User Manual - BuildModel

This document provides all the required information to build a 3D Model of the object, using RGB-D Sensors (such as Microsoft Kinect, Orbec Astra Pro and Intel Euclid RealSense).

1 Download

The code to build a 3D model of an object can be found here.

2 Prerequisites

The following prerequisites are required to build and run the code.

- OpenCV 2.4.8 or higher
- PCL 1.7.1 or higher
- ROS Indigo or higher
- CMAKE 2.8.9 or higher

3 CMake and Build

To build the code execute the following steps :

1. Create a build directory in *BuildModel* folder.
   $ cd Object-Pose-Estimation-master/BuildModel
   $ mkdir build
   $ cd build

2. CMAKE and Build
   $ cmake ..
   $ make

4 Command Line Interface

Usage : ./BuildModel [-parameters=value]

Parameter	Default Value	Description
h help	false	Prints the help
o object		Name of the object to be built
l limits	-0.2,0.3,-0.5,0.1,0.5,1.05	Xmin,Xmax,Ymin,Ymax,Zmin,Zmax for field of processing of sensor (in m) No Spaces!
d directory	empty	Provide a directory of point clouds to built a model
s sensor	kinect	3D Sensor(kinect, astra, euclid)
cd corrdist	0.005	Maximum correspondence distance (in m) for correspondence estimation
t threshold	0.7	Correspondence rejection threshold (*arccos* of angle between correspondences)
m maxiter	500	Maximum iterations for ICP to converge

5 Working and Example

1. To make a 3D model of an object, the object should be placed on a plane. We need to launch the sensor using ROS packages to get the input from the sensors. The moment this code is launched the viewer starts and press 'c' to capture the frames and 'f' to finish capturing the different views of the object to reconstruct it.
   Our job is done!
   The frames captured are saved on the system in the folder same as object name, one level above the build directory. The program will automatically segment the object placed on plane and apply ICP to align the frames to make a 3D model of the object. The model is saved in the same folder as frames are saved. It also builds a mesh of the aligned model and saves in the same folder. Various processes happening during this program are displayed on the terminal.

   For example, if we want to reconstruct a cup with different limits using astra sensor.
   $ ./BuildModel -o=cup -l=-0.5,0.5,-0.5,0.3,0.5,1.6 -s=astra

2. If we are not satisfied with the built model using the parameters. We can provide the required parameters and the directory of saved frames instead of scanning the object again!
   For example,
   $ ./BuildModel -d=path_to_directory -t=0.3 -m=100