Assignment 3: Object Recognition

ETH Zurich

Individual Demonstration: Tuesday, 24.11.2015 at 16:15
Individual Software Due: Tuesday, 24.11.2015 at 23:00
Group Work Demonstration: Monday, 30.11.2015 at 16:15
Group Work Software Due: Monday, 30.11.2015 at 23:00

1 Object recognition

1.1 Background

Object recognition is the process of identifying known objects in an environment. Various object recognition algorithms exist, including appearance-based algorithms and feature-based algorithms. Appearance-based algorithms recognize test objects by matching them to example images of the known objects. As these algorithms assume the appearance of objects in the training set and the test to be similar, success of recognition depends on similarity of the conditions in which data are collected. Feature-based algorithms, on the other hand, find objects by matching object features and image features. Features represent images more compactly, and with well-devised features, feature-based algorithms can recognize objects in different conditions.

Spin image [1] is a 3D feature descriptor that represents the distribution of points around a point of interest as a 2D histogram. For a given point with a normal $\vec{n}$, a spin image is computed by spinning a sheet around the normal and collecting its neighboring points into a 2D histogram. Given a 2D histogram indexed by α by β, the spin image is generated for a point by looping through all points within the support of the spin image, and for each point, incrementing the bin that corresponds to the coordinate (α, β) of the point. The resulting 2D histogram (or spin image) is then normalized so that it can be compared to the model spin images for recognition.

Several parameters affect the spin image. One parameter is bin size. Bin size is the geometric width of bins in spin image and is often set as a multiple of the resolution of the points. This eliminates the dependency of bin size on object scale and resolution. Bin size determines the storage size of the spin image and has an effect on the descriptiveness of the spin images. Another parameter is image width, which is the number of rows or columns in a square spin image. Multiplying image width by bin size gives spin image support distance. Support distance determines the amount of space swept out by a spin image. Last parameter is support angle. Support angle is the maximum angle between the direction of the oriented point basis of a spin image and the surface normal of points that are allowed to contribute to the spin image. A neighboring point with a normal $\mathbf{n}'$ makes a contribution towards the spin image with normal $\mathbf{n}$ if $acos(\mathbf{n}', \mathbf{n}) < A_{threshold}$. Support angle limits the effect of self occlusion and clutter during spin image matching. More detail on spin image generation can be found in chapter 2.3 of `http://www.ri.cmu.edu/pub_files/pub2/johnson_andrew_1997_3/johnson_andrew_1997_3.pdf`.

Object recognition can be performed as follows: first, we compute a set of spin images for the models. Then, for a given scene, we can first segment the scene to extract foreground. The foreground extraction enables us to focus our computational power only on most relevant areas of the scene. We can then compute spin images at some random points of the foreground object and match the extracted spin images to the model's spin images. We then filter the matches to find a plausible transformation from the model to the object. The match can then be verified

by transforming all the points and evaluating the overlap or by ensuring that spin images from other points of the foreground object also match well to the model.

1.2 Task

Implement an object recognition algorithm using spin image as the feature descriptor. The Point Cloud Library (PCL) [2] provides everything you need, including segmentation algorithms, spin image computation, and correspondence. Your task is to put different algorithms together so that your system can recognize objects that are in the visible range of the camera.

There will be two classes of objects: wooden toy dolls ("human") and rubber ducks ("duck"). Your recognition system should be able to label toy dolls and rubber ducks correctly. In addition, it should label an object as unknown if it is very different from the two known classes ("human" or "duck").

1.2.1 Practical Help

First, open your openni2.launch file

```
sudo gedit /opt/ros/indigo/share/openni2_launch/launch/openni2.launch
```

and change the following lines

```
<arg name="rgb_frame_id" default="$(arg tf_prefix)/$(arg camera)_rgb_optical_frame" />
<arg name="depth_frame_id" default="$(arg tf_prefix)/$(arg camera)_depth_optical_frame" />
```

to

```
<arg name="rgb_frame_id" default="$(arg camera)_rgb_optical_frame" />
<arg name="depth_frame_id" default="$(arg camera)_depth_optical_frame" />
```

by removing `$(arg tf_prefix)/`. Then, connect your Carmine 1.09 sensor to a USB 2.0 port and run:

```
roslaunch openni2_launch openni2.launch
```

Run RViz and set the fixed frame to `camera_link`. Add `PointCloud2` type, subscribe to `/camera/depth_registered/points` topic, and set color transformer to `RGB8`. You should be able to see RGBD data. You can also visualize `/camera/depth/points` and set color transformer to `Intensity`. This topic contains only depth data.

Your recognition node should subscribe to `/camera/depth_registered/points` (XYZRGB) or `/camera/depth/points` (only XYZ) of type `sensor_msgs::PointCloud2` and convert the data to PCL. http://wiki.ros.org/pcl_ros describes how to publish and subscribe point clouds in ROS. Once the image is converted to PCL data type, you can utilize various functionalities that PCL provides to perform the necessary task. http://www.pointclouds.org/documentation/tutorials/ contains many tutorials on PCL. Some examples that may be useful include:

- Segmentation: plain model segmentation, Euclidean cluster, region growing
- Filtering: pass through filter, statistical outlier removal
- Recognition: correspondence grouping

The only requirement is that your recognition system uses SpinImage as the 3D feature descriptor. Outside of this requirement, you are free to use other functionality that PCL offers to enhance your recognition system.

http://docs.pointclouds.org/trunk/classpcl_1_1_spin_image_estimation.html describes `pcl::SpinImageEstimation` class in detail. Parameters you need to set include `radius` for the *spin image support distance*, `image width` for the *bin size*, and `support angle`. You may also want to set `min point count` to ensure that every spin image is created from a minimum number of neighboring points. Note that terms used in the API are not necessarily the same as those used in the original paper. See http://goo.gl/IBKxym for a tutorial on SpinImage in PCL.

Lastly, the `data` repository contains sample PCD files of objects you are to recognize in this assignment. The I/O section of PCL tutorials has an example on how to read point cloud data from PCD files.

2 Grading

2.1 In-class demonstration (20 points)

The goal of this assignment is to demonstrate that your robot can recognize different objects. The recognition system should place a bounding box around each recognized object using `visualization_msgs::Marker::CUBE` or `visualization_msgs::Marker::CYLINDER` and display the bounding boxes in RViz. Each bounding box should be centered at the corresponding object's center and its radius/width and height should equal to those of the object. For more information on Marker type, read http://wiki.ros.org/rviz/Tutorials/Markers:BasicShapes and http://wiki.ros.org/rviz/DisplayTypes/Marker.

2.1.1 Individual In-class demonstration (10 points): Tuesday, 24.11.2015 at 16:15

In the individual portion, you will demonstrate how your system can detect and label four different objects. Three object classes are "human", "duck", and unknown. Objects will be placed in front of the Carmine 1.09 sensor, up to two at a time. The environment will contain nothing within the visible range of the sensor except the object to be recognized. You will display a bounding box around the object in RViz and set the color based on the object's label – green for "human", blue for "duck", and red for "unknown".

- Every correct recognition (2.5 points)
  - Label (1.5 points)
  - Bounding box (1 point)

Note that if your system assigns a fixed color (without recognition), you will not get any credit for the labeling.

2.1.2 Group In-class demonstration (10 points): Monday, 30.11.2015 at 16:15

In the group portion, you will demonstrate how your robot can go from start to goal, visiting two rooms on the way. The environment is as defined in testenvironment.png plus one obstacle. You will start at (0.5, 0.5), visit two rooms, (1.5, 0.5) and (1.5, 1.5), and finish at (0.5, 1.5). Each room will have zero to two objects of type person, duck, or unknown. As in the individual assignment, the recognition system should publish a colored bounding box around each recognized object.

You can utilize the knowledge about the environment, i.e., the wall and ground are planar and white, to ease the segmentation and foreground extraction. You can either run the recognition at regular interval or detect the change of the environment, e.g., when the color distribution of the scene deviates a lot from the expected background color distribution, and run the recognition only then.

- Object recognition (3 objects, 2 points each)
  - Correct placement of the bounding box in the map coordinate frame (1 point)
  - Correct label (1 point)
- Speed (2 points)
  - Within 5 seconds of the fastest robot (2 points)
  - Every 5 seconds thereafter (-0.1 point)
- Accuracy (2 points)
  - Within $1cm$ of the closest robot (2 points)
  - Every $1cm$ thereafter (-0.1 point)
- Bumping (-1 point)
- You will get two attempts. Every extra attempt will cost 1 point.

2.2 Software quality (20 points)

On the due date at 23:00, we will collect your code through your SVN repository. Every file that should be considered for grading must be in the repository at that time. Note that EIFGENs folder in your project contains auxiliary files and binaries after compilation. Please, DO NOT include EIFGENs folder into your svn repository.

2.2.1 Individual evaluation (10 points): Tuesday, 24.11.2015 at 23:00

- Choice of abstraction and relations (3 points)
- Correctness of implementation (4 points)
- Extendibility and reusability (2 points)
- Comments and documentation, including "README" (1 points)

2.2.2 Group evaluation (10 points): Monday, 30.11.2015 at 23:00

- Choice of abstraction and relations (3 points)
- Correctness of implementation (4 points)
- Extendibility and reusability (2 points)
- Comments and documentation, including "README" (1 points)

References

[1] Johnson, A. and Herber, M. "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes". IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5). 1999.

[2] http://wiki.ros.org/pcl_ros and http://www.pointclouds.org.