CS 547 Josh Villbrandt
Project – Final Report 28 November 2011

identify_duplos (ROS Package)

The goal of this project is to "count the Duplos on the table." The tools available include a point cloud-producing sensor (the Kinect), ROS, the Robotic Operating system from Willow Garage, and PCL, the Point Cloud library. An end goal for this project is to provide some usual information to another agent such as a robotic arm motion planner. The information provided could help the motion planner manipulate LEGO Duplos.

Two methods of counting Duplos are presented here. The first method is very naïve and separates a point cloud into clusters of bricks only by color and distance segmentation. It will be shown that this method is very reliable and very fast in a controlled situation. However, this method is shown to be imprecise. The second method improves upon the first by trying to fit known models of Duplo blocks to the point clouds clusters. With a known model is aligned to a target cloud, its shape and full 6DOF can be determined. While much more descriptive in its results, this method is slower and requires training the model before use.

Algorithm ("Quick Mode")

Starting with an input point cloud (either from a Kinect or from a PCD file via the read_pcd program), RANSAC is used to find the largest continuous plane in the cloud. This is assumed to be the floor or table, and it is extracted. Next, all points behind the floor / table plane are removed using the equation of a plane provided by the RANSAC algorithm. Next, statistical outlier filter is executed to remove noise.

This is especially helpful when working with carpet. Now, the point cloud is segmented by color. It is important to do this step before distance segmentation to ensure that each block is fully separated. The colors to be segmented against are supplied via a configuration file. RGB values for each color are given as well as a range value for each channel. Points get allocated to the most appropriate color or get thrown out.

Finally, each color cluster is segmented using a Euclidean distance search. At this point it is assumed that blocks are separated into individual clusters. If multiple blocks get stuck in a single cluster, the identified shape will most likely read "2x4 block" in quick mode. Quick mode essentially identifies block shapes by the number of points in the individual cloud. (A voxel grid with a constant resolution could be used here to turn raw point count into a comparable number.) The orientation of clusters in quick mode are given as the normal vector of the largest plane in the cluster (again given by RANSAC.) The color and position of the block are given by averaging the entire cluster.

Algorithm (Full-Pose Mode)

The full-pose mode picks up where quick mode leaves off and provides a much more thorough analysis of a blocks properties. This is achieved by testing each cluster against block templates using a sample consensus initial-alignment algorithm to find the best fit. (Future versions will assign block shape "unknown" for poor alignments.) Since the pose of the block templates is set during training, there is now a much more accurate estimate of a blocks properties. The orientation usually respects whether a block is right side up or upside down, position information is now given as the blocks centroid, and block shape is inherently point-cloud resolution independent.

The training program uses the same sample consensus initial-alignment to build block templates. The goal of this program was to be able to slowly and continuously rotate a block in ones hand and for the program to automatically learn the shape. Unfortunately, due to the symmetrical nature of Duplo blocks, the FPFH descriptors only ever align to three sides reliably. In future versions, the rotation during the alignment phase could be constrained to force the block template to continue building on all sides. Once a suitable block template has been made, a “write” command is given to the trainer. Then a PCD file is created and the programmer makes an corresponding entry into the configuration file for the “identify” program.

Usage

Please view the README.txt file located in the ROS package for more specific usage details. The code is also well documented, and more information can be gathered from it.

Results

Photographs of this project can be found in the final presentation PowerPoint show. As stated before, the “quick” version of this program works fast and well, albeit in a controlled scenario. While much effort was placed into the full-pose version of this program, time constraints prevented this portion from fully maturing. With that said, the template alignment algorithm works well enough to build strong block templates in the training program.

Notes for the Future

During the development of the project, much time was wasted on the identification and re-identification of block colors because of varying ambient light levels. It is desired to have an HSV representation of color instead of an RGB representation so that hue can be used independently of brightness.

The full-pose version of this program is an order of magnitude slower than the quick version. It is noted that many OpenMP drop-in replacements are available within PCL. Running this program in a parallel manner would also for significant speed gains.

Finally, much thought was given to including a Bayesian filter. While it ended up being unnecessary to complete the task, such a filter could provide for must increased capabilities, especially when introducing a robot arm that could block the point cloud sensor or when there are so many Duplos that the cannot be observed all at once. An intelligent filter could maintain the status of previously identified blocks even when that could not be seen.