1. **OVERVIEW**

This document describes in more detail the various sections of the poster, as well as some additional work that did not make it into the deliverable package, but which we regard as promising for future extensions to our project.

1. **SEGMENTATION PIPELINE**

A sequence of algorithms, aka. pipeline, is needed to achieve object segmentation. The segmentation pipeline starts with capturing an image from a 3D depth camera. At the end of the segmentation pipeline, we obtain the location and boundary information of the object of interest in the scene. The segmentation pipeline passes through the following stages sequentially:

1. **Downsampling**

The raw point clouds incoming from the camera, have a resolution which is far too high for segmentation to be feasible within the allotted 500 ms. The basic technique to solving this problem is called ‘voxel filtering’, and it entails compressing several nearby points into a single point. In other words, all points in some specified cubical region of volume will be combined into a single point. The parameter which specifies the size of this volume element is called the ‘leaf size’. Fig. 1 shows an example of applying the voxel filter with several different leaf sizes.

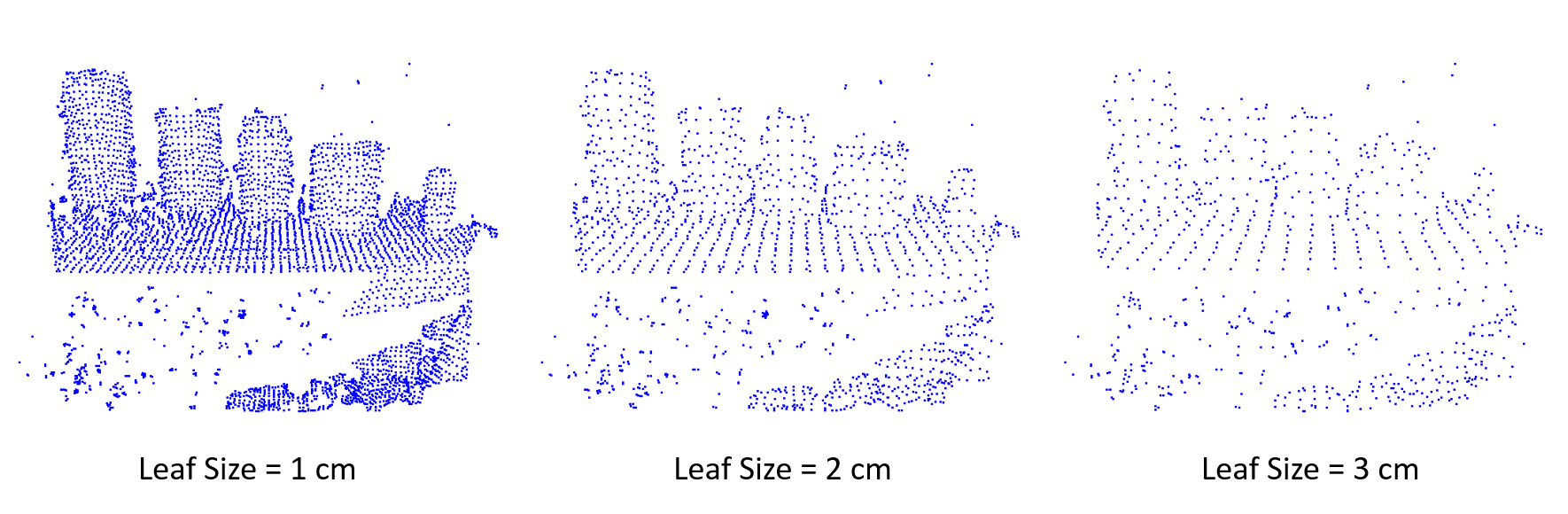


Figure 1. PCD images of 5 objects on a table for different leaf sizes

1. **Plane/Prism extraction using RANSAC**

RANSAC is a quick method of finding mathematical models. In the case of a plane, the RANSAC method will create a virtual plane that is then rotated and translated throughout the scene looking for the plane with the data points that fit the model (aka. Inliers). The two parameters used are the threshold distance and the number of iterations. The greater the threshold, the thicker the plane can be. The more iteration RANSAC is allowed to do, the greater the probability of finding the plane with the most inliers. In Fig. 2, one can see what happens as the number of iterations is changed. The blue points represent the original data, the red points represent the plane inliers, and the magenta points represent the noise (aka. Outliers) remaining after prism extraction. As can be seen, the image on the left shows how the plane of the table was not found due to RANSAC not being given enough iterations. The image on the **r**ight shows the plane being found and the objects above the plane being properly segmented from the original data.

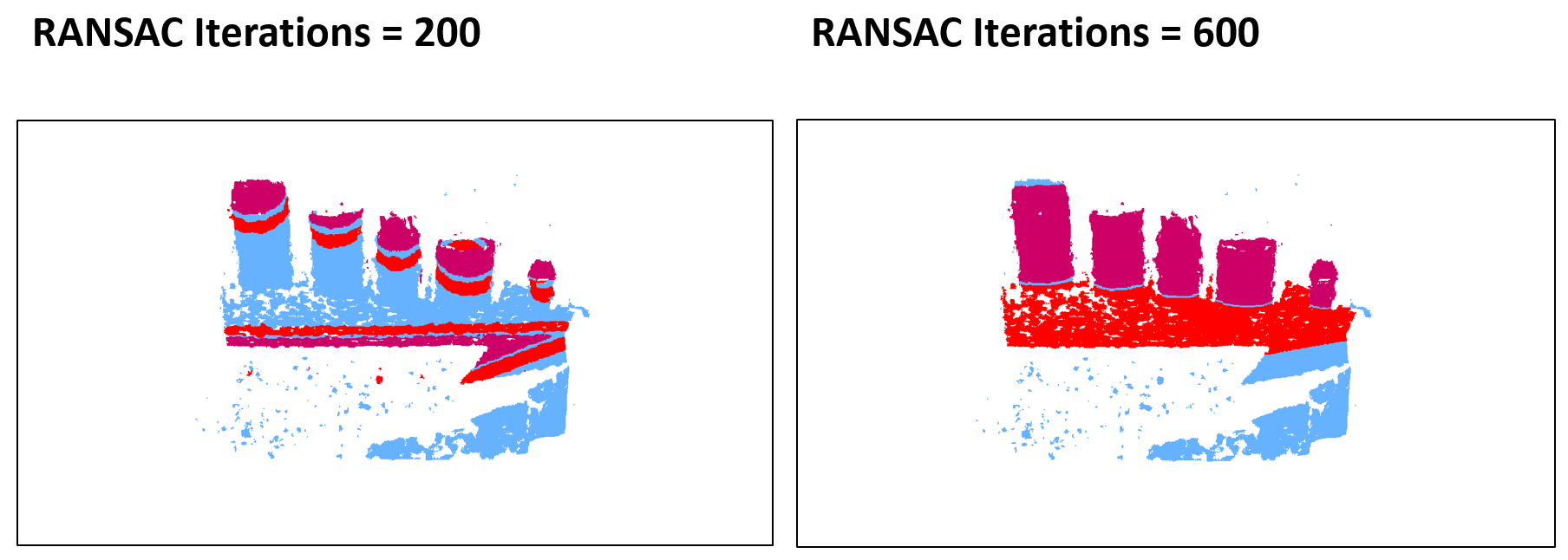


Figure 2. PCD images showing the difference in RANSAC iterations

1. **Clustering**

The last stage in the segmentation process is ***Euclidean clustering***, which takes the down sampled point cloud, minus the plane and its convex hull, and breaks it into clusters, with each cluster hopefully corresponding to one of the objects on the table. This is accomplished by first creating a ***kd-tree*** data structure, which stores the remaining points in the cloud in a way that can be searched efficiently, and then the cloud points are iterated over, with a radius search being performed for each point. Neighboring points within the threshold radius are then added to the current cluster, and marked as processed. This continues until all points in the cloud have been marked as processed.

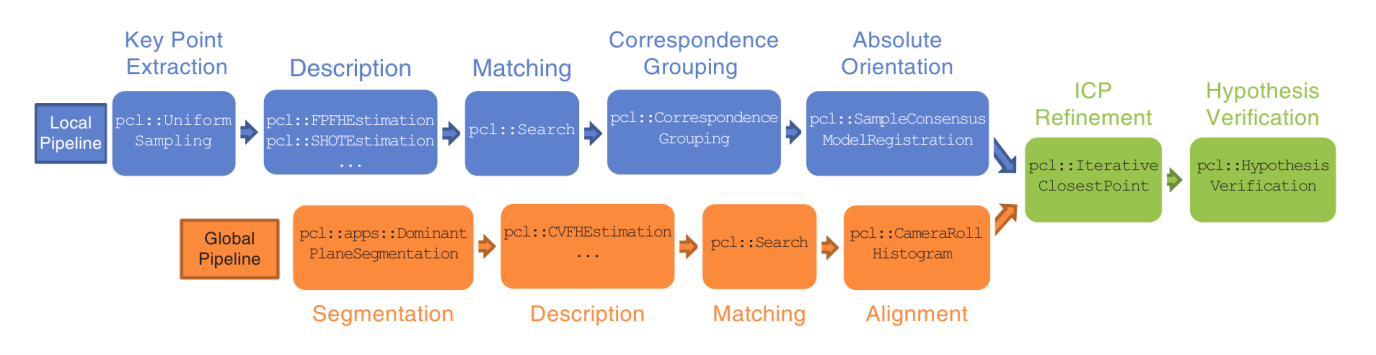
1. **OBJECT RECOGNITION PIPELINE**

Figure 3. Figure 3: Local (top) and global (bottom) object recognition pipelines. Image credit: Aldoma et al [4], Vienna University of Technology

The objective of object recognition is to identify an object of interest out of a scene to get the information of its position and orientation. For our research, we are interested on identifying the position of the robot hand at a particular instant of time so that the robot hand can be guided to pick up the segmented object. This requires us to have a model of the robot hand and compare it against the scene. There are two standard approaches in 3D object recognition to compare the model and the scene: global pipeline and local pipeline. As shown in Fig. 3, these two techniques achieve the goal of object comparison/matching through the implementation of two distinct sequences (pipelines) of feature recognition algorithms. Particularly, they use different descriptors to compare the objects. A descriptor encodes information on model and scene images to make it possible to compare them. The more insights on our global and local pipelines are provided below:

1. **GLOBAL PIPELINE**

The global pipeline works by taking the object clusters from the earlier clustering step and computing a histogram for each of them. This histogram is then compared against a database of pre-calculated histograms for some object of interest, in our case, the robot’s hand. These pre-calculated histograms are simply loaded at program boot-time. There are many histograms in the database for each object of interest because one needs to have data representing an object from many different camera positions.

There are several descriptors that are available to store the data: Viewpoint Feature Histogram (VFH), Clustered Viewpoint Feature Histogram (CVFH), Oriented, Unique and Repeatable CVFH (OUR-CVFH), Ensemble of Shape Functions (ESF), and the Global Radius-based Surface Descriptor (GRSD). All of these variants were tested by our group, but we ended up choosing the OUR-CVFH descriptor, mainly because that was the approach favored by some of the original authors of PCL.

* 1. **VFH**

The VFH has a viewpoint dependent part, which stores data on the angles between the normal vectors of the points in the cloud and the vector from the camera viewpoint to the cluster centroid. Another portion of the histogram stores data on the distances of the points in the cluster to the cluster’s centroid. There are other components to the histogram as well, but the full details are somewhat beyond the scope of this handout, and the reader is referred to [4] for a detailed description.

* 1. **CVFH**

The Clustered VFH descriptor is similar to VFH except that it first divides the cluster into several regions using region-growing segmentation. An example would be a milk carton being divided into clusters corresponding to the sides of the carton. From here VFHs are computed for each region. This offers a significant advantage over VFH, in that only one of the regions need be visible for the object to be identified. So, for example, if the desired object is occluded by some other object standing in front of it, the desired object can still be picked out based on the visible part.

* 1. **OUR-CVFH**

The Oriented, Unique, Repeatable CVFH descriptor extends CVFH by adding one additional processing stage after the region segmentation ￚ it further filters the points in each region according to their difference from the region’s average normal, thereby further refining knowledge of the region’s shape.

After this step, a Semi-Global Unique Reference Frame (SGURF), a.k.a, a local coordinate system, is computed for each region. It is these reference frames which make up the content of the descriptor. The full detail of this process can be found in [7].

The result of these extra processing phases is an 82% detection rate which is a notable improvement over other global descriptors.

1. **LOCAL PIPELINE**

Local object segmentation techniques extract key points information in the model and the scene images and associate them in a local 3D descriptor neighborhood [1]. A local keypoint information provides information on the local geometric features such as image position and coverage area. A local descriptor compares the keypoints between the model and the scene.Unlike global descriptors that describe an entire object, local descriptors use multiple points in the image for comparison. Since local feature recognition technique can operate on image patches, it gives promising recognition even for occluded and cluttered scenes in which global feature recognition technique fails.

We designed correspondence grouping and object alignment technique based on local feature descriptors to evaluate the performance of local pipeline [5][4].In each of the pipeline, we downsample the point cloud, take the keypoints associate with them, and use the them for 3D local descriptor Signature of Histograms of Orientations (SHOT) and Fast Point Feature Histogram (FPFH) descriptors. Then, matching is performed to determine the presence of model instance in a scene.

Although we were able to achieve high accuracy of object recognition using local pipeline, its computation time was higher than our execution time requirement, hence we abandon this approach.

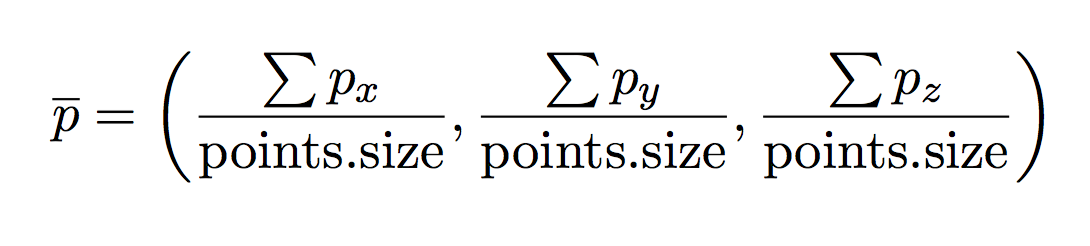
**IV. Bounding Box Calculation**

The bounding box of a cloud is obtained by

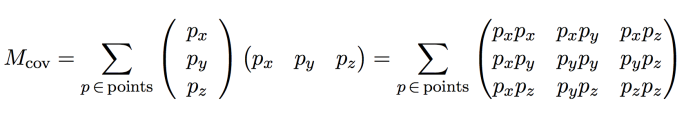
calculating the minimal and maximal points in the

cloud as well as its center of mass. We take the following steps to get the minimum and maximum points of the bounding box:

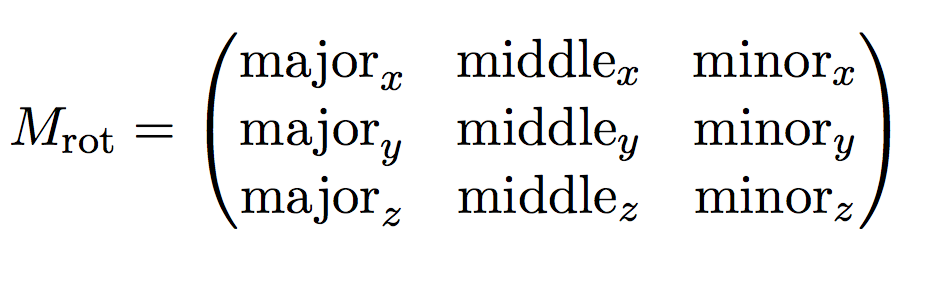
1. Compute mean value vector of the points in the cloud, p-bar:
2. Compute the covariance matrix, *M*cov



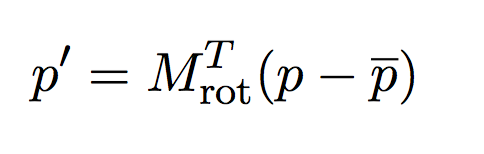
***Equation 1. Mean value of the point cloud***



Equation 2. Covariance Matrix

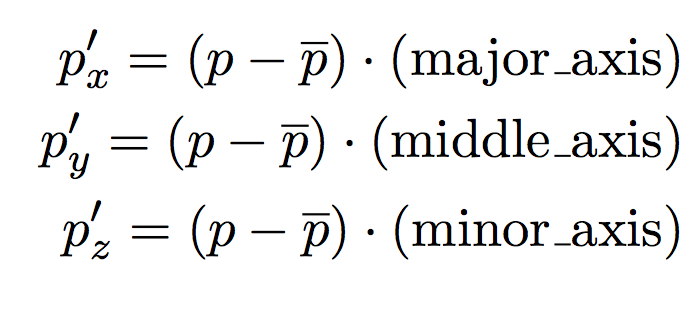
1. Solve for the eigenvectors and eigenvalues of the covariance matrix. The three eigenvectors, major, middle, and minor, then make up the columns, a.k.a., the axes of the rotation matrix. 

Equation 3. Rotation Matrix

1. Compute the max and min points for the bounding box: for every point *p* in the cloud, compute the difference from the mean value vector, rotated by the rotation matrix, i.e., 

Equation 4. Point after transformation

Which is equivalent to



Equation 5. x, y, z component of Eq. 4

And then compare these to the current max/min point values. If they are greater/lesser respectively, then overwrite them with these new values.

1. Compute the midpoint between the max and min points. This is the shift vector.

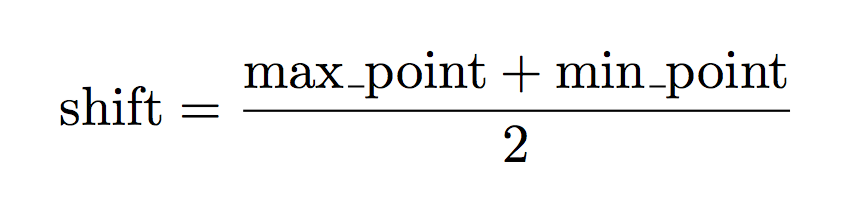
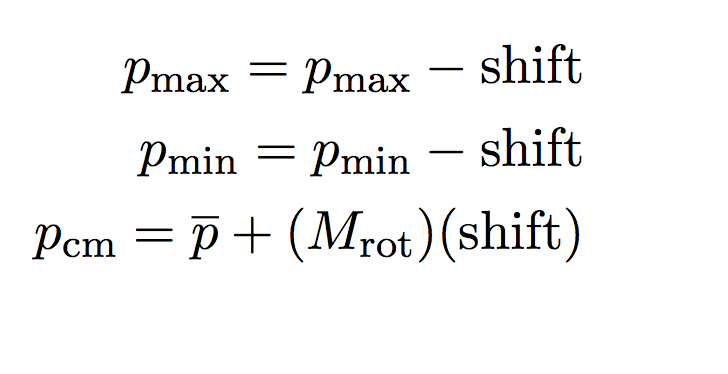


Figure 4. Voxel filter execution time (top-left). Polynomial prism extraction on using RANSAC (top-right). Polynomial prism extraction threshold testing(bottom-left). Range of execution times from parameter testing (bottom-right).

Equation 6. Midpoint calculation

1. Finally, adjust the max/min points by the shift, as well as setting the position vector to the center of the box. This give the complete description of the bounding box.



Equation 7. Min and max point of the bounding box

**V. TESTING**

To test the performance of the segmentation pipeline, we first needed to find the effect each function parameter had on the execution time of the program. To do this, the program was run by only changing one parameter at a time over a range of values. As shown in Figure 4 **(bottom-right)**, not all functions in the segmentation pipeline have an equal impact on the execution time. This led to some functions being optimized for performance over the resulting point cloud data.

The images in Figure 4 show the results of each individual test. All parameters, with the exception of the one being tested, were held constant for each test. Also, the source point cloud data used was a RAW unedited scene from the Realsense camera. For each graph the black line represents the execution time in comparison to the parameter being tested. The orange line shows how many points are left in the resulting PCD file.

**VI. APPLICATIONS**

There are several practical applications of object segmentation. Machine vision, medical imaging, human machine interaction, place recognition, object detection, video surveillance and manufacturing industry are some of the fields where the implemented object segmentation and recognition algorithms can be used.

**VII. FUTURE WORK**

**Accuracy Benchmarking**

Some work was done on a method for measuring the accuracy of bounding boxes calculated by the segmentation pipeline. The idea is to first analyze an image without downsampling so that the resulting bounding box is regarded as an ideal model. Then, the segmentation parameters can be adjusted and the degradation in accuracy caused by these adjustments can be measured quantitatively.

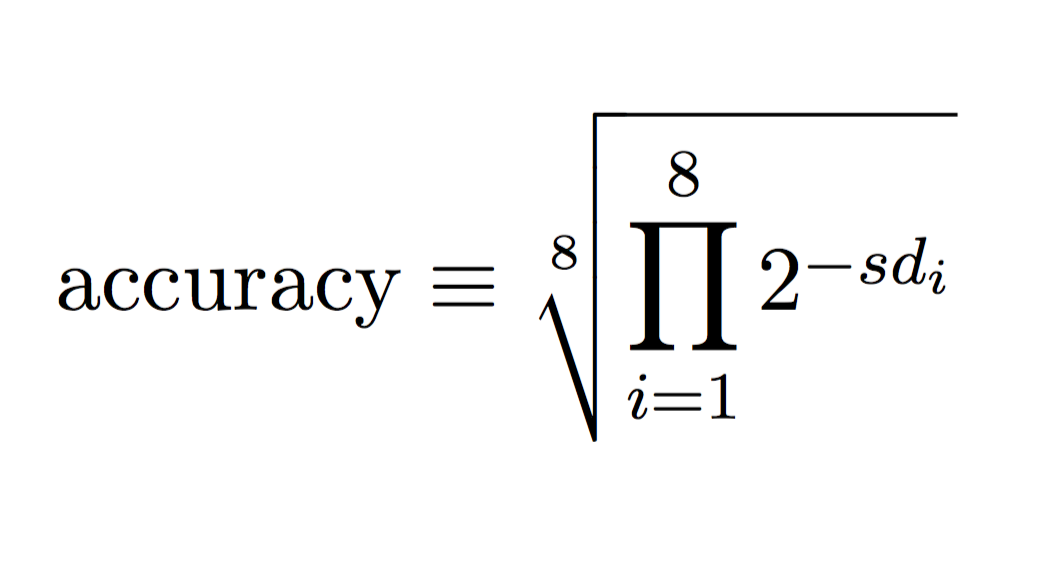
The first step in the process is to find the best fit pairing between the corners of the cube. While this operation is O(n!), where n = 8, our initial tests

revealed that it was not a significant part of the execution process. Additionally, the expense of this operation is mitigated by skipping the square root

operation since minimizing the square of the distance implies minimizing the distance as well.

Once this best fit matching of the corners has been obtained, we can calculate the accuracy of a box with respect to the ideal box-model. This calculation was motivated with the following reasoning .A corner of the box is 100% accurate if it has the exact position of the corner in the box which is being compared, and the accuracy will fall off exponentially as the distance between the corners increases, reaching zero if the distance between them is infinity. So, it is reasonable to say that the accuracy will be halved for each unit of distance between the 2 corners. This distance unit should be chosen based on the context, e.g., if the application is dealing with small objects, a distance unit of a few centimeters might make sense.

Since there are 8 corners to compare between the 2 bounding boxes, the total accuracy will be the geometric mean of the product of the accuracies for the 8 corners. Symbolically,



Equation 8. Mathematical definition of accuracy

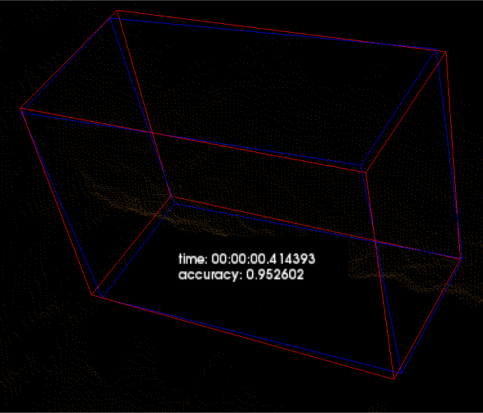
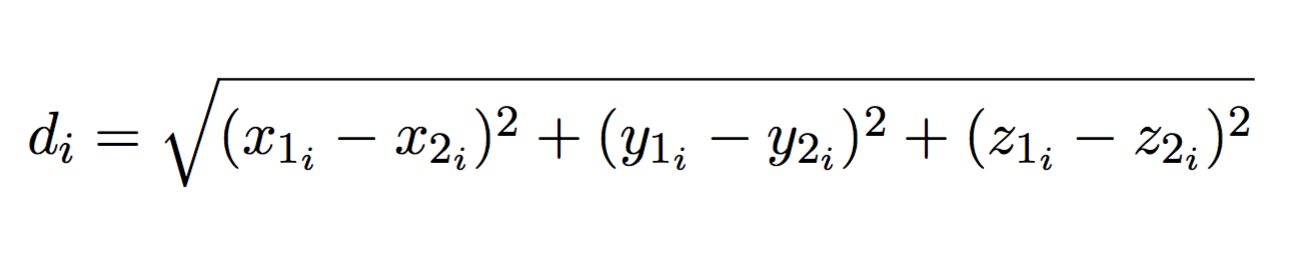


Figure 5. an example accuracy calculation, showing 2 boxes which match with 95% accuracy

Where *di* , i.e., the distance between the *i*’th corner in the first box, and the *i*’th corner in the second box, is given by the standard Euclidean distance formula:



Equation 10. Distance between points in 3D space

This algorithm has been implemented in our code base, but it took second priority to getting the segmentation pipeline working, and for that reason has not been extensively tested.

**VIII. CONCLUSION**

We were able to successfully implement object segmentation and hand recognition pipeline. We publish the information of the segmented object and robot hand on ROS nodes which allows other parties to move the robot hand without the knowledge of the implementation detail.

**REFERENCES**

[1] Aldoma, Aitor, Federico Tombari, Luigi Di Stefano, and Markus Vincze. "A global hypotheses verification method for 3D object recognition." In *Computer Vision–ECCV 2012*, pp. 511-524. Springer Berlin Heidelberg, 2012.

[2] “PCL/OpenNI tutorial 5: 3D object recognition (pipeline).”

Robotics Group of the University of Leon.

<http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_5:_3D_object_recognition_(pipeline)>

Updated Nov 1, 2015. Accessed May 29, 2015.

[3] “Cluster Recognition and 6DOF Pose Estimation using VFH descriptors.” *pointclouds.org*. Perception Foundation.

<http://pointclouds.org/documentation/tutorials/vfh_recognition.php>

Accessed May 29, 2015.

[4] Aldoma, Aitor et al. *Three-Dimensional Object Recognition and 6 DoF Pose Estimation*. IEEE Robotics & Automation Magazine, September 2012.

[5] "Aligning Object Templates to a Point Cloud." - Point Cloud Library (PCL). Accessed May 30, 2016. http://pointclouds.org/documentation/tutorials/template\_alignment.php.

[6] Aldoma, Aitor, and Federico Tombari. "3D Object Recognition Based on Correspondence Grouping." - Point Cloud Library (PCL). Accessed May 30, 2016. http://www.pointclouds.org/documentation/tutorials/correspondence\_grouping.php.

[7] <http://vision.deis.unibo.it/fede/papers/dagm12.pdf> (TODO finish this)