

- `~input (jsk_recognition_msgs/RectArray)`
  Input rectangles.

18.72.3 Publishing Topic

- `~output (jsk_recognition_msgs/RectArray)`
  Thinned out rectangles.
- `~output/count (std_msgs/Int64)`
  Number of output rectangles.
18.72.4 Parameters

- `~threshold` (Float, default: 0.0)
  Maximum allowed value of IoU (intersect over union) for each pair of input rectangles.

18.72.5 Advertising Service

- `non_maximum_suppression` (jsk_perception/NonMaximumSuppression)
  Service API of non-maximum suppression described above.

18.72.6 Sample

```
roslaunch jsk_perception sample_non_maximum_suppression.launch
```

18.73 ocr_node.py

18.73.1 What is this?

Optical Character Reader (OCR) node from image and text area. We recommend the python3 environment. For python2 users, please do pip install `pytesseract==0.3.1` manually because the latest version of pytesseract is not installed automatically.

18.73.2 Subscribing Topic

- `~input` (sensor_msgs/Image)
  Raw image.

- `~input/polygons` (jsk_recognition_msgs/PolygonArray)
  Polygon array representing a text area. If `~subscribe_polygon` is True, subscribe `~input/polygons`.

- `~input/rects` (jsk_recognition_msgs/RectArray)
  Rect array representing a text area. If `~subscribe_polygon` is False, subscribe `~input/rects`.
18.73.3 Publishing Topic

- ~output (std_msgs/String)
  Recognized text.
- ~output/viz (sensor_msgs/Image)
  Visualized OCR results.
- ~output/labels (jsk_recognition_msgs/LabelArray)
  The recognized text is output as the label name in the order of the input polygon array or rect array.
- ~output/debug/viz (sensor_msgs/Image)
  Tile image of cropped input image.
- ~output/debug/binary_viz (sensor_msgs/Image)
  Tile image of a binarized cropped input image.

18.73.4 Parameters

- ~approximate_sync (Bool, default: False)
  Whether to use approximate for input topics.
- ~queue_size (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- ~slop (Double, Default: 0.1)
  Slope for approximate sync.
- ~language (Int, Default: eng)
  OCR target language. In current ocr_node.py uses tesseract-ocr.
  If you want to use a language other than English, please install the appropriate language data from packages and change the language argument.
  For example, if you want to use Japanese, please install tesseract-ocr-jpn (apt install tesseract-ocr-jpn in Ubuntu) and pass jpn as the language argument.
- ~number_of_jobs (Int, Default: -1)
  Number of jobs for ocr. If this value is -1, use the value of multiprocessing.cpu_count().
- ~font_path (String, default: "")
  Font path. Specify the font with the characters you want to display.
- ~font_size (Int, default: 16)
  Font size for visualization
- ~box_thickness (Int, default: 2)
  Thickness of bounding box (text areas).
- ~resolution_factor (Double, default: 2.0)
  Factor for resolution of output image. When this option is set as 1.0, an output image has the same resolution as an input image.
• `~interpolation_method` (Enum[Int], default: INTER_LANCZOS4)
  Method for interpolation on input image resizing.

• `~subscribe_polygon` (Bool, Default: False)
  If `~subscribe_polygon` is True, subscribe `~input/polygons`. If `~subscribe_polygon` is False, subscribe `~input/rects`.

18.73.5 Sample

```bash
roslaunch jsk_perception sample_craft_node.launch
```

18.74 Opening

See MorphologicalOperator

18.75 OverlayImageColorOnMono

18.75.1 What is this?

Publish the overlayed image of input color image on input mono image.

18.75.2 Subscribing Topic

• `~input/color` (sensor_msgs/Image)
  Raw image that is overlayed as color image.

• `~input/mono` (sensor_msgs/Image)
  Raw image that converted to mono image and being overlayed by the color image.
18.75.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Overlayed image of input color image on input mono image.

18.75.4 Parameters

- ~color_alpha (Double, default: 0.3)
  The weight for pixel value of color image.
- ~approximate_sync (Bool, default: False)
  Whether to use approximate for input topics.
- ~queue_size (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.

18.75.5 Sample

```bash
roslaunch jsk_perception sample_overlay_image_color_on_mono.launch
```

18.76 paper_finder

Detect papers (rectangle) and publish posearray.
18.76.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image.
- ~input/depth (sensor_msgs/Image)
  Input depth image. Subscribed if ~with_depth is true.
- ~input/camera_info (sensor_msgs/CameraInfo)
  Camera info. Subscribed if ~with_depth is true.

18.76.2 Publishing Topics

- ~output/viz (sensor_msgs/Image)
  Visualization of recognition result in image.
- ~output/pose (geometry_msgs/PoseArray)
  Recognized paper pose.
- ~output/boxes (jsk_recognition_msgs/BoundingBoxArray)
  Recognized Bounding Box.
- ~output/length (std_msgs/Float32MultiArray)
  Recognized the rectangle sides of paper.

18.76.3 Parameters

- ~angle_tolerance (float default: 286.4788975654116)
  286.4788975654116 = np.rad2deg(5.0)
- ~area_tolerance (float default: 0.1)
- ~rect_x (float default: 0.210)
  0.210 [m] is the width of A4 paper.
- ~rect_y (float default: 0.297)
  0.297 [m] is the height of A4 paper.
- ~length_tolerance (float default: 0.04)
- ~queue_size (int default: 10)
- ~with_depth (bool default: true)
  If true, use the depth of the detected square apex position for pose estimation.
- ~approximate_sync (bool default: true)
  If true, approximately synchronize inputs. Required if ~with_depth:=true.
- ~slop (float default: 0.1)
  The slop time in second for message_filters.ApproximateTimeSynchronizer. Required if
  ~with_depth:=true and ~approximate_sync:=true.
18.76.4 Sample

```
roslaunch jsk_perception sample_paper_finder.launch
```

18.77 people_mask_publisher.py

18.77.1 What is this?

Create people mask image according to people pose information.

18.77.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input image.
- `~input/pose (jsk_recognition_msgs/PeoplePoseArray)`
  Input people pose array.

18.77.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  People mask image.
- `~debug/output (sensor_msgs/Image)`
  Debug image which has visualized rectangle of the masked limb part.
18.77.4 Parameters

- ~person_indices (Int, Default: -1)
  Which person to create mask of. -1 represents creating masks of all people.

- ~limb_part (String or List of String, Default: all)
  Which limb part to create mask of.
  RHand, LHand, Nose and all is allowed.

- ~arms_score_threshold (Float, Default: 0.25)
  Threshold of arm score.
  Used when limb_part includes RHand or LHand.

- ~hand_ratio (Float, Default: 0.33)
  Ratio that is used to predict hand position from elbow and wrist positions.
  Used when limb_part includes RHand or LHand.

- ~hand_width_ratio (Float, Default: 0.8)
  Ratio that is used to predict hand region from arm length.
  Used when limb_part includes RHand or LHand.

- ~face_ratio (Float, Default: 0.6)
  Ratio that is used to predict face position from nose and neck positions.
  Used when limb_part includes Nose.

- ~face_shoulder_ratio (Float, Default: 0.5)
  Ratio that is used to predict face position from shoulder positions.
  Used when limb_part includes Nose.

- face_width_margin_ratio (Float, Default: 1.3)
  Ratio that is used to decide face width margin.
  Used when limb_part includes Nose.

- ~approximate_sync (Bool, Default: False)
  Use approximate synchronization policy.

- ~queue_size (Int, Default: 10)
  Queue size for synchronization.

- ~slop (Float, Default: 0.1)
  Slop for approximate sync.
18.77.5 Example

```bash
text
```
roslaunch jsk_perception sample_people_pose_estimation_2d.launch GPU:=0 LIMB_PART:=\'RHand\'
roslaunch jsk_perception sample_people_pose_estimation_2d.launch GPU:=0 LIMB_PART:=\'Nose\'
```

18.78 people_pose_estimation_2d.py

18.78.1 What is this?
Estimate people pose and hand pose in 2d. Please refer to original paper.

In order to use this feature, you need to install chainer. To install chainer with GPU support, please refer to this page.

### 18.78.2 Subscribing Topic

- `~input(sensor_msgs/Image)`
  
  Input image.

- `~input/depth(sensor_msgs/Image)`
  
  Input depth image.

- `~input/info(sensor_msgs/CameraInfo)`
  
  Input camera info.
18.78.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Detected people pose image.
- ~pose (jsk_recognition_msgs/PeoplePoseArray)
  If with_depth is true, publish 3D joint position.
  If with_depth is false, publish 2D joint position in image.
- ~pose2d (jsk_recognition_msgs/PeoplePoseArray)
  If with_depth is true, publish 2D joint position.
- ~skeleton (jsk_recognition_msgs/HumanSkeletonArray)
  If with_depth is true, publish 3D human skeletons.
Rviz visualization is available in jsk-ros-pkg/jsk_visualization #740.

18.78.4 Parameters

- ~gpu (Int, Default: -1)
  GPU id. -1 represents CPU mode.
- ~scales (Float, Default: 0.38)
  Resize image scale.
- ~stride (Int, Default: 8)
  Stride of image.
- ~pad_value (Int, Default: 128)
  Value of padding area.
- ~thre1 (Float, Default: 0.1)
  Threshold of heatmap value.
- ~thre2 (Float, Default: 0.05)
  Threshold of score.
- ~hand/enable (Bool, Default: False)
  If True, estimate hand pose.
- ~hand/gaussian_ksize (Int, Default: 17)
  Size of gaussian kernel.
- ~hand/gaussian_sigma (Float, Default: 2.5)
  Value of gaussian sigma.
- ~hand/thre1 (Float, Default: 20)
  Threshold of width of hand area.
- ~hand/thre2 (Float, Default: 0.1)
  Threshold of hand score.
• ~hand/width_offset (Int, Default: 0)
  Offset of hand area’s width.
• ~visualize (Bool, Default: True)
  If ~visualize is true, draw an estimated pose.
• ~model_file (String, Required)
  Trained model file.
• ~with_depth (Bool, Default: False)
  If true, subscribe ~input/depth and ~input/info.
• ~sync_camera_info (Bool, Default: False)
  Synchronize ~input/info if enabled, otherwise the last received camera info message is used.
• ~approximate_sync (Bool, Default: True)
  Use approximate synchronization policy.
• ~queue_size (Int, Default: 10)
  Queue size for synchronization.
• ~slop (Float, Default: 0.1)
  Slop for approximate sync.

18.78.5 Example

roslaunch jsk_perception sample_people_pose_estimation_2d.launch gpu:=0
18.79 PointPoseExtractor

Calcute object pose compared to template.

18.79.1 Subscribing Topic

- `/ImageFeature0D (posedetection_msgs::ImageFeature0D)
  Image, camera and template feature information. You can use ImageShift to get it.

18.79.2 Publishing Topics

- `/ObjectDetection (posedetection_msgs::ObjectDetection)
  Detected object pose with time stamp.
- `/ObjectDetection_agg (posedetection_msgs::ObjectDetection)
  Detected object pose with time stamp.
- `/object_pose (geometry_msgs::PoseStamped)
  Detected Object Pose.
- `~debug_image (cv_bridge::CvImage)
  Output image for debug.
- `/tf (tf2_msgs/TFMessage)
  Detected Object Frame when ~publish_tf is set to true.
18.79.3 Parameters

- `~template_filename` (str default: "/sample/opencv-logo2.png")
  path to template image
- `~object_width` (float default: 0.06)
  Width of template image
- `~object_height` (float default: 0.0739)
  Height of template image
- `~relative_pose` (str default: "0 0 0 0 0 0 1")
  Coordinate of the object relative to the texture
- `~reprojection_threshold` (float default: 3.0)
- `~distanceratio_threshold` (float default: 0.49)
- `~error_threshold` (float default: 50.0)
- `~theta_step` (float default: 5.0)
- `~phi_step` (float default: 5.0)
- `~viewer_window` (bool default: true)
- `~window_name` (str default: "sample1")
- `~autosize` (bool default: false)
  The window size is automatically adjusted to fit the displayed image, and you cannot change the window size manually.
- `~publish_null_object_detection` (bool default: false)
- `~publish_tf` (bool default: false)
  If set to true, detected object pose is also broadcasted as tf frame.
- `~child_frame_id` (string default: "matching")
  frame_id of detected object when `~publish_tf` is set to true.

18.79.4 Service

- `/SetTemplate (SetTemplate)`
  Used to add another template.
18.79.5 Mouse Event

Set template from viewer window.

To specify the range of template, left-click four corners clockwise from upper left. Selected points are reset by right-clicking.

After all four points are selected, you can input template’s width, height and filename. The filename should have an extension.

18.79.6 Sample

```bash
rosrun jsk_perception sample_point_pose_extractor.launch
rosrun jsk_perception sample_point_pose_extractor.launch
rosrun jsk_perception sample_point_pose_extractor.launch
```

Example of how to run set_template service

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import cv2
import cv_bridge
import rospy
from jsk_perception.srv import SetTemplate, SetTemplateRequest

rospy.init_node('point_pose_extractor_sample')

client = rospy.ServiceProxy('SetTemplate', SetTemplate)
```

(continues on next page)
rospy.sleep(1)
req= SetTemplateRequest()
im = cv2.imread('./sample/rosdiamondback.jpg')
bridge = cv_bridge.CvBridge()
imgmsg = bridge.cv2_to_imgmsg(im, encoding='bgr8')
imgmsg.header.frame_id = 'dummy_camera'
imgmsg.header.stamp = rospy.Time.now()
req.type = 'img0001'
req.image = imgmsg
req.dimx = 0.1
req.dimy = 0.05
req.savefilename = 'img0001.png'
res = client.call(req)
print(res)

18.80 pointit.py

Select object bounding box by human pointing his/her finger toward one of them.
18.80.1 Subscribing Topics

- ~input (jsk_recognition_msgs/PeoplePoseArray)
  Pose of human pointing his/her finger toward some objects.
- ~input/boxes (jsk_recognition_msgs/BoundingBoxArray)
  Object bounding boxes.
- ~input/class (jsk_recognition_msgs/ClassificationResult)
  Label of objects.
  This topic is subscribed only when ~use_classification_result is True.

18.80.2 Publishing Topics

- ~output (jsk_recognition_msgs/BoundingBoxArray)
  Pointed object bounding box.
- ~output/marker_array (visualization_msgs/MarkerArray)
  Finger marker.

18.80.3 Parameters

- ~use_classification_result (Bool, default: False)
  If True, this node will subscribe ~input/class.
  Also, parameter ~approximate_sync and ~queue_size will be enabled.
- ~approximate_sync (Bool, default: True)
  Allow approximate synchronization of ~input/boxes and ~input/class.
- ~queue_size (Int, default: 10 (when ~approximate_sync is True) or 100 (when ~approximate_sync is False))
  Maximum number of messages stored into subscriber for synchronization.
- ~slop (Float, default: 0.1)
  Maximum allowed time for approximate synchronization in [sec].
  This parameter is enabled only when ~approximate_sync is True.
- ~min_dist_threshold (Float, default: 0.0)
  Minimum allowed distance from left/right hand to the nearest object.
- ~max_dist_threshold (Float, default: 0.1)
  Maximum allowed distance from left/right hand to the nearest object.
- ~min_norm_threshold (Float, default: 0.2)
  Minimum allowed distance from left/right finger to object candidates.
- ~use_tf2_buffer_client (Bool, default: True)
  Whether to use tf2_ros.BufferClient or not.
  If false, tf2_ros.Buffer and tf2_ros.TransformListener will be used.
18.80.4 Sample

```bash
roslaunch jsk_perception sample_pointit.launch
```

18.81 PolygonArrayColorHistogram

Compute color histogram of the region which is specified by 3-D Polygon.

18.81.1 Subscribing Topics

- `~input (jsk_recognition_msgs/PolygonArray)`
  
  Input 3-D polygon array.

- `~input/image (sensor_msgs/Image)`
  
  Input image.

- `~input/info (sensor_msgs/CameraInfo)`
  
  Input camera info.
18.81.2 Publishing Topics

- `~output (jsk_recognition_msgs/HistogramWithRangeArray)`
  Histogram array. The order of the array is the same as `~input polygon array`.
- `~debug/polygon_image (sensor_msgs/Image)`
  Debug image representing polygons seen from camera.

18.81.3 Parameter

- `~bin_size (Int, default: 10)`
  The number of histogram bins.
- `~pixel_min_value (Int, default: 0)`
- `~pixel_max_value (Int, default: 180)`
  Minimum and maximum value of the histogram.
- `~debug_line_width (Int, default: 2)`
  Line width of debug image.
- `~max_queue_size (Int, default: 10)`
  Queue size of subscriber
- `~synchronizer_queue_size (Int, default: 100)`
  Queue size of message filter

18.81.4 Sample

```bash
roslaunch jsk_perception sample_polygon_array_color_histogram.launch
```
18.82 PolygonArrayColorLikelihood

Compute polygon likelihood based on distance of histograms.

18.82.1 Subscribing Topics

- ~input/polygons (jsk_recognition_msgs/PolygonArray)
  Input polygons.
- ~input/histograms (jsk_recognition_msgs/HistogramWithRangeArray)
  Color histogram of input polygons.
- ~input/reference (jsk_recognition_msgs/HistogramWithRange)
  Reference color histogram.

18.82.2 Publishing Topics

- ~output (jsk_recognition_msgs/PolygonArray)
  Output polygons with updated likelihood field.
18.82.3 Parameters

- `~approximate_sync` (Bool, default: false)
  Approximately synchronize `~input/polylines` and `~input/histograms`.

- `~max_queue_size` (Int, default: 10)
  Queue size of subscriber

- `~synchronizer_queue_size` (Int, default: 100)
  Queue size of message filter

- `~reference_file` (String, default: "")
  If this parameter is specified, `PolygonArrayColorLikelihood` reads reference histogram from a yaml file instead of subscribing `~input/reference`.

  The yaml file format is

  ```yaml
  bins:
    -
      min_value: xx
      max_value: xx
      count: xx
    -
      min_value: xx
      max_value: xx
      count: xx
  ```

- `~coefficient_method` (Int, default: 0)
  Method to compute coefficient between two histograms.

  Choose from `correlancy` (0), `chi_squared` (1), `intersect` (2), `bhattacharyya` (3), `EMD_Manhattan` (4) and `EMD_Euclid` (5).

  This parameter can be changed by `dynamic_reconfigure`.

18.82.4 Sample

```bash
roslaunch jsk_perception sample_polygon_array_color_likelihood.launch
```
18.83 PolygonArrayToLabelImage

Convert polygon array into label image. Label starts with 1 and 0 mean the pixel does not belong to any polygons. No z-buffer is taken into account and occlusion is not solved correctly.

Frame ID of polygons and camera_info must be the same.

18.83.1 Subscribing Topic

- `~input (jsk_recognition_msgs/PolygonArray)`
  Input 3-D polygon array.
- `~input/camera_info (sensor_msgs/CameraInfo)`
  Input camera info to project 3-D polygon.

18.83.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  Label image filled with `~input` polygons. Currently only convex polygon is supported.
18.83.3 Sample

```bash
tools@robot:~$ roslaunch jsk_perception sample_polygon_array_to_label_image.launch
```

18.84 PolygonToMaskImage

Convert polygon into mask image.

18.84.1 Subscribing Topic

- `~input (geometry_msgs/PolygonStamped)`
  
  Input 3-D polygon.

- `~input/camera_info (sensor_msgs/CameraInfo)`
  
  Input camera info to project 3-D polygon.

  *frame_id of `~input` and `~input/camera_info` should be the same.*

18.84.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Mask image filled with `~input` polygon. Currently only convex polygon is supported.

18.84.3 Sample

```bash
tools@robot:~$ roslaunch jsk_perception sample_polygon_to_mask_image.launch
```
18.85 probability_image_classifier.py

18.85.1 What is this?

Classify from probability image.
Return label whose sum of probability is largest.

18.85.2 Subscribing Topic

- ~input(sensor_msgs/Image)
  Input probability image.

18.85.3 Publishing Topic

- ~output(jsk_recognition_msgs/ClassificationResult)
  Classification result of input image.

18.85.4 Parameters

- ~ignore_labels(List of Int, Default: [])
  Ignoring labels.
18.85.5 Example

```bash
roslaunch jsk_perception sample_probability_image_classifier.launch
```

18.86 ProjectImagePoint

Convert image local coordinates (represented as `geometry_msgs/PointStamped`) into 3-D point. Z value of the point is specified via dyanmci_reconfigure API.
18.86.1 Subscribing Topic

- ~input (geometry_msgs/PointStamped)
  Input point in image local coordinates.
- ~input/camera_info (sensor_msgs/CameraInfo)
  Camera parameter of the original image.

18.86.2 Publishing Topic

- ~output (geometry_msgs/PointStamped)
  Output point and the value is scaled to satisfy specified z value.
- ~output/ray (geometry_msgs/Vector3Stamped)
  3-D ray vector of the point of image local coordinates.

18.86.3 Parameters

- ~z (Double, default: 2.0)
  Z value of projected point.

18.86.4 Sample

roslaunch jsk_perception sample_project_image_point.launch

18.87 random_forest_server.py

18.87.1 What is this?
Service server of Random Forest classifier.

### 18.87.2 Advertising Service

- **predict (ml_classifiers/ClassifyData)**
  
  Returns classification result as a list of string, according to requested points.

  Currently, only `ClassifyData/data/point` field is used for request.

### 18.87.3 Parameters

- **~random_forest_train_file (String, required)**
  
  Path to training data file used for building forest of tree.

  If the file name ends with ‘pkl’, then this node will treat it as a built forest and try to deserialize it.

  If not, the file should contain training data as a list of float in each line.

- **~random_forest_train_class_file (String)**
  
  When ~random_forest_train_file does not end with ‘pkl’, this parameter will be enabled.

  The file should contain ground-truth class label data as a float number in each line.

### 18.87.4 Sample

```bash
roslaunch jsk_perception random_forest_sample.launch
```

### 18.88 RectArrayActualSizeFilter

Filter `jsk_recognition_msgs/RectArray` with actual size based on depth image.

Sample is `selective_search_depth_image_filter.launch`.
18.88.1 Subscribing Topics

- ~input (jsk_recognition_msgs/RectArray)
  Input array of rectangle regions.
- ~input/depth_image (sensor_msgs/Image)
  Input depth image.
- ~input/info (sensor_msgs/CameraInfo)
  Intrinsic camera parameter of the depth image.

18.88.2 Publishing Topics

- ~output (jsk_recognition_msgs/RectArray)
  Filtered array of rectangle regions.

18.88.3 Parameters

- ~approximate_sync (bool, default: False)
  Synchronize timestamps of input topics approximately.
- ~kernel_size (Integer, default: 3)
  Kernel size to take average of distance.
- ~min_x
- ~max_x
- ~min_y
- ~max_y
  Minimum and maximum size of x (width) and y (height) axis in meter unit.

18.88.4 Sample

```bash
cat
```

18.89 rect_array_in_panorama_to_bounding_box_array.py

This node calculates distances to RectArray in a panorama image with FOV of the image, heights of rects and heights of each labels with the equation \((\text{object height [m]}) = (\text{distance [m]}) \times (\text{object height in panorama image [rad]})\), and publish BoundingBoxArray.

Sample is sample_rect_array_in_panorama_to_bounding_box_array.launch
18.89.1 Subscribing Topics

- ~panorama_info (jsk_recognition_msgs/PanoramaInfo)
  Panorama info topic.
- ~input_class (jsk_recognition_msgs/ClassificationResult)
  ClassificationResult from object detection with the panorama image topic
- ~input_rects (jsk_recognition_msgs/RectArray)
  RectArray from object detection with the panorama image topic

18.89.2 Publishing Topics

- ~bbox_array (jsk_recognition_msgs/BoundingBoxArray)
  BoundingBoxArray of each object of output of object detection.

18.89.3 Parameters

- ~frame_fixed (string, default: fixed_frame)
  The frame_id of BoundingBoxArray. Assumed to be a fixed frame.
- ~dimensions_labels (dict of string to lists of float values, default: {})
  Dimensions for each labels. These are used for BoundingBoxArray and also calculation of distance.
  e.g. {'person':[0.5,0.5,1.5]}
- ~duration_timeout (float, default: `0.05)
  Duration of timeout for lookup transform

18.89.4 Sample

roslaunch jsk_perception sample_rect_array_in_panorama_to_bounding_box_array.launch
18.90 RectArrayToDensityImage

18.90.1 What is this?

Convert rect array (jsk_recognition_msgs/RectArray) to density image (sensor_msgs/Image, 32FC1).

18.90.2 Subscribing Topic

- ~input/image(sensor_msgs/Image)
  Raw image.
- ~input/rect_array(jsk_recognition_msgs/RectArray)
  Array of rectangle.

18.90.3 Publishing Topic

- ~output(sensor_msgs/Image, 32FC1)
  Density image computed based on the number of rectangles overlaid.

18.90.4 Parameters

- ~approximate_sync(Boolean, default: False)
  Whether to use approximate for input topics.
- ~queue_size(Integer, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
18.90.5 Sample

```
roslaunch jsk_perception sample_rect_array_to_density_image.launch
```

18.90.6 Test

```
rostest jsk_perception rect_array_to_density_image.test
```

18.91 `rect_array_to_image_marker.py`

Convert `jsk_recognition_msgs/RectArray` to `image_view2/ImageMarker`.

18.91.1 Subscribing Topics

- `~input(jsk_recognition_msgs/RectArray)`

18.91.2 Publishing Topics

- `~output(image_view2/ImageMarker)`

18.91.3 Sample

```
roslaunch jsk_perception sample_rect_array_actual_size_filter.launch
```
18.92 RectToMaskImage

Convert 2D rectangle (geometry_msgs/Polygon), whose position and size are represented in [pixel], into mask image (sensor_msgs/Image).

We expect it will be used with image_view2.

18.92.1 Subscribing Topic

• ~input (geometry_msgs/PolygonStamped)
  PolygonStamped to represent rectangle region of image.
• ~input/camera_info (sensor_msgs/CameraInfo)
  Original camera info.

18.92.2 Publishing Topic

• ~output (sensor_msgs/Image)
  Mask image.
18.92.3 Sample

```bash
roslaunch jsk_perception sample_rect_to_mask_image.launch
```

18.93 RectToROI

Convert 2D rectangle (geometry_msgs/Polygon), whose position and size are represented in [pixel], into camera info with ROI (sensor_msgs/CameraInfo).

We expect it will be used with image_view2.

18.93.1 Subscribing Topic

- `~input (geometry_msgs/PolygonStamped)`
  
  Polygon to represent rectangle region of image.

- `~input/camera_info (sensor_msgs/CameraInfo)`
  
  Original camera info.
18.93.2 Publishing Topic

- `~output(sensor_msgs/CameraInfo)`
  camera info with ROI filled by `~input`.

18.93.3 Sample

```
roslaunch jsk_perception sample_rect_to_roi.launch
```

18.94 regional_feature_based_object_recognition.py

18.94.1 What is this?

Classify object image from input image and mask using regional feature outputted by ResNet.
18.94.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input label image.
- `~input/mask (sensor_msgs/Image)`
  Region of interest.

18.94.3 Publishing Topic

- `~output (jsk_recognition_msgs/ClassificationResult)`
  Classification result of input image.

18.94.4 Parameters

- `~db_file (String, required)`
  DB file which has the pairs of object label and ResNet feature vector.
- `~gpu (Int, default: 0)`
  GPU id to be used.

18.94.5 Example

The sample classifies 39 objects which is used Amazon Picking Challenge 2016.

```bash
roslaunch jsk_perception sample_regional_feature_based_object_recognition.launch # CPU mode
roslaunch jsk_perception sample_regional_feature_based_object_recognition.launch --gpu:=0 # GPU mode
```

18.94.6 How to create db_file?

You can create the DB file form pairs of object image and mask for each object you’d like to recognize. In the sample, the db_file is automatically downloaded, but you can try to create it again in your environment.

```bash
rosrun jsk_perception create_db_for_regional_feature_based_object_recognition.py $(rospack find jsk_perception)/sample/data/apc2016_object_imgs_and_masks_templates $(rospack find jsk_perception)/sample/data/resnet_features_apc2016.npz
```
18.95 RemoveBlurredFrames

18.95.1 What Is This

The node for removing blurred images from image topics. When a blurred image is subscribed, nothing is published, and when a non-blurred image is subscribed, it is published as is. It detects blurred images by thresholding the value of variance of a Laplacian filtered image.

18.95.2 Subscribing Topic

- ~input(sensor_msgs/Image)

  Image input.

18.95.3 Publishing Topic

- ~output(sensor_msgs/Image)

  Non-blurred image output.

- ~output/mask(sensor_msgs/Image)

  Laplacian filtered image output.

- ~output/var(std_msgs/Float64)

  Laplacian filtered image’s variance.

18.95.4 Parameters

- ~min_laplacian_var(Double, default: 400.0)

  The threshold of laplacian variance. Increasing this value removes more frames.

18.95.5 For use

```bash
roslaunch jsk_perception remove_blurred_frames.launch INPUT_IMAGE:=<your image>
```

18.95.6 Sample

```bash
roslaunch jsk_perception sample_remove_blurred_frames.launch
```
18.96 RGBDecomposer

Split the original color image into 3 greyscale image of each color (R,G,B) strength.

18.96.1 Subscribing Topic

- \texttt{~input (sensormsgs/Image)}
  
  Divide input camera image into 3 image (R,G,B) according to its color info.

18.96.2 Publishing Topic

- \texttt{~output/red (sensormsgs/Image)}
- \texttt{~output/green (sensormsgs/Image)}
- \texttt{~output/blue (sensormsgs/Image)}
18.96.3 Parameters

none

18.96.4 Samples

roslaunch jsk_perception sample_rgb_decomposer.launch

18.97 RobotToMaskImage

Convert robot model into mask image.

18.97.1 Subscribing Topic

- `~input/camera_info` (sensor_msgs/CameraInfo)

  Input camera info to project 3-D polygon.
18.97.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  Mask image to fill `~input` polygon.
  The size of the output image will be resized to fit the `roi` and `binning`.
- `~output/info (sensor_msgs/CameraInfo)`
  CameraInfo for the output image.

18.97.3 Parameters

- `~max_robot_dist (Double, default: 10)`
  Maximum distance of robot from camera.
  Distance is in z-axis direction of the camera coordinates.
- `robot_description (String, required)`
  robot_description of the mask-generated robot.
- `~self_see_default_padding (Double, default: 0.001)`
  Same as the parameter in `pr2_navigation_self_filter`. Padding of robot link for mask generation.
- `~self_see_default_scale (Double, default: 1.0)`
  Same as the parameter in `pr2_navigation_self_filter`. Scale of robot link for mask generation.
- `~self_see_links (Array of link configuration, required)`
  Same as the parameter in `pr2_navigation_self_filter`. Configuration of links for mask generation. Link configuration consists of name (required), padding (optional), and scale (optional).

18.97.4 Samples

```
roslaunch jsk_perception sample_robot_to_mask_image.launch
```
18.98 ROItoMaskImage

Convert camera info with ROI to mask image.

18.98.1 Subscribing Topic

- `~input(sensor_msgs/CameraInfo)`
  
  Input camera info with ROI filled.

18.98.2 Publishing Topic

- `~output(sensor_msgs/Image)`
  
  Output mask image.

18.98.3 Sample

```
roslaunch jsk_perception sample_roi_to_mask_image.launch
```
18.99 ROItoRect

Convert camera info with ROI to 2D polygon.

18.99.1 Subscribing Topic

• ~input(sensor_msgs/CameraInfo)
  Input camera info with ROI filled.

18.99.2 Publishing Topic

• ~output(geometry_msgs/PolygonStamped)
  Output rectangle region.
  Position of each vertex is represented in [pixel].

18.99.3 Sample

roslaunch jsk_perception sample_roi_to_rect.launch
18.100 SaliencyMapGenerator

This nodelet is used to compute the saliency of an image.

The image can be composed of any feature space. However, the current nodelet takes only 3 channel BGR image and computes the saliency map.

The original code is implemented by the authors of the paper: Human detection using a mobile platform and novel features derived from a visual saliency mechanism.

Please cite this paper should you use the code.

However, there are minor changes from the original code including the MultiThreading support using OpenMP.
18.100.1 Subscribing Topic

• ~input(sensor_msgs/Image)
  Currently only supports BGR8 encoding (3 channel RGB intensity image)

18.100.2 Publishing Topic

• /saliency_map_generator/output/saliency_map(sensor_msgs/Image)
  Output saliency map.

18.100.3 Parameters

• ~num_threads(Int, default: 2)
  Number of threads used for execution
• ~fps(Boolean, default: true)
  Prints the frame rate to the output image if true.

18.100.4 Sample

roslaunch jsk_perception sample_saliency_map_generator.launch

18.101 selective_search.py

Segment image using Selective Search algorithm. Segmentation result is published as jsk_recognition_msgs/RectArray
18.101.1 Subscribing Topics

• ~input(sensor_msgs/Image)
  Input image

18.101.2 Published Topics

• ~output(jsk_recognition_msgs/RectArray)
  Segmentated bounding box
• ~debug(sensor_msgs/Image)
  Debug image

18.101.3 Sample

roslaunch jsk_perception sample_selective_search.launch

18.102 SingleChannelHistogram

Compute histogram of single channel image.
18.102.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image. It should has CV_8UC1 as encoding.
- ~input/mask (sensor_msgs/Image)
  Mask image. if ~use_mask is true, histogram is computed with this mask image.

18.102.2 Publishing Topic

- ~output (jsk_recognition_msgs/ColorHistogram)
  Histogram of ~input image.

18.102.3 Parameters

- ~use_mask (Boolean, default: false)
  If this parameter is set true, histogram is computed with mask image.
- ~hist_size (Integer, default: 10)
  The number of bins of histogram.
  This parameter can be changed by dynamic_reconfigure.
- ~min_value (Double, default: 0.0)
- ~max_value (Double, default: 256.0)
  Minimum and maximum value of histogram.
  These parameters can be changed by dynamic_reconfigure.

18.102.4 Sample

roslaunch jsk_perception sample_single_channel_histogram.launch
18.103 Skeletonization

The nodelet is used to obtain the single pixel skeleton of either edges, contours or regions. An important use of this nodelet is to fix broken edges on 2D image by first applying Dilation operation to the edges such that the broken edges intersect.

By using skeletonization, the dilated image single pixel skeleton structure can be obtained.

Note that this is different from erosion, as erosion erodes the edges without preserving the structure.

For more information refer to Wikipedia - Zhang-Suen thinning algorithm

18.103.1 Subscribing Topic

- `~input(sensor_msgs/Image)`
  
  Currently only supports “MONO8” (single channel 8 bit grayscale)

18.103.2 Publishing Topic

- `~image_output(sensor_msgs/Image)`
  
  float image containing skeleton info of the input image. 1 presents the skeleton.
  
  The output is the message with encoding type of 32FC1.
  
  You may need rviz or latest image_view to display them.
18.103.3 Parameters

- `~num_threads` (Int, default: 2)
  Number of threads for multi-threading computation.

18.103.4 Sample

```
roslaunch jsk_perception sample_skeletonization.launch
```

18.104 `sklearn_classifier.py`

18.104.1 What is this?

Load trained classifier model and classify with input vector array.

18.104.2 Subscribing Topic

- `~input (jsk_recognition_msgs/VectorArray)`
  Input vector array to do classification.

18.104.3 Publishing Topic

- `~output (jsk_recognition_msgs/ClassificationResult)`
  Classification result.

18.104.4 Parameters

- `~clf_path` (type: String, required)
  Trained classifier path. Currently we support `*.pkl.gz` file. To train the classifier, use `jsk_perception/sklearn_classifier_trainer.py`.

18.104.5 Sample

See here.
18.105 SLICSuperPixels


Output of this node is an image and each value means label index.

18.105.1 Subscribing Topic

- image (sensor_msgs/Image)
  Input image.

18.105.2 Publishing Topic

- ~output (sensor_msgs/Image)
  Output image. The encoding of image is CV_32SC1 and each element value means label index.
- ~debug (sensor_msgs/Image)
  Debug image, each border of cluster drawn by red contour. (optional, publish if ~publish_debug_images is true)
- ~debug/mean_color (sensor_msgs/Image)
  Debug image, each cluster is drawn by mean color of the cluster. (optional, publish if ~publish_debug_images is true)
- ~debug/center_grid (sensor_msgs/Image)
  Debug image, Center of each cluster is plotted by red dot. (optional, publish if ~publish_debug_images is true)

18.105.3 Parameters

- ~number_of_super_pixels (Integer, default: 100)
  The number of super pixels.
- ~weight (Integer, default: 4)
  Weight of metrics between color and pixel distance.
- ~publish_debug_images (Bool, default: false)
  Publish debug images (~debug, ~debug/mean_color, ~debug/centr_grid)
18.105.4 Sample

roslaunch jsk_perception sample_slic_super_pixel.launch

18.106 SlidingWindowObjectDetector

18.106.1 What is this?
This nodelet performs supervised object detection through binary support vector machine trained object classifier. The Nodelet uses a sliding window detection method as a raster scanning of the image. Currently, the detector is trained on Histogram of Oriented Gradients and HS Color histogram. To train this detector please use the custom implemented jsk_perception/sliding_window_object_detector_trainer_node node bundled in jsk_perception pkg. This manifest file contains specific configurations of the trainer, the feature dimensionalities, the detector window size and the output directories with filenames.

18.106.2 Usage
The nodelet can be configured to run as either:

- 1) a detector (DETECTOR)
   The nodelet simply loads the manifest and trainer and performs object detection.

- 2) a bootstraper (BOOTSTRAPER)
   This mode, is used to refine the trained classifier by accumulating the false positive detection in the environment to re-train the detector.

This method helps reduce the false positives.
Note when doing bootstrapping make sure that the object of interest is NOT in the environment.
The nodelet before doing bootstrapping will load the negative training bag file, read, and will write using the same name and will append the images from bootstrapping to the bag.
This is done to set all training set to similar time stamps.
18.106.3 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input image.

18.106.4 Publishing Topic

- `~output/image (sensor_msgs/Image)`
  Raw image marked with bounding boxes of detected objects.
- `~output/rects (jsk_recognition_msgs/RectArray)`
  Array of detected bounding boxes

18.106.5 Parameters

- `~run_type (string, required)`
  Run the nodelet as a detector or bootstraper. See above.
- `~trainer_manifest (string, default: sliding_window_trainer_manifest.xml)`
  Manifest file containing training parameters, which is one of outputs of `sliding_window_object_detector_trainer_node`.
- `override_manifest (bool, default: false)`
  Override parameters shown below after loading `~trainer_manifest`. Overridable parameters are …
- \~trainer\_path (string)
  Trained SVM model.
- \~swindow\_x (int)
- \~swindow\_y (int)
  Kernel size of sliding window.
- \~dataset\_path (string)
  Path to directory which contains dataset rosbag files.
  It should end with /
- \~object\_dataset\_filename (string)
- \~nonobject\_dataset\_filename (string)
  Rosbag file name of the object/background training set.

- \~image\_downsize (Int, default: 2)
  Reduces the image by this factor. (Smaller image dimensions makes processing faster)
  This parameter can be changed by dynamic\_reconfigure.

- \~scaling\_factor (float, default: -0.06)
  Scale factor for pyramidical scaling of the window. + value indices increase while - reduce.
  This parameter can be changed by dynamic\_reconfigure.

- \~stack\_size (int, default: 2)
  Speficies the number of times a window is to be changed for raster scanning.
  The changed factor for each traversal is determined by \~scaling\_factor
  This parameter can be changed by dynamic\_reconfigure.

- \~sliding\_window\_increment (int, default: 16)
  Speficies the number of pixels to shift the window for next detection (a.k.a. stride).
  This parameter can be changed by dynamic\_reconfigure.
Note that these parameters are critical in determining the detection rate and the speed of execution. Please fine tune the parameters accordingly to get the best performance.

### 18.106.6 Sample

```bash
roslaunch jsk_perception sample_sliding_window_object_detector.launch
```

### 18.107 sliding_window_object_detector_trainer_node

#### 18.107.1 What is this?

Node to train jsk_perception/SlidingWindowObjectDetector using binary support vector machine.

The object is assigned a label of +1 and -1 otherwise. The SVM used is from the OpenCV Library with default set to RBF Kernel and 10-Fold Cross Validations.
18.107.2 Parameters

- `~dataset_path` (string, required)
  
  Folder name where `~object_dataset_filename` and `~nonobject_dataset_filename` resides. It should end with `/`.
- `~object_dataset_filename` (string, required)
  
  Rosbag file name of the object (positive) training set. The bag file must contain `~object_dataset_topic` topic.
- `~object_dataset_topic` (string, default: `/dataset/roi`)
  
  Topic name of `sensor_msgs/Image` which is a set of positive training examples.
- `~nonobject_dataset_filename` (string, required)
  
  Rosbag file name of the non-object (negative) training set. The bag file must contain `~nonobject_dataset_topic` topic.
- `~nonobject_dataset_topic` (string, default: `/dataset/background/roi`)
  
  Topic name of `sensor_msgs/Image` which is a set of negative training examples.
- `~classifier_name` (string, required)
  
  Path to trained SVM classifier output file. `.xml` or `.yaml` format is supported.
- `~manifest_filename` (string, default: `sliding_window_trainer_manifest.xml`)
  
  Path to manifest file which contains parameters of the trainer such as trainer window size, save directory, etc. `.xml` or `.yaml` format is supported.
- `~swindow_x` (int, required)
- `~swindow_y` (int, required)
  
  Images in training dataset are resized to this size (width, height) before training SVM.
18.107.3 Sample

```bash
roslaunch jsk_perception sample_sliding_window_object_detector_trainer.launch
```

and wait a few minutes until “Trained Successfully” message appears.

18.108 Snakesegmentation

Snake segmentation based on cvSnakeImage.
This feature is not supported in OpenCV >= 3.

18.108.1 Subscribing Topics

- `~input` (sensor_msgs/Image)
  Input image.

18.108.2 Publishing Topics

- `~debug` (sensor_msgs/Image)
  Debug image.

18.108.3 Parameters

- `~alpha` (double, default: 0.1)
  weight of connectivity energy
- `~beta` (double, default: 0.1)
  weight of curvature energy
- `~gamma` (double, default: 0.1)
  weight of image energy
- `~window_size` (Integer, default: 3)
- `~max_iterations` (Integer, default: 1000)
- `~epsilon` (double, default: 0.1)
  
  epsilon value of convergence

### 18.108.4 Sample

```
roslaunch jsk_perception sample_snake_segmentation.launch
```

### 18.109 solidity_rag_merge.py

#### 18.109.1 What is this?

(red region in top image is segmented using euclidean clustering, yellow, purple and red region is segmented by solidity_rag_merge.)

Segment image region which has high solidity, using RAG. It firstly segments image with SLIC Superpixels, generates RAG based on solidity, and then merges regions.
### 18.109.2 Subscribing Topic

- ~input (sensor_msgs/Image)
  
  Raw image. In most cases, this is mask or depth image.
- ~input/mask (sensor_msgs/Image)
  
  Region where the label should be ignored.

### 18.109.3 Publishing Topic

#### Default Topics

- ~output (sensor_msgs/Image, encoding: 32SC1)
  
  Label image of region with high solidity.

#### Optional Topics

Following topics are published when ~debug is True, see Parameters for more detail.
- ~debug/slic (sensor_msgs/Image, encoding: 32SC1)
  
  Label image segmented with SLIC Superpixels.
- ~debug/rag (sensor_msgs/Image)
  
  RAG overlaid image.
- ~debug/label_viz (sensor_msgs/Image)
  
  Label color overlaid image. This function can be replaced by label_image_decomposer.py but I added for easy debugging.

### 18.109.4 Parameters

- ~approximate_sync (Bool, default: False)
  
  Whether to use approximate for input topics.
- ~debug (Bool, default: True)
  
  Debug mode. In this mode, the node publishes ~debug/* topics.
18.110 SparseImageDecoder

18.110.1 What is this?

Decode `jsk_recognition_msgs/SparseImage` and publish it as a mask image. The combination of `jsk_perception/SparseImageEncoder` and this node provides almost the same function as image thresholding.

18.110.2 Subscribing Topic

- `sparse_image (jsk_recognition_msgs/SparseImage)`
  Input indices to decode.

18.110.3 Publishing Topic

- `sparse/image_decoded (sensor_msgs/Image)`
  Decoded mask image.
  The encoding is `MONO8`.

18.110.4 Sample

```bash
roslaunch jsk_perception sample_sparse_image_encoder_decoder.launch
```
18.111 SparseImageEncoder

18.111.1 What is this?

Pick up bright pixels from an image and encode the indices.
The combination of this node and jsk_perception/SparseImageDecoder provides almost the same function as image thresholding.
Note that this node is irrelevant to sparse coding or auto encoder.

18.111.2 Subscribing Topic

- image (sensor_msgs/Image)
  Input image to encode.

18.111.3 Publishing Topic

- sparse_image (jsk_recognition_msgs/SparseImage)
  Encoded indices.
  If either width or height of the input image is greater than 256, data32 field is used.
  If it is less than or equal to 256, then data16 field is used instead.

18.111.4 Parameters

- ~rate (Float, default: 3.0)
  Maximum publishing rate [Hz].
- ~print_point_num (Bool, default: False)
  Print number of encoded pixels as NODELET_INFO.
18.112.2 Subscribing Topic

- `~input/label(sensor_msgs/Image)`
  Label image to find target labels.
- `~input/image(sensor_msgs/Image)`
  Raw image which is used to speak with `jsk_gui_msgs/SlackMessage`.

The `speak` means:

- Speak with `sound_play`.
- Publish `std_msgs/String` message.
- Publish `jsk_gui_msgs/SlackMessage`.

And this node speaks if target labels are found in input label image.
18.112.3 Publishing Topic

- `~output/string(std_msgs/String)`
  Output message with string.
- `~output/slack_msg(jsk_gui_msgs/SlackMessage)`
  Output message to post to slack. This node can be easily integrated with `jsk_tools/post_to_slack_server`.

18.112.4 Parameters

- `~label_names (List of String, required)`
  The name of labels.
- `~target_labels (List of Int, required if ~target_label_names is not specified)`
- `~target_label_names (List of String, required if ~target_labels is not specified)`
  Target label.
- `~sound (Bool, default: true)`
  Flag of using sound_play.
- `~min_label_region (Float, default: 0.1)`
  Threshold of label region to recognize as label is detected.
- `~approximate_sync (Bool, default: False)`
  Whether to use approximate for input topics.
- `~queue_size (Int, default: 10)`
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- `~slop (Float, default: 0.1)`
  How many seconds you allow about the difference of timestamp when you specify `~approximate_sync`.

18.112.5 Sample

```
roslaunch jsk_perception sample_speak_when_label_found.launch
```
18.113 split_fore_background.py

18.113.1 What is this?

Publish foreground and background mask image splitted by local maximum of depth image.

18.113.2 Subscribing Topic

- `~input` ([`sensor_msgs/Image`, `encoding: 16UC1` or `32FC1`])
  
  Depth image.
18.113.3 Publishing Topic

- `~output/fg_mask` (sensor_msgs/Image, encoding: 8UC1)
  Mask image to extract foreground.
- `~output/bg_mask` (sensor_msgs/Image, encoding: 8UC1)
  Mask image to extract background.

18.114 split_image.py

18.114.1 What Is This

Split the input image at equal intervals vertically and horizontally.

18.114.2 Subscribing Topic

- `~input` (sensor_msgs/Image)
  Input image to be split.
18.114.3 Publishing Topic

- `~output/vertical(vertical_index)/horizontal(horizontal_index)` (sensor_msgs/Image)

  Split images. The number of published topics is `~vertical_splits * ~horizontal_splits`.

18.114.4 Parameters

- `~vertical_splits` (int, default: 1)
  The number of parts the input image is split vertically.
- `~horizontal_splits` (int, default: 1)
  The number of parts the input image is split horizontally.

18.114.5 Sample

```
roslaunch jsk_perception sample_split_image.launch
```

18.115 SubtractMaskImage

Subtract one mask image from another.

18.115.1 Subscribing Topic

- `~input/src1` (sensor_msgs/Image)
- `~input/src2` (sensor_msgs/Image)

  Input mask images.
18.115.2 Publishing Topic

- \texttt{\~output} (\texttt{sensor_msgs/Image})
  Subtracted mask image. (output = input/src1 - input/src2)

18.115.3 Parameters

- \texttt{\~approximate_sync} (Bool, default: false)
  Approximately synchronize \texttt{\~input/src1} and \texttt{\~input/src2} if it's true.
- \texttt{\~queue_size} (Int, default: 100)
  Queue size

18.115.4 Sample

\texttt{roslaunch jsk_perception sample_subtract_mask_image.launch}

18.116 TabletopColorDifferenceLikelihood

Compute likelihood of object on table based on color space histogram. In tabletop object detection, color is a good feature to detect objects. Idea is simple.

First compute “average” color of the pixels of the polygon. Second compute “distance” between the “average” color and the pixel for each pixels in the polygon.

However, there is problem in “average” and “distance”. If polygon is clean, we can get expected result from simple average and distance computation. However if polygon is cluttered, we will get unexpected result because average of pixels is feasible feature.

TabletopColorDifferenceLikelihood uses histogram to resolve the issue.

1. Compute histogram of the pixels in a polygon.
2. Mark top N histograms which covers the area which is computed by multiplying given ratio and all the area of histograms.
3. Compute minimum distance between a pixel and the N histograms for each pixel in the polygon.
18.116.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Single channel color image
- `~input/polylons (jsk_recognition_msgs/PolygonArray)`
  Polygon messages
- `~input/camera_info (sensor_msgs/CameraInfo)`
  Input camera info

18.116.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  Result distance image.
- `~debug/histogram_image (sensor_msgs/Image)`
  Image to visualize histogram. Red bars are extracted top N histograms.
- `~debug/polygon_image (sensor_msgs/Image)`
  Image to visualize polygons projected on 2-D image.

18.116.3 Parameters

- `~tf_queue_size (Int, default: 10)`
  Queue length of tf message filters
- `~cyclic_value (Bool, default: True)`
  Set to true if the channel is circular value.
- `~bin_size (Int, default: 30)`
  The number of bins in histogram.
  This parameter can be changed by dynamic_reconfigure.
- `~pixel_min_value (Int, default: 0)`
- `~pixel_max_value (Int, default: 180)`
  Minimum and maximum value in the color space.
  This parameter can be changed by dynamic_reconfigure.
- `~histogram_top_n_ratio (Float, default: 0.5)`
  Ratio of extracting top N histograms.
  This parameter can be changed by dynamic_reconfigure.
18.116.4 Sample

```bash
roslaunch jsk_perception sample_tabletop_color_difference_likelihood.launch
```

18.117 tile_image.py

18.117.1 What is this?

Publish an image topic by tiling image topics specified.
### 18.117.2 Subscribing Topic

The subscribing topics should be specified with rosparam. See Parameters.

### 18.117.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Tiled image.

### 18.117.4 Parameters

- **input_topics (type: StringArray, required)**
  
  Input topic names should be specified like:

  ```
  <node name="tile_image" pkg="jsk_perception" type="tile_image.py" output="screen">
    <rosparam>
      input_topics: [img1/output, img2/output, img3/output, img4/output]
    </rosparam>
  </node>
  ```

- **no_sync (type: Bool, default: False)**
  
  Set `no_sync` parameter true if you do not want to synchronize timestamps of `input_topics`.

- **queue_size (type: Int, default: 10)**
  

- **slop (type: Float, default: 1.0)**
  
  The slop time in second for `message_filters.ApproximateTimeSynchronizer`.

- **draw_topic_name (type: Bool, default: False)**
  
  Draw topic name on each image.

- **font_scale (type: Float, default: 0.8)**
  
  Font size to draw topic names.

- **shape (type: FloatArray, default: None)**
  
  Tile shape like `[2, 4]` in `[Y, X]` order. If `None`, shape is automatically decided to be square as much as possible.

### 18.117.5 CLI

```
rosrun jsk_perception tile_image.py _input_topics:='[img1/output, img2/output]'
```
18.117.6 Sample

```bash
roslaunch jsk_perception tile_image.launch
```

18.118 TopHat

See MorphologicalOperator

18.119 UnapplyMaskImage

Unapply mask image to the size of original image.

18.119.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Masked image.

- `~input/mask (sensor_msgs/Image)`
  
  Mask image.
18.119.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  Unmasked image. The region outside of mask image is filled by black (0).

18.119.3 Parameters

- `~approximate_sync (Bool, default: false)`
  Approximately synchronize inputs if it’s true.

18.119.4 Sample

```
roslaunch jsk_perception sample_unapply_mask_image.launch
```

18.120 unwrap_histogram_with_range_array.py

![Histogram Plot](image)
18.120.1 What is this?

Unwrap a HistogramWithRangeArray message and publish its element.

18.120.2 Subscribing Topic

- ~input (jsk_recognition_msgs/HistogramWithRangeArray)
  Histogram array.

18.120.3 Publishing Topic

- ~output (jsk_recognition_msgs/HistogramWithRange)
  Histogram

18.120.4 Parameters

- ~index (Int, default: 0)
  Histogram index to be extracted from ~input.

18.120.5 Sample

```
roslaunch jsk_perception sample_unwrap_histogram_with_range_array.launch
```
18.121 vgg16_object_recognition.py

18.121.1 What is this?

Recognize object with VGG16 net by resizing input image to 224 x 224. This node requires pretrained chainer model. For training VGG16 net, please refer to wkentaro/vgg16

18.121.2 Subscribing Topic

• ~input (sensor_msgs/Image)
  Input image.

18.121.3 Publishing Topic

• ~output (jsk_recognition_msgs/ClassificationResult)
  Classification result of input image.
• ~debug/net_input (sensor_msgs/Image)
  Resized image to 224 x 224.
18.121.4 Parameters

- `~gpu` (Int, Default: -1)
  GPU id. -1 represents CPU mode.
- `~target_names` (List of String, Required)
  Target names for classification.
- `~model_name` (String, Required)
  Currently vgg16 or vgg16_batch_normalization is only supported. See models in $(rospack find jsk_recognition_utils)/python/jsk_recognition_utils/chainermodels.
- `~model_file` (String, Required)
  Trained model file.
- `use_mask` (Bool, Default: False)
  If true, topic `~input/mask` is enabled.
- `~approximate_sync` (Bool, Default: False)
  Use approximate synchronization policy.
- `~queue_size` (Int, Default: 10)
  Queue size for synchronization.
- `~slop` (Float, Default: 0.1)
  Slop for approximate sync.

18.121.5 Example

```
roslaunch jsk_perception sample_vgg16_object_recognition.launch
```
18.122 VirtualCameraMono

Calculate perspective transformation from TF frame and apply it to the input image.

18.122.1 Subscribing Topics

- **image (sensor_msgs/Image)**
  Input image.
- **camera_info (sensor_msgs/CameraInfo)**
  Input camera info.
  This topic name is resolved from image by image_transport.
- **view_point (geometry_msgs/TransformStamped, optional)**
  Pose of virtual camera relative to ~frame_id.
  This pose is initialized from ~initial_pos and ~initial_rot.
- **target_polygon (geometry_msgs/PolygonStamped, optional)**
  Target plane polygon.
  The polygon is initialized from \([0, 1, 0], [0, -1, 0], [0, -1, -1], [0, 1, -1]\).
18.122.2 Publishing Topics

- ~image(sensor_msgs/Image)
  Output image to which perspective transformation was applied.
- ~camera_info(sensor_msgs/CameraInfo)
  Camera info of virtual camera.

18.122.3 Parameters

- ~frame_id(String, default: /elevator_inside_panel)
  Target frame ID.
- ~child_frame_id(String, default: /virtual_camera_frame)
  Frame ID of virtual camera used in published topics.
- ~queue_size(Int, default: 1)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- ~initial_pos(List of Float, default: [0.7, 0.0, 0.0])
  Initial position of virtual camera relative to ~frame_id.
- ~initial_rot(List of Float, default: [0.5, 0.5, -0.5, -0.5])
  Initial rotation of virtual camera relative to ~frame_id in quaternion.
- ~interpolation_method(Int, default: 1)
  Choose from INTER_NEAREST (0), INTER_LINEAR (1), INTER_AREA (2), INTER_CUBIC (3) and INTER_LANCZOS4 (4).
  This parameter can be changed by dynamic_reconfigure.

18.122.4 Sample

```bash
roslaunch jsk_perception sample_virtual_camera_mono.launch
```
18.123 vqa_node.py

The ROS node for Visual Question Answering (VQA).

Q: what does this image describe?
A: a young man wearing glasses holding up his hand
18.123.1 System Configuration

This node requires to work with the Docker Container for inference. Please build the container at first following Setup instruction.

18.123.2 Setup

Prerequisite

This node requires NVIDIA GPU and more than 4GB GRAM to work properly. You have to install nvidia-container-toolkit for using GPU with docker. Please follow official instruction.

Build the docker image

You have to build the docker image of OFA

```bash
roscd jsk_perception/docker
make
```
18.123.3 Subscribing topic

- ~image (sensor_msgs/Image)
  Input image

18.123.4 Publishing topic

- ~result (jsk_recognition_msgs/VQAResult)
  VQA result
- ~result/image (sensor_msgs/Image)
  Images used for inference
- ~visualize (std_msgs/String)
  VQA question and answer to visualize

18.123.5 Action topic

- ~inference_server/goal (jsk_recognition_msgs/VQATaskActionGoal)
  VQA request with custom questions and image
- ~inference_server/result (jsk_recognition_msgs/VQATaskActionResult)
  VQA result of ~inference_server/goal

18.123.6 Parameters

- ~host (String, default: localhost)
  The host name or IP of inference container
- ~port (Integer, default: 8080)
  The HTTP port of inference container

18.123.7 Dynamic Reconfigure Parameters

- ~questions (string, default: what does this image describe?)
  Default questions used for subscribing image topic.

18.123.8 Sample

Run inference container on another host or another terminal

In the remote GPU machine,

cd jsk_recognition/jsk_perception/docker
./run_jsk_vil_api --port (Your vacant port) --ofa_task caption --ofa_model_scale huge
--ofa_task should be caption or vqa. Empirically, the output results are more natural for VQA tasks with the Caption model than with the VQA model in OFA.

If you encounter GPU ram shortage, please make ofa_model_scale_large.

In the ROS machine,

```bash
roslaunch jsk_perception vqa.launch port:=(Your inference container port) host:=(Your inference container host) VQA_INPUT_IMAGE:=(Your image topic name) gui:=true
```

Run both inference container and ros node in single host

```bash
roslaunch jsk_perception vqa.launch run_api:=true VQA_INPUT_IMAGE:=(Your image topic name) gui:=true
```

### 18.124 YCCDecomposer

Decompose BGR/RGB image into separate planes in YCbCr color space.
18.124.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input image.

18.124.2 Publishing Topic

- ~output/y (sensor_msgs/Image)
- ~output/cr (sensor_msgs/Image)
- ~output/cb (sensor_msgs/Image)
  Y, Cr and Cb separated planes. Each image has CV_8U encoding.

18.124.3 Sample

```bash
roslaunch jsk_perception sample_ycc_decomposer.launch
```
JSK_PCL_ROS

jsk_pcl_ros is a package to provide some programs using pcl.
This package provides some programs as nodelet.

19.1 Depth Camera Calibration(Kinect,Xtion,Primesense)
19.1.1 Two Main Steps:

• 1. Camera Intrinsic Calibration: Intrinsic Calibration Please refer to ros RGB camera calibration tutorial >> ros wiki
• 2. Depth Calibration: Depth Calibration related to the distance z, optical frame pixel u v and global pixel offset.

19.1.2 You need:

• Chessboard: Make sure ur Application Range, if u wanna use camera in short range(0.5m~2m) choose small chessboard, otherwise plz choose a larger one.
• Depth Camera: Kinect one, Xtion, Primesense. (There may be some problem when using primesense, check here to install the newest openni2, perhaps u need to do apt-get remove libopenni2-0 first)
• Good PC with ubuntu and ros installed: We only tested in Lenovo thinkpad series.
• jsk_pcl_ros: jsk package

19.1.3 Camera Intrinsic Calibration:

• Please follow this tutorial and when u finished calibration(4 features become green), wait patiently until u can click upload, calibration file will be right there in ~/.ros/camera_info/***.yaml waiting. Check the openni2_launch.launch file or openni2_local.launch if u use JSK package to edit the path.

19.1.4 Depth Calibration(Available only in jsk_pcl_ros package):

We assume the intrinsic calibration has been performed well.

• Plug in ur depth camera to your favourite USB port and run roslaunch jsk_pcl_ros openni2_local.launch and roslaunch jsk_pcl_ros openni2_remote.launch (Load the camera intrinsic calibration file)
• Do roscd jsk_pcl_ros and cd launch, find file depth_error.launch and edit param rect0_size_x rect0_size_y and grid0_size_x grid0_size_y according to your chessboard. Then roslaunch jsk_pcl_ros depth_error.launch
• Do rosrun rviz rviz and subscribe to 3 topics(two pointcloud2 and one Pose)
    1. Pose /checkerdetector/objectdetection_pose
    2. Raw Pointcloud /camera_remote_uncalibrated/depth_registered/points
    3. Calibrated Pointcloud /camera_remote/depth_registered/points
You will see the Error between Pose(Estimated by rgb camera while looking at chessboard) and uncalibrated point-cloud.

• Open another Terminal and run rosrun jsk_pcl_ros depth_error_calibration.py --model quadratic-uv-quadratic-abs and move the chessboard slowly while watching to the image window. The edges of the image should be covered and the range(due to your application) should also be covered as more as possible.
• Open new Terminal and run `rosrun image_view image_view image:=/depth_error_logistic_regression/frequency_image`. You can observe which point the chessboard passes on the window. If it never appears, setting `approximate_sync` in `jsk_pcl_ros/launch/depth_error.launch` to true might help you. (Please see the example below.)

```xml
<node pkg="jsk_pcl_ros" type="depth_image_error" name="depth_image_error" output="screen">
  <remap from="~image" to="$(arg DEPTH_IMAGE)" />
  <remap from="~point" to="/checkerdetector/corner_point" />
  <remap from="~camera_info" to="$(arg CAMERA_INFO_TOPIC)" />
  <rosparam>
    approximate_sync: true
  </rosparam>
</node>
```

• Checking the window output and the Rviz output when you find the calibrated pointcloud overlaps the Pose vector. Ctrl+c in this Terminal and enter y to save the calibration file. Edit `openni2_remote.launch` file and find the param `depth_calibration_file`, add the path of your calibration file.

• Finish and Check it again.
19.2 Triple RGB-D Fusion using Two Cameras

Kentaro Wada, Yuto Uchimi, and Shun Hasegawa, University of Tokyo, JSK Laboratory

19.2.1 Why?

Depth information came from a mono RGB-D camera has many lacks of depth in each frame, because of the noise of IR sensors and light reflection in dynamic scenes. This leads the use of stereo sensor, but it also has difficulty of pixel matching of textureless regions in a frame. These problems motivate us to fuse RGB-D sensor inputs using multiple RGB-D cameras.

19.2.2 How?

The system overview is shown below and the input is 2 rgb images, 2 depths, and camera parameters of both two cameras.

All sensor informations, rgb images and depths, is transformed to the frame of left camera, and fused in the coordinate. The rgb image and depth from left camera $I_{left}(c_{left}), D_{left}(c_{left})$ is inheritly in left frame $(c_{left})$, so we don’t need to transform it. The right ones, $I_{right}(c_{right})$ and $D_{right}(c_{right})$, is converted to point cloud $P_{right}(c_{right})$ in right frame $(c_{right})$. The point cloud is converted to rgb image $I_{right}(c_{left})$ and depth $D_{right}(c_{left})$ in left frame. Stereo depth is given from stereo matching using camera parameters and rgb images from the two cameras, and usually the output rgb image and depth is in left frame: $I_{stereo}(c_{left}), D_{stereo}(c_{left})$.

In above paragraph, we describe the way to acquire rgb images and depths from ir sensors in two rgb-d cameras and stereo matching. Now we have 3 rgb images: $I_{left}(c_{left}), I_{right}(c_{left}), I_{stereo}(c_{left})$, and 3 depths: $D_{left}(c_{left}), D_{right}(c_{left}), D_{stereo}(c_{left})$, in left frame, then we explain the operation of fusions of them. For rgb images, we use
or operation, and avg operation for depths.

\[
I_{\text{fused}}(c_{\text{left}}) = Op_{\text{or}}(I_{\text{left}}(c_{\text{left}}), I_{\text{right}}(c_{\text{left}}), I_{\text{stereo}}(c_{\text{left}}))
= Op_{\text{or}}(I_{\text{left}}, I_{\text{right}}, I_{\text{stereo}})
\]

\[
D_{\text{fused}}(c_{\text{left}}) = Op_{\text{avg}}(D_{\text{left}}(c_{\text{left}}), D_{\text{right}}(c_{\text{left}}), D_{\text{stereo}}(c_{\text{left}}))
= Op_{\text{avg}}(D_{\text{left}}, D_{\text{right}}, D_{\text{stereo}})
= \left( D_{\text{left}} + D_{\text{right}} + D_{\text{stereo}} \right) / 3
\]

The right bottom pictures in above figure of system figure shows fused result of rgb image and depth: \( I_{\text{fused}}(c_{\text{left}}) \) and \( D_{\text{fused}}(c_{\text{left}}) \). We use or operation for rgb image to avoid cracks in the output image caused by error in calibration, and avg for depth to reduce noise assuming that depth is changed smoothly in frame.

### 19.2.3 Sample

You can see implementations by using sample in `jsk_recognition`. There are rosbag, and you can see triple depth fusion by streaming.

```bash
roslaunch jsk_pcl_ros sample_fuse_depth_images.launch
```

### Hardware

For real world use, you need hardware of stereo rgb-d camera. Any rgb-d cameras should be fine, but for your information, we write here we used two Astra Mini S cameras in the experiment.
19.2.4 Acknowledgement

Yuto Uchimi contributed this work by calibrating both intrinsic and extrinsic camera parameters for stereo, and Shun Hasegawa created hardware attachment to combine two cameras. Kentaro Wada proposed, implemented the stereo rgb-d system and wrote this document.

19.3 To Test Some Samples

Please be careful about the nodelet manager name when execute some sample launches.

Because the nodelet manager name is different between groovy version and hydro version in openni.launch, you have to replace the nodelet manager name when use in groovy as below.

From

```
/camera_nodelet_manager
```

To

```
/camera/camera_nodelet_manager
```

19.4 AddColorFromImage

Add color to pointcloud (no need to be organized) from image and camera info.
19.4.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud to be colorized, which is not necessarily organized.
- `~input/image (sensor_msgs/Image)`
  Input image for the colors.
- `~input/camera_info (sensor_msgs/CameraInfo)`
  Input camera info.

19.4.2 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Output colorized pointcloud.

19.4.3 Sample

```
roslaunch jsk_pcl_ros add_color_from_image.launch
```

19.5 AddColorFromImageToOrganized

![Input Image](image1.png) + ![Input Cloud](image2.png) = ![Output Cloud](image3.png)

Add color to organized pointcloud from image. Use AddColorFromImage for unorganized pointcloud.

19.5.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input organized pointcloud to be colorized.
- `~input/image (sensor_msgs/Image)`
  Input image for the colors.
19.5.2 Publishing Topic

- `~output(sensor_msgs/PointCloud2)`
  Output colorized pointcloud.

19.5.3 Sample

```bash
roslaunch jsk_pcl_ros add_color_from_image_to_organized.launch
```

19.6 AttentionClipper

19.6.1 What Is This

It retrieves `sensor_msgs/CameraInfo` and publish `sensor_msgs/CameraInfo` with ROI filled and retrieves `sensor_msgs/PointCloud2` and publish `pcl_msgs/PointIndices`.

You can specify the pose and size of the interest bounding box and jsk_pcl/AttentionClipper returns ROI to see the object.
19.6.2 Note

AttentionClipper does not work properly on Kinetic + PCL 1.8.0, and you can avoid the problem with SSE disabled PCL.

Please see here for more information.

Original issue: attention_clipper does not work properly on Kinetic + PCL1.8.0

19.6.3 Subscribing Topic

- `~input (sensor_msgs/CameraInfo)`
  Original camera info. (You don’t need camera info when you only use pointcloud.)

- `~input/points (sensor_msgs/PointCloud2)`
  Original pointcloud.

- `~input/pose (geometry_msgs/PoseStamped)`

- `~input/box (jsk_recognition_msgs/BoundingBox)`
  Specify the pose of the bounding box. Timestamp will be ignored and camera info’s timestamp will be used. If you use `~input/box`, you can change the size of attention region. There callbacks are only enabled if `~use_multiple_attention` is false.

- `~input/pose_array (geometry_msgs/PoseArray)`

- `~input/box_array (jsk_recognition_msgs/BoundingBoxArray)`
  It’s an array version of `~input/pose` and `~input/box`. There callbacks are only enabled if `~use_multiple_attention` is true.

19.6.4 Publishing Topic

- `~output (sensor_msgs/CameraInfo)`
  This camera info is same with `~input` except for roi field. (only when `~input` is published.)

- `~output/box_array (jsk_recognition_msgs/BoundingBoxArray)`
  Array of bounding boxes representing the interest regions.

- `~output/mask (sensor_msgs/Image)`
  Mask image to mask the regions of specified interest. (only when `~input` is published.)

- `~output/point_indices (pcl_msgs/PointIndices)`
  Indices of `~input/points` which are inside of interest regions.

- `~output/cluster_point_indices (jsk_recognition_msgs/ClusterPointIndices)`
  Cluster point indices of `~input/points` which are inside of interest regions.
19.6.5 Parameter

- \texttt{~use\_multiple\_attention} (Boolean, default: False)
  
  If you want to enable multiple attentions, please set this variable True.

- \texttt{~dimension\_x} (Double, default: 0.1)
- \texttt{~dimension\_y} (Double, default: 0.1)
- \texttt{~dimension\_z} (Double, default: 0.1)

  Size of bounding box. Available only if \texttt{~use\_multiple\_attention} is false.

- \texttt{~frame\_id} (String, default: base\_link)

  Frame id of attention region. Available only if \texttt{~use\_multiple\_attention} is false.

- \texttt{~initial\_pos} (Array of double, default: None):

  Initial pose of interesting region. Available only if \texttt{~use\_multiple\_attention} is false.

- \texttt{~initial\_rot} (Array of double, default: None):

  Initial orientation of interesting region. The value should be represented in [roll, pitch, yaw]. Available only if \texttt{~use\_multiple\_attention} is false.

- \texttt{~initial\_pos\_list} (Array of array of double, default: None)
- \texttt{~initial\_rot\_list} (Array of array of double, default: None)
- \texttt{~frame\_id\_list} (Array of string, default: None)
- \texttt{~dimensions} (Array of array of string, default: None)
- \texttt{~prefixes} (Array of string, default: None)

  Position, Rotation, frame id, prefix and Dimensions of multiple attention regions respectively. \texttt{~initial\_pos\_list} should follow \([[[x, y, z], ...]]\), \texttt{~initial\_rot\_list} should follow \([[[rx, ry, rz], ...]]\) and \texttt{~dimensions} should follow \([[[x, y, z], ...]]\). \texttt{~prefixes} \([prefix1, prefix2, ...]\). These prefixes will add to the /point\_indices and advertise Available only if \texttt{~use\_multiple\_attention} is true.

- \texttt{~negative} (Boolean, default: False)

  Publish points which are not inside of attention regions if this parameter is true.

19.6.6 For use

If you want to get PointCloud2 which are inside of interest regions, you should use ExtractIndices node like in sample\_attention\_clipper.launch.
19.6.7 Sample

```
roslaunch jsk_pcl_ros sample_attention_clipper.launch
```

19.7 BilateralFilter

19.7.1 What Is This

Apply bilateral filter to specified point cloud.
The filter reduces noise, preserves edges and smooth point cloud.

19.7.2 Subscribing Topic

- `~input` (sensor_msgs/PointCloud2)
  Original point cloud.

19.7.3 Publishing Topic

- `~output` (sensor_msgs/PointCloud2)
  Filtered point cloud.

19.7.4 Parameter

- `~sigma_s` (Float, default: 15.0)
  Standard deviation of Gaussian (a.k.a. spatial sigma).
  This parameter can be changed by `dynamic_reconfigure`.
- `~sigma_r` (Float, default: 0.05)
Standard deviation of gaussian used to control how much an adjacent pixel is downweighted because of the intensity difference (a.k.a. range sigma).
This parameter can be changed by `dynamic_reconfigure`.

### 19.7.5 Sample

```
roslaunch jsk_pcl_ros sample_bilateral_filter.launch
```

### 19.8 BorderEstimator

Convert pointcloud into range image and detect border

#### 19.8.1 Subscribing Topic

- `~input` (sensor_msgs/PointCloud2)
  - Input pointcloud
- `~input_camera_info` (sensor_msgs/CameraInfo)
  - Camera info is used to reconstruct organized range image when `~model_type=planar`
19.8.2 Publishing Topic

• ~output_cloud(sensor_msgs/PointCloud)
  Pointcloud converted as range image. If ~model_type is planar, it should be equal to original organized pointcloud. When ~model_type is laser, original pointcloud and ~output_cloud is not same.
• ~output_border_indices(pcl_msgs/PointIndices)
  Indices of border
• ~output_veil_indices(pcl_msgs/PointIndices)
  Indices of veil points
• ~output_shadow_indices(pcl_msgs/PointIndices)
  Indices of shadow edge
• output_range_image(sensor_msgs/Image)
  Range image

19.8.3 Parameters

• ~model_type(String, default: planar)
  Choose model type of range image by this topic. planar, laser or sphere is allowed.
• ~angular_resolution(Double, default: 0.5)
  Angular resolution of range image. Only does laser model use this parameter.
• ~noise_level(Double, default: 0.1)
  Noise level of pointcloud. Only does laser model use this parameter.
• ~min_range(Double, default: 0.0)
  Minimum distance to take into account range image. Only does laser model use this parameter.
• ~border_size(Integer, default: 0)
  Border size to remove from range image.
• ~max_angle_height(Double, default: 2pi)
  Maximum angle height of range image. Only does laser model use this parameter.
• ~max_angle_width(Double, default: 2pi)
  Maximum angle width of range image. Only does laser model use this parameter.

19.8.4 Sample

roslaunch jsk_pcl_ros sample_border_estimator.launch
19.9 BoundingBoxFilter

Filter bounding boxes (jsk_recognition_msgs/BoundingBoxArray) with threshold of bounding box dimensions. Optionally with indices (jsk_recognition_msgs/ClusterPointIndices).

19.9.1 Publishing Topics

- `~output_box (jsk_recognition_msgs/BoundingBoxArray)`
  
  Filtered bounding boxes.

- `~output_indices (jsk_recognition_msgs/ClusterPointIndices)`
  
  Cluster indices filtered with indices of filtered bounding boxes. Published only with `~with_indices`. 
19.9.2 Subscribing Topics

- ~input_box (jsk_recognition_msgs/BoundingBoxArray)
  Input bounding boxes.

- ~input_indices (jsk_recognition_msgs/ClusterPointIndices)
  Input cluster point indices. Subscribed only with ~with_indices.

19.9.3 Parameters

- ~with_indices (bool, default: true, optional)
  Flag to enable filtering cluster indices also.

- ~use_x_dimension, ~use_y_dimension, ~use_z_dimension (bool, default: false, optional, dynparam)
  Flag to filtering with x, y, or z dimensions.

- ~x_dimension_min, ~y_dimension_min, ~z_dimension_min (float, default: 0.1, optional, dynparam)
  Minimum threshold of x, y, or z dimensions.

- ~x_dimension_max, ~y_dimension_max, ~z_dimension_max (float, default: 0.1, optional, dynparam)
  Maximum threshold of x, y, or z dimensions.

- ~filter_limit_negative (bool, default: false, optional, dynparam)
  Set to true if we want to return the data outside [filter_limit_min; filter_limit_max].

19.9.4 Sample

roslaunch sample_bounding_box_filter.launch
19.10 BoundingBoxOcclusionRejector

Rejects bounding boxes which occlude target object.

19.10.1 Publishing Topics

- `~output (jsk_recognition_msgs/BoundingBoxArray)`
  Occlusion free candidate bounding boxes.
- `~output/target_image (sensor_msgs/Image)`
  Rendered mask image of target bounding boxes.
- `~output/candidate_image (sensor_msgs/Image)`
  Rendered mask image of candidate bounding boxes.

19.10.2 Subscribing Topics

- `~input/camera_info (sensor_msgs/CameraInfo)`
  CameraInfo of sensor.
- `~input/target_boxes (jsk_recognition_msgs/BoundingBoxArray)`
  BoundingBox array to represent target objects to see.
- `~input/candidate_boxes (jsk_recognition_msgs/BoundingBoxArray)`
  BoundingBox array of candidate objects.

Note that frame ID of all 3 input topics should be the same.
19.10.3 Sample

```bash
roslaunch jsk_pcl_ros sample_boundingbox_occlusion_rejector.launch
```

19.11 `calculate_polygon_from_imu.py`

19.11.1 What Is This

It retrieves `sensor_msgs/Imu` and publish perpendicular plane as polygon array.

19.11.2 Subscribing Topic

- `imu_data (sensor_msgs/Imu)`
  
  Imu data. Only `linear_acceleration` field will be used to calculate polygon.

19.11.3 Publishing Topic

- `polygon_array (jsk_recognition_msgs/PolygonArray)`
  
  Output plane polygon array.

- `model_coefficients_array (jsk_recognition_msgs/ModelCoefficientsArray)`
  
  Normal vector of plane.
19.11.4 Parameters

None.

19.11.5 Sample

```
roslaunch jsk_pcl_ros sample_calculate_polygon_from_imu.launch
```

19.12 CaptureStereoSynchronizer

19.12.1 What Is This

A nodelet to capture training data of stereo cameras. It subscribes several messages with synchronizing timestamp and republish them into `~output` namespace.

19.12.2 Subscribing Topic

- `~input/pose` (geometry_msgs/PoseStamped)
  Pose of checkerboard
- `~input/mask` (sensor_msgs/Image)
  Mask image of the object
- `~input/mask_indices` (pcl_msgs/PointIndices)
  Pointcloud indices of the object
• ~input/left_image(sensor_msgs/Image)
  Left camera image
• ~input/left_camera_info(sensor_msgs/CameraInfo)
  Left camera parameter
• ~input/right_camera_info(sensor_msgs/CameraInfo)
  Right camera parameter
• ~input/disparity(stereo_msgs/DisparityImage)
  Disparity image of the stereo camera.

19.12.3 Publishing Topic

• ~output/pose(geometry_msgs/PoseStamped)
• ~output/mask(sensor_msgs/Image)
• ~output/mask_indices(pcl_msgs/PointIndices)
• ~output/left_image(sensor_msgs/Image)
• ~output/left_camera_info(sensor_msgs/CameraInfo)
• ~output/right_camera_info(sensor_msgs/CameraInfo)
• ~output/disparity(stereo_msgs/DisparityImage)
  These topics are the same message to the ~input/foo messages but all of them are republished only if input messages are synchronized.
• ~output/count(std_msgs/Int32)
  Number of sample.
  If ~input/pose is near from any previously stored poses, then republishing input topics will be skipped.

19.12.4 Parameters

• ~rotational_bin_size(Float, default: 0.175)
  Minimum allowable rotational pose difference between new pose and all stored poses in radians.
• ~positional_bin_size(Float, default: 0.1)
  Minimum allowable positional pose difference between new pose and all stored poses in meters.

19.12.5 Sample

roslaunch jsk_pcl_ros sample_capture_stereo_synchronizer.launch
19.13 ClusterPointIndicesDecomposer

19.13.1 What is this

Decompose jsk_recognition_msgs/ClusterPointIndices into array of topics of sensor_msgs/PointCloud like ~output00, ~output01 and so on. It also publishes tf of centroids of each cluster and oriented bounding box of them. The direction of the bounding box are aligned on to the nearest planes if available.

19.13.2 Subscribing topics

- ~input (sensor_msgs/PointCloud2):
  Input pointcloud.
- ~target (jsk_recognition_msgs/ClusterPointIndices):
  Input set of indices to represent clusters.
- ~align_planes (jsk_recognition_msgs/PolygonArray):
- ~align_planes_coefficients (jsk_recognition_msgs/ModelCoefficientsArray):
  The planes for bounding box to be aligned on.
19.13.3 Publishing topics

Default Topics

- `~debug_output (sensor_msgs/PointCloud2)`:
  Concatenate all the clusters into one pointcloud and colorize each cluster to see the result of segmentation.

- `~boxes (jsk_recognition_msgs/BoundingBoxArray)`:
  Array of oriented bounding box for each segmented cluster.
  If `~align_boxes`, `~align_boxes_with_plane` and `~fill_boxes_label_with_nearest_plane_index` are True, each box (jsk_recognition_msgs/BoundingBox)’s label indicates nearest plane index.

- `~label (sensor_msgs/Image)`:
  Label image for each cluster point indices. You can visualize it with `jsk_perception/ColorizeLabels`

- `~mask (sensor_msgs/Image)`:
  Mask image generated from cluster point indices.

- `~centroid_pose_array (geometry_msgs/PoseArray)`
  Each cloud’s centroid poses.

- `~negative_indices (pcl_msgs/PointIndices)`
  Point indices which are not included in input indices.

- `~cluster_indices (jsk_recognition_msgs/ClusterPointIndices)`
  Sorted cluster point indices.

Optional Topics

- `~output%02d (sensor_msgs/PointCloud2)`:
  Series of topics for each pointcloud cluster. This is published only when `publish_clouds` is true.

- `/tf (tf2_msgs/TFMessage)`
  Transformation of each decomposed points’ center of gravity.

19.13.4 Parameters

- `~approximate_sync (Boolean, default: False)`:
  Policy of synchronization, if False it synchronizes exactly, else approximately.

- `~queue_size (Int, default: 100)`:
  Queue size of topic msgs for synchronization.

- `~publish_tf (Boolean, default: False)`:
  Toggle tf publishing.

- `~publish_clouds (Boolean, default: False)`:
  Toggle `~output%02d` topics.

- `~align_boxes (Boolean, default: False)`:
  - `~align_boxes_with_plane (Boolean, default: True)`:
    * Is enabled only if `~align_boxes` is True.
If `~align_boxes` is True and `~align_boxes_with_plane` is True:

- Topics `~align_planes` and `~align_planes_coefficients` are enabled.

If `~align_boxes` is True and `~align_boxes_with_plane` is False:

- Parameter `~target_frame_id` is required, and bounding boxes are aligned with the target frame.

See `sample/sample_cluster_point_indices_decomposer.launch` also.

- `~use_pca` (Boolean, default: False):
  Run PCA algorithm on each cluster to estimate x and y direction. The x-axis indicates the first principal component and y-axis indicates the second one.

- `~force_to_flip_z_axis` (Boolean, default: True)
  Flip z axis direction if this value is true.

- `~max_size`, `~min_size` (Int, default: -1, -1)
  If positive value, ignores cluster which points size is external from (clusterPointsSize < `~min_size` or clusterPointsSize > `~max_size`).

- `~sort_by` (String, default: `input_indices`)
  Sort output indices. Currently below options are supported:
  - `input_indices`: same order as the input cluster indices
  - `z_axis`: sort by z axis of cloud
  - `cloud_size`: sort by size of cloud

- `~fill_boxes_label_with_nearest_plane_index` (Boolean, default: False):
  If `~align_boxes`, `~align_boxes_with_plane` and this value are True, each box (jsk_recognition_msgs/BoundingBox)’s label indicates nearest plane index.

### 19.13.5 Sample

```bash
roslaunch jsk_pcl_ros sample_cluster_point_indices_decomposer.launch
roslaunch jsk_pcl_ros sample_cluster_point_indices_decomposer_sort_by.launch
```

### 19.14 ClusterPointIndicesDecomposerZAxis (Deprecated)

Please use `ClusterPointIndicesDecomposer` with rosparam `_sort_by:=z_axis`.

### 19.14.1 What Is This

This nodelet is almost same to jsk_pcl/ClusterPointIndicesDecomposer, however it always sort clusters in z direction.
19.15 CollisionDetector
19.15.1 What Is This

Collision check between robot model and point cloud.
This node has service interface. You can request collision check by service.

19.15.2 Subscribing Topics

- ~input (sensor_msgs/PointCloud2):
  input pointcloud.

19.15.3 Publishing Topics

None.

19.15.4 Advertising Services

- ~check_collision (jsk_recognition_msgs/CheckCollision):
  Service interface to check collision. joint is the joint state of robot. pose is the pose of root link, which is specified by the parameter ~root_link_id.

```
sensor_msgs/JointState joint
geometry_msgs/PoseStamped pose
---
bool result
```

19.15.5 Parameters

- ~world_frame_id (String, default: map)
  World frame_id.
- ~robot_description (String, required)
  robot_description of the collision-checked robot. The namespace is private, so you can specify the robot, which is different from global robot_description.
- ~root_link_id (String, default: BODY)
  The name of robot root link.
- ~self_see_default_padding (Double, default: 0.01)
  Same as the parameter in self_filter. Padding of robot link in collision check.
- ~self_see_default_scale (Double, default: 1.0)
  Same as the parameter in self_filter. Scale of robot link in collision check.
- ~self_see_links (Array of link configuration, required)
  Same as the parameter in self_filter. Configuration of links for collision check. Link configuration consists of name (required), padding (optional), and scale (optional).
- ~publish_tf (Boolean, default: False)
  Publish tf of robot links. This is useful to display robot model in Rviz.
19.15.6 Sample

```
roslaunch jsk_pcl_ros sample_collision_detector.launch
```

19.16 ColorBasedRegionGrowingSegmentation

![Image](image.jpg)

`jsk_pcl/ColorBasedRegionGrowingSegmentation` segments regions based on the color.

19.16.1 Subscribing Topics

- `~input (sensor_msgs/PointCloud2)`:
  
  Input pointcloud.

19.16.2 Publishing Topics

- `~output (jsk_recognition_msgs/ClusterPointIndices)`:

  Result of region growing segmentation.

19.16.3 Parameters

- `~distance_threshold (Integer, default: 10)`
  
  Distance threshold to determine whether the point is neighbouring or not.

- `~point_color_threshold (Integer, default: 6)`
  
  Color threshold is used for testing points color.

- `~region_color_threshold (Integer, default: 5)`
  
  This threshold used when the merging process takes place.

- `~min_cluster_size (Integer, default: 600)`
  
  Minimum number of cluster size.
19.17 ColorHistogram

hue_sat
Compute color histogram for each cluster point indices.

19.17.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  Input point cloud
- ~input/indices (jsk_recognition_msgs/ClusterPointIndices)
  Input point indices

19.17.2 Publishing Topics

- ~output (jsk_recognition_msgs/ColorHistogramArray)
  Color histogram for each point indices. Each histogram contains either Hue, Saturation (as 1-d vector) or both of two values (as 2-d matrix). (configurable by parameter ~histogram_policy) Bin size is also configurable by parameter ~bin_size.

19.17.3 Parameters

- ~queue_size (Int, default: 100)
  Queue size for message synchronization
- ~bin_size (Int, default: 100)
  Bin size for histogram
  If a parameter ~histogram_policy is set as HUE_AND_SATURATION, actual vector length of each histogram is bin_size * bin_size, otherwise bin_size.
- ~histogram_policy (Enum[Int], default: HUE_AND_SATURATION)
  Policy for histogram values to accumulate
  - 0: HUE
    * Use hue only
  - 1: SATURATION
    * Use saturation only
  - 2: HUE_AND_SATURATION
    * Use both hue and saturation as 2-d matrix
19.17.4 How to visualize

See ColorHistogramVisualizer

19.17.5 Sample

roslaunch jsk_pcl_ros sample_color_histogram.launch

19.18 ColorHistogramClassifier

Classify point indices using color histogram by comparing with reference histogram array.

Methods for histogram comparison is configurable from multiple methods. (See parameter \texttt{~compare\_policy})  
After computing distance between input histograms and reference, classify by their labels (See parameter \texttt{~detection\_threshold})  
Reference histograms are loaded from rosparam on start.
19.18.1 Subscribing Topics

• ~input (jsk_recognition_msgs/ColorHistogram)
  Input color histogram to be classified
• ~input/array (jsk_recognition_msgs/ColorHistogramArray)
  Input color histogram array to be classified

19.18.2 Publishing Topics

• ~output (jsk_recognition_msgs/ClassificationResult)
  Class from color histogram array

19.18.3 Parameters

• ~queue_size (Int, default: 100)
  Queue size for message synchronization
• ~compare_policy (Enum[Int], default: CORRELATION)
  Policy for histogram values to compare
    - 0: CORRELATION
      * Use correlation
    - 1: BHATTACHARYYA
      * Use bhattacharyya distance
    - 2: INTERSECTION
      * Use vector intersection
    - 3: CHISQUARE
      * Use chi-square between two vectors
    - 4: KL_DIVERGENCE
      * Use Kullback-Leibler divergence for comparing two vectors
• ~label_names (String[], required)
  Reference class names
  This parameter is required on start
• ~histograms/<label name> (Double[], required)
  Reference histogram vector for each class
  Length of all histograms must be the same.
• ~detection_threshold (Double, default: 0.8)
  Color histograms and point cloud indices whose similarities are above this value are published as filtered topics.
19.18.4 Collecting Reference Color Histogram

1. First, launch color histogram without classifier

   roslaunch jsk_pcl_ros sample_color_histogram.launch use_classifier:=false

2. Put one object on a plane

   Once you put an object, you can see color histogram in rqt_image_view.

3. Open another terminal and get a histogram

   Now you can get actual histogram data by rostopic echo.

   rostopic echo -n1 /color_histogram/color_histogram/output/histograms/histogram[0]
   [0.22604790329933167, 0.026946106925606728, 0.01646706648170948, 0.009730539284646511, 0.010479042306542397, 0.024700598791241646, 0.08757484704256058, 0.13173653185367584, 0.07335329055786133, 0.040419161319732666, 0.0359281450510025, 0.2365269511938095, 0.08008982241153717, 0.0]

Write this vector data into yaml file so that classifier nodelet can load the histogram as reference.

```yaml
# labels.yaml
label_names:
  - coffee
histograms:
  coffee: [0.22604790329933167, 0.026946106925606728, 0.01646706648170948, 0.009730539284646511, 0.010479042306542397, 0.024700598791241646, 0.08757484704256058, 0.13173653185367584, 0.07335329055786133, 0.040419161319732666, 0.0359281450510025, 0.2365269511938095, 0.08008982241153717, 0.0]``
4. Load reference histograms to classifier

Now you can register reference histograms to classifier in launch file

```xml
<node name="color_histogram_classifer"
 pkg="jsk_pcl_ros" type="color_histogram_classifer">
 <rosparm command="load" file="labels.yaml" />
</node>
```

5. Run and get result

You will be able to get classification result as `jsk_recognition_msgs/ClassificationResult`. See `jsk_pcl_ros/sample/sample_color_histogram.launch` for detail.

### 19.19 ColorHistogramFilter

Filter point indices using color histogram by comparing with reference histogram

Methods for histogram comparison is configurable from multiple methods. (See parameter `~compare_policy`)
After computing distance between input histograms and reference, filter by thresholding (See parame-
Reference histogram can be set as `~reference` topic or as a parameter `~reference_histogram`.

### 19.19.1 Subscribing Topics

- `~input (jsk_recognition_msgs/ColorHistogramArray)`
  Input color histogram array. The order of each histograms must be the same as the order of input cluster point indices.
- `~input/indices (jsk_recognition_msgs/ClusterPointIndices)`
  Input point indices
- `~input/reference (jsk_recognition_msgs/ColorHistogram)`
  Reference histogram
  It can be set as a parameter. See parameter `~reference_histogram`.

### 19.19.2 Publishing Topics

- `~output (jsk_recognition_msgs/ColorHistogramArray)`
  Filtered color histogram array
- `~output/indices (jsk_recognition_msgs/ClusterPointIndices)`
  Filtered cluster point indices

### 19.19.3 Parameters

- `~queue_size (Int, default: 100)`
  Queue size for message synchronization
- `~bin_size (Int, default: 100)`
  Bin size for histogram
- `~compare_policy (Enum[Int], default: CORRELATION)`
  Policy for histogram values to compare
  - 0: CORRELATION
    - Use correlation
  - 1: BHATTACHARYYA
    - Use bhattacharyya distance
  - 2: INTERSECTION
    - Use vector intersection
  - 3: CHISQUARE
    - Use chi-square between two vectors
  - 4: KL_DIVERGENCE
    - Use Kullback-Leibler divergence for comparing two vectors
• `~distance_threshold` (Double, default: 0.6)
  Color histograms and point cloud indices whose similarities are above this value are published as filtered topics.

• `~flip_threshold` (Bool, default: false)
  Publish indices whose distance from reference is higher than `~distance_threshold` if this value is false. If this value is true, publish indices whose is lower than threshold.

• `~reference_histogram` (Float[])
  Reference histogram
  It can also be set as topic. See `~input/reference` topic.

### 19.20 ColorHistogramMatcher

#### 19.20.1 What is this

Finds objects similar to a selected object as reference based on bhattacharyya distance of color histogram.

#### 19.20.2 Subscribing Topics

• `~input` (sensor_msgs::PointCloud2):
  Input point cloud.

• `~input_indices` (jsk_recognition_msgs::ClusterPointIndices):
  Input indices related to input point cloud.

• `~input_reference_cloud` (sensor_msgs::PointCloud2):
  Input target point cloud as reference.
• `~input_reference(jsk_recognition_msgs::ColorHistogram)`:
  Input reference color histogram.

### 19.20.3 Publishing Topic

• `~output(jsk_recognition_msgs::ClusterPointIndices)`:
  Output Indices of satisfying color histogram similarity.
• `~output_reference(jsk_recognition_msgs::ColorHistogram)`:
  Output of histogram from input referenced point cloud.
• `~output_histograms(jsk_recognition_msgs::ColorHistogramArray)`:
  Output of all histograms from clustered input point cloud.
• `~coefficient_points(sensor_msgs/PointCloud2)`:
  Output of heat map point cloud.
• `~best_match(geometry_msgs/PoseStamped)`:
  Output of best match indices points pose.

### 19.20.4 Parameters

• `coefficient_threshold(Double, default: 0.9)`
  Threshold for determining color histogram similarity
• `bin_size(Double, default: 10)`
  Number of bins of histogram
• `histogram_method(Int, default: 3)`
  Histogram Comparing method.
  – 0: HUE
  – 1: SATURATION
  – 2: VALUE
  – 3: HUE AND SATURATION

### 19.20.5 Sample

```
roslaunch jsk_pcl_ros sample_color_histogram_matcher.launch
```

Please refer to `SelectedClusterPublisher` on how to select a referenced object.
19.21 ColorHistogramVisualizer

Visualize 1D / 2D color histogram

19.21.1 Run

```bash
gerun jsk_pcl_ros color_histogram_visualizer.py```

19.21.2 Subscribing Topics

- `~input (jsk_recognition_msgs/ColorHistogram)`
  Input color histogram
- `~input/array (jsk_recognition_msgs/ColorHistogramArray)`
  Input color histogram array
19.21.3 Publishing Topics

- `~output (sensor_msgs/Image)`
  Image of histogram bar graph

19.21.4 Parameters

- `~histogram_policy`:
  See `ColorHistogram`
- `~histogram_index (Int, default: 0)`:
  Index of input array to show histogram (Valid only when input is `ColorHistogramArray`)
- `~histogram_scale (Double, default: 1.0)`:
  Scale factor for each color histogram (Scale up when bar is too short to visualize)

19.22 ColorizeMapRandomForest

19.22.1 What Is This

Extract FPFH (Fast Point Feature Histograms) from input point cloud, and classify each cluster into 2 categories using Random Forest classifier.

This node internally applies x, y and z filter as preprocessing.

In contrast to `jsk_pcl_ros/ColorizeRandomForest`, this node does not run euclidean clustering as preprocessing.
19.22.2 Subscribing Topic

- ~input(sensor_msgs/PointCloud2)
  Input point cloud from which specified part will be extracted.

19.22.3 Publishing Topic

- ~output/debug_points(sensor_msgs/PointCloud2)
  Output point cloud for debugging. Red means classification result == 0, blue means the other.

19.22.4 Parameter

- ~rs (Float, default: 0.03)
  Radius in meters used for searching FPFH.
- ~po (Float, default: 0.03)
  Offset in meters used for pass through filter.
- ~po2 (Float, default: 0.06)
  Another offset in meters used for pass through filter.
- ~sum_num (Int, default: 100)
  Number of points used for averaging FPFH histogram.

19.22.5 Calling Service

- ~classify_server(ml_classifiers/ClassifyData)
  Random Forest server for classification.

19.22.6 Sample

```bash
roslaunch jsk_pcl_ros sample_colorize_map_random_forest.launch
```
19.23 ColorizeRandomForest

19.23.1 What Is This

Extract FPFH (Fast Point Feature Histograms) from input point cloud, and classify each cluster into 2 categories using Random Forest classifier.

This node internally applies x, y and z filter and euclidean clustering as preprocessing.

19.23.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  
  Input point cloud from which specified part will be extracted.

19.23.3 Publishing Topic

- ~output/zero (sensor_msgs/PointCloud2)
  
  Output point cloud whose classification result == 0

- ~output/nonzero (sensor_msgs/PointCloud2)
  
  Output point cloud whose classification result != 0
19.23.4 Parameter

- \(\text{\~r}_s\) (Float, default: 0.03)
  Radius in meters used for searching FPFH.
- \(\text{\~p}_o\) (Float, default: 0.03)
  Offset in meters used for pass through filter.
- \(\text{\~p}_o^2\) (Float, default: 0.06)
  Another offset in meters used for pass through filter.
- \(\text{\~s}_{\text{um}_\text{num}}\) (Int, default: 100)
  Number of points used for averaging FPFH histogram.

19.23.5 Calling Service

- \(\text{\~c}_{\text{lassify}_\text{server}}\) (ml_classifiers/ClassifyData)
  Random Forest server for classification.

19.23.6 Sample

```bash
roslaunch jsk_pcl_ros sample_colorize_random_forest.launch
```
19.24 ContainerOccupancyDetector

19.24.1 What Is This

It subscribes containers’ boxes (jsk_recognition_msgs/BoundingBoxArray) and point cloud in the each boxes (sensor_msgs/PointCloud2, jsk_recognition_msgs/ClusterPointIndices) then publish their occupancies (jsk_recognition_msgs/BoundingBoxArray). Their occupancy rates are in each boxes’ value field.
19.24.2 Subscribing Topic

- ~container/boxes (jsk_recognition_msgs/BoundingBoxArray)
  Containers input.
- ~container/points (sensor_msgs/PointCloud2)
  Original pointcloud.
- ~container/point_indices (jsk_recognition_msgs/ClusterPointIndices)
  The indices of point cloud in each box.

19.24.3 Publishing Topic

- ~container/occupancies (jsk_recognition_msgs/BoundingBoxArray)
  Containers’ occupancies.

19.24.4 Parameters

- ~approximate_sync (Bool, default: false)
  Approximately synchronize inputs if it’s true.
- ~queue_size (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.

19.24.5 For use

It is strongly recommended to use with original launch file like

```
roslaunch jsk_pcl_ros container_occupancy_detector.launch POINTCLOUD_INPUT:=-<your point cloud> CONTAINER_BOXES_INPUT:=<your containers>
```

19.24.6 Sample

```
roslaunch jsk_pcl_ros sample_container_occupancy_detector.launch
```
19.25 ConvexConnectedVoxels

Merges the voxels initially segmented using SuperVoxel Segmentation into high level object representations by merging the voxels based on convexity measurements.

19.25.1 Subscribing Topics

- ~input/indices (jsk_recognition_msgs/ClusterPointIndices)
- ~input/cloud (sensor_msgs/PointCloud2)

Input is set of voxel indices and point cloud from the supervoxel_segmentation nodelet

19.25.2 Publishing Topics

- ~output/indices (jsk_recognition_msgs/ClusterPointIndices)

Output is set of merged voxel indices

19.25.3 Sample

roslaunch jsk_pcl_ros sample_convex_connected_voxels.launch
19.26 DepthCalibration

This nodelet applies calibration model to depth image.

19.26.1 What Is This

This nodelet applies calibration model to depth image.

19.26.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input depth image.
  The encoding should be `32FC1`.
- `~camera_info (sensor_msgs/CameraInfo)`
  Input camera_info of depth camera.

19.26.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  Output depth image.

19.26.4 Advertising Service

- `~set_calibration_parameter (jsk_recognition_msgs/SetDepthCalibrationParameter)`
  Update parameters below except for `~uv_scale`.
19.26.5 Parameter

- `~coefficients2` (Array of double, default: \[0, 0, 0, 0, 0\])
  coefficients of calibration model.
- `~coefficients1` (Array of double, default: \[0, 0, 0, 0, 1.0\])
  coefficients of calibration model.
- `~coefficients0` (Array of double, default: \[0, 0, 0, 0\])
  coefficients of calibration model.
- `~use_abs` (Boolean, default: False)
  If you want to use absolute value in applying calibration model, please set this variable True.
- `~uv_scale` (Double, default: 1.0)
  If you want to scale value in applying calibration model, please set this variable.

19.26.6 Sample

```bash
roslaunch jsk_pcl_ros sample_depth_calibration.launch
```

19.27 depth_error_calibration.py

19.27.1 What Is This

This nodelet stores result of depth error calculated by \textit{DepthImageError} and dump calibration parameter into CSV file (calibration-%Y-%m-%d-%H-%M-%S.csv).
19.27.2 Subscribing Topic

- depth_image_error/output (jsk_recognition_msgs/DepthErrorResult)
  
  Result of depth error.

19.27.3 Publishing Topic

- ~frequency_image (sensor_msgs/Image)
  
  Frequency map.

- ~error_plot_image (sensor_msgs/Image)
  
  Plot of relation between Z from depth image and Z from checker board.

19.27.4 Internally Calling Service

- /camera_remote/depth_calibration/set_calibration_parameter
  (jsk_recognition_msgs/SetDepthCalibrationParameter)
  
  Set depth calibration parameters.

19.27.5 Parameter

- ~u_min (Int, default: 0)
- ~u_max (Int, default: 4096)
- ~v_min (Int, default: 0)
- ~v_max (Int, default: 4096)
  
  Minimum/maximum limit of (u, v) of checkerboard corner.

19.27.6 Sample

```
roslaunch jsk_pcl_ros sample_depth_error_calibration.launch
```
19.28 DepthImageCreator

19.28.1 What is this

Create organized pointcloud from non-organized pointcloud. Currently it supports pcl::PointXYZ and pcl::PointXYZRGB as the input.

19.28.2 Subscribing Topics

- `~input` (sensor_msgs/PointCloud2):
  
  The input pointcloud to be reconstructed as organized pointcloud.

- `~info` (sensor_msgs/CameraInfo):
  
  Put a simulated camera according to `~info` and generate organized pointcloud.
19.28.3 Publishing Topics

- **~output** (sensor_msgs/Image): Publish organized pointcloud as depth image.
- **~output_image** (sensor_msgs/Image): Publish image colorized according to the input cloud.
- **~output_cloud** (sensor_msgs/PointCloud2) Output pointcloud.
  See ~organize_cloud parameter.
- **~output_disp** (sensor_msgs/DisparityImage) Publish organized pointcloud as disparity image.

19.28.4 Parameters

- **~scale_depth** (Double, default: 1.0) scale depth value.
- **~use_fixed_transform** (Boolean, default: False): ~use_fixed_transform (Boolean, default: False):
- **~translation** (Array of double, default: [0, 0, 0])
- **~rotation** (Array of double, default: [0, 0, 0, 1])
  If ~use_fixed_transform is set to True, transformation between ~input and ~info is not resolved via tf but fixed transformation is used according to ~rotation and ~translation.
- **~use_asynchronous** (Boolean, default: False)
  If this parameter is set to True, this node process incoming PointCloud regardless of the synchronization between ~input and ~info.
- **~use_approximate** (Boolean, default: False)
  Synchronize ~input and ~info approximately if this parameter is set to True. If this parameter is set to False, and the timestamps of the ~input and ~info are not strictly matched, the callback will not be processed. (See http://wiki.ros.org/message_filters/ApproximateTime)
- **~info_throttle** (Integer, default: 0)
  The number of ~info messages to skip to generate depth image.
- **~max_queue_size** (integer, default: 100): Queue length for synchronization of topics. According to the message_filters synchronization policy, in case of ~input(1Hz) and ~info(30Hz) comes, message_filters needs at least over 30 queue_size for searching a synchronized set of the two. (You can also check like this rosrun jsk_topic_tools is_synchronized --approximate-sync --queue-size 3 /kinect_head/depth_registered/camera_info /kinect_head/depth_registered/points)
- **~max_pub_queue_size** (integer, default: ~max_queue_size_):
  Queue length of topic publishers. Default is value set for max_queue_size_.
- **~max_sub_queue_size** (integer, default: ~max_queue_size_):
  Queue length of topic subscribers. Default is value set for max_queue_size_.

19.28. DepthImageCreator
• ~fill_value (float, default: nan):
  Initial value of depth image. The pixels where there is no corresponding point are filled by this value.

• ~organize_cloud (Boolean, default: False)
  Whether to organize ~output_cloud or not.

• ~tf_duration (float, default: 0.001):
  TF Lookup transform duration. This value is only used when use_fixed_transform is false.

### 19.28.5 Sample

```
roslaunch jsk_pcl_ros sample_depth_image_creator.launch
```

### 19.29 DetectGraspablePosesPcabase

#### 19.29.1 What Is This

Detect_graspable_poses_pcabase.py is a program which publishes pose array where a robot can grasp, using point cloud of a target object and PCA algorithm. In the above image, the grasp poses are visualized with the axies. Please transform input cloud to a flame which direction is the same as the robot (e.g. /BODY) before using this program.

The above image illustrates an example way to use this node. Axies represent poses where the robot can grasp.
19.29.2 Example videos using a real robot (only allowed for the jsk members)

19.29.3 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  
  Input pointcloud. RGB field is required.

19.29.4 Publishing Topic

- ~output/can_grasp_poses (geometry_msgs/PoseArray)
  
  PoseArray where a robot can grasp.

19.29.5 Parameters

- ~direction (Character, default: x)
  
  From which direction a robot try to grasp a target object. (‘x’: from the front, ‘z’: from the top)

- ~hand_width [m] (Float, default: 0.13)
  
  How long a robot can spread its hand.

- ~interval_m [m] (Float, default: 0.04)
  
  Interval between target poses. Please decrease this number if you want to detect more possible grasp poses.
19.30 display-bounding-box-array.l

Show `jsk_recognition_msgs/BoundingBoxArray` in IRT viewer in Euslisp.
19.30.1 Subscribing Topics

- /cluster_decomposer/boxes (jsk_recognition_msgs/BoundingBoxArray)
  Input bounding box array.

19.30.2 Sample

roslaunch jsk_pcl_ros sample_display_bounding_box_array.launch

19.31 dump_depth_error.py

19.31.1 What Is This

This node retrieves jsk_recognition_msgs/DepthErrorResult, and publish (true_depth, observed_depth) as a 2D plot data.

It also dump the data to CSV file.
19.31.2 Subscribing Topic

- `/depth_image_error/output (jsk_recognition_msgs/DepthErrorResult)
  Depth error result calculated by `jsk_pcl_utils/DepthImageError`

19.31.3 Publishing Topic

- `~scatter (jsk_recognition_msgs/PlotData)
  2D plot data of true_depth and observed_depth`

19.31.4 Parameters

- `~csv_path (String, default: output.csv)
  Path to output CSV file`

19.31.5 Sample

```
roslaunch jsk_pcl_ros sample_dump_depth_error.launch
```

19.32 EdgeDepthRefinement

![Before](image1.png) ![After](image2.png)

**Here is a jump**

**Removed**
19.32.1 What is this?

Refine edges based on depth connectivity.
In order to take continuity into account, it run RANSAC and remove outliers.

19.32.2 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
- ~input_indices (jsk_recognition_msgs/ClusterPointIndices)

19.32.3 Publishing Topics

- ~output (jsk_recognition_msgs/ClusterPointIndices)
- ~output_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
- ~output_outlier_removed (jsk_recognition_msgs/ClusterPointIndices)
- ~output_outlier_removed_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)

19.32.4 Parameters

- ~outlier_distance_threshold (Double, default: 0.01)
- ~min_inliers (Int, default: 10)
- ~duplication_angle_threshold (Double, default: 0.1)
- ~duplication_distance_threshold (Double, default: 0.01)

19.32.5 Sample

```bash
roslaunch jsk_pcl_ros sample_edge_depth_refinement.launch
```
19.33 EdgebasedCubeFinder

19.33.1 What Is This

Detect cubes by estimating parallel and perpendicular planes from parallel edge pairs.

19.33.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input point cloud.
- `~input_edges (jsk_recognition_msgs/ParallelEdgeArray)`
  Input parallel edge pairs.

19.33.3 Publishing Topic

- `~output (jsk_recognition_msgs/BoundingBoxArray)`
  Detected cubes.
- `~output_pose_array (geometry_msgs/PoseArray)`
  Poses of detected cubes.
- `~debug_filtered_cloud (sensor_msgs/PointCloud2)`
  Point cloud on edges for debugging.
- `~debug_clusters (jsk_recognition_msgs/ClusterPointIndices)`
  This topic is advertised, but not published now.
• `~debug_polygons (jsk_recognition_msgs/PolygonArray)`
  This topic is advertised, but not published now.

• `~debug_marker (visualization_msgs/Marker)`
  This topic is advertised, but not published now.

19.33.4 Parameter

• `~outlier_threshold (Float, default: 0.01)`
  Threshold to remove outliers in meters.
  This parameter can be changed by `dynamic_reconfigure`.

• `~parallel_edge_distance_min_threshold (Float, default: 0.1)`
  Minimum distance between parallel edges in meters.
  This parameter can be changed by `dynamic_reconfigure`.

• `~parallel_edge_distance_max_threshold (Float, default: 0.4)`
  Maximum distance between parallel edges in meters.
  This parameter can be changed by `dynamic_reconfigure`.

• `~min_inliers (Int, default: 1000)`
  This parameter is not used for now.

• `~convex_area_threshold (Float, default: 0.01)`
  This parameter is not used for now.

• `~convex_edge_threshold (Float, default: 0.1)`
  This parameter is not used for now.

19.33.5 Sample

```sh
roslaunch jsk_pcl_ros sample_edgebased_cube_finder.launch
```
19.34 EnvironmentPlaneModeling

19.34.1 What Is This

Make environment plane model as occupancy grid from point cloud, plane indices, plane polygons and plane normal coefficients.

19.34.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Input point cloud which contains normal for each point.
  This topic is used for converting plane indices and coefficients into convex polygon.
- ~input/full_cloud (sensor_msgs/PointCloud2)
  Input point cloud which contains normal for each point.
  This topic is used for building grid map.
- ~input/indices (jsk_recognition_msgs/ClusterPointIndices)
  Input point indices of plane.
- ~input/polygons (jsk_recognition_msgs/PolygonArray)
  Input polygons of plane.
- ~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
  Input normal coefficients of plane.

These 5 topics described above must be synchronized.
• ~input/leg_bounding_box (jsk_recognition_msgs/BoundingBox)
  Optional input bounding box.
  Used only when ~complete_footprint_region is true.
• /move_base_simple/goal (geometry_msgs/PoseStamped)
  Optional input pose which a user wants robot to move to.

19.34.3 Publishing Topic

• ~output (jsk_recognition_msgs/SimpleOccupancyGridArray)
  Occupancy grid of plane.
• ~output/non_plane_indices (pcl_msgs/PointIndices)
  Point indices of non-plane region.
• ~debug/magnified_polygons (jsk_recognition_msgs/PolygonArray)
  Magnified plane polygons for debugging.
• ~debug/convex_cloud (sensor_msgs/PointCloud2)
  Vertex point cloud of convex polygons.
• ~debug/plane_poses (geometry_msgs/PoseArray)
  Pose of plane for debugging.
• ~debug/magnified_plane_poses (geometry_msgs/PoseArray)
  Pose of magnified plane for debugging.
• ~debug/raw_grid_map (jsk_recognition_msgs/SimpleOccupancyGridArray)
  Occupancy grid of plane before applying morphological dilation & erosion.
• ~debug/noeroded_grid_map (jsk_recognition_msgs/SimpleOccupancyGridArray)
  Occupancy grid of plane to which morphological filter is applied, but not enough erosion.
• /footstep_simple/goal (geometry_msgs/PoseStamped)
  Output pose of suitable grid.
  Only published when /move_base_simple/goal is subscribed.

19.34.4 Parameter

• ~complete_footprint_region (Bool, default: False)
  Set to true if you want to complete grid map by ~input/leg_bounding_box.
• ~footprint_frames (List of String, default: [])
  Frame ID used for looking up ground plane for footprint.
  Used only when ~complete_footprint_region is true.
19.34.5 Sample

```bash
roslaunch jsk_pcl_ros sample_environment_plane_modeling.launch
```

19.35 EuclideanClustering

![EuclideanClustering Diagram](image)

19.35.1 What Is This

Segment pointcloud based euclidean metrics, which is based on `pcl::EuclideanClusterExtraction`. This nodelet has topic interface and service interface.

The result of clustering is published as `jsk_recognition_msgs/ClusterPointIndices`.

If the number of the cluster is not changed across different frames, `EuclideanClustering` tries to track the segment.
19.35.2 Subscribing Topics

- `~input (sensor_msgs/PointCloud2)`: input pointcloud. If `~multi` is `false`, this input is only enough.
- `~input/cluster_indices (jsk_recognition_msgs/ClusterPointIndices)`: input indices. If `~multi` is `true`, synchronized `~input` and `~input/cluster_indices` are used.

19.35.3 Publishing Topics

- `~output (jsk_recognition_msgs/ClusterPointIndices)`: Result of clustering.
- `~cluster_num (jsk_recognition_msgs/Int32Stamped)`: The number of clusters.

19.35.4 Advertising Services

- `~euclidean_clustering (jsk_pcl_ros/EuclideanSegment)`: Service interface to segment clusters.

```
sensor_msgs/PointCloud2 input
float32 tolerance
---
sensor_msgs/PointCloud2[] output
```

19.35.5 Parameters

- `~tolerance (Double, default: 0.02)`: Max distance for the points to be regarded as same cluster.
- `~label_tracking_tolerance (Double, default: 0.2)`: Max distance to track the cluster between different frames.
- `~max_size (Integer, default: 25000)`: The maximum number of the points of one cluster.
- `~min_size (Integer, default: 20)`: The minimum number of the points of one cluster.
- `~multi (Boolean, default: false)`: Flag of applying euclidean clustering for each pointcloud's indices(`~input/cluster_indices's cluster_indices`).
  If `~multi` is `true`, synchronized `~input` and `~input/cluster_indices` are used.
- `~approximate_sync (Boolean, default: False)`: Policy of synchronization, if `false` it synchronizes exactly, else approximately. This value is only valid in case of `~multi` is `true`.
• ~queue_size (Int, default: 20):
  Queue size of topic msgs for synchronization.
• ~downsample_enable (Boolean, default: false)
  Flag of VoxelGrid downsampling. If ~downsample_enable is true, ~input is downsampled.
• ~leaf_size (Double, default: 0.01)
  Leaf size of voxel grid downsampling. This value is only valid in case of ~downsample_enable is true.
• ~cluster_filter (Int, default: 0)
  Specify cluster filtering methods.
  0: Passthrough the all clustering result for each cluster.
  1: Take a cluster which has the maximum size of those for each cluster. The length of ~output's cluster_indices, ~cluster_num, is length of ~input/cluster_indices's cluster_indices.

19.35.6 Sample

Plug the depth sensor which can be launched by openni.launch and run the below command.

```bash
roslaunch jsk_pcl_ros euclidean_segmentation.launch
```

19.36 ExtractCuboidParticlesTopN

Extract top-N particles of pcl::tracking::ParticleCuboid by comparing weight of them.
19.36.1 Publishing Topics

- `~output (pcl_msgs/PointIndices)`
  Top-N particles indices.
- `~output/box_array (jsk_recognition_msgs/BoundingBoxArray)`
  Top-N particles as BoundingBoxArray.
- `~output/pose_array (jsk_recognition_msgs/WeightedPoseArray)`
  Top-N particles as WeightedPoseArray.

19.36.2 Subscribing Topics

- `~input (sensor_msgs/PointCloud2)`
  Particle cloud of `pcl::tracking::ParticleCuboid`. All the weights are expected to be normalized.

19.36.3 Parameters

- `~top_n_ratio (Float, default: 0.9)`
  Ratio of top-N.

19.36.4 Sample

```bash
roslaunch jsk_pcl_ros sample_extract_cuboid_particles_top_n.launch
```

19.37 ExtractIndices

19.37.1 What Is This
ExtractIndices extracts point cloud with input point indices.

Upstream package pcl_ros has similar node which extracts point cloud with point indices. (see here)

The pros of this node compared to it are:

- supports keep_organized option when extract cloud.
- connected-based system. (does not subscribe without child subscriber)
- other params. max_queue_size, approximate_sync
- support simple command line interface. (you can run with rosrun)

### 19.37.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Original depth information from which you extract some of them.
- ~indices (pcl_msgs/PointIndices)
  Indices for point cloud you extract.

### 19.37.3 Publishing Topic

- ~output (sensor_msgs/PointCloud2)
  Extracted point cloud.

### 19.37.4 Parameter

- keep_organized (Boolean, default: false)
  Set keep_organized when extract indices.
- negative (Boolean, default: false)
  Set negative when extract indices.
- max_queue_size (Int, default: 10)
  Max queue size for subscribers.
- approximate_sync (Boolean, default: false)
  If this parameter is true, ~input and ~indices are synchronized with approximate time policy.

### 19.37.5 Sample

```sh
roslaunch jsk_pcl_ros sample_attention_clipper.launch
```
19.38 `extract_top_polygon_likelihood.py`

19.38.1 What Is This

Extract maximum likelihood polygon from input polygons.

19.38.2 Subscribing Topic

- `~input (jsk_recognition_msgs/PolygonArray)`
  Polygons which have likelihood field.
- `~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Normal vector of polygons.

19.38.3 Publishing Topic

- `~output (jsk_recognition_msgs/PolygonArray)`
  Extracted plane polygon array.
- `~output/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Extracted coefficients.

19.38.4 Parameters

- `~min_likelihood (Float, default: 0.7)`
  Do not publish anything if all likelihood are smaller than this value.
  This parameter can be changed by `dynamic_reconfigure`.  

19.38.5 Sample

```bash
roslaunch jsk_pcl_ros sample_extract_top_polygon_likelihood.launch
```

19.39 FeatureRegistration

Align pointcloud using 3d feature. Currently only FPFH is supported.

### 19.39.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud. The type of point is `pcl::PointNormal`.
- `~input/feature (sensor_msgs/PointCloud2)`
  Input feature. The type of point is `pcl::FPFHSignature33`.
- `~input/reference/cloud (sensor_msgs/PointCloud2)`
  Reference pointcloud. The type of point is `pcl::PointNormal`.
- `~input/reference/feature (sensor_msgs/PointCloud2)`
  Reference feature. The type of point is `pcl::FPFHSignature33`. 
19.39.2 Publishing Topic

- `~output (geometry_msgs/PoseStamped)`
  Transformation to align reference cloud to input cloud.
- `~output/cloud (sensor_msgs/PointCloud2)`
  Reference pointCloud which is aligned to input cloud.

19.39.3 Parameters

- `~max_iterations (Integer, default: 1000)`
  Maximum number of iterations.
- `~correspondence_randomness (Integer, default: 2)`
  Number of neighbors to use when selecting a random feature correspondence.
  A higher value will add more randomness to the feature matching.
- `~similarity_threshold (Double, default: 0.9)`
  Similarity threshold in [0,1] between edge lengths of the underlying polygonal correspondence rejector object, where 1 is a perfect match.
- `~max_correspondence_distance (Double, default: 0.0075)`
  Maximum distance threshold between two correspondent points in source <-> target.
- `~inlier_fraction (Double, default: 0.25)`
  Required inlier fraction of the input in [0, 1]
- `~transformation_epsilon (Double, default: 0.1)`
  Maximum allowable difference between two consecutive transformations in order for an optimization to be considered as having converged to the final solution.

These parameters can be changed by `dynamic_reconfigure`.

19.39.4 Sample

```bash
roslaunch jsk_pcl_ros sample_feature_registration.launch
```
19.40 FindObjectOnPlane

19.40.1 What Is This

List up bounding parallelogram of input 2D mask image using 3D plane normal coefficients, and choose the minimum area from them.

19.40.2 Subscribing Topic

- ~input(sensor_msgs/Image)
  Input mask image.
- ~input/camera_info(sensor_msgs/CameraInfo)
  Input camera info.
- ~input/coefficients(pcl_msgs/ModelCoefficients)
  Input normal coefficients of plane.

19.40.3 Publishing Topic

- ~debug/min_area_rect_image(sensor_msgs/Image)
  Visualization of minimum bounding parallelogram of mask.
  Red rectangle is aligned to x and y axis.
  Green parallelograms are bounding parallelograms of mask.
  Yellow parallelogram is the bounding parallelogram which is estimated to be minimum.
19.40.4 Sample

```bash
to launch jsk_pcl_ros sample_find_object_on_plane.launch
```

19.41 FisheyeSpherePublisher

Show the sphere point cloud generated from fisheye image. This was tested with Prosilica GC 2450C + nm30 lens

19.41.1 Subscribing Topics

- `~input(sensor_msgs/Image)`
  
  Fisheye Image

19.41.2 Publishing Topics

- `~output(sensor_msgs/PointCloud2)`
  
  Sphere pointcloud.
19.41.3 Sample

```
roslaunch jsk_pcl_ros sample_fisheye_sphere_publisher.launch
```

19.42 FuseDepthImages

Do sensor fusions by multiple depth images ignoring nan region in each image. For transformation of depth from one to another, you can use `jsk_pcl_ros/DepthImageCreator`. See `sample_fuse_images.launch` for detail.

19.42.1 Subscribing Topic

See rosparam `~input_topics`.

19.42.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Output fused depth image.
19.42.3 Parameters

Required

• ~input_topics (String array, required)
  Input depth image topics.

Optional

• ~approximate_sync (Boolean, default: False):
  Policy of synchronization, if False it synchronizes exactly, else approximately.
• ~queue_size (Int, default: 100):
  Queue size of topic msgs for synchronization.
• ~averaging (Bool, default: true)
  Average image values while sensor fusion.

19.42.4 Sample

roslaunch jsk_pcl_ros sample_fuse_images.launch

19.43 FuseRGBImages

Fuse rgb images ignoring black region in each image. See jsk_pcl_ros/FuseDepthImages for other info.
19.44 GeometricConsistencyGrouping

Estimate model position using Geometric Consistency Grouping technique

19.44.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)
  Scene pointcloud. The type is pcl::PointNormal.
- `~input/feature (sensor_msgs/PointCloud2)
  Scene feature. currently SHOT352 is supported.
- `~input/reference (sensor_msgs/PointCloud2)
  Model pointcloud. The type is pcl::PointNormal.
- `~input/reference/feature (sensor_msgs/PointCloud2)
  Model feature. currently SHOT352 is supported.

19.44.2 Publishing Topic

- `~output (geometry_msgs/PoseStamped)
  Pose of recognized object

19.44.3 Parameters

- `~gc_size (Double, default: 0.01)
  Size of cluster
- `~gc_thresh (Double, default: 5.0)
  Threshold of clustering
19.45 GridSampler

19.45.1 What is this?

Sample clusters from point cloud with grid size.
19.45.2 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  Input point cloud.

19.45.3 Publishing Topics

- ~output (jsk_recognition_msgs/ClusterPointIndices)
  Clusters of point indices which represent each grid sample.

19.45.4 Parameters

- ~grid_size (Double, default: 0.2)
  Grid size of each cluster in [m].
- ~min_indices (Int, default: 0)
  Minimum number of point indices in each cluster.

19.45.5 Sample

```
roslaunch jsk_pcl_ros sample_grid_sampler.launch
```
19.46 HandleEstimator

19.46.1 What Is This

Estimate 6-DOF grasp pose candidates using bounding box.
This node is similar to jsk_pcl_ros/detect_graspable_poses_pcabase.py, but the differences are ...

- Publish pre-approach pose or not.
- Candidates are the same position but different pose (former), or different position but the same pose (latter).

This node can also publish only selected pose.

19.46.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input point cloud.
  Currently this topic is not used for estimation, but required.
  It must be synchronized with `~input_box`.
- `~input_box (jsk_recognition_msgs/BoundingBox)`
  Input bounding box.
  Dimensions of bounding box are used to estimate handle type internally.
  So it is recommended that the box is aligned to the object using PCA or so.
• \texttt{\~selected\_index} (\texttt{jsk\_recognition\_msgs/Int32Stamped})
  Pose index chosen from indices of \texttt{\~output} and \texttt{\~output\_preapproach}.

19.46.3 Publishing Topic

• \texttt{\~output} (\texttt{geometry\_msgs/PoseArray})
  Grasp pose candidates.
• \texttt{\~output\_preapproach} (\texttt{geometry\_msgs/PoseArray})
  Pre-approach poses of \texttt{\~output}.
• \texttt{\~output\_best} (\texttt{geometry\_msgs/PoseStamped})
  Suggested best grasp pose chosen from \texttt{\~output}.
• \texttt{\~output\_selected} (\texttt{geometry\_msgs/PoseStamped})
  Grasp pose selected by \texttt{\~selected\_index}.
• \texttt{\~output\_selected\_preapproach} (\texttt{geometry\_msgs/PoseStamped})
  Pre-approach poses of \texttt{\~output\_selected\_preapproach}.

19.46.4 Parameter

• \texttt{\~gripper\_size} (Float, default: 0.08)
  Gripper width of robot in meters.
  If all dimensions of input box are greater than this parameter, then the box will be estimated as ungraspable.
• \texttt{\~approach\_offset} (Float, default: 0.1)
  Offset from grasp point in meters.
  This parameter is used for calculating \texttt{\~output\_preapproach}.
• \texttt{\~angle\_divide\_num} (Int, default: 6)
  Number of grasp pose candidates.

19.46.5 Sample

```bash
roslaunch jsk_pcl_ros sample_handle_estimator.launch
```
19.47 HeightmapConverter

Convert a pointcloud (sensor_msgs/PointCloud2) into heightmap representation (sensor_msgs/Image).

19.47.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud

19.47.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  fields of the image is CV_32FC2(float). Channel0 of the image represents heightmap and Channel1 of the image represents quality/intensity/reliability of value. If a pixel is not observed, it is filled by -FLT_MAX.

- `~output/config (jsk_recognition_msgs/HeightmapConfig)`
  Config topic.
19.47.3 Parameters

- `~use_projected_center` (Bool, default: False)
  
  If true, `~fixed_frame_id`, `~center_frame_id` and `~projected_center_frame_id` will be enabled and heightmap will be created at `~center_frame_id`.
  
  If false, heightmap will be created at input point cloud frame.

- `~fixed_frame_id` (String, default: map)
  
  Parent frame ID of `~projected_center_frame_id`.

- `~center_frame_id` (String, default: BODY)
  
  Frame ID of center of output heightmap.

- `~projected_center_frame_id` (String, default: BODY_on_map)
  
  Name of new frame ID used in `~output`.

- `~resolution_x` (Integer, default: 400)

- `~resolution_y` (Integer, default: 400)

  Width and height of the output height map in pixels.

  These parameters can be changed by `dynamic_reconfigure`.

- `~min_z` (Double, default: -inf)

- `~max_z` (Double, default: inf)

  Minimum and maximum value to specify z range to be considered. Outside of this range will be ignored.

  These parameters can be changed by `dynamic_reconfigure`.

- `~min_x` (Double, default: -2.0)

- `~max_x` (Double, default: 2.0)

- `~min_y` (Double, default: -2.0)

- `~max_y` (Double, default: 2.0)

  Minimum and maximum value of heightmap dimension in meters.

  These parameters can be changed by `dynamic_reconfigure`.

- `~duration_transform_timeout` (Double default: 1.0)

  Duration of timeout for transform looking up of tf frames

- `~initial_probability` (Double, default: 1.0)

  Initial value to be set to Channel1 of heightmap image

  This parameter can be changed by `dynamic_reconfigure`.
19.47.4 Sample

```bash
roslaunch jsk_pcl_ros sample_heightmap_converter.launch
```

19.48 HeightmapMorphologicalFiltering

Apply morphological filtering and average filter to fill small holes in pointcloud which is represented as heightmap.

19.48.1 Subscribing Topic

- `~input (sensor_msgs/Image)`
  Input heightmap. Hole should be represented as -FLT_MAX or nan.
  Encoding should be 32FC2.
- `~input/config (jsk_recognition_msgs/HeightmapConfig)`
  Config topic.
  This topic name is automatically resolved by `~input` topic name.

19.48.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  Output heightmap.
- `~output/config (jsk_recognition_msgs/HeightmapConfig)`
  Config topic.
  This node just relays `~input/config`. 
19.48.3 Parameters

- `~max_queue_size` (Integer, default: 10):
  Max queue size of subscription callback.
- `~mask_size` (Integer, default: 2):
  Size of kernel operator of average filtering.
- `~max_variance` (Double, default: 0.1):
  Allowable max variance in kernel operator
- `~smooth_method` (String, default: `average_variance`)
  You can choose method of smoothing from `average_variance` and `average_distance`.
- `~use_bilateral` (Bool, default: false)
  use bilateral filtering after smooth(interpolation) method
- `~bilateral_filter_size` (Integer, default: 5)
  Kernel size of bilateral filtering.
- `~bilateral_sigma_color` (Double, default: 0.04)
  filter sigma of color space.
- `~bilateral_sigma_space` (Double, default: 5)
  filter sigma of coordinate space.

19.48.4 Sample

```
roslaunch jsk_pcl_ros sample_heightmap_morphological_filtering.launch
```

19.49 HeightmapTimeAccumulation

Accumulate heightmap in time series and construct a new heightmap.
19.49.1 Subscription Topic

- `~input (sensor_msgs/Image)`
  Input new heightmap(t=k).
- `~input/prev_pointcloud (sensor_msgs/PointCloud2)`
  Accumulated heightmap represented in pointcloud from 0 to k-1 step.
- `~input/config (jsk_recognition_msgs/HeightmapConfig)`
  Config topic.

19.49.2 Publishing Topic

- `~output (sensor_msgs/Image)`
  Accumulated heightmap.
- `~output/config (jsk_recognition_msgs/HeightmapConfig)`
  Config topic.

19.49.3 Advertising Service

- `~reset (std_srvs/Empty)`
  Reset heightmap cache.

19.49.4 Parameters

- `~fixed_frame_id (String, required)`
  Fixed frame ID used for transforming pointcloud from previous coordinate to current coordinate.
- `~center_frame_id (String, required)`
  Center frame ID used for transforming pointcloud from previous coordinate to current coordinate.
- `~tf_queue_size (Int, default: 10)`
  Queue size for `tf::MessageFilter`.
- `~use_offset (Bool, default: false)`
  Use averaging height offset to fit input and `prev_pointcloud`.
  This parameter can be changed by `dynamic_reconfigure`.
- `~use_bilateral (Bool, default: false)`
  Use bilateral filtering after accumulation.
  This parameter can be changed by `dynamic_reconfigure`.
- `~bilateral_filter_size (Integer, default: 5)`
  Kernel size of bilateral filtering.
  This parameter can be changed by `dynamic_reconfigure`.
• \texttt{~bilateral\_sigma\_color} (Double, default: 0.04)
  filter sigma of color space.
  This parameter can be changed by \texttt{dynamic\_reconfigure}.
• \texttt{~bilateral\_sigma\_space} (Double, default: 5)
  filter sigma of coordinate space.
  This parameter can be changed by \texttt{dynamic\_reconfigure}.

19.49.5 Sample

\texttt{roslaunch jsk\_pcl\_ros sample\_heightmap\_time\_accumulation\.launch}

19.50 HeightmapToPointCloud

Convert a heightmap to pointcloud.
19.50.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input heightmap.
  Encoding should be 32FC2.
- ~input/config (jsk_recognition_msgs/HeightmapConfig)
  Config topic.

19.50.2 Publishing Topic

- ~output (sensor_msgs/PointCloud2)
  Output pointcloud.
- ~output/config (jsk_recognition_msgs/HeightmapConfig)
  Config topic.

19.50.3 Parameters

- ~keep_organized (Bool, default: False)
  Whether to keep pointcloud organized or not.

19.50.4 Sample

```
roslaunch jsk_pcl_ros sample_heightmap_to_pointcloud.launch
```
19.51 HintedHandleEstimator

Detect a handle grasp pose from pointcloud and point as hint.

19.51.1 Subscribing Topic

- `~cloud(sensor_msgs/PointCloud2)`
  
  Input pointcloud

- `~point(geometry_msgs/PointStamped)`
  
  3D Point (You can get from rviz “Publish Point” or image_view2)
19.51.2 Publishing Topic

- \texttt{~handle\_pose} (geometry\_msgs/PoseStamped)
  estimated handle pose
- \texttt{~handle\_length} (std\_msgs/Float64)
  This topic is advertised but not published for now.
- \texttt{~handle} (jsk\_recognition\_msgs/SimpleHandle)
  Estimated handle pose with handle width.
- \texttt{~debug\_marker} (visualization\_msgs/Marker)
  the result of calculating handle direction
- \texttt{~debug\_marker\_array} (visualization\_msgs/MarkerArray)
  estimated handle visualization

19.51.3 Parameters

- \texttt{~finger\_l} (Float, default: 0.03)
- \texttt{~finger\_w} (Float, default: 0.01)
- \texttt{~finger\_d} (Float, default: 0.02)
• ~arm_l (Float, default: 0.05)
• ~arm_w (Float, default: 0.1)
• ~arm_d (Float, default: 0.02)

19.51.4 Sample

roslaunch jsk_pcl_ros sample_hinted_handle_estimator.launch

19.52 HintedPlaneDetector

19.52.1 What Is This

Estimate plane parameter from small ‘hint’ pointcloud and grow it to detect larger plane.

Algorithm is:

1. Detect hint plane from small hint pointcloud using RANSAC
2. Filter ~input pointcloud based on distance and normal direction with hint plane.
3. Detect plane from the pointcloud using RANSAC
4. Segment clusters out of the inliers of the detected plane based on euclidean metrics
5. Apply density filter
6. Extract points from the nearest segmented clusters to the centroid of hint plane
7. Compute convex hull of the extracted points
19.52.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud. It is required to have normal and xyz fields.
- `~input/hint/cloud (sensor_msgs/PointCloud2)`
  Hint pointcloud to estimate plane parameter and only xyz fields are required.

19.52.3 Publishing Topic

- `~output/polygon (geometry_msgs/PolygonStamped)`
- `~output/polygon_array (jsk_recognition_msgs/PolygonArray)`
- `~output/inliers (pcl_msgs/PointIndices)`
- `~output/coefficients (pcl_msgs/ModelCoefficients)`
  Result of detection.
- `~output/hint/polygon (geometry_msgs/PolygonStamped)`
- `~output/hint/polygon_array (jsk_recognition_msgs/PolygonArray)`
- `~output/hint/inliers (pcl_msgs/PointIndices)`
- `~output/hint/coefficients (pcl_msgs/ModelCoefficients)`
  Result of detection of hint pointcloud.
- `~output/hint_filtered_indices (pcl_msgs/PointIndices)`
- `~output/plane_filtered_indices (pcl_msgs/PointIndices)`
- `~output/density_filtered_indices (pcl_msgs/PointIndices)`
- `~output/euclidean_filtered_indices (pcl_msgs/PointIndices)`
  Candidate point indices filtered by each filtering phase.

19.52.4 Parameters

Parameters for detecting hint plane

- `~hint_outlier_threshold (Double, default: 0.1)`
  Outlier threshold to detect hint plane using RANSAC
- `~hint_max_iteration (Integer, default: 100)`
  Maximum iteration number to detect hint plane using RANSAC
- `~hint_min_size (Integer, default: 100)`
  Minimum number of inliers in hint plane
Parameters for filtering pointcloud with hint

- `~enable_distance_filtering` (Bool, default: True)
  Whether to filter `~input` by distance from hint convex polygon.
  The distance is defined by `~outlier_threshold`.
- `~enable_normal_filtering` (Bool, default: True)
  Whether to filter `~input` by angular difference from normal of hint plane.
- `~normal_filter_eps_angle` (Double, default: 0.01)
  Maximum allowable angle in radians to filter candidate points before detecting larger plane.
  Normal direction is computed from hint plane.

Parameters for detecting larger plane

- `~outlier_threshold` (Double, default: 0.1)
  Outlier threshold to detect larger plane using RANSAC
- `~max_iteration` (Integer, default: 100)
  Maximum iteration number to detect larger plane using RANSAC
- `~eps_angle` (Double, default: 0.01)
  Maximum allowable angle in radians to detect larger plane and normal direction is computed from hint plane.
- `~min_size` (Integer, default: 100)
  Minimum number of inliers in larger plane

Parameters for filtering by euclidean clustering

- `~enable_euclidean_filtering` (Bool, default: True)
  Whether to filter `~input` by euclidean clustering.
- `~euclidean_clustering_filter_tolerance` (Double, default: 0.001)
  Tolerance distance in meters in euclidean clustering to filter far points.
- `~euclidean_clustering_filter_min_size` (Integer, default: 20)
  Minimum cluster size in euclidean clustering to filter far points.
  This parameter is not used but `~min_size` is used instead.
Parameters for filtering by density

- `~enable_density_filtering` (Bool, default: True)
  Whether to filter `~input` by density (radius search).
- `~density_radius` (Double, default: 0.1)
- `~density_num` (Integer, default: 10)
  These parameters are used in density filtering.
  The only points which have at least `~density_num` neighbors within `~density_radius` distance [m] will pass.

19.52.5 Sample

```
roslaunch jsk_pcl_ros sample_hinted_plane_detector.launch
```

19.53 HintedStickFinder

Detect a cylinder from pointcloud and line in 2-D image as hint.

19.53.1 Subscribing Topic

- `~input` (sensor_msgs/PointCloud2)
  Input pointcloud
- `~input/camera_info` (sensor_msgs/CameraInfo)
  Camera parameter where hint line is defined
- `~input/hint/line` (geometry_msgs/PolygonStamped)
  Hint line described in 2-D image.
  Only the first and second points in the polygon will be used.
19.53.2 Publishing Topic

- `~debug/line_filtered_indices (pcl_msgs/PointIndices)`
  Indices of input pointcloud which is filtered by hint line.
- `~debug/line_filtered_normal (sensor_msgs/PointCloud2)`
  Normal pointcloud of filtered pointcloud.
  This topic is advertised but not published for now.
- `~debug/cylinder_marker (visualization_msgs/Marker)`
  Marker topic to visualize detected stick
- `~output/cylinder_pose (geometry_msgs/PoseStamped)`
  Pose of detected stick.
- `~output/inliers (pcl_msgs/PointIndices)`
  Inliers of detected stick.
- `~output/coefficients (pcl_msgs/ModelCoefficients)`
  Coefficients of detected stick. The coefficients are \([cx, cy, cz, dx, dy, dz, r, h]\).

19.53.3 Parameters

- `~min_radius (Double, default: 0.05)`
- `~max_radius (Double, default: 0.2)`
  Minimum and maximum radius of cylinder fitting in meters.
- `~filter_distance (Double, default: 0.2)`
  Maximum distance in meters from hint line to points which can be cylinder candidate.
- `~outlier_threshold (Double, default: 0.01)`
  Outlier threshold in cylinder fitting in meters.
- `~max_iteration (Integer, default: 100)`
  Maximum number of iterations in cylinder fitting
- `~eps_angle (Double, default: 0.1)`
  This parameter is not used for now.
- `~min_probability (Integer, default: 0.8)`
  Required minimum probability of cylinder fitting
- `~cylinder_fitting_trial (Integer, default: 3)`
  The number of cylinder fitting trials when no cylinder is found
- `~min_inliers (Integer, default: 10)`
  Minimum number of inliers in cylinder fitting.
- `~eps_2d_angle (Double, default: 0.1)`
  Maximum allowable angle difference in radians between hint line and detected stick. This evaluation is done in 2-D coordinate system.
Parameters above can be changed by `dynamic_reconfigure`.

- `~not_synchronize` (Boolean, default: False)
  Do not synchronize `~input`, `~input/camera_info` and `~input/hint/line` if this parameter is True. `~input/camera_info` and `~input/hint/line` are stored in nodelet and latest of the messages are used for new `~input pointcloud`.

- `~use_normal` (Boolean, default: False)
  Do not run normal estimation inside of the nodelet and normal fields of `~input` are used instead.

### 19.53.4 Sample

```
roslaunch jsk_pcl_ros sample_hinted_stick_finder.launch
```

### 19.54 HSIColorFilter

**Input Cloud**

**Filtered Cloud**

Filter pointcloud based on HSI range.

#### 19.54.1 Subscribing Topic

- `~input` (sensor_msgs/PointCloud2)
  Input pointcloud. rgb field is required.

- `~indices` (pcl_msgs/PointIndices)
  Indices of pointcloud. only available if `~use_indices` is true.
19.54.2 Publishing Topic

- `~output (sensor_msgs/PointCloud2)
  Filtered pointcloud.
- `~color_space (sensor_msgs/PointCloud2)
  Color space visualization for debugging

19.54.3 Parameters

- `~h_limit_max (Integer, default: 127)
- `~h_limit_min (Integer, default: -128)
- `~s_limit_max (Integer, default: 255)
- `~s_limit_min (Integer, default: 0)
- `~i_limit_max (Integer, default: 255)
- `~i_limit_min (Integer, default: 0)
  Color range to filter.
19.55 ICPRegistration
Register two pointclouds based on icp like registration technique.

### 19.55.1 Subscribing Topics

- **~input (sensor_msgs/PointCloud2)**
  Target pointcloud.

- **~input/camera_info (sensor_msgs/CameraInfo)**
  Camera info. This topic is always subscribed but needed only when ~correspondence_algorithm == Projective (1).

- **~input_reference (sensor_msgs/PointCloud2)**
  Reference pointcloud. frame_id of this pointcloud is ignored.
  This topic is subscribed only when ~synchronize_reference is false. Only one of ~input_reference and ~input_reference_array can be used.

- **~input_reference_array (jsk_recognition_msgs/PointsArray)**
  Array of reference pointcloud. ICPRegistration uses the reference which provides the best fitting score.
  This topic is subscribed only when ~synchronize_reference is false. Only one of ~input_reference and ~input_reference_array can be used.

- **~input_reference_add (sensor_msgs/PointCloud2)**
  Reference pointcloud in addition to ~input_reference or ~input_reference_array.
  This topic is subscribed only when ~synchronize_reference is false.

- **~input_box (jsk_recognition_msgs/BoundingBox)**
  Bounding box to align pointcloud with.
  This topic is subscribed only when ~synchronize_reference is false and ~align_box is true.

- **~input_offset (geometry_msgs/PoseStamped)**
  Offset of pose.
  It will only be subscribed if ~synchronize_reference is false and ~align_box is false and ~use_offset_pose is true.

- **~reference (sensor_msgs/PointCloud2)**
  Reference pointcloud.
  This topic is subscribed only when ~synchronize_reference is true.

### 19.55.2 Publishing Topics

- **~output (sensor_msgs/PointCloud2)**
  Reference pointcloud aligned with target pointcloud.

- **~output_pose (geometry_msgs/PoseStamped)**
  Result pose of alignment.

- **~debug/source (sensor_msgs/PointCloud2)**

- **~debug/target (sensor_msgs/PointCloud2)**
• `~debug/result (sensor_msgs/PointCloud2)`
  Pointcloud for debugging.

• `~debug/flipped (sensor_msgs/PointCloud2)`
  This topic is advertised but not published for now.

• `~icp_result (jsk_recognition_msgs/ICPResult)`
  Result pose of alignment with score and ID of best fitted reference pointcloud.

• `~output/latest_time (std_msgs/Float32)`
  latest computation time

• `~output/average_time (std_msgs/Float32)`
  average computation time

### 19.55.3 Advertising Services

• `~icp_align (jsk_recognition_msgs/ICPAlign)`
  Service API of registration using target and reference pointcloud.

• `~icp_service (jsk_recognition_msgs/ICPAlignWithBox)`
  Service API of registration using target pointcloud and bounding box.

  Reference pointcloud should be stored in advance from `~input_reference`, `~input_reference_array` or `~input_reference_add`.

### 19.55.4 Parameters

Parameters for subscribing topics

• `~synchronize_reference (Bool, default: false)`
  If true, `~input` and `~reference` will be subscribed with synchronization.

  If false, parameter `~align_box` and `~use_offset_pose` are enabled. Also, `~input_reference`, `~input_reference_array` and `~input_reference_add` will be subscribed separately.

• `~align_box (Bool, default: false)`
  If true, `~input` and `~input_box` will be subscribed with synchronization.

• `~use_offset_pose (Bool, default: false)`
  If `~align_box` is false and `~use_offset_pose` is true, `~input` and `~input_offset` will be subscribed with synchronization.

  If both of `~align_box` and `~use_offset_pose` are false, `~input` will be subscribed.
Parameters for ICP alignment

- ~use_normal (Bool, default: false)
  Use normal information in registration. In order to use this feature, reference and target pointcloud should have
  valid normal fields.
- ~transform_3dof (Bool, default: false)
  Add constraint to transform estimation on 3D (1D rotation + 2D translation) from header frame of input cloud.
  See TfTransformCloud to change header frame of point cloud.

Parameters below can be changed by dynamic_reconfigure.

- ~algorithm (Int, default: 0)
  Should be one of ICP (0), GICP (1) or NDT (2).
- ~correspondence_algorithm (Int, default: 0)
  Should be one of NN (0) or Projective (1).
- ~max_iteration (Int, default: 100)
  Maximum iterations of ICP alignment.
- ~correspondence_distance (Float, default: 10)
  Maximum correspondence distance in meters.
- ~transform_epsilon (Float, default: 1e-9)
  Maximum allowable difference between two consecutive transformations for an optimization to be considered
  as having converged to the final solution.
- ~euclidean_fitness_epsilon (Float, default: 0.01)
  Maximum allowed Euclidean error between two consecutive steps in the ICP loop, before the algorithm is
  considered to have converged.
- ~ransac_iterations (Int, default: 1000)
  Number of iterations RANSAC should run for.
- ~ransac_outlier_threshold (Float, default: 0.05)
  Outlier distance threshold for the internal RANSAC outlier rejection loop.
- ~rotation_epsilon (Float, default: 2e-3)
  Maximum allowable difference between two consecutive rotations for an optimization to be considered as having
  converged to the final solution.
  This parameter is used only when ~algorithm is GICP (1).
- ~correspondence_randomness (Int, default: 20)
  Number of neighbors used when selecting a point neighborhood to compute covariances.
  This parameter is used only when ~algorithm is GICP (1).
- ~maximum_optimizer_iterations (Int, default: 20)
  Maximum number of iterations at the optimization step.
  This parameter is used only when ~algorithm is GICP (1).
• ~ndt_resolution (Float, default: 1.0)
• ~ndt_step_size (Float, default: 0.05)
• ~ndt_outlier_ratio (Float, default: 0.35)

These parameters are not used for now.

Other parameters

• ~use_flipped_initial_pose (Bool, default: true)

Whether to consider flipped initial pose.

19.55.5 Sample Launch

roslaunch jsk_pcl_ros sample_icp_registration.launch

19.56 ImageRotateNodelet

19.56.1 @Deprecated

image_rotate in jsk_pcl_ros is deprecated.

Please use image_pipeline’s image_rotate(http://wiki.ros.org/image_rotate).

19.57 in_hand_recognition_manager.py

![Image of in_hand_recognition_manager.py](image)
19.57.1 What Is This
This node concatenate transformations for recognizing object in hand.

19.57.2 Subscribing Topic
- ~input (geometry_msgs/PoseStamped)
  Teacher pose.
- ~input/result (geometry_msgs/PoseStamped)
  Difference pose.

19.57.3 Publishing Topic
- ~output (geometry_msgs/PoseStamped)
  Concatenated transformations.
- ~output/recognition (geometry_msgs/PoseStamped)
  This topic is just a relay of ~input.

19.57.4 Sample
roslaunch jsk_pcl_ros sample_in_hand_recognition_manager.launch

19.58 IncrementalModelRegistration

19.58.1 What Is This
Build a full-model from sequential captured data.

### 19.58.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud. RGB field is required.
- `~input/pose (geometry_msgs/PoseStamped)`
  Initial pose to estimate accurate pose of the pointcloud.
- `~input/indices (pcl_msgs/PointIndices)`
  Indices to mask object in `~input` pointcloud.

### 19.58.3 Publishing Topic

- `~output/non_registered (sensor_msgs/PointCloud2)`
  Pointcloud just concatenated according to `~input/pose`
- `~output/registered (sensor_msgs/PointCloud2)`
  Pointcloud refined by ICP.

### 19.58.4 Internally Using Services

- `~icp_service (jsk_recognition_msgs/ICPAlign)`
  ICP service interface to refine model.

### 19.58.5 Advertising Services

- `~start_registration (std_srvs/Empty)`
  Trigger to start registration.

### 19.58.6 Parameters

- `~frame_id (String, default: multisense/left_camera_optical_frame)`
  Frame ID used for output topics.

### 19.58.7 Sample

```bash
roslaunch jsk_pcl_ros sample_incremental_model_registration.launch
```
19.59 InteractiveCuboidLikelihood

19.59.1 What Is This

Compute cuboid likelihood at given point.

19.59.2 Subscribing Topic

- `~input(sensor_msgs/PointCloud2)

  Input point cloud.

19.59.3 Publishing Topic

- `~output(std_msgs/Float32)

  Cuboid likelihood.

19.59.4 Parameter

- `~frame_id(String, default: odom)

  Frame ID of interactive marker, which is generated internally.

- `~sensor_frame(String, default: odom)

  Frame ID of sensor.

- `~initial_pos(List of Float, default: [0, 0, 0])

  Initial x, y, z of interactive marker in meters.
• ~initial_rot (List of Float, default: [0, 0, 0])
  Initial roll, pitch, yaw of interactive marker in radians.

Parameters below can be changed by dynamic_reconfigure.

• ~dx (Float, default: 0.1)
• ~dy (Float, default: 0.1)
• ~dz (Float, default: 0.1)
  Dimension of interactive marker in meters.

• ~use_range_likelihood (Bool, default: False)
  Set to true if you want to update likelihood based on geometry respected to plane.

• ~range_likelihood_local_min_z (Float, default: 0.0)
• ~range_likelihood_local_max_z (Float, default: 0.0)
  Allowed minimum and maximum distance from plane in meters.

• ~outlier_distance (Float, default: 0.1)
  Threshold to regard points as inlier in meters.

• ~min_inliers (Int, default: 10)
  Minimum number of inliers.

• ~use_occullusion_likelihood (Bool, default: False)
  Set to true if you want to take occlusion into account when compute likelihood.

• ~plane_distance_error_power (Float, default: 2.0)
  Power used for computing error at each candidate point.

• ~use_inside_points_distance_zero (Bool, default: False)
  Set to true if you want to treat plane inside points as distance = 0.
  This parameter is used when ~use_occullusion_likelihood is false.

• ~expected_density (Float, default: 0.01)
  Expected average side in meters of cuboid one point occupies.

• ~use_inliers (Bool, default: False)
  Set to true if you want to take inlier likelihood into account.

• ~inliers_power (Float, default: 2.0)
  Power used for computing inlier likelihood.
  This parameter is used when ~use_inliers is true.

• ~use_support_plane_angular_likelihood (Bool, default: False)
  Set to true if you want to take angular likelihood of support plane into account.

• ~support_plane_angular_likelihood_weight_power (Float, default: 1.0)
  Power used for computing angular likelihood of support plane.
  This parameter is used when ~use_support_plane_angular_likelihood is true.
- `~use_surface_area_likelihood` (Bool, default: False)
  Set to true if you want to take surface area likelihood into account.
- `~surface_area_error_power` (Float, default: 1.0)
  Power used for computing surface area likelihood.
- `~use_polygon_likelihood` (Bool, default: False)
  Set to true if you want to take polygon likelihood into account.

19.59.5 Sample

```bash
roslaunch jsk_pcl_ros sample_plane_supported_cuboid_estimator.launch
```
19.60 IntermittentImageAnnotator

19.60.1 What Is This

![Image of IntermittentImageAnnotator](image_url)

Blue axis is the direction of interest region.
1. Store images when ~shutter service is called
2. Publish snapshots as one concatenated image
3. Subscribe ~output/screenrectangle to get ROI.
4. Publish ROI information to ~output namespace.
5. Publish pointcloud inside of the ROI to ~output/cloud if ~store_pointcloud is set

19.60.2 Subscribing Topic

- ~input/image and ~input/camera_info (sensor_msgs/Image and sensor_msgs/CameraInfo)
  Input image and camera info.
- ~input/cloud (sensor_msgs/PointCloud2)
  Input pointcloud to be clipped by specified ROI.
- ~output/screenrectangle (geometry_msgs/PolygonStamped)
  ROI. We expect to use image_view2.

19.60.3 Publishing Topic

- ~output (sensor_msgs/Image)
  Snapshots as one concatenated image.
- ~output/direction (geometry_msgs/PoseStamped)
  Direction of ROI as PoseStamped. z-axis directs the center of ROI.
- ~output/roi (jsk_recognition_msgs/PosedCameraInfo)
  Publish ROI of specified region as PosedCameraInfo.
- ~output/cloud (sensor_msgs/PointCloud2)
  Pointcloud inside of ROI. pointcloud is stored when ~shutter service is called and its timestamp will be updated according to the latest image.
- ~output/marker (visualization_msgs/Marker)
  Marker to visualize ROI (~output/roi).

19.60.4 Parameters

- ~fixed_frame_id (String, default: odom)
  Fixed frame id to resolve tf.
- ~max_image_buffer (Integer, default: 5)
  The maximum number of images to store in this nodelet.
- ~rate (Float, default: 1.0)
  Publishing rate of concatenated images in [Hz].
• ~store_pointcloud(Boolean, default: false)
  Store pointcloud if it’s true
• ~keep_organized(Boolean, default: false)
  Keep pointcloud organized after clipping by specified ROI.

19.60.5 Advertising Service

• ~shutter(std_srvs/Empty)
  Take a snapshot
• ~clear(std_srvs/Empty)
  Clear images stored in the nodelet.
• ~request(std_srvs/Empty)
  Request publishing concatenated image to ~output.

19.60.6 Sample

```
roslaunch jsk_pcl_ros sample_intermittent_image_annotator.launch
```

19.61 JointStateStaticFilter

19.61.1 What Is This

Pass through input point cloud only when specified robot joints are static.

19.61.2 Subscribing Topic

• ~input(sensor_msgs/PointCloud2)
  Input point cloud to which you want to apply this filter.
• ~input_joint_state(sensor_msgs/JointState)
  Input joint state of the robot.

19.61.3 Publishing Topic

• ~output(sensor_msgs/PointCloud2)
  Filtered point cloud.
19.61.4 Parameter

- `~joint_names` (List of String, default: `[]`)
  Target joint list to check if static.
  If no joint is specified, this node does not publish anything.

19.61.5 Sample

```
roslaunch jsk_pcl_ros sample_joint_state_static_filter.launch
```

19.62 KeypointsPublisher

19.62.1 What Is This

This nodelet will calculate the NARF keypoints and publish.
19.62.2 Subscribing Topic

• ~input(sensor_msgs/PointCloud2)
  Input point cloud. Point type should be pcl::PointXYZ.
  The point cloud should be organized.

19.62.3 Publishing Topic

• ~nerf_keypoints(sensor_msgs/PointCloud2)
  Point cloud of keypoints.

19.62.4 Parameters

None.

19.62.5 Sample

roslaunch jsk_pcl_ros sample_keypoints_publisher.launch

19.63 Kinfu

Use kinfu (kinect fusion) for model generation and SLAM.

19.63.1 Note

Kinfu is available only with CUDA enabled PCL.
Please see here for more information.

19.63.2 Subscribing Topics

• ~input/camera_info(sensor_msgs/CameraInfo)
  Intrinsic camera parameter of depth image
• ~input/depth(sensor_msgs/Image)
  Input depth image.
• ~input/color(sensor_msgs/Image)
  RGB color image (subscribed if ~integrate_color is true).
19.63.3 Publishing Topics

- `~output (geometry_msgs/PoseStamped)`
  Pose of camera
- `~output/cloud (sensor_msgs/PointCloud2)`
  Generated point cloud of kinfu model.
- `~output/depth (sensor_msgs/Image)`
  Generated depth of kinfu model in current camera view.
- `~output/rendered_image (sensor_msgs/Image)`
  Rendered image of kinfu model in current camera view.
- `~output/status (jsk_recognition_msgs/TrackerStatus)`
  Status of icp tracking. Succeeding or lost.

19.63.4 Advertising Services

- `~reset (std_srvs/Empty)`
  Reset tracking and mapping of kinfu.
- `~save_mesh (std_srvs/Empty)`
  Convert tsdf to mesh using marching cubes algorithm, saved mesh.obj under `~save_dir`.

19.63.5 Parameters

- `~queue_size (Int, default: 10)`
  Size of message queue for synchronization.
- `~auto_reset (Boolean, default true)`
  Flag to auto reset if ICP tracking is lost.
- `~integrate_color (Boolean, default: false)`
  Flag to integrate color for tracking and mapping. If true, `~input/color` is also Subscribed.
- `~slam (Boolean, default: false)`
  Flag to publishing tf map relative to `~fixed_frame_id`.
- `~fixed_frame_id (String, default: odom_init)`
  Used when `~slam` is true.
- `~save_dir (String, default: .)`
  Save directory for mesh and texture images.
- `~n_textures (Int, default: -1)`
  The number of textures to be used to create texture mesh with below logics:
  - `-1`: all textures are used and texture mesh is saved
  - `0`: no textures are used and polygon mesh is saved
- >0: n_textures_ textures are used and texture mesh is saved

- volume_size (Float, default: 3.0)
  TSDF Volume size for Kinect Fusion.

### 19.63.6 Sample

#### Without SLAM

| roslaunch jsk_pcl_ros sample_kinfu.launch |
| rosservice call /kinfu/save_mesh # saves mesh model below |
| rosservice call /save_mesh_server/request # saves mesh with context (bbox) |

#### With SLAM

| gdown https://drive.google.com/uc?id=0B9P1L--7Wd2vMDA4NW9ySEpoczQ -O $(rospack find -- jsk_pcl_ros)/sample/data/hrp2_apc_2016-07-27-22-08-02.bag |
| export ROBOT=HRP2JSKNTS |
| # you need to resolve dependencies on private projects here to see robot model on rviz. |
| roslaunch jsk_pcl_ros sample_kinfu_hrp2_apc.launch |
19.64 LineSegmentCollector

19.64.1 What is this?
Collect line segments.

19.64.2 Subscribing Topics

- `~input` (sensor_msgs/PointCloud2)
  Input pointcloud.
- `~input_indices` (jsk_recognition_msgs/ClusterPointIndices)
  Cluster point indices of lines.
- `~input_coefficients` (jsk_recognition_msgs/ModelCoefficientsArray)
  Coefficients of lines.
- `~trigger` (jsk_recognition_msgs/TimeRange)
  Trigger for resetting collected line segments buffer.
  Only start field will be used.
19.64.3 Publishing Topics

- `~output/cloud(sensor_msgs/PointCloud2)`
  Pointcloud of collected line segments.
- `~output/coefficients(jsk_recognition_msgs/ModelCoefficientsArray)`
- `~output/inliers(jsk_recognition_msgs/ClusterPointIndices)`
- `~output/polygons(jsk_recognition_msgs/PolygonArray)`
  These topics are advertised but not published for now.
- `~debug/connect_segments/inliers(jsk_recognition_msgs/ClusterPointIndices)`
  Connected inliers of collected lines.

19.64.4 Parameters

- `~rotate_type(String, default: tilt_two_way)`
  This parameter is not used for now.
- `~segment_connect_normal_threshold(Double, default: 0.9)`
  Threshold of dot product of normal to connect clusters.
- `~ewma_tau(Double, default: 0.2)`
  Tau parameter of EWMA to connect clusters.
- `~outlier_threshold(Double, default: 0.01)`
  This parameter is not used for now.

19.64.5 Sample

```
roslaunch jsk_pcl_ros sample_line_segment_collector.launch
```
19.65 LineSegmentDetector

19.65.1 What is this?
Detect lines in a point cloud.

19.65.2 Subscribing Topics
- ~input (sensor_msgs/PointCloud2)
  Input point cloud.
- ~input_indices (jsk_recognition_msgs/ClusterPointIndices)
  Input indices of the cluster in a point cloud.

19.65.3 Publishing Topics
- ~debug/line_marker (visualization_msgs/Marker)
  Marker topic to visualize detected line.
- ~output/inliers (jsk_recognition_msgs/ClusterPointIndices)
- ~output/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
  Result of detection.
19.65.4 Parameters

- `~approximate_sync (Bool, default: false)`
  Whether to allow approximate synchronization of input topics.
- `~method_type (Int, default: 0)`
  The type of sample consensus method to use.
  - 0: SAC_RANSAC
  - 1: SAC_LMEDS
  - 2: SAC_MSAC
  - 3: SAC_RRANSAC
  - 4: SAC_RMSAC
  - 5: SAC_MLESAC
  - 6: SAC_PROSAC
- `~outlier_threshold (Double, default: 0.005)`
  Outlier threshold in meters to detect plane using RANSAC.
- `~max_iterations (Int, default: 1000)`
  Maximum iteration number to detect larger plane using RANSAC.
- `~min_indices (Int, default: 1000)`
  Minimum number of points which construct a line.
- `~min_length (Double, default: 0.1)`
  Minimum length of each line in meters.
- `~line_width (Double, default: 0.01)`
  Width of line marker published to `~debug/line_marker`.

19.65.5 Sample

```
roslaunch jsk_pcl_ros sample_line_segment_detector.launch
```
19.66 LINEMODDetector

19.66.1 What Is This

A nodelet to detect object using LINEMOD.

19.66.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Input pointcloud.

19.66.3 Publishing Topic

- ~output (sensor_msgs/PointCloud2)
  Result of detection as pointcloud.
- ~output/mask (sensor_msgs/Image)
  Result of detection as mask image.
- ~output/pose (geometry_msgs/PoseStamped)
  Pose of detected template
- ~output/template (sensor_msgs/PointCloud2)
  Template pointcloud at identity pose.
19.66.4 Parameters

- `~template_file(String, default: template)`
  Path to template files (.pcd and _poses.yaml).
  e.g.) If the templates are /foo/bar.pcd and /foo/bar_poses.yaml, then this parameter should be /foo/bar.

- `~gradient_magnitude_threshold(Double, default: 10.0)`
  Gradient magnitude threshold

- `~detection_threshold(Double, default: 0.75)`
  Detection threshold

19.66.5 Sample

```
roslaunch jsk_pcl_ros sample_linemod_detector.launch
```

19.67 LINEMODTrainer

19.67.1 What Is This

A nodelet to train LINEMOD data from pointcloud and indices to mask the objects. This nodelet stores data of pointcloud and if you call `~start_training` service, it will train the data and dump the templates into lmt file.
19.67.2 Subscribing Topic

• ~input (sensor_msgs/PointCloud2)
  This pointcloud should be able to be converted into pcl::PointXYZRGBA data.
• ~input/indices (pcl_msgs/PointIndices)
  Indices to mask object in ~input pointcloud.
• ~input/info (sensor_msgs/CameraInfo)
  Camera parameter to sample viewpoint.

19.67.3 Publishing Topic

• ~output/range_image (sensor_msgs/Image)
• ~output/colored_range_image (sensor_msgs/Image)
• ~output/sample_cloud (sensor_msgs/PointCloud2)
  Image and pointcloud generated by viewpoint sampling.

19.67.4 Advertising Services

• ~start_training (std_srvs/Empty)
  Start training and dump result into a file.
• ~clear (std_srvs/Empty)
  Clear stored data.

19.67.5 Parameters

• ~output_file (String, default: template)
  A file path to dump trained data.
  e.g.) If this parameter is set to /foo/bar, then /foo/bar.linemod, /foo/bar.pcd and /foo/bar_poses.yaml will be created.
• ~sample_viewpoint (Bool, default: True)
  Generate training data by samplngenerating viewpoint if this parameter is set to true.
• ~sample_viewpoint_angle_step (Double, default: 40.0)
• ~sample_viewpoint_angle_min (Double, default: -80.0)
• ~sample_viewpoint_angle_max (Double, default: 80.0)
• ~sample_viewpoint_radius_step (Double, default: 0.2)
• ~sample_viewpoint_radius_min (Double, default: 0.4)
• ~sample_viewpoint_radius_max (Double, default: 0.8)
  Viewpoint sampling parameters. Pose of model is sampled by golden ratio spatial technique.
- \texttt{-n\_points (Int, default: 150)}
  
  Number of viewpoint set.

19.67.6 Sample

\begin{verbatim}
roslaunch jsk_pcl_ros sample_linemod_trainer.launch
\end{verbatim}

19.68 \texttt{marker\_appender.py}

Concatenate all subscribed \texttt{visualization\_msgs/Marker} into one \texttt{visualization\_msgs/MarkerArray}.

19.68.1 Subscribing Topics

- \texttt{marker (visualization\_msgs/Marker)}
  
  Marker.
19.68.2 Publishing Topics

- marker_array (visualization_msgs/MarkerArray)
  Marker array.

19.68.3 Sample

```bash
roslaunch jsk_pcl_ros sample_marker_appender.launch
```

19.69 MaskImageClusterFilter

Segment Clouds with mask_image and Clustering Methods (example. SuperVoxel).

19.69.1 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Input point cloud.
  It must be synchronized with ~target.
- ~target (jsk_recognition_msgs/ClusterPointIndices)
  Result of Some Clustering methods
• ~input/mask (sensor_msgs/Image)
  Mask image which is used to filter the input segmented clusters.

• ~input/camera_info (sensor_msgs/CameraInfo)
  Camera parameters of the image.

19.69.2 Publishing Topic

• ~output (pcl_msgs/PointIndices)
  Indices of the points masked with ~input/mask and ~target.

19.69.3 Parameters

None.

19.69.4 Sample

```bash
roslaunch jsk_pcl_ros sample_mask_image_cluster_filter.launch
```

19.70 MaskImageFilter

Extract indices of pointcloud which is masked by mask image. The pointcloud is no need to be organized.
19.70.1 Subscribing Topic

- `~input(sensor_msgs/PointCloud2)`
  Input point cloud to be filtered.
- `~input/mask(sensor_msgs/Image)`
  Mask image.
- `~input/camera_info(sensor_msgs/CameraInfo)`
  Camera parameters of the mask image.

19.70.2 Publishing Topic

- `~output(pcl_msgs/PointIndices)`
  Indices of the points masked by `~input/mask`.

19.70.3 Parameters

- `~negative(Bool, default: false)`
  Flip the max region of mask image or not.

19.70.4 Sample

```bash
roslaunch jsk_pcl_ros sample_mask_image_filter.launch
```

19.71 MovingLeastSquareSmoothing
19.71.1 What Is This

This nodelet will subscribe point cloud, and publish smoothed one. Please refer about MovingLeastSquare in http://pointclouds.org/documentation/tutorials/resampling.php

19.71.2 Subscribing Topics

- `~input` (sensor_msgs/PointCloud2)
  
  Input pointcloud.

19.71.3 Publishing Topics

- `~output` (sensor_msgs/PointCloud2)
  
  Smoothed pointcloud.

19.71.4 Parameters

- `~gauss_param_set` (Boolean, default: False)
  
  Whether set gauss param or not
- `~search_radius` (Double, default: 0.03)
  
  Nearest Neighbor Search param
- `~use_polynomial_fit` (Boolean, default: False)
  
  Whether use polynomial_fit or not
- `~polynomial_order` (Int, default: 2)
  
  Set polynomial order
- `~calc_normal` (Boolean, default: True)
  
  calc_normal

19.71.5 Sample

```bash
roslaunch jsk_pcl_ros sample_moving_least_square_smoothing.launch
```
19.72 MultiPlaneExtraction

Extract the points above the planes between ~min_height and ~max_height.

19.72.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2):
  Input pointcloud.
- ~indices (jsk_recognition_msgs/ClusterPointIndices)
- ~input_polygons (jsk_recognition_msgs/PolygonArray)
- ~input_coefficients (jsk_recognition_msgs/ModelCoefficientsArray):
  The input planes.
  If ~use_indices parameter is false, ~indices is used.
  If ~use_coefficients parameter is false, ~coefficients is not used. (Instead ~use_sensor_frame should be enabled in order to determine normal direction of planes.)
19.72.2 Publishing Topics

- `~output` (sensor_msgs/PointCloud2):
  Pointcloud above the planes between `~min_height` and `~max_height`.
- `~output_nonplane_cloud` (sensor_msgs/PointCloud2):
  Pointcloud above the planes is not between `~min_height` and `~max_height`.
- `~output/indices` (pcl_msgs/PointIndices)
  PointIndices of points which are between `~min_height` and `~max_height`.

19.72.3 Parameters

- `~keep_organized` (Bool, default: True)
  Keep organized point cloud or not.
- `~min_height` (Double, default: 0.0)
- `~max_height` (Double, default: 0.5)
  Minimum and maximum height of 3-D polygonal region to extract points.
- `~max_queue_size` (Integer, default: 100)
  Queue length for subscribing topics.
- `~use_indices` (Bool, default: True)
  Use indices of planar regions to filter if it’s set true. You can disable this parameter to filter pointcloud which is not the same pointcloud to segment planes.
- `~magnify` (Double, default: 0.0)
  Magnify planes by this parameter. The unit is m.
- `~use_async` (Boolean, default: False)
  Approximate sync input topics.
- `~use_coefficients` (Bool, default: False)
  Use coefficients topic to determine viewpoint. For avoiding plane flipping, either this parameter or `~use_sensor_frame` should be enabled.
- `~use_sensor_frame` (Boolean, default: False)
  Set viewpoint as a TF sensor frame. If this parameter is enabled, `~use_coefficients` is ignored and handled as false.
- `~sensor_frame` (String, default: head_root)
  Specify frame_id of sensor origin. This parameter is enabled only when `~use_sensor_frame` is enabled.
19.72.4 Issues

- Normal direction of planes

Normal directions of each planes are computed as followings:

Both ~use_sensor_frame and ~use_coefficients is disabled by default, which means the normal directions are computed individually on every callback and can be flipped on the continuous time domain.

Generally it’s better to set either ~use_sensor_frame or ~use_coefficients as true for stable detection.

19.72.5 Sample

```bash
roslaunch jsk_pcl_ros sample_multi_plane_extraction.launch
```

19.73 MultiPlaneSACSegmentation

19.73.1 What Is This

Segment planes from point cloud.

This node can be applied to unorganized point cloud.

If you have organized one, we recommend you to use jsk_pcl/OrganizedMultiPlaneSegmentation for faster segmentation.
19.73.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input point cloud.
- `~input_normal (sensor_msgs/PointCloud2)`
  Input point cloud which contains normal for each point.
- `~input_clusters (jsk_recognition_msgs/ClusterPointIndices)`
  Input clusters for each of which planes are segmented.
- `~input_imu (sensor_msgs/Imu)`
  Input IMU.
  `linear_acceleration` field is used for plane segmentation.
  Please see `~use_imu_parallel` and `~use_imu_perpendicular`.

19.73.3 Publishing Topic

- `~output_indices (jsk_recognition_msgs/ClusterPointIndices)`
  Plane indices.
- `~output_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Normal coefficients of each plane.
- `~output_polygons (jsk_recognition_msgs/PolygonArray)`
  Plane Polygons.

19.73.4 Parameter

- `~use_normal (Bool, default: False)`
- `~use_clusters (Bool, default: False)`
- `~use_imu_parallel (Bool, default: False)`
- `~use_imu_perpendicular (Bool, default: False)`

If `~use_normal` is set to True, `~input_normal` will be subscribed.
If either `~use_imu_parallel` or `~use_imu_perpendicular` is set to True, `~input_imu` will be subscribed.
`~use_imu_parallel` should be enabled when you want to determine a plane parallel to the `linear_acceleration` axis.
On the other hand, `~use_imu_perpendicular` should be enabled when you want to determine a plane perpendicular to the `linear_acceleration` axis.
`~use_clusters` can be enabled when none of parameters above is enabled, and then `~input_clusters` will be subscribed.

So, the possible input patterns are below:
1. `~input, ~input_normal, ~input_imu`
2. `~input, ~input_normal`
3. `~input,~input_imu`
4. `~input,~input_clusters`
5. `~input`

- `~outlier_threshold` (Float, default: 0.01)
  Distance threshold in meters to remove outliers.
  This parameter can be changed by `dynamic_reconfigure`.
- `~max_iterations` (Int, default: 1000)
  Maximum number of iterations in plane segmentation.
  This parameter can be changed by `dynamic_reconfigure`.
- `~min_inliers` (Int, default: 5000)
  Minimum number of points you allow about segmented planes.
  This parameter can be changed by `dynamic_reconfigure`.
- `~min_points` (Int, default: 1000)
  Minimum total number of outliers in each segmentation used for defining inputs at next loop.
  This parameter can be changed by `dynamic_reconfigure`.
- `~min_trial` (Int, default: 3)
  Number of times for which this node tries to segment planes when number of inlier points < `~min_inliers`.
  This parameter can be changed by `dynamic_reconfigure`.
- `~eps_angle` (Float, default: 0.01)
  Maximum allowed difference between the model normal and the given axis in radians.
  This parameter is enabled only when `~use_imu_parallel` or `~use_imu_perpendicular` is enabled.
  This parameter can be changed by `dynamic_reconfigure`.
- `~normal_distance_weight` (Float, default: 0.1)
  Relative weight (between 0 and 1) to give to the angular distance (0 to pi/2) between point normals and the plane normal.
  This parameter can be changed by `dynamic_reconfigure`.

### 19.73.5 Sample

```bash
roslaunch jsk_pcl_ros sample_multi_plane_sac_segmentation.launch
```
19.74 NormalDirectionFilter

`jsk_pcl/NormalDirectionFilter` filters pointcloud based on the direction of the normal. It can filter pointcloud based on 1) static direction and 2) direction based on imu linear_acceleration.

19.74.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  
  input normal pointcloud. Point type is `pcl::Normal`.

- ~input_imu (sensor_msgs/Imu)
  
  imu message, which is enabled if ~use_imu parameter is true

19.74.2 Publishing Topics

- ~output (pcl_msgs/PointIndices)
  
  result of filtering as indices. You can use `pcl/ExtractIndices` to get pointcloud of the indices.
19.74.3 Parameters

- `~use_imu` (Boolean, default: False):
  Enable `~input_imu` topic and set target direction based on imu linear acceleration.
- `~eps_angle` (Double, default: 0.2):
  Allowed angle difference in [rad] to regard the normal as required direction.
  This parameter can be changed by `dynamic_reconfigure`.
- `~angle_offset` (Double, default: 0.0):
  Offset parameter to the angle difference.
  This parameter can be changed by `dynamic_reconfigure`.
- `~direction` (Double Array, required):
  if `~use_imu` is false, the direction should be specified with this parameter.
- `~queue_size` (Integer, default: 200):
  The length of input queue.

19.74.4 Sample

```
roslaunch jsk_pcl_ros sample_normal_direction_filter.launch
```
19.75 NormalEstimationIntegralImage

19.75.1 What is this?

Compute normals for an organized point cloud using integral images.

19.75.2 Subscribing Topics

- \texttt{~input (sensor\_msgs/PointCloud2)}
  
  Input point cloud. (point type is \texttt{pcl::PointXYZRGB})

19.75.3 Publishing Topics

- \texttt{~output (sensor\_msgs/PointCloud2)}
  
  Output normals. (point type is \texttt{pcl::Normal})

- \texttt{~output\_with\_xyz (sensor\_msgs/PointCloud2)}
  
  Output point cloud with normal field. (point type is \texttt{pcl::PointXYZRGBNormal})
19.75.4 Parameters

- `max_depth_change_factor` (Double, default: 0.02)
  max depth change factor
- `normal_smoothing_size` (Double, default: 20.0)
  normal smoothing size parameter
- `estimation_method` (Int, default: 1)
  Estimation method.
  - 0: AVERAGE_3D_GRADIENT
  - 1: COVARIANCE_MATRIX
  - 2: AVERAGE_DEPTH_CHANGE
- `depth_dependent_smoothing` (Boolean, default: false)
  Use depth dependent smoothing.
- `border_policy_ignore` (Boolean, default: true)
  Ignore border policy.

Parameters above can be changed by `dynamic_reconfigure`.

19.75.5 Sample

```
roslaunch jsk_pcl_ros sample_normal_estimation_integral_image.launch
```
This nodelet is almost the same as `pcl/NormalEstimationOMP` of `pcl_ros` package, but it can handle timestamp correctly.

### 19.76.1 Subscribing Topic

- `~input` (sensor_msgs/PointCloud2)
  
  Input pointcloud. Point type is `pcl::PointXYZRGB`.

### 19.76.2 Publishing Topic

- `~output` (sensor_msgs/PointCloud2)
  
  Output pointcloud, point type is `pcl::Normal`.
- `~output_with_xyz` (sensor_msgs/PointCloud2)
  
  Output pointcloud, point type is `pcl::PointXYZRGBNormal`.
- `~output/latest_time` (std_msgs/Float32)
  
  latest computation time
- `~output/average_time` (std_msgs/Float32)
  
  average computation time
19.76.3 Parameters

- `~number_of_threads` (Int, default: 0)
  Number of hardware threads to use.
  If 0 is specified, the number will be automatically determined.
- `~k_search` (Int, default: 10)
  Number of k-nearest neighbors to search for.
  This parameter can be changed by `dynamic_reconfigure`.
- `~radius_search` (Float, default: 0.0)
  Sphere radius for nearest neighbor search.
  This parameter can be changed by `dynamic_reconfigure`.

Note that either `~k_search` or `~radius_search` must be set to 0.

19.76.4 Sample

```bash
roslaunch jsk_pcl_ros sample_normal_estimation_omp.launch
```
19.77 OctomapServerContact
19.77.1 What Is This

Occupancy grid map in which the contact sensor model is available. OctomapServerContact class is derived from OctomapServer class[1]. [1] http://wiki.ros.org/octomap_server

This node has topic interface. You can pass the output of vision and contact sensors.

19.77.2 Subscribing Topics

- `contact_sensors_in (jsk_recognition_msgs::ContactSensorArray)`: input contact sensor data.
- `cloud_in (sensor_msgs/PointCloud2)`: input pointcloud.
- `proximity_in (sensor_msgs/PointCloud2)`: input pointcloud from proximity sensors.
19.77.3 Publishing Topics

- **octomap_full**(octomap_msgs/Octomap)
  The complete maximum-likelihood occupancy map as compact OctoMap binary stream, encoding free and occupied space. The full message contains the complete probabilities and all additional data stored in the tree. See octomap_msgs for deserializing the message. (cited from http://wiki.ros.org/octomap_server)

- **octomap_binary**(octomap_msgs/Octomap)
  The complete maximum-likelihood occupancy map as compact OctoMap binary stream, encoding free and occupied space. The binary message only distinguishes between free and occupied space but is smaller. See octomap_msgs for deserializing the message. The complete maximum-likelihood occupancy map as compact OctoMap binary stream, encoding free and occupied space. The binary message only distinguishes between free and occupied space but is smaller. See octomap_msgs for deserializing the message. (cited from http://wiki.ros.org/octomap_server)

- **octomap_point_cloud_centers**(sensor_msgs/PointCloud2)
  The centers of all occupied voxels as point cloud, useful for visualization. Note that this will have gaps as the points have no volumetric size and OctoMap voxels can have different resolutions! Use the MarkerArray topic instead. (cited from http://wiki.ros.org/octomap_server)

- **occupied_cells_vis_array**(visualization_msgs/MarkerArray)
  All occupied voxels as “box” markers for visualization in RViz. Be sure to subscribe to the topic occupied_cells_vis in RViz! (cited from http://wiki.ros.org/octomap_server)

- **free_cells_vis_array**(visualization_msgs/MarkerArray)
  All free voxels as “box” markers for visualization in RViz. Be sure to subscribe to the topic free_cells_vis in RViz!

- **unknown_cells_vis_array**(visualization_msgs/MarkerArray)
  All unknown voxels as “box” markers for visualization in RViz. Be sure to subscribe to the topic unknown_cells_vis in RViz!

- **octomap_unknown_point_cloud_centers**(sensor_msgs/PointCloud2)
  The centers of all unknown voxels as point cloud, useful for visualization. Note that this will have gaps as the points have no volumetric size, but all unknown voxels are divided into resolution size.

- **frontier_cells_vis_array**(visualization_msgs/MarkerArray)
  All frontier voxels as “box” markers for visualization in RViz. Be sure to subscribe to the topic frontier_cells_vis in RViz!

- **octomap_frontier_point_cloud_centers**(sensor_msgs/PointCloud2)
  The centers of all frontier voxels as point cloud, useful for visualization. Note that this will have gaps as the points have no volumetric size, but all frontier voxels are divided into resolution size.

- **projected_map**(nav_msgs/OccupancyGrid)
  Downprojected 2D occupancy map from the 3D map. Be sure to remap this topic if you have another 2D map server running. New / changed in octomap_mapping 0.4.4: The topic is now projected_map by default to avoid collisions with static 2D maps (cited from http://wiki.ros.org/octomap_server)

19.77.4 Advertising Services

- **octomap_binary**(octomap_msgs/GetOctomap)
  Same with original OctomapServer. The complete maximum-likelihood occupancy map as compact OctoMap binary stream, encoding free and occupied space.

- **~clear_bbx**(octomap_msgs/BoundingBoxQuery)
  Same with original OctomapServer. Clears a region in the 3D occupancy map, setting all voxels in the region to “free”

- **~reset**(std_srvs/Empty)
  Same with original OctomapServer. Resets the complete map
19.77.5 Parameters

- `~self_see_default_padding` (Double, default: 0.001)
  Same as the parameter in pr2_navigation_self_filter. Padding of contact sensor link.

- `~self_see_default_scale` (Double, default: 1.0)
  Same as the parameter in pr2_navigation_self_filter. Scale of contact sensor link.

- `~self_see_links` (Array of link configuration, required)
  Same as the parameter in pr2_navigation_self_filter. Configuration of links for contact sensor. Link configuration consists of name (required), padding (optional), and scale (optional).

- `~frame_id` (String, default: /map)
  Same with original OctomapServer. Static global frame in which the map will be published. A transform from sensor data to this frame needs to be available when dynamically building maps.

- `~resolution` (float, default: 0.05)
  Same with original OctomapServer. Resolution in meter for the map when starting with an empty map. Otherwise the loaded file's resolution is used.

- `~base_frame_id` (string, default: base_footprint)
  Same with original OctomapServer. The robot's base frame in which ground plane detection is performed (if enabled)

- `~height_map` (bool, default: true)
  Same with original OctomapServer. Whether visualization should encode height with different colors

- `~color/r/g/b/a` (float)
  Same with original OctomapServer. Color for visualizing occupied cells when ~height_map=False, in range [0:1]

- `~color_free/r/g/b/a` (float)
  Color for visualizing free cells

- `~color_unknown/r/g/b/a` (float)
  Color for visualizing unknown cells.

- `~color_frontier/r/g/b/a` (float)
  Color for visualizing frontier cells.

- `~sensor_model/max_range` (float, default: -1 (unlimited))
  Same with original OctomapServer. Maximum range in meter for inserting point cloud data when dynamically building a map. Limiting the range to something useful (e.g. 5m) prevents spurious erroneous points far away from the robot.

- `~sensor_model/[hit|miss]` (float, default: 0.7 / 0.4)
  Same with original OctomapServer. Probabilities for hits and misses in the sensor model when dynamically building a map

- `~sensor_model/[min|max]` (float, default: 0.12 / 0.97)
  Same with original OctomapServer. Minimum and maximum probability for clamping when dynamically building a map
- `~latch` (bool, default: True for a static map, false if no initial map is given)
  
  Same with original OctomapServer. Whether topics are published latched or only once per change. For maximum performance when building a map (with frequent updates), set to false. When set to true, on every map change all topics and visualizations will be created.

- `~filter_ground` (bool, default: false)
  
  Same with original OctomapServer. Whether the ground plane should be detected and ignored from scan data when dynamically building a map, using pcl::SACMODEL_PERPENDICULAR_PLANE. This clears everything up to the ground, but will not insert the ground as obstacle in the map. If this is enabled, it can be further configured with the `~ground_filter/…` parameters.

- `~ground_filter/distance` (float, default: 0.04)
  
  Same with original OctomapServer. Distance threshold for points (in z direction) to be segmented to the ground plane

- `~ground_filter/angle` (float, default: 0.15)
  
  Same with original OctomapServer. Angular threshold of the detected plane from the horizontal plane to be detected as ground

- `~ground_filter/plane_distance` (float, default: 0.07)
  
  Same with original OctomapServer. Distance threshold from z=0 for a plane to be detected as ground (4th coefficient of the plane equation from PCL)

- `~pointcloud_[min|max]_[x|y|z]` (float, default: −/+ infinity)
  
  Minimum and maximum height of points to consider for insertion in the callback. Any point outside of this interval will be discarded before running any insertion or ground plane filtering. You can do a rough filtering based on height with this, but if you enable the ground_filter this interval needs to include the ground plane.

- `~occupancy_[min|max]_[x|y|z]` (float, default: −/+ infinity)
  
  Minimum and maximum height of occupied cells to be consider in the final map. This ignores all occupied voxels outside of the interval when sending out visualizations and collision maps, but will not affect the actual octomap representation.

### 19.77.6 Sample

**Simulation**

```bash
roslaunch jsk_pcl_ros sample_octomap_server_contact_pr2.launch
```

**Real**

Plug the depth sensor which can be launched by openni.launch and run the below command.

```bash
roslaunch jsk_pcl_ros octomap_server_contact.launch
```

If you want to see how frontier grids works, run the below command.

```bash
roslaunch jsk_pcl_ros sample_octomap_contact.launch
```
19.78 OctreeChangePublisher

19.78.1 What Is This

This nodelet will publish the difference of sequential pointcloud. You can get the newly generated pointclouds. Difference with pcl_ros/SegmentDifference refer https://github.com/jsk-ros-pkg/jsk_recognition/pull/67

19.78.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  
  Input point cloud.

19.78.3 Publishing Topic

- `~octree_change_result (sensor_msgs/PointCloud2)`
  
  Point cloud which did not exist in previous time.
19.78.4 Parameters

- \~resolution (Float, default: 0.02)
  Octree resolution at lowest octree level in meters.
- \~noise_filter (Int, default: 2)
  Minimum amount of points required within leaf node to become serialized.

These parameters can be changed by `dynamic_reconfigure`.

19.78.5 Sample

```bash
roslaunch jsk_pcl_ros sample_octree_change_publisher.launch
```

19.79 OctreeVoxelGrid
19.79.1 What Is This

This is a nodelet to get voxel grided cloud from point cloud. VoxelGrid[1] and ApproximateVoxelGrid also generate voxel filtered cloud, but those clouds are not aligned.

i.e. This nodelet outputs the center position of each aligned voxel grid, and other voxel filtered functions output the other position such as the centroid of belonging points.


19.79.2 Subscribing Topics

- ~input (sensor_msgs/PointCloud2):
  input pointcloud.

19.79.3 Publishing Topics

- ~output (sensor_msgs/PointCloud2):
  Voxel grided clouds.
- ~output_marker (visualization_msgs/Marker):
- ~output_marker_array (visualization_msgs/MarkerArray):
  Voxel grided clouds.
- ~output_resolution (std_msgs/Float32)
  Resolution of octree.

19.79.4 Parameters

- ~publish_marker (Boolean, default: true):
  Publish voxel grided clouds as visualization_msgs/Marker topic.

19.79.5 Dynamic Reconfigure Parameters

- ~resolution (Double, default: 0.1):
  Resolution of voxel grid. If resolution is set as 0, it just relays input pointcloud.
- ~marker_color (String, default: z)
  Method to decide color of marker. z, x, y or flat can be selected.
- ~marker_color_alpha (Double, default: 0.8)
  Alpha value of marker color.
19.79.6 Sample

Plug the depth sensor which can be launched by openni.launch and run the below command. Voxel grided clouds are published with topics /octree_voxel_grid/octgrid/output and /octree_voxel_grid/octgrid/output_marker.

roslaunch jsk_pcl_ros octree_voxel_grid.launch

19.80 OrganizedEdgeDetector

19.80.1 What is this?

Detect edges from organized point cloud.

Note: This node requires PCL > 1.7.2.
19.80.2 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  
  Input organized point cloud.

19.80.3 Publishing Topics

- ~output_normal (sensor_msgs/PointCloud2)
  
  Estimated normal of whole point cloud.
  
  Point type is pcl::Normal.
  
  This topic is published only when ~publish_normal is true.

- ~output_nan_boundary_edge_indices (pcl_msgs/PointIndices)
- ~output_occluding_edge_indices (pcl_msgs/PointIndices)
- ~output_occluded_edge_indices (pcl_msgs/PointIndices)
- ~output_curvature_edge_indices (pcl_msgs/PointIndices)
- ~output_rgb_edge_indices (pcl_msgs/PointIndices)
- ~output_indices (pcl_msgs/PointIndices)
  
  Output edge indices.
  
  ~output_indices means all of the edge indices above.

- ~output_nan_boundary_edge (sensor_msgs/PointCloud2)
- ~output_occluding_edge (sensor_msgs/PointCloud2)
- ~output_occluded_edge (sensor_msgs/PointCloud2)
- ~output_curvature_edge (sensor_msgs/PointCloud2)
- ~output_rgb_edge (sensor_msgs/PointCloud2)
- ~output (sensor_msgs/PointCloud2)
  
  Output point cloud of edge.
  
  ~output means all of the edge point cloud above.

- ~output_straight_edges_indices (jsk_recognition_msgs/ClusterPointIndices)
  
  Cluster indices containing straight edges.
  
  This topic is published only when ~use_straightline_detection is true.

- ~edge_image (sensor_msgs/Image)
- ~hough_image (sensor_msgs/Image)
  
  Debug image. Visualize the input and output image of cv::HoughLinesP.
  
  These topics are published only when ~use_straightline_detection is true and ~publish_debug_image is true.
### 19.80.4 Parameters

**Parameters for estimating normal**

- `~publish_normal (Boolean, default: false)
  
  Publish normal pointcloud or not.

- `~max_depth_change_factor, (Double, default: 0.02)
  
  Max depth change factor.

- `~normal_smoothing_size, (Double, default: 20.0)
  
  Normal smoothing size parameter.

- `~estimation_method, (Int, default: 1)
  
  Estimation method.

  Choose from `AVERAGE_3D_GRADIENT(1), `COVARIANCE_MATRIX(1) and
  `AVERAGE_DEPTH_CHANGE(2).

- `~depth_dependent_smoothing (Boolean, default: false)
Use depth dependent smoothing.

- `~border_policy_ignore` (Boolean, default: true)
  Ignore border policy.

**Parameters for estimating edge**

- `~max_search_neighbors` (Int, default: 100)
  The maximum search distance for deciding occluding and occluded edges
- `~depth_discontinuation_threshold` (Double, default: 0.04)
  Threshold about depth discontinuation between neighboring points in meters.
- `~use_nan_boundary` (Boolean, default: false)
  Add NAN Boundary Edge to estimating edge type
- `~use_occluding` (Boolean, default: true)
  Add Occluding Edge to estimating edge type
- `~use_occluded` (Boolean, default: true)
  Add Occluded Edge to estimating edge type
- `~use_curvature` (Boolean, default: true)
  Add High Curvature Edge to estimating edge type
- `~use_rgb` (Boolean, default: false)
  Add RGB Canny Edge to estimating edge type

**Parameters for estimating straight edge**

- `~use_straightline_detection` (Boolean, default: true)
  Estimate Straight Lines or not.
- `~rho` (Double, default: 1)
  Distance resolution of the accumulator in pixels.
- `~theta` (Double, Default: 1)
  Angle resolution of the accumulator in degrees.
- `~straightline_threshold` (Int, default: 50)
  The minimum number of intersections to ‘detect’ a line.
- `~min_line_length` (Double, default: 50)
  The minimum number of points that can form a line.
  Lines with less than this number of points are disregarded.
- `~max_line_gap` (Double, default: 10)
  The maximum gap between two points to be considered in the same line.
- `~publish_debug_image` (Boolean, default: true)
  Publish Debug Images.
19.80.5 Sample

```bash
roslaunch jsk_pcl_ros sample_organized_edge_detector.launch
```

19.81 OrganizedMultiPlaneSegmentation

19.81.1 What Is This

This nodelet segments multiple planes from organized pointcloud. It estimates planes based on connected-component analysis using `pcl::OrganizedMultiPlaneSegmentation`. 
1. Estimate normal using integral image.
2. Conduct connected component analysis to estimate planar regions.
3. Connect neighbor planes if the normal directions and the borders of the planes are near enough.
4. Refine plane coefficients of connected planes based on RANSAC. If the areas of the planes after refinement are too small, they will be removed.

These process is implemented in one nodelet in order not to convert pointcloud between PCL and ROS.

19.81.2 Subscribing Topics

- `~input` (sensor msgs/PointCloud2):
  Input pointcloud. This should be organized pointcloud.

19.81.3 Publishing Topics

- `~output` (jsk_recognition_msgs/ClusterPointIndices):
- `~output_polygon` (jsk_recognition_msgs/PolygonArray):
- `~output_coefficients` (jsk_recognition_msgs/ModelCoefficientsArray)

  The inliers, coefficients and convex polygons of the connected polygons.

- `~output_nonconnected` (jsk_recognition_msgs/ClusterPointIndices):
- `~output_nonconnected_polygon` (jsk_recognition_msgs/PolygonArray):
- `~output_nonconnected_coefficients` (jsk_recognition_msgs/ModelCoefficientsArray)

  The inliers, coefficients and polygons of the polygons of connected components analysis.

- `~output_refined` (jsk_recognition_msgs/ClusterPointIndices):
• ~output_refined_polygon (jsk_recognition_msgs/PolygonArray):
The inliers, coefficients and convex polygons of the refined polygons.
• ~output_refined_coefficients (jsk_recognition_msgs/ModelCoefficientsArray):
The inliers, coefficients and convex polygons of the refined polygons.
• ~output_normal (sensor_msgs/PointCloud2):
The pointcloud of normal of ~input pointcloud.
  Point type is pcl::Normal.

19.81.4 Parameters

• ~estimate_normal (Boolean, default: True):
  Estimate normal if it is set to True
• ~publish_normal (Boolean, default: False):
  Publish the result of normal to ~output_normal
• ~max_depth_change_factor (Double, default: 0.02):
  The depth change threshold for computing object borders in normal estimation.
• ~normal_smoothing_size (Double, default: 20.0):
  the size of the area used to smooth normals (depth dependent if ~depth_dependent_smoothing is true)
• ~depth_dependent_smoothing (Boolean, default: False)
  Smooth normal depending on depth
• ~estimation_method (Integer, default: 1)
  Estimation method of normal. You can choose one of AVERAGE_3D_GRADIENT(0),
  COVARIANCE_MATRIX(1) and AVERAGE_DEPTH_CHANGE(2).
• ~border_policy_ignore (Boolean, default: True)
  Ignore border if this is True
• ~min_size (Integer, default: 2000)
  Minimum number of the points on a planar region during connected component analysis. We recommend
  smaller size for this parameter in order to get stable result.
• ~angular_threshold (Double, default: 0.05)
• ~distance_threshold (Double, default: 0.01)
  Distance and angular threshold in connected component analysis.
• ~max_curvature (Double, default: 0.001)
  The maximum curvature allowed for a planar region
• ~connect_plane_angle_threshold (Double, default: 0.2)
• ~connect_distance_threshold (Double, default: 0.01)
  These parameters affect near plane connection. OrganizedMultiPlaneSegmentation connects planes which have
  near normal direction and whose boundaries are near enough.
• ~ransac_refine_coefficients (Boolean, default: True)
  Conduct RANSAC refinement for each plane if it is true.
- `~ransac_refine_outlier_distance_threshold` (Double, default: 0.1)
  Outlier threshold of RANSAC refinement for each plane.
- `~min_refined_area_threshold` (Double, default: 0.04)
- `~max_refined_area_threshold` (Double, default: 10000)
  Minimum and maximum area threshold for each convex polygon.

19.81.5 Sample

```
roslaunch jsk_pcl_ros sample_organized_multi_plane_segmentation.launch
```

19.82 OrganizedPassThrough

19.82.1 What Is This?

Filter organized pointcloud based on specified index range.
19.82.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Organized point cloud.

19.82.3 Publishing Topic

- ~output (sensor_msgs/PointCloud2)
  Filtered point cloud.

19.82.4 Parameter

- ~min_index (Int, default: 0)
  Minimum index along ~filter_field axis that are included.
- ~max_index (Int, default: 640)
  Maximum index along ~filter_field axis that are included.
- ~keep_organized (Boolean, default: true)
  Set keep_organized when extract indices.
- ~remove_nan (Boolean, default: false)
  Remove points with x, y, or z equal to NaN.
- ~filter_limit_negative (Boolean, default: false)
  Set negative when convert indices to point cloud.
- ~filter_field (Int, default: 0)
  - 0: Filter based on x field.
  - 1: Filter based on y field.

These parameters can be changed by dynamic_reconfigure.

19.82.5 Sample

```
roslaunch jsk_pcl_ros sample_organized_pass_through.launch
```
19.83 OrganizedStatisticalOutlierRemoval

Organized version of statistical outlier removal.
### 19.83.1 Subscribing Topic

- ~input (sensormsgs/PointCloud2)
  
  Input pointcloud to be removed outlier.

### 19.83.2 Publishing Topic

- ~output (sensormsgs/PointCloud2)
  
  Output pointcloud.

### 19.83.3 Parameter

- ~keep_organized (Boolean, default: True)
  
  keep organized point cloud or not
- ~negative (Boolean, default: False)
  
  remove outlier or remove other than outlier.
- ~mean_k (Int, default: 2)
  
  mean k value for statistical outlier removal.
- ~stddev (Double, default: 0.0)
  
  std deviation multipelier for statistical outlier removal.

### 19.83.4 Sample

```
roslaunch jsk_pcl_ros organized_statistical_outlier_removal.launch
```
19.84 ParallelEdgeFinder

19.84.1 What is this?

Find parallel edges from multiple lines.

19.84.2 Subscribing Topics

- `~input_indices (jsk_recognition_msgs/ClusterPointIndices)`
  Cluster point indices of multiple lines.
- `~input_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Coefficients of lines for each cluster.

19.84.3 Publishing Topics

- `~output_edges_groups (jsk_recognition_msgs/ParallelEdgeArray)`
  Array of parallel edge pairs.
- `~output_clusters (jsk_recognition_msgs/ClusterPointIndices)`
  Parallel edge clusters extracted from `~input_indices`. 
19.84.4 Parameters

- `~angular_threshold` (Double, default: 0.1)
  
  Allowable angle of inclination of pair of edges in radians.

19.84.5 Sample

```
roslaunch jsk_pcl_ros sample_parallel_edge_finder.launch
```

19.85 ParticleFilterTracking

19.85.1 What Is This

This nodelet tracks the target pointcloud.
19.85.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Input pointcloud
- ~input_change (sensor_msgs/PointCloud2)
  Input change pointcloud, which is only enabled when ~use_change_detection is true and should be synchronized with ~input.
- ~renew_model (sensor_msgs/PointCloud2)
  Reference pointcloud to track.
- ~renew_model_with_marker (visualization_msgs/Marker)
  Reference marker model to track. This will convert marker model to pointcloud. You need to pass the marker whose type is TRIANGLE_LIST and it should have the color.
  This topic is subscribed only when ~align_box is false.
- ~renew_box (jsk_recognition_msgs/BoundingBox)
  Bounding box information to align reference pointcloud model. Available only if ~align_box parameter is true, and it should be synchronized with ~renew_model.

19.85.3 Publishing Topic

- ~track_result (sensor_msgs/PointCloud2)
  Reference pointcloud which is transformed by tracking result.
- ~tracking_result_pose (geometry_msgs/PoseStamped)
  Tracking result as pose of reference pointcloud.
- ~particle (sensor_msgs/PointCloud2)
  Particles during tracking. Only x, y and z are available.
- ~output/latest_time (std_msgs/Float32)
  latest computation time
- ~output/average_time (std_msgs/Float32)
  average computation time
- ~output/rms_angle_error (std_msgs/Float32)
- ~output/rms_distance_error (std_msgs/Float32)
  Root mean squared error of angle/distance.
- ~output/velocity (geometry_msgs/TwistStamped)
  Velocity of object.
- ~output/velocity_norm (std_msgs/Float32)
  Norm of velocity of object.
- ~output/no_move (std_msgs/Bool)
• ~output/no_move_raw (std_msgs/Bool)
  These topics will be true if object looks stable.
• ~output/skipped (std_msgs/Bool)
  This topic is advertised but not published for now.
• ~output/change_marker (visualization_msgs/MarkerArray)
  This topic is advertised only when ~use_change_detection is true, but not published for now.
• ~output/tracker_status (jsk_recognition_msgs/TrackerStatus)
  Current tracking status.
  This topic is published only when ~use_change_detection is true.

19.85.4 Advertising Services

• ~renew_model (jsk_recognition_msgs/SetPointCloud2)
  Service interface to set reference pointcloud.

19.85.5 Parameters

• ~thread_nr (Integer, default: cpu num)
  The number of thread used in tracking
• ~particle_num (Integer, default: ~max_particle_num - 1)
  The number of initial particles
• ~use_normal (Boolean, default: false)
  Use normal information to track or not.
• ~use_hsv (Boolean, default: true)
  If it’s true, tracker uses color information in HSV color space to evaluate likelihood.
• ~track_target_name (String, default: track_result)
  The name of the target, it is used as frame_id of tf.
• ~octree_resolution (Double, default: 0.01)
  Octree resolution to search.
• ~align_box (Bool, default: false)
  If it’s true, tracker subscribes ~renew_box topic and align reference model against the box.
• ~BASE_FRAME_ID (String, default: NONE)
  Coordinate system of the tracker. NONE means “same to frame_id of input pointcloud”.
• ~default_initial_mean (Array of double, default: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
  Mean value of initial sampling.
• ~initial_noise_covariance (Array of double, default: [0.00001, 0.00001, 0.00001, 0.00001, 0.00001, 0.00001])
  Covariance value of initial sampling.
• ~max_particle_num (Integer, default: 1000)
  Maximum number of particles
• ~iteration_num (Integer, default: 1)
  The number of iteration per one frame.
• ~resample_likelihood_thr (Double, default: 0.0)
  Threshold of likelihood to resample (re-initialize) all the particles.
• ~delta (Double, default: 0.99)
  Delta value for KLD sampling.
• ~epsilon (Double, default: 0.2)
  epsilon parameter for KLD sampling.
• ~bin_size_x (Double, default: 0.01)
• ~bin_size_y (Double, default: 0.01)
• ~bin_size_z (Double, default: 0.01)
• ~bin_size_roll (Double, default: 0.01)
• ~bin_size_pitch (Double, default: 0.01)
• ~bin_size_yaw (Double, default: 0.01)
  Size of bin for KLD sampling. Larger value means smaller number of particles.
• ~default_step_covariance_x (Double, default: 0.0001)
• ~default_step_covariance_y (Double, default: 0.0001)
• ~default_step_covariance_z (Double, default: 0.0001)
• ~default_step_covariance_roll (Double, default: 0.004)
• ~default_step_covariance_pitch (Double, default: 0.004)
• ~default_step_covariance_yaw (Double, default: 0.004)
  Covariance value of noise in resampling phase.
• ~reversed (Boolean, default: false)
  Reverse relationship between reference and input. If this parameter is true, tracker transforms input pointcloud instead of reference pointcloud. It is useful when input pointcloud is smaller than reference pointcloud.
  If this parameter is true, KLDSampling is disabled.
• ~not_use_reference_centroid (Boolean, default: false)
  If this parameter is true, tracker does not use centroid of reference pointcloud as the origin of reference pointcloud.
• ~not_publish_tf (Boolean, default: false)
  If this parameter is true, do not publish tf frame.
• ~enable_cache (Boolean, default: false)
  Enable caching of nearest-neighbor search.
  At most one of ~enable_cache and ~enable_organized can be set to true.
• `~enable_organized` (Boolean, default: false)
  Enable using Organized nearest-neighbor search
  At most one of `~enable_cache` and `~enable_organized` can be set to true.
• `~cache_size_x` (Double, default: 0.01)
• `~cache_size_y` (Double, default: 0.01)
• `~cache_size_z` (Double, default: 0.01)
  Resolution of cache voxel grid.
  These parameters are enabled only when `~enable_cache` is true.
• `~max_distance` (Double, default: 0.01)
  Maximum distance between points to take into account when computing likelihood
• `~use_change_detection` (Bool, default: false)
  Use change detection to skip tracking when no change in pointcloud.
• `~static_velocity_thr` (Double, default: 0.1)
  Velocity threshold to regard object is stable.
• `~change_cloud_near_thr` (Double, default: 0.2)
  Distance threshold to trigger tracking when `~use_change_detection` is true.

**19.85.6 Sample**

```bash
roslaunch jsk_pcl_ros sample_particle_filter_tracking.launch
```

... or you can visualize tracking status by using `tracking_info.py` and `tracker_status_info.py`

```bash
roslaunch jsk_pcl_ros sample_particle_filter_tracking_change_detection.launch
```

... or you can try service API `renew_tracking.py` to renew reference pointcloud instead of topic API.

```bash
roslaunch jsk_pcl_ros sample_particle_filter_tracking_service_renew.launch
```
19.86 PeopleDetection

19.86.1 What is this?
Detecting people on a ground plane with RGB-D data.

19.86.2 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
- ~input/info (sensor_msgs/CameraInfo)
- ~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
  Gound plane’s coefficients.

19.86.3 Publishing Topics

- ~boxes (jsk_recognition_msgs/BoundingBoxArray)
  Bounding Box of detected people.
19.86.4 Parameters

- `~filename` (String, default: "")
  training SVM filename.
- `~people_height_threshold` (Double, default: 0.5)
- `~min_confidence` (Double, default: -1.5)
- `~queue_size` (Int, default: 1)
- `~voxel_size` (Double, default: 0.03)
- `~box_width` (Double, default: 0.5)
- `~box_depth` (Double, default: 0.5)

19.86.5 Sample

```bash
roslaunch jsk_pcl_ros sample_people_detection.launch
```

19.87 PlaneSupportedCuboidEstimator

Estimate a cuboid on a plane. Plane information is used as hint. It uses particle filter to estimate pose of cuboid.

`jsk_pcl/InteractiveCuboidLikelihood` is a helper nodelet to confirm likelihood function behaves as expected.
19.87.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  Input pointcloud
- ~fast_input (sensor_msgs/PointCloud2)
  Faster input pointcloud
- ~input/polygon (jsk_recognition_msgs/PolygonArray)
  Planes which may support cuboid object
- ~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)

19.87.2 Publishing Topics

- ~output/result (jsk_recognition_msgs/BoundingBoxArray)
  Result of estimation as bounding box.
- ~output/result_pose (geometry_msgs/PoseStamped)
  Estimated cuboid pose.
- ~output/candidate_cloud (sensor_msgs/PointCloud2)
  Candidate pointcloud extracted from polygons.
- ~output/particles (sensor_msgs/PointCloud2)
  Particles as pointcloud (xyzi)
- ~output/histogram/global/x (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/global/y (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/global/z (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/global/roll (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/global/pitch (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/global/yaw (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/dx (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/dy (jsk_recognition_msgs/HistogramWithRange)
- ~output/histogram/dz (jsk_recognition_msgs/HistogramWithRange)
  Histograms of particles for each dimension
19.87.3 Advertising Services

- \texttt{~reset (std_srvs/Empty)}
  
  Reset particles filters.

19.87.4 Parameters

- \texttt{~sensor\_frame (String, default: odom)}
  
  Frame ID of sensor frame. It is used to compute viewpoint and occlusion.

Parameters below can be changed by \texttt{dynamic\_reconfigure}.

- \texttt{~init\_local\_position\_z\_min (Float, default: 0.0)}
- \texttt{~init\_local\_position\_z\_max (Float, default: 1.0)}
  
  Height limits of initial random polygonal prism at local coordinates in meters.

- \texttt{~init\_local\_orientation\_roll\_mean (Float, default: 0.0)}
- \texttt{~init\_local\_orientation\_roll\_variance (Float, default: 0.005)}
- \texttt{~init\_local\_orientation\_pitch\_mean (Float, default: 0.0)}
- \texttt{~init\_local\_orientation\_pitch\_variance (Float, default: 0.005)}
- \texttt{~init\_local\_orientation\_yaw\_mean (Float, default: 0.0)}
- \texttt{~init\_local\_orientation\_yaw\_variance (Float, default: 0.01)}
- \texttt{~init\_dx\_mean (Float, default: 0.1)}
- \texttt{~init\_dx\_variance (Float, default: 0.001)}
- \texttt{~init\_dy\_mean (Float, default: 0.1)}
- \texttt{~init\_dy\_variance (Float, default: 0.001)}
- \texttt{~init\_dz\_mean (Float, default: 0.1)}
- \texttt{~init\_dz\_variance (Float, default: 0.001)}
  
  Parameters of random gaussian of initial particle cuboid along each axis at local coordinates.

- \texttt{~use\_init\_world\_position\_z\_model (Bool, default: False)}
  
  Whether to enable \texttt{~init\_world\_position\_z\_min} and \texttt{~init\_world\_position\_z\_max} to filter initial particle cuboid.

- \texttt{~init\_world\_position\_z\_min (Float, default: 0.0)}
- \texttt{~init\_world\_position\_z\_max (Float, default: 1.0)}
  
  Height limits of initial particle cuboid at world coordinates in meters.

- \texttt{~disable\_init\_roll (Bool, default: False)}
- \texttt{~disable\_init\_pitch (Bool, default: False)}
  
  Force to set roll/pitch of initial particle cuboid at world coordinates to 0 if True.

- \texttt{~use\_global\_init\_yaw (Bool, default: False)}
  
  Enable \texttt{~init\_global\_orientation\_yaw\_mean} and \texttt{~init\_global\_orientation\_yaw\_variance} to decide initial particle states.
- \texttt{~init\_global\_orientation\_yaw\_mean} (Float, default: 0.0)
- \texttt{~init\_global\_orientation\_yaw\_variance} (Float, default: 0.01)
  Parameters of random gaussian of initial particle cuboid along yaw at world coordinates.
- \texttt{~particle\_num} (Int, default: 1000)
  Number of points in particle.
- \texttt{~step\_x\_variance} (Float, default: 0.0001)
- \texttt{~step\_y\_variance} (Float, default: 0.0001)
- \texttt{~step\_z\_variance} (Float, default: 0.0001)
- \texttt{~step\_roll\_variance} (Float, default: 0.0001)
- \texttt{~step\_pitch\_variance} (Float, default: 0.0001)
- \texttt{~step\_yaw\_variance} (Float, default: 0.0001)
- \texttt{~step\_dx\_variance} (Float, default: 0.0001)
- \texttt{~step\_dy\_variance} (Float, default: 0.0001)
- \texttt{~step\_dz\_variance} (Float, default: 0.0001)
  Parameters of random gaussian for sampling and tracking particle cuboid.
- \texttt{~min\_dx} (Float, default: 0.001)
- \texttt{~min\_dy} (Float, default: 0.001)
- \texttt{~min\_dz} (Float, default: 0.001)
  Minimum limits of random value generated from \texttt{~step\_d?\_variance}.
- \texttt{~use\_init\_polygon\_likelihood} (Bool, default: False)
  Whether to use likelihood of input polygon.
- \texttt{~fast\_cloud\_threshold} (Float, default: 2.0)
  If distance to object is smaller than \texttt{~fast\_cloud\_threshold}, \texttt{~fast\_input} will be used.

19.87.5 Sample

```bash
roslaunch jsk_pcl_ros sample_plane_supported_cuboid_estimator.launch
```
19.88 plane_time_ensync_for_recognition.py

19.88.1 What Is This

This node force `jsk_recognition_msgs/PolygonArray` and `jsk_recognition_msgs/ModelCoefficientsArray` topics to synchronize.

19.88.2 Subscribing Topic

- `planes (jsk_recognition_msgs/PolygonArray)`
- `planes_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  
  Plane polygons and their normal coefficients.
- `timer (sensor_msgs/PointCloud2)`
  
  Trigger topic for publishing synchronized polygons and coefficients.

19.88.3 Publishing Topic

- `ensynced_planes (jsk_recognition_msgs/PolygonArray)`
- `ensynced_planes_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  
  Synchronized polygons and coefficients.
19.88.4 Parameters

None.

19.88.5 Sample

roslaunch jsk_pcl_ros sample_plane_time_ensync_for_recognition.launch

19.89 PointCloudLocalization

Localize 6d pose of robot using ICP registration of pointcloud. It publishes tf transformation from global frame to odometry frame like amcl does.

19.89.1 Subscribing Topic

• ~input (sensor_msgs/PointCloud2)
  Input pointcloud to align.

19.89.2 Publishing Topic

• ~output (sensor_msgs/PointCloud2)
  Concatenated pointcloud.
19.89.3 Parameters

- `~global_frame` (String, default: `map`)
  Frame ID of output pointcloud.
- `~odom_frame` (String, default: `odom`)
  Frame ID of broadcasted transform of localization.
- `~initialize_from_tf` (Bool, default: `false`)
  Whether to initialize transform from `~initialize_tf` frame.
- `~initialize_tf` (String, default: `odom_on_ground`)
  Frame ID used for initialization of transform.
  This parameter is enabled only when `~initialize_from_tf` is true.
- `~clip_unseen_pointcloud` (Bool, default: `false`)
  Whether to filter out pointcloud which cannot be seen by a sensor.
- `~sensor_frame` (String, default: `BODY`)
  Frame ID used for filtering pointcloud before running ICP.
  This parameter is enabled only when `~clip_unseen_pointcloud` is true.
- `~tf_rate` (Double, default: `20.0`)
  Frequency to publish tf transformations [Hz].
- `~cloud_rate` (Double, default: `10.0`)
  Frequency to publish `~output` topic [Hz].
- `~leaf_size` (Double, default: `0.01`)
  Resolution of voxel grid downsampling in meters.
- `~use_normal` (Bool, default: `false`)
  Support normal field in `~input` pointcloud.

19.89.4 Internally Using Services

- `~icp_align` (jsk_pcl_ros/ICPAlign)
  ICP service to align pointcloud

19.89.5 Advertising Services

- `~localize` (std_srvs/Empty)
  Run localization
- `~update_offset` (jsk_recognition_msgs/UpdateOffset)
  Update transformation between odom frame and global frame manually. Currently no tf is resolved.
19.89.6 Sample

```bash
roslaunch jsk_pcl_ros sample_pointcloud_localization.launch
```

19.90 PointcloudDatabaseServer

![PointcloudDatabaseServer Image]

Load PCD or STL model files and publish pointcloud array.

19.90.1 Publishing Topic

- `~output (jsk_recognition_msgs/PointsArray)`
  
  Output pointcloud array.

- `~cloud(sensor_msgs/PointCloud2)`
  
  Output pointcloud.

19.90.2 Parameters

- `~models` (Array of string, required)
  
  This parameter is an array of PCD/STL model file path.

- `~duration` (Float, default: 1.0)
  
  Time between each publish in seconds.

- `~use_array` (Bool, default: True)
  
  If true, pointcloud array will be published to `~output`.
If false, first index of pointcloud array will be published to `~cloud`.

### 19.90.3 Sample

```bash
roslaunch jsk_pcl_ros sample_pointcloud_database_server.launch
```

### 19.91 PointcloudScreenpoint

#### 19.91.1 What is this

Use pointcloud from kinect
Use pointcloud from laser
Use amplified pointclouds published by laser

This is a nodelet to convert (u, v) coordinate on a image to 3-D point. It retrieves 3-D environment as pointcloud. pointcloud_screenpoint_sample.launch is a sample launch file.

19.91.2 Note

This class inherits jsk_topic_tools::ConnectionBasedNodelet, which can start subscribing topics if published topics are subscribed by the other node.

19.91.3 Subscribing Topics

- ~points (sensor_msgs/PointCloud2):
  Pointcloud source to estimate 3D points that the user wanted to specify on a 2D screen
- ~point (geometry_msgs/PointStamped):
  Input point to represent (u, v) image coordinate. Only x and y fields are used and the header frame_id is ignored. If ~synchronization parameter is set True, ~points and ~point are synchronized.
- ~rect (geometry_msgs/PolygonStamped):
  Input rectangular region on image local coordinates. Only x and y fields are used and the header frame_id is ignored. And the region should be rectangular. If ~synchronization parameter is set True, ~points and ~rect are synchronized.
- ~poly (geometry_msgs/PolygonStamped):
Input polygonal region in image local coordinates. If ~synchronization parameter is set True, ~points and ~poly are synchronized.

- ~point_array (sensor_msgs/PointCloud2):
  Input points to represent series of (u, v) image coordinate. Only x and y fields are used and the header frame_id is ignored. If ~synchronization parameter is set True, ~points and ~point_array are synchronized.

### 19.9.1.4 Publishing Topics

- ~output_point (geometry_msgs/PointStamped):
  The topic to be used to publish one point as a result of screenpoint.
- ~output (sensor_msgs/PointCloud):
  The topic to be used to publish series of points as a result of screenpoint.
- ~output_polygon (geometry_msgs/PolygonStamped):
  Projected points of ~rect or ~poly.

### 19.9.1.5 Advertising Services

- ~screen_to_point (jsk_pcl_ros::TransformScreenpoint):
  ROS Service interface to convert (u, v) image coordinate into 3-D point.
  The definition of jsk_pcl_ros::TransformScreenpoint is:

```plaintext
# screen point
float32 x
float32 y

# position in actual world
std_msgs/Header header
geometry_msgs/Point point
geometry_msgs/Vector3 vector
```

With int this service, the latest pointcloud acquired by ~points is used to convert (u, v) into 3-D point.

### 19.9.1.6 Parameters

- ~synchronization (Boolean, default: False):
  If this parameter is set to True, the timestamps of 3-D pointcloud and the target point/rectangle/point array are synchronized.
- ~approximate_sync (Boolean, default: False):
  If this parameter is set to True, approximate synchronization is enabled. This parameter is valid only when ~synchronization is True.
- ~queue_size (Integer, default: 1):
  Queue length of subscribing topics.
- ~crop_size (Integer, default: 10):
  The size of approximate region if ~points pointcloud has nan holes.
• ~search_size (Integer, default: 16):
  Size to search normal of point cloud.
• ~timeout (Double, default: 3.0):
  Timeout to wait for point cloud.

19.92 pointcloud_screenpoint.l

19.92.1 What Is This

This node is a service client of PointcloudScreenPoint.

19.92.2 Subscribing Topic

  • $(param ~sensor_topic)/screenpoint (geometry_msgs/PointStamped)
    Screen point from image_view2.
19.92.3 Publishing Topic

- ray_marker_array (visualization_msgs/MarkerArray)
  Marker of touched point which can be visualized in rviz.
- image_marker (image_view2/ImageMarker2)
  Marker of touched point which can be visualized in image_view2.
- ray_coords (geometry_msgs/PoseStamped)
  Pose of 3D touched point.

19.92.4 Internally Calling Service

- $(param ~ray_srv) (jsk_recognition_msgs/TransformScreenpoint)
  Service server.

19.92.5 Parameter

- ~sensor_topic (String, required)
  Name space of subscribed topic.
  See Subscribing Topic.
- ~ray_srv (String, required)
  Name of service server.
  See Internally Calling Service.
- ~base_frame (String, default: base_footprint)
  Frame ID of ray_coords.

19.92.6 Sample

```bash
roslaunch jsk_pcl_ros sample_pointcloud_screenpoint.launch
```

19.93 PPFRegistration

19.93.1 What is this?
Compute normals for an organized point cloud using integral images.

### 19.93.2 Subscribing Topics

- `~input/cloud(sensor_msgs/PointCloud2)`
  - Input point cloud. (PointXYZ)
- `~input/reference_cloud(sensor_msgs/PointCloud2)`
  - Reference point cloud. (PointXYZ)
  - Subscribe when parameter `use_array` is `false`.
- `~input/reference_array(jsk_recognition_msg/PointsArray)`
  - Reference point cloud array. (PointXYZ)
  - Subscribe when parameter `use_array` is `true`.

### 19.93.3 Publishing Topics

- `~output/cloud(sensor_msgs/PointCloud2)`
  - Output aligned reference point cloud.
  - Publish when parameter `use_array` is `false`.
- `~output/pose_stamped(geometry_msg/PoseStamped)`
  - Output aligned reference pose.
  - Publish when parameter `use_array` is `false`.
• ~output/points_array (jsk_recognition_msg/PointsArray)
  Output aligned reference point cloud array.
  Publish when parameter use_array is true.
• ~output/pose_array (geometry_msg/PoseArray)
  Output aligned reference pose array.
  Publish when parameter use_array is true.

19.93.4 Parameters

• ~use_array (Boolean, default: false)
  Determine whether reference format is jsk_recognition_msg/PointsArray or sensor_msgs/PointCloud2
• ~queue_size (Int, default: 100)
  Queue size
• ~approximate_sync (Boolean, default false)
  Select to use approximate sync policy
• search_radius (Double, default 0.05)
  Search radius for normal calculation
• sampling_rate (Int, default 10)
  Sampling rate for registration
• position_clustering_threshold (Double, default 0.2)
  Position clustering threshold for registration
• rotation_clustering_threshold (Double, default 30)
  Rotation clustering threshold for registration

19.93.5 Sample

roslaunch jsk_pcl_ros sample_ppf_registration.launch
19.94 PrimitiveShapeClassifier

Classify shape for each cluster point indices on planes.

19.94.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  Input XYZRGB point cloud
- ~input/normal (sensor_msgs/PointCloud2)
  Input normal point cloud
- ~input/indices (jsk_recognition_msgs/ClusterPointIndices)
  Input cluster point indices
- ~input/polygons (jsk_recognition_msgs/PolygonArray)
  Input supporting planes
19.94.2 Publishing Topics

- `~output (jsk_recognition_msgs/ClassificationResult)`
  Classification result
- `debug/boundary_indices (jsk_recognition_msgs/ClusterPointIndices)`
  Cluster point indices of boundary points for each clustered objects
- `debug/projected_cloud (sensor_msgs/PointCloud2)`
  Projected boundary points on supporting planes

19.94.3 Parameters

- `~queue_size (Int, default: 100)`
  Queue size for message synchronization
- `~min_points_num (Int, default: 10)`
  Minimum number of points for each cluster.
- `~sac_max_iterations (Int, default: 500)`
  Maximum iteration number for SAC segmentation
- `~sac_distance_threshold (Double, default: 0.005)`
  Distance threshold for SAC Segmentation
- `~sac_radius_limit_min (Double, default: 0.025)`
  Minimum radius for circle estimation
- `~sac_radius_limit_max (Double, default: 0.13)`
  Maximum radius for circle estimation
- `~box_threshold (Double, default: 0.70)`
  Threshold to classify objects as boxes
- `~circle_threshold (Double, default: 0.30)`
  Threshold to classify objects as circles
19.95 publish_clicked_point_bbox.py

19.95.1 What Is This

This node subscribes geometry_msgs/PointStamped message and publishes jsk_recognition_msgs/BoundingBoxArray at that point.

19.95.2 Subscribing Topic

- clicked_point (geometry_msgs/PointStamped)
  Input point whose position is used for bounding box.

19.95.3 Publishing Topic

- bbox_with_clicked_point (jsk_recognition_msgs/BoundingBoxArray)
  Output bounding box array.
  Dimensions of the box is set to 1 [m] and frame ID is set to odom.
  So the transformation between clicked_point/header/frame_id and odom should be resolvable.
19.95.4 Sample

```bash
gerun rosrun jsk_pcl_ros sample_publish_clicked_point_bbox.launch
```

19.96 RearrangeBoundingBox

19.96.1 What is this?

Rearrange and Rescale BoundingBoxArray.

19.96.2 Subscribing Topics

- `~input (jsk_recognition_msgs/BoundingBoxArray)`

19.96.3 Publishing Topics

- `~output (jsk_recognition_msgs/BoundingBoxArray)`
19.96.4 Parameters

- `~offset_x` (Double, default: 0.0)
- `~offset_y` (Double, default: 0.0)
- `~offset_z` (Double, default: 0.0)
- `~scale_x` (Double, default: 1.0)
- `~scale_y` (Double, default: 1.0)
- `~scale_z` (Double, default: 1.0)
- `~rotate_x` (Double, default: 0.0)
- `~rotate_y` (Double, default: 0.0)
- `~rotate_z` (Double, default: 0.0)

19.96.5 Sample

```
roslaunch jsk_pcl_ros sample_rearrange_bounding_box.launch
```
19.97 RegionGrowingMultiplePlaneSegmentation

jsk_pcl/RegionGrowingMultiplePlaneSegmentation estimates multiple planes from pointcloud. It extracts planes based on region growing and evaluation function of connectivity if based on the following equation:

19.97.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2):
  input pointcloud.
  Point type should be pcl::PointXYZRGB.
- ~input_normal (sensor_msgs/PointCloud2):
  normal pointcloud of ~input.
  Point type should be pcl::Normal.
19.97.2 Publishing Topics

- ~output/clustering_result (jsk_recognition_msgs/ClusterPointIndices):
  Result of region growing as cluster.
- ~output/inliers (jsk_recognition_msgs/ClusterPointIndices):
  Set of indices of the polygons.
- ~output(coefficients (jsk_recognition_msgs/ModelCoefficientsArray):
  Array of coefficients of the polygons.
- ~output/polygons (jsk_recognition_msgs/PolygonArray):
  Plane polygons.
- ~output/latest_time (std_msgs/Float32)
  latest computation time
- ~output/average_time (std_msgs/Float32)
  average computation time

19.97.3 Parameters

- ~angular_threshold (Double, default: 0.04)
  Angular threshold in [rad] to connect two points in one cluster.
- ~distance_threshold (Double, default: 0.01)
  Distance threshold in [m] to connect two points in one cluster.
- ~max_curvature (Double, default: 0.1)
  Before extracting planes, filtering out the points which have higher curvature than this value.
- ~min_size (Integer, default: 100)
  The minimum number of the points of each plane.
- ~max_size (Integer, default: 25000)
  The maximum number of the points of each plane.
  Currently this parameter is disabled and unlimited is set instead.
- ~min_area (Double, default: 0.1)
  The minimum area of the convex planes in [m^2].
- ~max_area (Double, default: 100)
  The maximum area of the convex planes in [m^2].
- ~cluster_tolerance (Double, default: 0.1)
  The spatial tolerance in [m] for new cluster candidates.
- ~ransac_refine_outlier_distance_threshold (Double, default: 0.1)
  Outlier threshold in [m] for plane estimation using RANSAC.
• ~ransac_refine_max_iterations (Integer, default: 100)
  The maximum number of the iterations for plane estimation using RANSAC.
  Note that these parameters can be changed by dynamic_reconfigure.

19.97.4 Sample

```bash
roslaunch jsk_pcl_ros sample_region_growing_multiple_plane_segmentation.launch
```

19.98 RegionGrowingSegmentation

19.98.1 What Is This

This node segment point cloud by using `pcl::RegionGrowing`.

19.98.2 Subscribing Topic

• ~input (sensor_msgs/PointCloud2)
  Point cloud to be segmented.
  It should have `xyz` and `normal` field and should not have NaN points.

19.98.3 Publishing Topic

• ~output (jsk_recognition_msgs/ClusterPointIndices)
  Result of clustering.
19.98.4 Parameter

- `~number_of_neighbors` (Int, default: 10)
  Number of the neighbors used in KdTree search.
- `~min_size` (Int, default: 20)
  Minimum number of points that a cluster needs to contain in order to be considered valid.
- `~max_size` (Int, default: 25000)
  Maximum number of points that a cluster needs to contain in order to be considered valid.
- `~smoothness_threshold` (Float, default: \( \frac{\pi}{2} \))
  Threshold value for the angle between normals in radians.
- `~curvature_threshold` (Float, default: 0.1)
  Threshold value for curvature testing.

19.98.5 Sample

```
roslaunch jsk_pcl_ros sample_region_growing_segmentation.launch
```

19.99 `renew_tracking.py`
Call `jsk_recognition_msgs/SetPointCloud2` service from subscribed point cloud.

### 19.99.1 Subscribing Topics

- `selected_pointcloud(sensor_msgs/PointCloud2)`
  
  Point cloud.

### 19.99.2 Calling Services

- `particle_filter_tracker/renew_model(jsk_recognition_msgs/SetPointCloud2)`
  
  Set new point cloud as a callback of `selected_pointcloud`.

### 19.99.3 Parameters

- `~track_target_name(String, default: track_result)`
  
  Dummy name used for calling service internally.

### 19.99.4 Sample

```bash
roslaunch jsk_pcl_ros sample_particle_filter_tracking_service_renew.launch
```

### 19.100 ResizePointsPublisher

![Image of ResizePointsPublisher](image.png)
19.100.1 What is this

ResizePointsPublisher resizes PointCloud generated from depth images. It keeps organized pointcloud. For example you can create QVGA pointcloud from VGA pointcloud of kinect like sensors.

19.100.2 Subscribing Topics

- `~input` (sensor_msgs/PointCloud2): Input PointCloud. The input should be organized pointcloud.
- `~input/mask` (sensor_msgs/Image): Mask image used for automatically updating `~step_x` and `~step_y`.
- `~indices` (pcl_msgs/PointIndices): Points of only these indices will be extracted and resized. To subscribe this topic, `~use_indices` must be set to true.

19.100.3 Publishing Topics.

- `~output` (sensor_msgs/PointCloud2): Output PointCloud. The output will be organized.

19.100.4 Parameters

- `~step_x`, `~step_y` (Double, default: 2): Binning step when resizing pointcloud. These parameters can be changed by `dynamic_reconfigure`.
- `~use_indices` (Boolean, default: false): If true, `~indices` is subscribed.
- `~not_use_rgb` (Boolean, default: false): If you want to resize pointcloud without RGB fields, you need to set this parameter to True.

19.100.5 Sample

```
roslaunch jsk_pcl_ros sample_resize_points_publisher.launch
```
19.101 RGBColorFilter

Filter pointcloud based on RGB range.

19.101.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud. rgb field is required.
- `~indices (pcl_msgs/PointIndices)`
  Indices of pointcloud. only available if `~use_indices` is true.

19.101.2 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Filtered pointcloud.

19.101.3 Parameters

- `~keep_organized (Bool, default: False)`
  Whether to publish organized point cloud if possible.
- `~use_indices (Bool, default: False)`
  If true, apply filter only in `~indices` region.
- `~r_max (Integer, default: 255)`
- `~r_min (Integer, default: 0)`
- `~g_max (Integer, default: 255)`
- `~g_min (Integer, default: 0)`
- `~b_max (Integer, default: 255)`
- `~b_min (Integer, default: 0)`
  Color range to filter.
19.101.4 Sample

```
roslaunch jsk_pcl_ros rgb_color_filter_sample.launch
```

19.102 ROIClipper

19.102.1 What Is This

It retrieves `sensor_msgs/Image` and `sensor_msgs/CameraInfo` and publish `sensor_msgs/Image` of ROI. It is similar to `image_proc/crop_decimate` but you can use `CameraInfo/roi` field to specify ROI.
We expect to use `jsk_pcl/ROIClipper` with `jsk_pcl/AttentionClipper` to get ROI image.

### 19.102.2 Subscribing Topic

- **~input/image (sensor_msgs/Image)**
  
  Input image.

- **~input/camera_info (sensor_msgs/CameraInfo)**
  
  Camera parameter and ROI field should be filled.

  ~input/image and ~input/camera_info should be synchronized if ~not_sync is false.

- **~input/cloud (sensor_msgs/PointCloud2)**
  
  Input point cloud ROI will be applied to.

  This topic is only enabled if ~not_sync is true.

### 19.102.3 Publishing Topic

- **~output (sensor_msgs/Image)**
  
  Image of ROI.

- **~output/cloud_indices (pcl_msgs/PointIndices)**
  
  The indices of the pointcloud which is inside of the interest 3-D region.

- **~output/cloud (sensor_msgs/PointCloud2)**
  
  PointCloud clipped from ~input/cloud and ~input/camera_info.

### 19.102.4 Parameter

- **~not_sync (Bool, default: False)**
  
  If ~not_sync is true, do not need to synchronize camera info and other input topics, and pointcloud clipping is enabled.

- **~keep_organized (Bool, default: False)**
  
  Whether to keep output point cloud organized or not.

### 19.102.5 Sample

```bash
roslaunch jsk_pcl_ros sample_roi_clipper.launch
```
19.103 SelectedClusterPublisher

19.103.1 What Is This

Extract specified point cloud from input point cloud and clusters.

This feature provides functionality which is similar to the combination of jsk_pcl_ros_utils/ClusterPointIndicesToPointIndices + jsk_pcl_ros/ExtractIndices.

The former (this node) subscribes a topic for specifying index, and the latter gets parameter.

19.103.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input point cloud from which specified part will be extracted.
- `~indices (jsk_recognition_msgs/ClusterPointIndices)`
  Input cluster indices.
- `~selected_index (jsk_recognition_msgs/Int32Stamped)`
  Index of cluster indices used for extraction.

19.103.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Selected part of input point cloud.
19.103.4 Parameter

- `~keep_organized` (Bool, default: False)
  Whether to keep organized for output point cloud.

19.103.5 Sample

```bash
roslaunch jsk_pcl_ros sample_selected_cluster_publisher.launch
```

19.104 Snapit

19.104.1 What Is This

This node finds the nearest plane from input pose and publish pose aligned to that plane.

19.104.2 Subscribing Topic

- `~input/polygons` (jsk_recognition_msgs/PolygonArray)
  Input latest plane polygon.
  This topic should be synchronized with `~input/polygon_coefficients`.

- `~input/polygon_coefficients` (jsk_recognition_msgs/ModelCoefficientsArray)
  This topic should be synchronized with `~input/polygons`, but no data in this topic is used for now.
• ~input/plane_align (geometry_msgs/PoseStamped)
  Input pose.
  When this topic is subscribed, ~output/plane_aligned will be published.
• ~input/convex_align (geometry_msgs/PoseStamped)
  Input pose.
  When this topic is subscribed, ~output/convex_aligned will be published.
• ~input/convex_align_polygon (geometry_msgs/PolygonStamped)
  Plane polygon.
  When this topic is subscribed, ~output/convex_aligned_pose_array will be published.

19.104.3 Publishing Topic
• ~output/plane_aligned (geometry_msgs/PoseStamped)
  Pose aligned to nearest plane.
• ~output/convex_aligned (geometry_msgs/PoseStamped)
  Similar to ~output/plane_aligned, but just relay ~input/convex_align when the foot of the perpendicular line is not in any of input planes.
• ~output/convex_aligned_pose_array (geometry_msgs/PoseArray)
  Array of pose of all vertices of input polygon.

19.104.4 Advertising Service
• ~align_footstep (jsk_recognition_msgs/SnapFootStep)
  Service API.
  Each pose is treated like ~input/convex_align.

19.104.5 Parameters
• ~use_service (Bool, default: False)
  Whether to enable ~align_footstep.

19.104.6 Sample
roslaunch jsk_pcl_ros sample_snapit.launch
19.105 SphericalPointCloudSimulator

Simulate a pointcloud which is acquired by spindle laser. Sensor model is spherical laser.

19.105.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  This topic is only used to synchronize timestamp of ~output pointcloud to certain pointcloud. If no ~frame_id is specified, frame_id of ~input is copied to ~output.

19.105.2 Publishing Topics

- ~output (sensor_msgs/PointCloud2)
  Simulated pointcloud.
19.105.3 Parameters

- `~frame_id` (String, default: None)
  frame_id of output pointcloud. If not specified, frame_id of `~input` is copied.

- `~r` (Double, default: 3.0)
  Radius of spherical model.

- `~min_phi` (Double, default: 0.0)
  Minimum angle of scanning plane.

- `~max_phi` (Double, default: 2pi)
  Maximum angle of scanning plane.

- `~scan_range` (Double, default: 4.7)
  Scan range of laser. The default value is same to hokuyo’s parameter.

- `~scan_num` (Integer, default: 1081)
  The number of points in one scan of laser. The default value is same to hokuyo’s parameter.

- `~fps` (Double, default: 40)
  Fps of laser sensor. The default value is same to hokuyo’s parameter.

19.105.4 Sample

```
roslaunch jsk_pcl_ros laser_camera_fov_sample.launch
```
19.106 store-pointcloud.l
19.106.1 What Is This
This node shows point cloud in IRT viewer and stores it into a list internally.

19.106.2 Subscribing Topic
- /openni/depth_registered/points_throttle(sensor_msgs/PointCloud2)
  Input point cloud.

19.106.3 Publishing Topic
None.

19.106.4 Sample
```
roslaunch jsk_pcl_ros sample_store_pointcloud.launch
```

19.107 Subscribing Topic
- ~input(sensor_msgs/PointCloud)
  Input point cloud to add color. No need to be an organized pointcloud.
- ~input/image(sensor_msgs/Image)
  BGR8 color image.
- ~input/camera_info(sensor_msgs/CameraInfo)
  Camera parameters of ~input/image
19.108 SupervoxelSegmentation

Segment pointcloud based on Supervoxel technique. see Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds (J. Papon et al. CVPR2013).

19.108.1 Subscribing Topic

- `~input(sensor_msgs/PointCloud2)`
  
  Input pointcloud. It should have rgb field.

19.108.2 Publishing Topic

- `~output/cloud(sensor_msgs/PointCloud2)`
  
  Output pointcloud downsampled by voxel grid.
- `~output/indices(jsk_recognition_msgs/ClusterPointIndices)`
  
  Clustering result.
19.108.3 Parameters

- ~color_importance (Double, default: 0.2)
  Color importance factor.
- ~spatial_importance (Double, default: 0.4)
  Spatial importance factor.
- ~normal_importance (Double, default: 1.0)
  Normal importance factor.
- ~use_transform (Boolean, default: True)
  Use single cloud transform
- ~seed_resolution (Double, default: 0.1)
  Seed resolution of super voxels.
- ~voxel_resolution (Double, default: 0.008)
  Voxel grid resolution of super voxels.

19.108.4 Sample

```bash
roslaunch jsk_pcl_ros sample_supervoxel_segmentation.launch
```

19.109 TargetAdaptiveTracking
19.109.1 What Is This

This node finds the nearest plane from input pose and publish pose aligned to that plane.

19.109.2 Subscribing Topic

- ~input_obj_cloud(sensor_msgs/PointCloud2)
  Object point cloud used for initialization.
- ~input_bkgd_cloud(sensor_msgs/PointCloud2)
  Background point cloud used for initialization.
- ~input_obj_pose(geometry_msgs/PoseStamped)
  Object pose used for initialization.

Note that ~input_obj_cloud, ~input_bkgd_cloud and ~input_obj_pose should be synchronized.

- ~input_cloud(sensor_msgs/PointCloud2)
  Target point cloud.
- ~input_pose(geometry_msgs/PoseStamped)
  Object pose used for updating motion displacement.

Note that ~input_cloud and ~input_pose should be synchronized.

19.109.3 Publishing Topic

- /selected_pointcloud(sensor_msgs/PointCloud2)
  Template point cloud to track.
- /target_adaptive_tracking/output/object_pose(geometry_msgs/PoseStamped)
  Result pose of tracking.
- /target_adaptive_tracking/output/cloud(sensor_msgs/PointCloud2)
  Relay of ~input_cloud.
- /target_adaptive_tracking/output/normal(sensor_msgs/PointCloud2)
  Point cloud of centroid with normal field.
- /target_adaptive_tracking/output/inliers(sensor_msgs/PointCloud2)
  Point cloud of inliers.
- /target_adaptive_tracking/output/centroids(sensor_msgs/PointCloud2)
  Point cloud of estimated position.
- /target_adaptive_tracking/output/probability_map(sensor_msgs/PointCloud2)
  Point cloud representing probability.
- /target_adaptive_tracking/supervoxel/cloud(sensor_msgs/PointCloud2)
- /target_adaptive_tracking/supervoxel/indices(jsk_recognition_msgs/ClusterPointIndices)
  Result point cloud / cluster point indices of supervoxel segmentation.
• /target_adaptive_tracking/supervoxel/tdp_indices (jsk_recognition_msgs/ClusterPointIndices)
  Filtered cluster point indices of supervoxel segmentation.

19.109.4 Parameters

• ~use_tf (Bool, default: False)
  Whether to look up transform between ~parent_frame_id and ~child_frame_id to update transform.
• ~parent_frame_id (String, default: /track_result)
  Parent frame ID.
  This parameter is used only when ~use_tf is True.
• ~child_frame_id (String, default: /camera_rgb_optical_frame)
  Child frame ID used for /target_adaptive_tracking/output/cloud and /target_adaptive_tracking/output/object_pose.
• ~color_importance (Float, default: 0.2)
• ~spatial_importance (Float, default: 0.4)
• ~normal_importance (Float, default: 1.0)
  Importances used for supervoxel segmentation.
• ~use_transform (Bool, default: True)
  This parameter is not used.
• ~seed_resolution (Float, default: 0.1)
  Average size in meters of resulting supervoxels.
• ~voxel_resolution (Float, default: 0.008)
  Resolution in meters of voxel used.
• ~min_cluster_size (Int, default: 20)
  Minimum number of supervoxels.
• ~threshold (Float, default: 0.4)
  Probability threshold.
• ~bin_size (Int, default: 18)
  Local structural rpy bin.
• ~eps_distance (Float, default: 0.03)
• ~eps_min_samples (Int, default: 3)
  These parameters are not used now.
• ~update_tracker_reference (Bool, default: False)
  Whether to update tracking model.
• ~vfh_scaling (Float, default: 0.7)
  Likelihood scaling factor for vfh matching.
• \texttt{\~color\_scaling} (Float, default: 0.5)
  Likelihood scaling factor for color matching.
• \texttt{\~structure\_scaling} (Float, default: 0.0)
  Likelihood scaling factor for local voxel adjacency.
• \texttt{\~update\_filter\_template} (Bool, default: False)
  Whether to update the particle filter tracking template.
• \texttt{\~history\_window\_size} (Int, default: 5)
  Number of frame after which the unmatched voxel is discarded.

19.109.5 Sample

\texttt{roslaunch jskpclros sample_target_adaptive_tracking.launch}

19.110 TfTransformCloud

19.110.1 Description

Transforms a point cloud in a given target TF frame. Refer discussion at Github issue

19.110.2 Subscribing Topic

• \texttt{\~input} (sensor\_msgs/PointCloud2)
  Input point cloud

19.110.3 Publishing Topic

• \texttt{\~output} (sensor\_msgs/PointCloud2)
  Transformed point cloud which has \texttt{\~target\_frame\_id} as header.frame_id

19.110.4 Parameters

• \texttt{\~target\_frame\_id} (String, default: base\_footprint)
  Parent TF frame to be transformed
19.110.5 Sample

```
<launch>
  <include file="$(find openni_launch)/openni_launch.launch" />
  <node pkg="tf" type="static_transform_publisher"
       args="10 20 30 20 10 20 /camera_link /tf_test_link 100" />
  <node name="pc_transformer" pkg="nodelet" type="nodelet"
        args="load jsk_pcl/TfTransformCloud /camera_nodelet_manager"
        output="screen" />
  <remap from="~input" to="/camera/depth/points" />
  <rosparam>
    target_frame_id: /tf_test_link
  </rosparam>
</node>
</launch>
```

19.111 TiltLaserListener

19.111.1 What is this

Listen to the joint states of tilt/spindle laser and publish time range to scan full 3-D space. You can choose several types of tilt/spindle lasers such as tilt-laser of PR2, infinite spindle laser of multisense.

19.111.2 Subscribing Topics

- ~input(sensor_msgs/JointState):
  Joint angles of laser actuator.

- ~input/cloud(sensor_msgs/PointCloud2):
  Input scan pointcloud. It only used if ~not_use_laser_assembler_service and ~use_laser_assembler are true.
19.111.3 Publishing Topics

• ~output (jsk_recognition_msgs/TimeRange):
  Time range to scan 3-D space.
• ~output_cloud (sensor_msgs/PointCloud2):
  Assembled pointcloud according to time range of ~output. This require ~assemble_scans2 service of laser_assembler.
  This topic is published only when ~use_laser_assembler is true.
• ~output_velocity (geometry_msgs/TwistStamped)
  Velocity of rotating laser. It is only published when ~twist_frame_id is provided.

19.111.4 Internally Using Services

• ~assemble_scans2 (laser_assembler/AssembleScans2):
  A service to build 3-D pointcloud from laser scan. It should be remapped to assemble_scans2 service of laser_assembler.

19.111.5 Advertising Service

• ~clear_cache (std_srvs/Empty)
  Clear cache and restart collecting data.

19.111.6 Parameters

• ~max_queue_size (Integer, default: 100):
  Queue size of subscription.
• ~clear_assembled_scans (Bool, default: false)
  Do not use assembled scans twice.
• ~skip_number (Integer, default: 1):
  Skip publishing and calling laser assembler per ~skip_number.
• ~twist_frame_id (String)
  Frame id used in ~output_velocity.
• ~use_laser_assembler (Boolean, default: False):
  Enable ~output_cloud and ~assemble_scans2.
• ~not_use_laser_assembler_service (Boolean, default: False)
  When it is true, do not use ~assemble_scans2 service but assemble scan pointcloud locally.
• ~joint_name (String, required):
  Joint name of actuator to rotate laser.
• ~laser_type (String, default: tilt_half_down):
  Type of rotating laser. You can choose one of the types:
1. **tilt**: A mode for tilting laser. In this mode, TiltLaserListener assumes the motor to be moved from minimum joint angle to maximum joint angle over again. TiltLaserListener publishes the minimum and latest time range to move tilting laser from minimum joint angle to maximum joint angle.

2. **tilt_half_down**: In this mode, TiltLaserListener publishes time range from maximum joint angle to minimum joint angle.

3. **tilt_half_up**: In this mode, TiltLaserListener publishes time range from minimum joint angle to maximum joint angle like `tilt_half_down`.

4. **infinite_spindle**: Infinite spindle laser. TiltLaserListener publishes time range to rotate laser 360 degrees.

5. **infinite_spindle_half**: Infinite spindle laser, but most of laser has over 180 degrees range of field. Therefore we don’t need to rotate laser 360 degrees to scan 3-D space, just 180 degree rotation is required. In this mode, TiltLaserListener publishes time range a time range of 180 degree rotation.

6. **periodic**: TiltLaserListener periodically publishes.

   - `~overwrap_angle` (Double, default: 0.0)
     overwrap angle offset in radians when detecting time range. Only available when `~laser_type` is `infinite_spindle` or `infinite_spindle_half`.

   - `~publish_rate` (Double, default: 1.0)
     rate of publishing assembled cloud [Hz]. Only available when `~laser_type` is `periodic`.

   - `~vital_rate` (Double, default: 1.0)
     Rate of publishing diagnostics [Hz].

### 19.111.7 Troubleshooting

- **Q**: TiltLaserListener doesn’t publish `~output_cloud` when `~laser_type` is `infinite_spindle` or `infinite_spindle_half`.

  **A**: Maybe velocity field of `~input` is not valid.

### 19.111.8 Sample

```bash
roslaunch jsk_pcl_ros sample_tilt_laser_listener.launch
```
19.112 TorusFinder

Find a torus out of pointcloud based on RANSAC with 3-D circle model.

19.112.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud. You may need to choose good candidates of pointcloud.
- `~input/polygon (geometry_msgs/PolygonStamped)`
  Input polygon. You can use this topic as well as `~input`.
  Vertices of the polygon will be used for detecting torus.
19.112.2 Publishing Topic

- `~output (jsk_recognition_msgs/Torus)`
  Output of detection.
- `~output/inliers (pcl_msgs/PointIndices)`
- `~output/coefficients (pcl_msgs/ModelCoefficients)`
  Inliers and coefficients which represents detection result.
- `~output/array (jsk_recognition_msgs/TorusArray)`
  Array of torus. It will be used for visualization.
- `~output/pose (geometry_msgs/PoseStamped)`
  Publish result of detection as geometry_msgs/PoseStamped
- `~output/with_failure (jsk_recognition_msgs/Torus)`
- `~output/with_failure/array (jsk_recognition_msgs/TorusArray)`
  Output of detection with failure information.
- `~output/latest_time (std_msgs/Float32)`
  latest computation time
- `~output/average_time (std_msgs/Float32)`
  average computation time

19.112.3 Parameters

- `~use_hint (Bool, default: False)`
  Whether to set hint axis which we need to search for a model perpendicular to.
- `~initial_axis_hint (Array of Float, default: [0, 0, 1])`
  The axis along which we need to search for a model perpendicular to.
  This parameter is enabled only when `~use_hint` is True.
- `~use_normal (Bool, default: False)`
  Whether to use normal field of input pointcloud.

Parameters below can be changed by dynamic_reconfigure.

- `~algorithm (String, default: RANSAC)`
  Type of sample consensus method to use.
  Choose one from RANSAC, LMEDS, MSAC, RRANSAC, RMSAC, MLESAC and PROSAC.
- `~min_radius (Double, default: 0.1)`
- `~max_radius (Double, default: 1.0)`
  Minimum and maximum radius of torus in meters.
- `~min_size (Integer, default: 10)`
  Minimum number of inliers.
\begin{itemize}
  \item \texttt{\textasciitilde outlier\_threshold} (Double, default: 0.01)
    \begin{quote}
      Outlier distance threshold used in RANSAC in meters.
    \end{quote}
  \item \texttt{\textasciitilde max\_iterations} (Integer, default: 100)
    \begin{quote}
      Maximum number of iterations of RANSAC.
    \end{quote}
  \item \texttt{\textasciitilde eps\_hint\_angle} (Double, default: 0.1)
    \begin{quote}
      Maximum allowed difference between the model normal and the given axis in radians.
      This parameter is used only when \texttt{\textasciitilde use\_hint} is True.
    \end{quote}
  \item \texttt{\textasciitilde voxel\_grid\_sampling} (Bool, default: False)
    \begin{quote}
      Whether to downsample \texttt{\textasciitilde input} before detection.
    \end{quote}
  \item \texttt{\textasciitilde voxel\_size} (Double, default: 0.02)
    \begin{quote}
      Leaf size of voxel grid in meters.
      This parameter is used only when \texttt{\textasciitilde voxel\_grid\_sampling} is True.
    \end{quote}
\end{itemize}

19.112.4 Sample

\texttt{roslaunch jsk_pcl_ros sample_torus\_finder\_launch}

19.113 tracker\_status\_info.py

[Image of a visualization showing Tracker Status]

Republish \texttt{jsk\_recognition\_msgs/TrackerStatus topic} as \texttt{jsk\_rviz\_plugins/OverlayText}.

This node is assumed to be used with \texttt{ParticleFilterTracking}
19.113.1 Subscribing Topics

- `~input(jsk_recognition_msgs/TrackerStatus)`
  Whether a node is tracking object or not.

19.113.2 Publishing Topics

- `~text(jsk_rviz_plugins/OverlayText)`
  Text message of `~input`.

19.113.3 Sample

```bash
gerun roslaunch jsk_pcl_ros sample_particle_filter_tracking_change_detection.launch
```

19.114 tracking_info.py

Republish two `std_msgs/Float32` topics as `jsk_rviz_plugins/OverlayText`.
This node is assumed to be used with `ParticleFilterTracking`
19.114.1 Subscribing Topics

- ~rms_angle_error(std_msgs/Float32)
- ~rms_distance_error(std_msgs/Float32)
  Root mean squared error of angle/distance.

19.114.2 Publishing Topics

- ~text(jsk_rviz_plugins/OverlayText)
  Text message of RMSEs.

19.114.3 Sample

```
roslaunch jsk_pcl_ros sample_particle_filter_tracking_change_detection.launch
```

19.115 UniformSampling

Sample pointcloud in the manner of uniform sampling.

19.115.1 Subscribing Topic

- ~input(sensor_msgs/PointCloud2)
  Input pointcloud

19.115.2 Publishing Topic

- ~output(pcl_msgs/PointIndices)
  Sampled indices

19.115.3 Parameters

- ~search_radius(Double, default: 0.01)
  Sampling radius to apply uniform sampling.
19.116 VoxelGridDownsampleDecoder

19.116.1 What Is This

Decode sliced point cloud encoded by `jsk_pcl_ros/VoxelGridDownsampleManager`, and publish it as a point cloud.

19.116.2 Subscribing Topic

- `~input (jsk_recognition_msgs/SlicedPointCloud)`
  
  Encoded sliced point cloud.

19.116.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  
  Decoded point cloud.

19.116.4 Sample

```
roslaunch jsk_pcl_ros sample_voxel_grid_downsample.launch
```
19.117 VoxelGridDownsampleManager

19.117.1 What Is This

Filter input point cloud by input marker size, and then downsample it.

19.117.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Original point cloud.
- `~add_grid (visualization_msgs/Marker)`
  Additional marker used for xyz pass through filter before downsampling.
  Only `frame_id`, `pose/position` and `scale` field are used for filtering.
  Also, leaf size [m] in voxel grid downsampling will be defined by `color/r` field.

Below marker is added internally by default.

```yaml
header:
  frame_id: ~base_frame
pose:
  position:
    x: 2.0
    y: 0.0
    z: -0.5
scale:
  x: 4.0
  y: 2.0
  z: 3.0
color:
  r: 0.05
```
19.117.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Downsampled point cloud for debugging.
- `~output_encoded (jsk_recognition_msgs/SlicedPointCloud)`
  Downsampled and sliced point cloud.
  All sliced clusters will be published in order with the `slice_index` info.
  Number of the clusters is calculated from `~max_points`.
  Messages of this topic can be decoded by `jsk_pcl_ros/VoxelGridDownsampleDecoder`.

19.117.4 Parameter

- `~base_frame (String, default: pelvis)`
  Frame ID of initial marker.
- `~max_points (Int, default: 300)`
  Number of maximum points in `~output_encoded`.
- `~rate (Float, default: 1.0)`
  Multiplicative inverse of duration between publishing `output_encoded`.
  The unit is [Hz].

19.117.5 Sample

```
roslaunch jsk_pcl_ros sample_voxel_grid_downsample.launch
```

19.118 VoxelGridLargeScale

VoxelGrid downsampler which can handle small leaf_size. Only supports `pcl::PointXYZ`.

VoxelGrid downsampler which can handle small leaf_size. Only supports `pcl::PointXYZ`.
19.118.1 Subscribing Topics

- `~input(sensor_msgs/PointCloud2)`
  Input cloud

19.118.2 Publishing Topics

- `~output(sensor_msgs/PointCloud2)`
  Output downsampled cloud

19.118.3 Parameters

- `~leaf_size(Float, default: 0.01)`
  Size of voxel grid.

19.118.4 Sample

```bash
roslaunch jsk_pcl_ros sample_voxel_grid_large_scale.launch
```
jsk_pcl_ros_utils is a package to provide some programs using pcl. This package provides some programs as nodelet.

## 20.1 AddPointIndices

Add two different `pcl_msgs/PointIndices` into one indices.

### 20.1.1 Subscribing Topic

- `~input/src1 (pcl_msgs/PointIndices)`
- `~input/src2 (pcl_msgs/PointIndices)`

  Input indices

### 20.1.2 Publishing Topic

- `~output (pcl_msgs/PointIndices)`

  Output indices

### 20.1.3 Parameters

- `approximate_sync (Boolean, default: false)`

  If this parameter is true, `~input/src1` and `~input/src2` are synchronized with approximate time policy.
20.2 BoundingBoxArrayToBoundingBox

20.2.1 What is this?

Convert `jsk_recognition_msgs/BoundingBoxArray` to `jsk_recognition_msgs/BoundingBox`.

20.2.2 Subscribing Topic

- `~input (jsk_recognition_msgs/BoundingBoxArray)`
  
  Bounding box array.
20.2.3 Publishing Topic

- ~output (jsk_recognition_msgs/BoundingBox)

20.2.4 Parameters

- ~index (Int, default: -1)

  Index value where bounding box is extracted from bounding box array. Please note that negative index is skipped.

20.3 CentroidPublisher

20.3.1 What Is This

This nodelet will subscribe the sensor_msgs::PointCloud2 or jsk_recognition_msgs/PolygonArray and calculate its centroid. This also broadcasts coordinates of cloud or each polygons as tf whose parent is cloud headers frame_id and whose child is the new centroid frame_id.
20.3.2 Subscribing Topics

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud.
- `~input/polylines (jsk_recognition_msgs/PolygonArray)`
  Input polygon.

20.3.3 Publishing Topics

- `/tf`
  Publish tf of the centroid of the input pointcloud.
- `~output/pose (geometry_msgs/PoseStamped)`
  Centroid of the pointcloud as `geometry_msgs/PoseStamped`.
- `~output/point (geometry_msgs/PointStamped)`
  Centroid of the pointcloud as `geometry_msgs/PointStamped`.
- `~output/pose_array (geometry_msgs/PoseArray)`
  Centroid poses of each polygons.

20.3.4 Parameters

- `~frame (String, default: node name)`
  A frame_id for centroid tf. For polygon array, suffix numbers are appended. (e.g. `frame00`)
- `~publish_tf (Boolean, default: False)` Set this parameter to `True` in order to publish tf frame. The invalid centroid is filtered and tf is not published in that case. Note that if this option is `True`, input topics are always subscribed.

20.3.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_centroid_publisher.launch
```
20.4 CloudOnPlane

Publishes true when a pointcloud is on a polygon.

20.4.1 Subscribing Topics

- `~input (sensor_msgs/PointCloud2)`
- `~input/polygon (jsk_recognition_msgs/PolygonArray)`
  Input pointcloud and polygons.

20.4.2 Publishing Topics

- `~output (jsk_recognition_msgs/BoolStamped)`
  True if distance between pointcloud and polygon is smaller than `~distance_thr` for `~buf_size` frames.

20.4.3 Parameters

- `~approximate_sync (Bool, default: False)`
  Whether to allow approximate synchronization for input topics.
- `~distance_thr (Float, default: 0.05)`
  Distance threshold between pointcloud and polygon.
  This parameter can be changed by `dynamic_reconfigure`. 
• `~buf_size` (Int, default: 2)
  CloudOnPlane only returns true if all the recent `~buf_size` results is true.
  This parameter can be changed by `dynamic_reconfigure`.

**20.4.4 Sample**

```bash
roslaunch jsk_pcl_ros_utils sample_cloud_on_plane.launch
```

**20.5 cloud_on_plane_info.py**

![Sample Cloud on Plane](image)

**20.5.1 What Is This**

Subscribe bool value representing whether object is on the plane or not, and publish a text.

**20.5.2 Subscribing Topic**

- `~input (jsk_recognition_msgs/BoolStamped)`
  Boolean value.
20.5.3 Publishing Topic

- ~text(jsk_rviz_plugins/OverlayText)
  
  Output text.
  
  This text can be visualized with Rviz.

20.5.4 Parameters

None.

20.5.5 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_cloud_on_plane.launch
```

20.6 ClusterPointIndicesLabelFilter

20.6.1 What Is This
This nodelet selects and republishes the ClusterPointIndices which belong to a certain label present in a LabelArray topic.

### 20.6.2 Subscribing Topics

- `~input/indices (jsk_recognition_msgs/ClusterPointIndices)`: Input indices.
- `~input/labels (jsk_recognition_msgs/LabelArray)`: Input labels.

### 20.6.3 Publishing Topics

- `~output (jsk_recognition_msgs/ClusterPointIndices)`: Filtered cluster point indices.

### 20.6.4 Parameters

- `~label_value (Int, default: 0)`: Label id to filter input cluster point indices.
- `~approximate_sync (Boolean, default: False)`: Policy of synchronization, if False it synchronizes exactly, else approximately.
- `~queue_size (Int, default: 100)`: Queue size of topic msgs for synchronization.

### 20.6.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_cluster_point_indices_label_filter.launch
```

### 20.7 ClusterPointIndicesToPointIndices

#### 20.7.1 What is this?

Convert cluster indices to point indices with specified index value.

#### 20.7.2 Subscribing Topic

- `~input (jsk_recognition_msgs/ClusterPointIndices)`: Cluster indices.
20.7.3 Publishing Topic

- `~output (pcl_msgs/PointIndices)`
  Output point indices.

20.7.4 Parameters

- `~index (Int, default: -1)`
  Index value where point indices is extracted from cluster indices. If `~index` is -1, concatenated cluster indices is published.

20.8 ColorizeDistanceFromPlane

20.8.1 What Is This

Colorize points based on distance from planes. This is usefull for calibration.
20.8.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input point cloud.

- `~input_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Input model coefficients. This topic is used only for synchronizing.

- `~input_polygons (jsk_recognition_msgs/PolygonArray)`
  Input plane.

20.8.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Output colorized point cloud.

20.8.4 Parameters

- `~min_distance (Float, default: 0.0)`
  Minimum distance of point cloud from planes.
  Minimum corresponds to blue, while maximum corresponds to red.

- `~max_distance (Float, default: 0.1)`
  These parameters can be changed by `dynamic_reconfigure`.

- `~only_projectable (Bool, default: False)`
  Draw points only if then can be projected inside of convex.
  This parameter can be changed by `dynamic_reconfigure`.

20.8.5 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_colorize_distance_from_plane.launch
```
20.9 ColorizeHeight2DMapping

Utility nodelet to visualize heightmap as pointcloud. It just set all the z of points 0 and assign z to intensity.

20.9.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  - Input pointcloud.

20.9.2 Publishing Topics

- ~output (sensor_msgs/PointCloud2)
  - Output pointcloud. z values of points are 0 and intensity of points has z value.

20.9.3 Sample

```
roslaunch jsk_pcl_ros_utils sample_colorize_height_2d_mapping.launch
```
20.10 DelayPointCloud

20.10.1 What is this?

Delay point cloud.

20.10.2 Subscribing Topics

• ~input (sensor_msgs/PointCloud2)
  Original point cloud.

20.10.3 Publishing Topics

• ~output (sensor_msgs/PointCloud2)
  Delayed point cloud.
20.10.4 Parameters

- ~delay_time (Double, default: 0.1)
  Delay time from original point cloud in [sec].
- ~queue_size (Int, int: 1000)
  Queue size of subscriber.
  It should be greater than ~delay_time * rate of ~input.

20.10.5 Sample

roslaunch jsk_pcl_ros_utils sample_delay_pointcloud.launch

20.11 DepthImageError

![Image](image.png)

Compute error of depth image and corner point of checker board.
20.11.1 Subscribing Topic

- ~image (sensor_msgs/Image)
  Input rectified and registered depth image whose encoding is 32FC1 (i.e. metric is [m]).
- ~camera_info (sensor_msgs/CameraInfo)
  Depth_registered camera info.
- ~point (geometry_msgs/PointStamped)
  Corner point of checkerboard whose frame_id is the same as rgb image.
  x and y field should represent (u, v) point in [pixels], while z field should represent depth in [m].

20.11.2 Publishing Topic

- ~output (jsk_recognition_msgs/DepthErrorResult)
  Error between depth image and ~point.

20.11.3 Parameters

- ~approximate_sync (boolean, default: false)
  Use approximate sync if it is true.

20.11.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_depth_image_error.launch
```

20.12 evaluate_box_segmentation_by_gt_box.py

20.12.1 What is this?

Evaluate three-dimensional box segmentation by a gt. (ground truth) bounding box.

20.12.2 Subscribing Topic

- ~input/box_gt (jsk_recognition_msgs/BoundingBox)
  GT. bounding box.
- ~input/box (jsk_recognition_msgs/BoundingBox)
  Input bounding box.
20.12.3 Publishing Topic

- ~output (jsk_recognition_msgs/Accuracy)
  
  Accuracy of box segmentation evaluated by gt. box.

\[
\text{accuracy} = \frac{\text{volume}^{gt} = \text{Volume}(\text{box}^{gt} \cap \text{box})}{\text{Volume}(\text{box}^{gt}) + \text{Volume}(\text{box}) - \text{volume}^{gt}}
\]

20.12.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_evaluate_box_segmentation_by_gt_box.launch
```

20.13 evaluate_voxel_segmentation_by_gt_box.py

20.13.1 What is this?

Evaluate three-dimensional voxel segmentation by a gt. (ground truth) bounding box.

20.13.2 Subscribing Topic

- ~input/box_gt (jsk_recognition_msgs/BoundingBox)
  
  GT. bounding box.
- ~input/markers (visualization_msgs/MarkerArray)
  
  Input voxel.

20.13.3 Publishing Topic

- ~output (jsk_recognition_msgs/Accuracy)
  
  Accuracy of voxel segmentation evaluated by gt. box.

\[
\text{accuracy} = \frac{\text{Volume}(<\text{voxels}_\text{tp}>)}{\text{Volume}(\text{box}^{gt}) + \text{Volume}(\text{voxels}) - \text{Volume}(<\text{voxels}_\text{tp}>)}
\]

20.13.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_evaluate_voxel_segmentation_by_gt_box.launch
```
20.14 LabelToClusterPointIndices

20.14.1 What is this?

Convert label image to cluster point indices with assumption of 0 is background label and it is published as an another topic.

20.14.2 Subscribing Topic

- `~input` (sensor_msgs/Image, encoding: 32SC1)
  Label image.

20.14.3 Publishing Topic

- `~output` (jsk_recognition_msgs/ClusterPointIndices)
  Set of point indices and each point indices means each label. label value 0 is recognized as background label, so $i + 1 = \text{label number}$. (ex. point indices in index 0 is region where label value is 1)

- `~output/bg_indices` (sensor_msgs/PointIndices)
  Point indices which means background label.
20.14.4 Parameters

- ~bg_label (Int, default: 0)
  Label value for which background point indices is published.
- ~ignore_labels (List of Int, default [])
  List of ignored labels. Indices of ignored labels are replaced by empty ones.

20.14.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_label_to_cluster_point_indices.launch
```

20.15 MarkerArrayVoxelToPointCloud

20.15.1 What is this?

A nodelet to convert marker array into point cloud.
20.15.2 Subscribing Topic

- ~input (visualization_msgs/MarkerArray)
  Input marker array.
  Position of each point will be copied from points field, while color from color field.

20.15.3 Publishing Topic

- ~output (sensor_msgs/PointCloud2)
  Output point cloud converted from ~input.

20.15.4 Parameters

None.

20.15.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_marker_array_voxel_to_pointcloud.launch
```

20.16 MaskImageToDepthConsideredMaskImage

20.16.1 What Is This

jsk_pcl/MaskImageToDepthConsideredMaskImage extracts directed area of mask image in the order of depth from sensor_msgs/PointCloud2 and sensor_msgs/Image of mask image. Example is at jsk_pcl_ros/launch/extract_only_directed_region_of_close_mask_image.launch.

20.16.2 Subscribing Topic

- ~input (sensor_msgs/PointCloud2)
  Depth information of image. Width and height of this data must be same with ~input/image.
- ~input/image (sensor_msgs/Image)
  Input mask image.
- ~input/maskregion (sensor_msgs/Image)
  Input mask region.(To use interactively, use interaction_mode:grabcut_rect of image_view2.)
20.16.3 Publishing Topic

- `~output (sensor_msg/Image)`
  
  Output mask Image. Points at close range is extracted.

20.16.4 Parameter

- `~extract_num (Int, default: 400)`
  
  Num of extract points in mask image.

- `~use_mask_region (Bool, default: True)`
  
  Whether use mask region option or not. If true, only selected region of mask image is extracted.

- `~in_the_order_of_depth (Bool, default: True)`
  
  Extracted points are in the order of the depth image if enabled.

- `~approximate_sync (Bool, default: False)`
  
  Use approximate synchronization policy instead of exact synchronization if enabled.

- `~queue_size (Int, default: 100)`
  
  Queue size of input topics

20.17 MaskImageToPointIndices

A nodelet to convert mask image (sensor_msgs/Image) to point indices (pcl_msgs/PointIndices or jsk_recognition_msgs/ClusterPointIndices) for organized pointcloud.
20.17.1 Subscribing Topic

- ~input (sensor_msgs/Image)
  Input mask image.

20.17.2 Publishing Topic

- ~output (pcl_msgs/PointIndices)
  Output indices converted from the mask image.
- ~output/all_indices (jsk_recognition_msgs/ClusterPointIndices)
  Output all indices converted from the mask image.

20.17.3 Parameters

- ~use_multi_channels (Bool, default: false)
  Handle input as multi-channeled mask image (a.k.a label image) if enabled this option.
- ~target_channel (Int, default -1)
  This option is valid only when ~use_multi_channels is enabled. Target channel of mask image which is to be published to ~output as indices. If the option is set as -1 (set by default), all channels are counted and published to ~output/all_indices instead of ~output.

20.17.4 Sample

roslaunch jsk_pcl_ros_utils sample_mask_image_to_point_indices.launch
20.18 NormalConcatenater

20.18.1 What Is This

Concatenate two input point cloud. One contains xyz and rgb field, and the other contains normal_xyz and curvature field.

20.18.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Point cloud which contains xyz and rgb field.
- `~normal (sensor_msgs/PointCloud2)`
  Point cloud which contains normal_xyz and curvature field.

20.18.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Concatenated point cloud.
20.18.4 Parameter

- ~use_async (Bool, default: False)
  Whether to enable approximate synchronization policy.
- ~max_queue_size (Int, default: 100)
  Queue size for message_filters synchronization.

20.18.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_normal_concatenater.launch
```

20.19 NormalFlipToFrame

Flip normal direction towards specified frame.

20.19.1 Subscribing Topics

- ~input (sensor_msgs/PointCloud2)
  Input pointcloud. It should have normal fields.

20.19.2 Publishing Topics

- ~output (sensor_msgs/PointCloud2)
  Output pointcloud whose normal vector orients toward specified frame by ~frame_id.
20.19.3 Parameters

- ~frame_id (String, required)
  Frame ID which is to be oriented by normal vectors.
- ~strict_tf (Bool, default: false)
  Do not take into account timestamp if this parameter is false.

20.19.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_normal_flip_to_frame.launch
```

20.20 PCDReaderWithPose

Publish cloud with given pose

20.20.1 Parameters

- ~pcd_file (Strng, required)
  file name of pcd for publish
20.20.2 Subscribing Topics

- ~input (geometry_msgs/PoseStamped)
  Pose for published pcd.

20.20.3 Publishing Topics

- ~output (sensor_msgs/PointCloud2)
  Point cloud whose position and orientation is filled from ~input.
  Also, frame ID will be the same as ~input.

20.20.4 Sample

roslaunch jsk_pcl_ros_utils sample_pcd_reader_with_pose.launch

20.21 PlanarPointCloudSimulator

Simulate a pointcloud which is acquired by cameras such as stereo camera and kinect. Sensor model is pinhole camera model.
20.21.1 Subscribing Topics

- ~input (sensor_msgs/CameraInfo)
  
  Camera info to simulate pointcloud.

20.21.2 Publishing Topics

- ~output (sensor_msgs/PointCloud2)
  
  Simulated planar pointcloud.

20.21.3 Parameters

- ~distance (Double, default: 1.0)
  
  Distance to pointcloud from origin along z-axis.

20.21.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_planar_pointcloud_simulator.launch
```

### 20.22 PlaneConcatenator

Concatenate near planes and build new set of planes.
20.22.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud.
- `~input/indices (jsk_recognition_msgs/ClusterPointIndices)`
- `~input/polygons (jsk_recognition_msgs/PolygonArray)`
- `~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Input planes.

20.22.2 Publishing Topics

- `~output/indices (jsk_recognition_msgs/ClusterPointIndices)`
- `~output/polygons (jsk_recognition_msgs/PolygonArray)`
- `~output/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Concatenated planes. Coefficients parameters are refined by RANSAC. If the coefficients are not concatenated, they will be output as is, but if they are refined by RANSAC, they will be forced to face the origin of the frame.

20.22.3 Parameters

- `~connect_angular_threshold (Double, default: 0.1)`
  Angular threshold to regard two planes as near.
- `~connect_distance_threshold (Double, default: 0.1)`
  Euclidean distance threshold to regard two planes as near.
- `~connect_perpendicular_distance_threshold (Double, default: 0.1)`
  Distance threshold to connect two planes in perpendicular direction.
- `~ransac_refinement_max_iteration (Integer, default: 100)`
  The maximum number of iteration of RANSAC refinement.
- `~ransac_refinement_outlier_threshold (Double, default: 0.1)`
  Outlier threshold of RANSAC refinement.
- `~ransac_refinement_eps_angle (Double, default: 0.1)`
  Eps angle threshold of RANSAC refinement using normal direction of the plane.
- `~min_size (default: 100)`
  Minimum inlier of concatenated polygons.
- `~min_area (default: 0.1)`
- `~max_area (default: 100.0)`
  Minimum and maximum area of concatenated polygons.
20.22.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_plane_concatenator.launch
```

20.23 PlaneReasoner

20.23.1 What Is This

Pass through horizontal or vertical planes from input planes.

20.23.2 Subscribing Topic

- `~input` (sensor_msgs/PointCloud2)
  Input point cloud.
  Only header of this topic is used for published topics.
- `~input_inliers` (jsk_recognition_msgs/ClusterPointIndices)
  Input cluster point indices of planes.
  This topic is not used for filtering, and only filtered plane indices will be published.
- `~input_polygons` (jsk_recognition_msgs/PolygonArray)
  Input plane polygons.
- `~input_coefficients` (jsk_recognition_msgs/ModelCoefficientsArray)
  Input normal coefficients of planes.
20.23.3 Publishing Topic

- `~output/vertical/inliers (jsk_recognition_msgs/ClusterPointIndices)`
  Filtered cluster point indices of vertical planes.
- `~output/vertical/polygons (jsk_recognition_msgs/PolygonArray)`
  Filtered vertical plane polygons.
- `~output/vertical/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Filtered normal coefficients of vertical planes.
- `~output/horizontal/inliers (jsk_recognition_msgs/ClusterPointIndices)`
  Filtered cluster point indices of horizontal planes.
- `~output/horizontal/polygons (jsk_recognition_msgs/PolygonArray)`
  Filtered horizontal plane polygons.
- `~output/horizontal/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Filtered normal coefficients of horizontal planes.

20.23.4 Parameter

- `~global_frame_id (String, default: odom)`
  Frame ID of filtering coordinates.
  This parameter can be changed by `dynamic_reconfigure`.
- `~horizontal_angular_threshold (Float, default: 0.1)`
  Angular difference you allow about horizontal planes in radians.
  This parameter can be changed by `dynamic_reconfigure`.
- `~vertical_angular_threshold (Float, default: 0.1)`
  Angular difference you allow about vertical planes in radians.
  This parameter can be changed by `dynamic_reconfigure`.

20.23.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_plane_reasoner.launch
```
20.24 PlaneRejector

20.24.1 What Is This

Remove plane if its normal is different from user-defined reference axis.

20.24.2 Subscribing Topic

- ~input_polygons (jsk_recognition_msgs/PolygonArray)
  Input plane polygons.
- ~input_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
  Input normal coefficients of planes.
- ~input_inliers (jsk_recognition_msgs/ClusterPointIndices)
  Input cluster point indices.
  This topic is not used for filtering, and only filtered plane indices will be published.
  Subscribed only when ~use_inliers is set to true.

20.24.3 Publishing Topic

- ~output_polygons (jsk_recognition_msgs/PolygonArray)
  Filtered plane polygons.
- ~output_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
  Filtered normal coefficients of planes.
- ~output_inliers (jsk_recognition_msgs/ClusterPointIndices)
  Filtered cluster point indices.
  Published only when ~use_inliers is set to true.
20.24.4 Parameter

- `~use_inliers` (Bool, default: False)
  Whether to subscribe `~input_inliers`.
- `~allow_flip` (Bool, default: False)
  Allow flipping before filtering.
- `~processing_frame_id` (String, required)
  Frame ID of `~reference_axis`.
- `~reference_axis` (List of Float, required)
  Axis for filtering plane.
  It must be composed of 3 float numbers.
  Planes whose normal coefficients are different from this parameter will be removed.
- `~angle` (Float, default: 0.0)
  Target angle difference between plane axis and reference axis in radians.
  This parameter can be changed by `dynamic_reconfigure`.
- `~angle_thr` (Float, default: 10.0 / 180.0 * pi)
  Threshold of allowed angular difference in radians.
  This parameter can be changed by `dynamic_reconfigure`.

20.24.5 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_plane_rejector.launch
```
20.25 PointCloudToClusterPointIndices

Just convert pointcloud to `jsk_recognition_msgs/ClusterPointIndices`. This nodelet is useful to compute bounding box of pointcloud by `jsk_pcl/ClusterPointIndicesDecomposer`.

20.25.1 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input pointcloud

20.25.2 Publishing Topic

- `~output (jsk_recognition_msgs/ClusterPointIndices)`
  Output cluster indices.

20.25.3 Parameter

- `~skip_nan (Bool, default: False)`
  Skip NaN points.
### 20.25.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_pointcloud_to_cluster_point_indices.launch
```

### 20.26 PointIndicesToClusterPointIndices

#### 20.26.1 What is this?

Convert point indices to cluster point indices.

#### 20.26.2 Subscribing Topic

- `~input` (pcl_msgs/PointIndices)
  
  Point indices.

#### 20.26.3 Publishing Topic

- `~output` (jsk_recognition_msgs/ClusterPointIndices)
  
  Cluster point indices, number of whose elements is 1 and it is the input indices.

### 20.27 PointIndicesToMaskImage

#### 20.27.1 What Is This

`jsk_pcl/PointIndicesToMaskImage` generates mask image from `pcl_msgs/PointIndices` of organized point-cloud and original `sensor_msgs/Image`.
20.27.2 Subscribing Topic

- ~input (pcl_msgs/PointIndices)
  Indices of the point cloud to mask.
- ~input/image (sensor_msgs/Image)
  In order to know width and height of the original image, jsk_pcl/PointIndicesToMaskImage requires input image. (Note If parameter ~static_image_size is True, this topic is not subscribed.)

20.27.3 Publishing Topic

- ~output (sensor_msg/Image)
  Mask image to get ~input indices from the original image.

20.27.4 Parameters

- ~approximate_sync (Bool, default: false)
  Approximately synchronize inputs if it’s true.
- ~queue_size (Int, default: 100)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- ~static_image_size (Bool, default: false)
  If this parameter is true, the topic ~input/image is not used and parameter ~height and ~width is used to generate mask image.

Optional

- ~height, ~width (Int)
  Size of mask image which will be generated.
20.28 PointCloudRelativeFromPoseStamped

Transform pointcloud relative from the specified pose stamped. It is useful for preprocessing of registration to detect initial pose.

20.28.1 Subscribing Topics

- ~input(sensor_msgs/PointCloud2)
  Original point cloud.
- ~input/pose(geometry_msgs/PoseStamped)
  Pose from which original point cloud will be transformed.

20.28.2 Publishing Topics

- ~output(sensor_msgs/PointCloud2)
  Transformed point cloud.

20.28.3 Parameters

- ~approximate_sync(Boolean, default: False)
  Whether to allow approximate synchronization of input topics.
20.28.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_pointcloud_relative_from_pose_stamped.launch
```

20.29 PointCloudToMaskImage

![pointcloud_to_mask_image](https://user-images.githubusercontent.com/1901008/34367264-aaa28e1e-eaeb-11e7-8699-c5e17dec71e8.gif)

20.29.1 What is this?

Convert point cloud to mask image with following rules:

1. NaN region of PointXYZ will be black (0).
2. Other region will be white (255).

20.29.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  
  Input pointcloud
  
  - `~input/depth (sensor_msgs/Image)`
    
    Input depth image

20.29.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  
  Output mask image.

20.29.4 Parameters

- `~z_near (Double, default: 0.0)`
  
  Lower limit of depth to be projected to mask image for each pixel (meter)

- `~z_far~ (Double, default: 10.0)`
  
  Upper limit of depth to be projected to mask image for each pixel (meter)
20.30 PointCloudToPCD

20.30.1 What is this?
Subscribe PointCloud2 topic and save in pcd file.

20.30.2 Subscribing Topic
- ~input(sensor_msgs/PointCloud2)

20.30.3 Advertising Services
- ~save_pcd(std_srvs/Empty)
  
  Convert PointCloud2 to pcd, saved under {prefix}/stamp.pcd.

20.30.4 Parameters
- prefix (String, default: Empty)
  
  prefix of PCD filenames. pcd files are saved as {prefix}/stamp.pcd.
- binary (Boolean, default False)
  
  save binary pcd files.
- compressed (Boolean, default False)
  
  save as compressed pcd files. this parameter is only effective when binary is True.
- fixed_frame (String, default: Empty)
  
  transform point cloud to fixed frame. when it is empty, point cloud is not transformed.
- duration (Double, default: 1.0)
  
  Saving duration. You can change saving frequency with this parameter. If the duration is greater than 0.0, the pcd data is stored continuously under the set duration. When you want to use this as ROS service, set the duration to 0.0.

20.31 PointCloudToPointIndices

A nodelet to convert PointCloud (sensor_msgs::PointCloud2) to pcl_msgs/PointIndices.
20.31.1 Subscribing Topic

- `~input(sensor_msgs/PointCloud2)`
  Input pointcloud.

20.31.2 Publishing Topic

- `~output(pcl_msgs/PointIndices)`
  Output indices converted from the pointcloud.

20.32 PointcloudToSTL

20.32.1 What Is This

This nodelet converts organized pointcloud to stl mesh using `pcl::OrganizedFastMesh` and generates a stl file.
20.32.2 Subscribing Topic

- `~input(sensor_msgs/PointCloud2)`
  Organized point cloud.

20.32.3 Publishing Topic

- `~pc_stl_mesh(visualization_msgs/Marker)`
  Marker of output mesh.

20.32.4 Advertising Services

- `~create_stl(jsk_recognition_msgs/SetPointCloud2)`
  Service API to create a stl file from pointcloud data.
  Returns output filename.

20.32.5 Parameters

- `~filename(String, default: /tmp/${ros::Time::now().toNSec()}_pointcloud.stl)`
  Path to STL mesh file.
- `~triangle_pixel_size(Float, default: 1.0)`
  Edge length (in pixels) used for constructing the fixed mesh.
- `~max_edge_length(Float, default: 4.5)`
  Maximum edge length.
- `~store_shadow_faces(Boolean, default: True)`
  Store shadowed faces or not.
- `~search_radius(Float, default: 0.05)`
- `~mu(Float, default: 3.5)`
- `~maximum_nearest_neighbors(Integer, default: 100)`
- `~maximum_surface_angle(Float, default: pi / 4)`
- `~minimum_angle(Float, default: pi / 18)`
- `~maximum_angle(Float, default: pi * 2 / 3)`
- `~normal_consistency(Boolean, default: False)`
  These parameters are not used now.
20.32.6 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_pointcloud_to_stl.launch
```

20.33 PointCloudXYZToXYZRGB

20.33.1 What is this?

Node to convert fields of `sensor_msgs/PointCloud2` from XYZ to XYZRGB.

20.33.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input cloud whose field is XYZ.

20.33.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Output cloud whose field is XYZRGB.

20.33.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_pointcloud_xyz_to_xyzrgb.launch
```

20.34 PointCloudXYZRGBToXYZ
20.34.1 What is this?
Node to convert fields of sensor_msgs/PointCloud2 from XYZRGB to XYZ.

20.34.2 Subscribing Topic
- `~input (sensor_msgs/PointCloud2)`
  Input cloud whose field is XYZRGB.

20.34.3 Publishing Topic
- `~output (sensor_msgs/PointCloud2)`
  Output cloud whose field is XYZ.

20.34.4 Sample
```bash
catkin_make
roslaunch jsk_pcl_ros_utils sample_pointcloud_xyzrgb_to_xyz.launch
```

20.35 PolygonAppender

20.35.1 What Is This
Concatenate two polygon arrays into one.
20.35.2 Subscribing Topic

- `~input0` ([jsk_recognition_msgs/PolygonArray](#))
  - Input polygons.
- `~input1` ([jsk_recognition_msgs/PolygonArray](#))
  - Input polygons.
- `~input_coefficients0` ([jsk_recognition_msgs/ModelCoefficientsArray](#))
  - Input normal coefficients of polygons.
- `~input_coefficients1` ([jsk_recognition_msgs/ModelCoefficientsArray](#))
  - Input normal coefficients of polygons.

20.35.3 Publishing Topic

- `~output` ([jsk_recognition_msgs/PolygonArray](#))
  - Concatenated polygons.
- `~output_coefficients` ([jsk_recognition_msgs/ModelCoefficientsArray](#))
  - Concatenated normal coefficients of polygons.

20.35.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_polygon_appender.launch
```

20.36 PolygonArrayAngleLikelihood

Compute likelihood based on angular distance. The nearer polygon is, the larger likelihood is.
The likelihood is determined by \( 1/(1+d^2) \) where \( d \) is an angular difference from \(~\text{target}_\text{frame}_\text{id}~\) to the polygon.

### 20.36.1 Subscribing Topic

- \(~\text{input} (\text{jsk\_recognition\_msgs/PolygonArray})~\)
  
  Input polygon array.

### 20.36.2 Publishing Topic

- \(~\text{output} (\text{jsk\_recognition\_msgs/PolygonArray})~\)
  
  Output polygon array.

### 20.36.3 Parameters

- \(~\text{target}_\text{frame}_\text{id}~(\text{String, required})~\)
  
  Frame id to compute polygon’s distance from.
- \(~\text{tf\_queue\_size}~(\text{Int, Default: 10})~\)
  
  Queue size of tf message filter
- \(~\text{axis}~(\text{List of float, Default: [1, 0, 0]})~\)
  
  Reference direction in \(~\text{target}_\text{frame}_\text{id}\) coordinates system.

### 20.36.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_polygon_array_angle_likelihood.launch
```
20.37 PolygonArrayAreaLikelihood

Compute likelihood based on area. The nearer polygon is, the larger likelihood is.

The likelihood is determined by \( \frac{1}{1 + d^2} \) where \( d \) is difference between area and expected area.

**20.37.1 Subscribing Topic**

- `~input (jsk_recognition_msgs/PolygonArray)`
  
  Input polygon array.

**20.37.2 Publishing Topic**

- `~output (jsk_recognition_msgs/PolygonArray)`
  
  Output polygon array.

**20.37.3 Parameters**

- `~area (Float, default: 1.0)`
  
  Expected size of area of polygon.
20.37.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_polygon_array_area_likelihood.launch
```

20.38 PolygonArrayDistanceLikelihood

Compute likelihood based on distance. The nearer polygon is, the larger likelihood is.
The likelihood is determined by $\frac{1}{1+d^2}$ where $d$ is a distance from ~target_frame_id to the polygon.

20.38.1 Subscribing Topic

- ~input (jsk_recognition_msgs/PolygonArray)
  
  Input polygon array.

20.38.2 Publishing Topic

- ~output (jsk_recognition_msgs/PolygonArray)
  
  Output polygon array.
## 20.38.3 Parameters

- `~target_frame_id` *(String, required)*
  Frame id to compute polygon's distance from.
- `~tf_queue_size` *(Int, default: 10)*
  Queue size of tf message filter

## 20.38.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_polygon_array_distance_likelihood.launch
```

## 20.39 PolygonArrayFootAngleLikelihood

Compute likelihood based on angular distance of foot from specified frame_id. The nearer polygon is, the larger likelihood is.

The likelihood is determined by $\frac{1}{1+d^2}$ where $d$ is a angular difference from `~target_frame_id` to the polygon.
20.39.1 Subscribing Topic

- `~input (jsk_recognition_msgs/PolygonArray)`
  Input polygon array.

20.39.2 Publishing Topic

- `~output (jsk_recognition_msgs/PolygonArray)`
  Output polygon array.

20.39.3 Parameters

- `~target_frame_id (String, required)`
  Frame id to compute polygon’s distance from
- `~tf_queue_size (Int, default: 10)`
  Queue size of tf message filter
- `~axis (List of float, Default: [1, 0, 0])`
  Reference direction in `~target_frame_id` coordinates system.

20.39.4 Sample

```bash
catkin_make
source install/setup.sh
rosrun jsk_pcl_ros_utils sample_polygon_array_foot_angle_likelihood.launch
```
20.40 PolygonArrayLikelihoodFilter

Filter `jsk_recognition_msgs/PolygonArray` by likelihood.

20.40.1 Subscribing Topics

- `~input_polygons(jsk_recognition_msgs/PolygonArray)`
  Input polygon array.

- `~input_coefficients(jsk_recognition_msgs/ModelCoefficientsArray)`
  Input coefficients array. (Enabled if `use_coefficients` is true. Expected the same order with input polygons)
20.40.2 Publishing Topics

- `~output_polygons (jsk_recognition_msgs/PolygonArray)`
  Filtered polygon array. (Polygons are sorted by their likelihood.)
- `~output_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Filtered coefficients array. (Published only if `use_coefficients` is true.)

20.40.3 Parameters

- `~use_coefficients (Bool, default: true)`
  If true, polygons and coefficients are subscribed and published synchronously.
- `~threshold (Double, default: 0.5)`
  Threshold for filtering polygons. See also description of `~negative` below for more detail.
- `~negative (Bool, default: false)`
  If false, published polygons whose likelihood is higher than `~threshold`, lower otherwise.

20.41 PolygonArrayTransformer

Change frame_id of `jsk_recognition_msgs/PolygonArray` according to tf.
20.41.1 Subscribe Topics

- `~input_polygons (jsk_recognition_msgs/PolygonArray)`
- `~input_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Input polygon array.

20.41.2 Publishing Topics

- `~output_polygons (jsk_recognition_msgs/PolygonArray)`
- `~output_coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Output polygon array.

20.41.3 Parameter

- `~frame_id (String, required)`
  Frame-id to transform to.

20.41.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_polygon_array_transformer.launch
```

20.42 PolygonArrayUnwrapper

20.42.1 What is this?

Convert `jsk_recognition_msgs/PolygonArray` to `geometry_msgs/PolygonStamped`, and `jsk_recognition_msgs/ModelCoefficientsArray` to `pcl_msgs/ModelCoefficients`.
20.42.2 Subscribing Topics

- `~input_polygons(jsk_recognition_msgs/PolygonArray)`
  Input polygon array.
- `~input_coefficients(jsk_recognition_msgs/ModelCoefficientsArray)`
  Input coefficients array. (expected the same order with input polygons)

20.42.3 Publishing Topics

- `~output_polygon(geometry_msgs/PolygonStamped)`
  Target polygon. (If `~use_likelihood` is true, the plane which has the highest likelihood is selected, otherwise the first plane in array is selected.)
- `~output_coefficients(pcl_msgs/ModelCoefficients)`
  Output coefficients.

20.42.4 Parameters

- `~use_likelihood(Boolean, default: false)`
  If true, likelihood is used to select plane to be unwrapped.
- `~plane_index(Int, default: 0)`
  If `~use_likelihood` is false, this index is used to select output plane.

20.42.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_polygon_array_unwrapper.launch
```

20.43 PolygonArrayWrapper
20.43.1 What is this?

Wrap `geometry_msgs/PolygonStamped` and `pcl_msgs/ModelCoefficients` into `jsk_recognition_msgs/PolygonArray` and `jsk_recognition_msgs/ModelCoefficientsArray`.

20.43.2 Subscribing Topics

- `~input_polygon(geometry_msgs/PolygonStamped)`
  Input polygon.
- `~input_coefficients(pcl_msgs/ModelCoefficients)`
  Input coefficients.

20.43.3 Publishing Topics

- `~output_polygons(jsk_recognition_msgs/PolygonArray)`
  Output polygon array.
- `~output_coefficients(jsk_recognition_msgs/ModelCoefficientsArray)`
  Output coefficients array.

20.43.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_polygon_array_wrapper.launch
```

20.44 PolygonFlipper

Flip `jsk_recognition_msgs/PolygonArray` to specified `sensor_frame`. 

20.44. PolygonFlipper
20.44.1 Subscribing Topic

- ~input/polys (jsk_recognition_msgs/PolygonArray)
- ~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
- ~input/indices (jsk_recognition_msgs/ClusterPointIndices)

Input polygons. If ~use_indices is disabled, ~input/indices is not used.

20.44.2 Publishing Topic

- ~output/polys (jsk_recognition_msgs/PolygonArray)
- ~output/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)
- ~output/indices (jsk_recognition_msgs/ClusterPointIndices)

Output flipped polygons which look at the origin of sensor_frame. If ~use_indices is disabled, ~output/indices is not published.

20.44.3 Parameter

- ~sensor_frame (String, Required):
  The frame_id of sensor for polygons to look at.
- ~queue_size (Int, default: 100):
  Queue size of subscribed messages for message synchronization.
- ~use_indices (Bool, default: true):
  Use indices if this parameter is enabled.

20.44.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_polygon_flipper.launch
```
20.45 PolygonMagnifier

Magnify polygons by specified length.
20.45.1 Subscribing Topic

• \texttt{~input (jsk\_recognition\_msgs/PolygonArray)}
  
  Input polygons

20.45.2 Publishing Topic

• \texttt{~output (jsk\_recognition\_msgs/PolygonArray)}
  
  Output magnified polygons

20.45.3 Parameters

• \texttt{~use\_scale\_factor (Bool, default: false)}
  
  If this is set \texttt{true}, use scale factor to magnify, otherwise use distance.

• \texttt{~magnify\_distance (Double, default: 0.2)}
  
  Length to scale polygons. Default value 0.2 means the distance of each corresponding edges will be 0.2 m. If this value is less than 0, output polygons are shrunked.

• \texttt{~magnify\_scale\_factor (Double, default: 0.8)}
  
  Factor to scale polygons. Default value 0.8 means the volume of magnified polygons are 80\% of the original polygons. This parameter is valid only when \texttt{~use\_scale\_factor} is set \texttt{true}.

20.45.4 Sample

\begin{verbatim}
roslaunch jsk_pcl_ros_utils sample_polygon_magnifier.launch
\end{verbatim}
20.46 PolygonPointsSampler

Sampling points with fixed grid size on polygons.

20.46.1 Subscribing Topic

- ~input/polys (jsk_recognition_msgs/PolygonArray)
- ~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)

Input polygons where be sampled

20.46.2 Publishing Topic

- ~output (sensor_msgs/PointCloud2)
  Sampled pointcloud (pcl::PointXYZRGBNormal).
- ~output_xyz (sensor_msgs/PointCloud2)
  Sampled pointcloud (pcl::PointXYZ).
20.46.3 Parameters

• \(\sim \text{grid\_size}\) (Double, default: 0.01)
  Sampling grid size in \([\text{m}]\).
  This parameter can be changed by \texttt{dynamic\_reconfigure}.

20.46.4 Sample

\texttt{roslaunch jsk\_pcl\_ros\_utils sample\_polygon\_points\_sampler.launch}

20.47 PoseWithCovarianceStampedToGaussianPointCloud

Visualize \texttt{geometry\_msgs/PoseWithCovarianceStamped} as gaussian pointcloud. Pointcloud is computed within a region of 3 sigma.

20.47.1 Subscribing Topics

• \(\sim \text{input}\) (\texttt{geometry\_msgs/PoseWithCovarianceStamped})
  Input pose

20.47.2 Publishing Topics

• \(\sim \text{output}\) (\texttt{sensor\_msgs/PointCloud2})
  Output pointcloud

20.47.3 Parameters

• \(\sim \text{cut\_plane}\) (String, default: \texttt{xy})
  You can choose a plane to compute gaussian distribution from \texttt{xy}, \texttt{yz}, \texttt{zx}, \texttt{flipped\_xy}, \texttt{flipped\_yz} or \texttt{flipped\_zx}.

• \(\sim \text{sampling\_num}\) (Int, default: 100)
  The number of sampling for each axis. The number of points will square of \(\sim \text{sampling\_num}\).

• \(\sim \text{normalize\_method}\) (String, default: \texttt{normalize\_area})
  \(\sim \text{normalize\_value}\) (Float, default: 1.0)
  You can choose \texttt{normalize\_area} or \texttt{normalize\_height} as a method to normalize gaussian distribution.
  If you choose \texttt{normalize\_area}, area of gaussian distribution will be \(\sim \text{normalize\_value}\).
  If you choose \texttt{normalize\_height}, the maximum height of gaussian distribution will be \(\sim \text{normalize\_value}\).
20.47.4 Sample

```bash
toolchain::roslaunch jsk_pcl_ros_utils sample_pose_with_covariance_stamped_to_gaussian_pointcloud.launch
```

20.48 SphericalPointCloudSimulator

Simulate a pointcloud which is acquired by 3D laser range finder such as tilt laser in PR2.

20.48.1 Subscribing Topics

- `~input (sensor_msgs/PointCloud2)`
  
  Trigger message to simulate pointcloud.
  
  Only timestamp of this message will be used.

20.48.2 Publishing Topics

- `~output (sensor_msgs/PointCloud2)`
  
  Simulated spherical pointcloud.
20.48.3 Parameters

- `~rate` (Float, default: `None`)
  If this parameter is specified, this node will publish `~output` at this rate [Hz], in addition to callback of `~input`.
- `~frame_id` (String, default: `~input.header.frame_id`)
  Frame ID of `~output`.
- `~r` (Float, default: `3.0`)
  Radius of spherical point cloud in [m].
  This parameter can be changed by `dynamic_reconfigure`.
- `~min_phi` (Float, default: `0.0`)
- `~max_phi` (Float, default: `3.14`)
  Minimum/maximum rotation angle of scanning in [rad].
  These parameters can be changed by `dynamic_reconfigure`.
- `~scan_range` (Float, default: `4.71`)
  Range of each scan in [rad].
  This parameter can be changed by `dynamic_reconfigure`.
- `~scan_num` (Int, default: `1081`)
  Number of distance observation in each scan.
  This parameter can be changed by `dynamic_reconfigure`.
- `~fps` (Float, default: `40.0`)
  Rate of observing each scan in [Hz].
  This parameter can be changed by `dynamic_reconfigure`.

20.48.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_spherical_pointcloud_simulator.launch
```
20.49 StaticPolygonArrayPublisher

20.49.1 What Is This

Publish static polygons.

20.49.2 Subscribing Topic

- `~input(sensor_msgs/PointCloud2)`
  Point cloud topic whose header is used for timestamp of output topics.
- `~trigger(jsk_recognition_msgs/Int32Stamped)`
  Trigger topic for publishing outputs.

20.49.3 Publishing Topic

- `~output_polygons(jsk_recognition_msgs/PolygonArray)`
  Polygon array.
- `~output_coefficients(jsk_recognition_msgs/ModelCoefficientsArray)`
  Normal array of polygons.
20.49.4 Parameter

- ~use_periodic (Bool, default: False)
  Enable timer publishing or not.
- ~use_message (Bool, default: False)
  Enable publishing when ~input is subscribed.
- ~use_trigger (Bool, default: False)
  Enable publishing when ~input and ~trigger are synchronously subscribed.

Note that at least one of these 3 parameters must be set to true.

- ~periodic_rate (Float, default: 10.0)
  Publishing rate [Hz] used when ~use_periodic is true.
- ~frame_ids (List of String, required)
  Frame ID of each polygon.
- ~polygon_array (List of List of List of Float, required)
  Polygon array.

Each polygon is composed of multiple (>=3) vertices, each of which is 3D point.

Example of ~polygon_array is below.

```
polygon_array:
- [[3, 0, 0], [0, 3, 0], [0, 1, 2], [1, 0, 2]]
- [[-1, -1, 1], [4, 3, 0], [3, 4, 0]]
```

20.49.5 Sample

```
roslaunch jsk_pcl_ros_utils sample_static_polygon_array_publisher.launch
```

20.50 SubtractPointIndices

20.50.1 What Is This

subtract one indices `pcl_msgs/PointIndices` from another indices.
20.50.2 Subscribing Topic

- ~input/src1 (pcl_msgs/PointIndices)
- ~input/src2 (pcl_msgs/PointIndices)

Input indices

20.50.3 Publishing Topic

- ~output (pcl_msgs/PointIndices)

Output indices

20.50.4 Parameters

- approximate_sync (Boolean, default: false)

If this parameter is true, ~input/src1 and ~input/src2 are synchronized with approximate time policy.

20.51 TfTransformBoundingBox

This nodelet will republish bounding box which is transformed with the designated frame_id.
20.51.1 Subscribing Topics

- ~input (jsk_recognition_msgs/BoundingBox)

  input bounding box.

20.51.2 Publishing Topics

- ~output (jsk_recognition_msgs/BoundingBox)

  output bounding box.

20.51.3 Parameters

- ~target_frame_id (string, required)

  The frame_id to transform bounding box.

- ~use_latest_tf (Bool, default: false)

  If this parameter is true, ignore timestamp of tf to transform bounding box.

- ~tf_queue_size (Int, default: 10)

  Queue size of tf message filter to synchronize tf and ~input topic.

20.51.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_tf_transform_bounding_box.launch
```
This nodelet will republish bounding box array which is transformed with the designated frame_id.

### 20.52.1 Subscribing Topics

- `~input (jsk_recognition_msgs/BoundingBoxArray)`
  
  input bounding box array.

### 20.52.2 Publishing Topics

- `~output (jsk_recognition_msgs/BoundingBoxArray)`
  
  output bounding box array.

### 20.52.3 Parameters

- `~target_frame_id (string, required)`
  
  The frame_id to transform bounding box array.

- `~use_latest_tf (Bool, default: false)`
  
  If this parameter is true, ignore timestamp of tf to transform bounding box array.

- `~tf_queue_size (Int, default: 10)`
  
  Queue size of tf message filter to synchronize tf and `~input` topic.
20.52.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_tf_transform_bounding_box_array.launch
```

20.53 TfTransformCloud

20.53.1 What Is This

This nodelet will republish the pointcloud which is transformed with the designated frame_id.

20.53.2 Topics

- **Input**
  - `~input(sensor_msgs/PointCloud2): input pointcloud`
- **Output**
  - `~output(sensor_msgs/PointCloud2): output pointcloud`. 
20.53.3 Parameters

- `~target_frame_id` (string, required)
  The frame_id to transform pointcloud.
- `~duration` (Double, default: 1.0)
  Second to wait for transformation
- `~use_latest_tf` (Bool, default: false)
  If this parameter is true, ignore timestamp of tf to transform pointcloud.
- `~tf_queue_size` (Int, default: 10)
  Queue size of tf message filter to synchronize tf and ~input topic.

20.53.4 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_tf_transform_cloud.launch
```

20.54 TransformPointcloudInBoundingBox

20.54.1 What Is This

Move input point cloud away from input bounding box.
20.54.2 Subscribing Topic

- `~input (sensor_msgs/PointCloud2)`
  Input point cloud to move.
- `~input_box (jsk_recognition_msgs/BoundingBox)`
  Input bounding box from which `~input` is moved away.

20.54.3 Publishing Topic

- `~output (sensor_msgs/PointCloud2)`
  Moved point cloud.
- `~output_offset (geometry_msgs/PoseStamped)`
  This topic is advertised, but currently not published.

20.54.4 Sample

```
roslaunch jsk_pcl_ros_utils sample_transform_pointcloud_in_bounding_box.launch
```

20.55 `xyz_to_screenpoint.py`

20.55.1 What Is This

Convert (x, y, z) 3-D coordinate to (u, v) coordinate on a image using camerainfo of sensor.

20.55.2 Subscribing Topic

- `~input (geometry_msgs/PointStamped)`
  Input point to represent (x, y, z) 3-D coordinate.
- `~input/camera_info (sensor_msgs/CameraInfo)`
  CameraInfo of sensor.

20.55.3 Publishing Topic

- `~output (geometry_msgs/PointStamped)`
  Output point to represent (u, v) image coordinate. Only x and y fileds are used. The header frame_id uses the information of `~input/camera_info`.
20.55.4 Parameters

None.

20.55.5 Sample

```bash
roslaunch jsk_pcl_ros_utils sample_xyz_to_screenpoint.launch
```
JSK_RECOGNITION_MSGS

This is a ROS message and service definition package for recognizing real world.

21.1 object_array_publisher.py

21.1.1 What is this?

Get mesh as rosparam and publish them as jsk_recognition_msgs/ObjectArray.
21.1.2 Publishing Topic

- `~output (jsk_recognition_msgs/ObjectArray)`
  Output meshes.

21.1.3 Parameters

- `~objects (List, required)`
  List of mesh.
- `~latch (Bool, default: False)`
  If true, oneshot publish is enabled and timestamp is fixed.

Example parameters are below.

```xml
<rosparam>
latch: true
objects:
  - id: 1
    name: avery_binder
    image_resources:
      - package://jsk_arc2017_common/data/objects/avery_binder/top.jpg
    mesh_resource: package://jsk_arc2017_common/data/objects/avery_binder/mesh/mesh.obj
  - id: 35
    name: tennis_ball_container
    class_id: 2
    class_name: cylinder
    image_resources:
      - package://jsk_arc2017_common/data/objects/tennis_ball_container/top.jpg
    mesh_resource: package://jsk_arc2017_common/data/objects/tennis_ball_container/mesh/mesh.obj
  - id: 35
    name: tennis_ball_container
    class_id: 2
    class_name: cylinder
    image_resources:
      - package://jsk_arc2017_common/data/objects/tennis_ball_container/top.jpg
    mesh_resource: package://jsk_arc2017_common/data/objects/tennis_ball_container/mesh/mesh.obj
</rosparam>
```

21.1.4 Sample

```bash
roslaunch jsk_recognition_msgs sample_object_array_publisher.launch JSK_INTERACTIVE_MARKER_INSTALLED:=true  # default: false
```
21.2 people_pose_array_to_pose_array.py

21.2.1 What is this?

Subscribe `jsk_recognition_msgs/PeoplePoseArray` and publish them as `geometry_msgs/PoseArray`.

21.2.2 Subscribing Topic

• `~input(jsk_recognition_msgs/PeoplePoseArray)`
  Input human poses.

21.2.3 Publishing Topic

• `~output(geometry_msgs/PoseArray)`
  Output poses.
21.2.4 Parameters

None.

21.2.5 Sample

```bash
catkin_ws/src/jsk_recognition/jsk_recognition_msgs/samples
```

21.3 plot_data_to_csv.py

21.3.1 What is this?

Subscribe `jsk_recognition_msgs/PlotData` and write data to file.

21.3.2 Subscribing Topic

- `~input (jsk_recognition_msgs/PlotData)`
  
  Input plot data.

21.3.3 Publishing Topic

None.

21.3.4 Parameters

- `~filename (String, default: output_%04d.csv)`

  Path to output file.
  
  Integer place holder like `%04d` can be used to separate file. If the path does not contain place holder, this node will overwrite the file on every callback.

21.3.5 Sample

```bash
catkin_ws/src/jsk_recognition/jsk_recognition_msgs/samples
```

21.4 save_mesh_server.py

21.4.1 What is this?

Subscribe `jsk_recognition_msgs/BoundingBox` and send request to mesh saving server. This node is a service server called from users, and at the same time a service client.

See also: `jsk_pcl/Kinfu`
21.4.2 Subscribing Topic

- ~input/bbox (jsk_recognition_msgs/BoundingBox)
  Bounding box used for request.

21.4.3 Publishing Topic

None.

21.4.4 Advertising Service

- ~request (std_srvs/Empty)
  Trigger to send request by users.

21.4.5 Calling Service

- ~save_mesh (jsk_recognition_msgs/SaveMesh)
  Service call from this node as a client.

21.4.6 Parameters

- ~ground_frame_id (String, default: "")
  Frame ID used for calling ~save_mesh.
jsk_recognition_utils a C++ library for sensor model, geometrical modeling and perception.

• Code API
  • *Nodes and Nodelets*

**22.1 add_bounding_box_array.py**

**22.1.1 What is this?**

Add multiple topics of *jsk_recognition_msgs/BoundingBoxArray* and republish them as a topic.

**22.1.2 Subscribing Topic**

See `~topics` in *Parameters*.

**22.1.3 Publishing Topic**

```
• `~output (jsk_recognition_msg/BoundingBoxArray)`
  Added bounding box array.
```

**22.1.4 Parameters**

**Required**

```
• `~topics (Array of String, required)`
  Topics to be added.
```

**Optional**

```
• `~approximate_sync (Bool, default: False)`
  Whether to use approximate for input topics.
• `~queue_size (Int, default: 10)`
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
```
• `~slop` (Float, default: 0.1)
  How many seconds you allow about the difference of timestamp when you specify `~approximate_sync`.

### 22.1.5 Usage

```bash
roslaunch jsk_recognition_utils sample_add_bounding_box_array.launch
```

### 22.2 add_cluster_indices.py

#### 22.2.1 What is this?

Add multiple topics of `jsk_recognition_msgs/ClusterPointIndices` and republish them as a topic.

#### 22.2.2 Subscribing Topic

See `~topics` in Parameters.

#### 22.2.3 Publishing Topic

- `~output (jsk_recognition_msgs/ClusterPointIndices)`
  Added cluster of point indices.

#### 22.2.4 Parameters

**Required**

- `~topics` (Array of String, required)
  Topics to be added.

**Optional**

- `~approximate_sync` (Bool, default: False)
  Whether to use approximate for input topics.
- `~queue_size` (Int, default: 10)
  How many messages you allow about the subscriber to keep in the queue. This should be big when there is much difference about delay between two topics.
- `~slop` (Float, default: 0.1)
  How many seconds you allow about the difference of timestamp when you specify `~approximate_sync`.  

22.2.5 Usage

```
rosrun jsk_recognition_utils add_cluster_indices.py _topics:='[/node_a/cluster_indices, /node_b/cluster_indices]'
```

22.3 bounding_box_array_publisher.py

22.3.1 What is this?
Add multiple topics of `jsk_recognition_msgs/ClusterPointIndices` and republish them as a topic.

22.3.2 Publishing Topic

- `~output (jsk_recognition_msgs/BoundingBoxArray)`
  Bounding boxes in the specified frame_id.

22.3.3 Parameters

Required

- `~frame_id (String, required)`
  Frame id of bounding boxes.
- `~boxes (Yaml, required)`
  Pose and dimension of bounding boxes. It is something like below:

```
<rosp param>
  boxes:
    - position: [-0.22, 0.280, 0.361] # required
      rotation: [0, 0, 1.57] # optional
      dimension: [0.37, 0.248, 0.218] # required
      label: 0 # optional
    - position: [-0.22, 0, 0.361]
      dimension: [0.37, 0.306, 0.218]
</rosp param>
```

Optional

- `~rate (Int, default: 1)`
  How many messages are published in a second.
22.3.4 Sample

```
roslaunch jsk_recognition_utils sample_bounding_box_array_publisher.launch
```

22.4 image_16uc1_to_32fc1.py

22.4.1 What is this?

Convert encoding of depth image: uint16 [mm] -> float32 [m].
It is recommended to use `depth_image_proc/convert_metric`.

22.4.2 Subscribing Topic

- `~input (sensor_msgs/Image)`
  
  Input depth image whose encoding is uint16.
22.4.3 Publishing Topic

- `~output (sensor_msgs/Image)`
  Output depth image whose encoding will be float32.

22.4.4 Sample

```
roslaunch jsk_recognition_utils sample_image_16uc1_to_32fc1.launch
```

22.5 polygon_array_publisher.py

![Polygon Array Publisher](image)

22.5.1 What is this?

Get polygons as rosparam and publish them as `jsk_recognition_msgs/PolygonArray`.

22.5.2 Publishing Topic

- `~output (jsk_recognition_msgs/PolygonArray)`
  Output polygons.
- `~output/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Output coefficients. This topic will be published if `~publish_coeffs` is True and each polygon is in planar.
22.5.3 Parameters

- **~polygons** (List, required)
  List of polygon.
  Each polygon must have `points` field, which is a list of vertices.
  You can also fill `label` and `likelihood` field for each polygon.
- **~frame_id** (String, required)
  Frame ID of polygons.
- **~publish_rate** (Float, default: 1.0)
  Publish rate [Hz].
- **~publish_coeffs** (Bool, default: False)
  If this value is true and each polygon is in planar, publish `jsk_recognition_msgs/ModelCoefficientsArray`.

Example parameters are below.

```xml
<rosparam>
  frame_id: base_link
  publish_coeffs: true
  polygons:
  - points:
    - [1.0, -1.0, 0.0]
    - [2.0, -1.0, 0.0]
    - [2.0, 1.0, 0.0]
    - [1.0, 1.0, 0.0]
    label: 1
    likelihood: 0.8
  - points:
    - [1.0, -1.0, 0.5]
    - [2.0, -1.0, 0.5]
    - [2.0, 1.0, 0.5]
    - [1.0, 1.0, 0.5]
    label: 2
    likelihood: 0.6
</rosparam>
```

22.5.4 Sample

```bash
roslaunch jsk_recognition_utils sample_polygon_array_publisher.launch
```
22.6 polygon_array_to_box_array.py

22.6.1 What is this?

Convert `jsk_recognition_msgs/PolygonArray` to `jsk_recognition_msgs/BoundingBoxArray` containing the polygons. For example, this can be used when users use polygon for collision avoidance programs.

22.6.2 Subscribing Topic

- `~input/polygons (jsk_recognition_msgs/PolygonArray)`
  Input polygon array.
- `~input/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Input coefficients array. We assume coefficients for plane detection.
22.6.3 Publishing Topic

- `~output/boxes (jsk_recognition_msgs/BoundingBoxArray)`
  Output boxes. Each box contains each input polygon. The long side of the bounding box is the x-axis, and the normal direction is the z-axis.
- `~output/polygons (jsk_recognition_msgs/PolygonArray)`
  Output polygons.
- `~output/coefficients (jsk_recognition_msgs/ModelCoefficientsArray)`
  Output coeffecients.

22.6.4 Parameters

- `~thickness (Float, default: 0.0001)`
  Thickness of bounding box.

22.6.5 Sample

```
roslaunch jsk_recognition_utils sample_polygon_array_to_box_array.launch
```

22.7 polygon_array_to_polygon.py

22.7.1 What is this?

Convert `jsk_recognition_msgs/PolygonArray` to `geometry_msgs/PolygonStamped`. 
22.7.2 Subscribing Topic

- `~input(jsk_recognition_msgs/PolygonArray)`
  Input polygon array.

22.7.3 Publishing Topic

- `~output(geometry_msgs/PolygonStamped)`

22.7.4 Parameters

- `~index(Int, default: -1)`
  Index value where polygon is extracted from polygon array. Please note that negative index is skipped.

22.7.5 Sample

```bash
roslaunch jsk_recognition_utils sample_polygon_array_to_polygon.launch
```

22.8 pose_array_to_pose.py

22.8.1 What is this?

Convert `geometry_msgs/PoseArray` to `geometry_msgs/PoseStamped`.
22.8.2 Subscribing Topic

- ~input (geometry_msgs/PoseArray)
  Input pose array.

22.8.3 Publishing Topic

- ~output (geometry_msgs/PoseStamped)

22.8.4 Parameters

- ~index (Int, default: -1)
  Index value where pose is extracted from pose array. Please note that negative index is skipped.

22.8.5 Sample

roslaunch jsk_recognition_utils sample_pose_array_to_pose.launch

22.9 rect_array_to_cluster_point_indices.py

22.9.1 What is this?

Convert jsk_recognition_msgs/RectArray to jsk_recognition_msgs/ClusterPointIndices

22.9.2 Subscribing Topic

- ~input (jsk_recognition_msgs/RectArray)
  Input rect array.
- ~input/info (sensor_msgs/CameraInfo)
  Input camera info. Subscribe only when use_info is true.

22.9.3 Publishing Topic

- ~output (jsk_recognition_msgs/ClusterPointIndices)
  Output cluster point indices.
22.9.4 Parameters

- ~use_info (Bool, Default: false)
  Subscribe camera_info topic or not
- ~img_width: (Int, required)
  Image width. Required only when use_info is false.
- ~img_height: (Int, required)
  Image height. Required only when use_info is false.
- ~queue_size (Int, Default: 10)
  Queue size for message_filters. Used only when use_info is true.
- ~approximate_sync (Bool, Default: false)
  Use approximate_sync or not. Used only when use_info is true.
- ~slop (Float, Default: 0.1)
  Slop size for message_filters. Used only when use_info is true.

22.9.5 Sample

```bash
roslaunch jsk_recognition_utils sample_rect_array_to_cluster_point_indices.launch
```

22.9.6 Sample with SSD object detector

SSD object detector with GPU or CPU
SSD object detector with TPU

```bash
# CPU
roslaunch jsk_recognition_utils sample_ssd_to_bounding_box_73b2_kitchen.launch gpu:=-1
# GPU
roslaunch jsk_recognition_utils sample_ssd_to_bounding_box_73b2_kitchen.launch gpu:=0

roslaunch jsk_recognition_utils sample_ssd_to_bounding_box.launch gpu:=-1
# coral ssd launch
source ~/coral_ws/devel/setup.bash
roslaunch jsk_recognition_utils sample_ssd_coral.launch
```
22.10 rect_array_to_polygon_array.py

22.10.1 What is this?

Convert `jsk_recognition_msgs/RectArray` to `jsk_recognition_msgs/PolygonArray`.

22.10.2 Subscribing Topic

- `~input (jsk_recognition_msgs/RectArray)`
  
  Input rect array.

22.10.3 Publishing Topic

- `~output (jsk_recognition_msgs/PolygonArray)`
  
  Output polygon array.
22.10.4 Sample

```
roslaunch jsk_recognition_utils sample_rect_array_to_polygon_array.launch
```

22.11 static_virtual_camera.py

22.11.1 What is this?

Publish static image and camera_info topic.
You can use more convenience node `jsk_perception/image_publisher.py`

22.11.2 Publishing Topic

- `~image_color(sensor_msgs/Image)`
  Image topic.
- `~camera_info(sensor_msgs/CameraInfo)`
  Camera info topic.
22.11.3 Sample

```bash
roslaunch jsk_recognition_utils sample_static_virtual_camera.launch
```
RESIZED_IMAGE_TRANSPORT

ROS nodes for resizing images.

23.1 image_resizer

Publish resized image and camera_info.

23.1.1 Subscribing Topic

- ~input/image(sensor_msgs/Image)
  Input image.
- ~input/camera_info(sensor_msgs/CameraInfo)
  Input camera info.

Note: The subscribing topic name is changed when you remap the one of input image, because it uses image_transport.

23.1.2 Publishing Topic

- ~output/image(sensor_msgs/Image)
  Resized image.
- ~output/camera_info(sensor_msgs/CameraInfo)
  Resized camera info.
23.1.3 Parameters

- ~resize_scale_x, ~resize_scale_y (Double, default: 0.25)
  Resizing scale.
- ~use_messages (Bool, default: true)
  If true, topic publishing rate will be limited, and it causes some problems on handling rostime: for example
  rosbag play --loop won’t work with this option, and the topic publication is stopped.
- ~msg_par_second (Double, default: 15.0)
  Topic publishing rate if ~use_messages is true.
- ~interpolation (String, default: LINEAR)
  Candidates: NEAREST, LINEAR, AREA, CUBIC, LANCZOS4
  See this page for more details

23.1.4 Sample

```bash
$ roslaunch resized_image_transport sample_image_resizer.launch
```

23.2 LogPolar

![LogPolar Image](image_url)
23.2.1 What is this?

Map input image to log polar space. 
This function is an imitation of central vision of human.

23.2.2 Subscribing Topic

- ~input/image(sensor_msgs/Image)
  Input image.
- ~input/camera_info(sensor_msgs/CameraInfo)
  Input camera info.

Note: The subscribing topic name is changed when you remap the one of input image, because it uses image_transport.

23.2.3 Publishing Topic

- ~output/image(sensor_msgs/Image)
  Log polar space image.
- ~output/camera_info(sensor_msgs/CameraInfo)
  Output camera info.

23.2.4 Parameters

- ~inverse_log_polar(Boolean, default: false)
  Whether to apply inverse mapping.
- ~log_polar_scale(Float, default: 100.0)
  Scaling coefficient. This parameter can be changed by dynamic_reconfigure.
- ~resize_scale_x, ~resize_scale_y(Float, default: 1.0)
  Resizing scale. This parameter can be changed by dynamic_reconfigure.
- ~msg_par_second(Float, default: 15.0)
  Topic publishing rate [Hz]. This parameter can be changed by dynamic_reconfigure.
- ~verbose(Boolean, default: false)
  This parameter is not used.
23.2.5 Sample

```bash
$ roslaunch resized_image_transport sample_log_polar.launch
```