3D OBJECT RECOGNITION AND POSE ESTIMATION

USING AN RGB-D CAMERA

by

Daniel Bray

Presented to the Graduate Faculty of

The University of Texas at San Antonio

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Mechanical Engineering

THE UNIVERSITY OF TEXAS AT SAN ANTON

Abstract

Identifying and determining the 6 degree of freedom pose of an object is essential for any robotic task where the robot has to interact with the world. This project focuses on developing and testing an algorithm that can accurately estimate an object's pose using a cheap and commercially available sensor like the Microsoft Kinect or ASUS Xtion RGB-D camera. This is achieved by developing a complete object recognition and pose estimation algorithm that is built around the Viewpoint Feature Histogram (VFH). The algorithm is capable of accurately estimating the pose of an object 90% of the time when at a distance of 1.25m or less from the camera. The computation time of the algorithm does not lend itself to real-time applications but does show the Kinect and Xtion sensors are cheap and viable alternatives for pose estimation at close range. However, their noise and measurement errors are an issue at greater distances.

Contents

List of Figures

List of Tables

1 – Introduction

One of the most fundamental requirements for mobile or manipulator robots is their ability to "see" their environment. Doing so enables robots to handle tasks such as navigating an unknown environment or locating and grasping an object by thinking and planning on their own based on the surroundings. To see the robot must interrupt the data given by any number of vision sensors such as LIDAR, RGB cameras, stereo cameras, structured light cameras, or a combination of these. Until recently the cost of these sensors, particularly those that produce 3D images, made robots that required vision cost prohibitive for many applications. However, sensors like the Microsoft Kinect and ASUS Xtion RGB-D cameras have greatly reduced the price of sensors that can produce a 3D image.

Figure 1: Microsoft Kinect and ASUS Xtion RGB-D cameras

1.1 – Project Overview

This project focuses on developing a vision system that is capable of estimating the 6 degree of freedom (DOF) pose of an object using the ASUS Xtion RGB-D camera as the sensor. An algorithm is presented that can accurately and reliably provide a pose estimate for an object. The algorithm takes advantage of several tools provide by the Point Cloud Library (PCL) [1] C++ open source libraries to develop a complete program that is able to estimate the pose from a single 3D image provide by the camera.

The process of the algorithm consists of the three distinct steps presented in Figure 2. First, the image goes through a preprocessing stage to prepare the raw image for object recognition. The second phase is to identify what objects are in the scene. Lastly, the identified objects pose is estimated and verified for correctness.

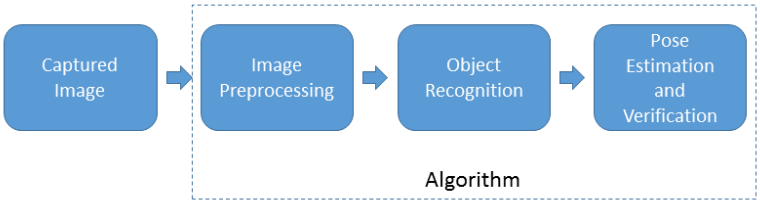

Figure 2: Algorithm flowchart

The algorithm is developed with the assumption that the object is not occluded and can be easily discerned from the rest of the environment. Ideally the algorithm would be able to make these estimates in real-time but the scope of this project is to only provide the pose estimate. Instead, the algorithm will be optimized to run as quickly as possible while still producing accurate results.

2 – Background

2.1 – The RGB-D Camera

The camera used is the ASUS Xtion Pro Live RGB-D camera and has become a popular sensor for a multitude of robotic applications for its benefits of low cost and open source SDK. The camera contains a standard RGB color sensor that can be found any other digital camera, but it also contains a structured light sensor that is capable of measuring the depth in an image. This algorithm will exclusively utilize the depth information to generate 3D images as the color data is often not present for CAD models.

Determining the depth is done by projecting an infrared grid pattern onto the scene. When viewed from a perspective other than the projector's this grid will appear distorted. By analyzing this distortion the depth in the image can be calculated and used to construct a set of 3D XYZ points known as a point cloud. Figure 3 shows a example point cloud image taken with the Xtion camera.

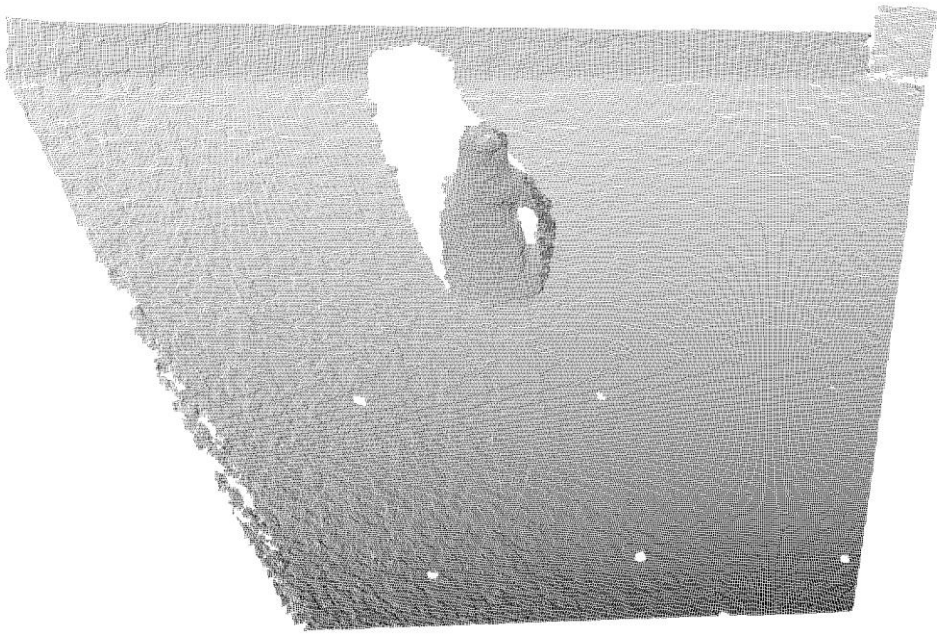

Figure 3: Example point cloud image

2.2 – Object Descriptors

To determine the pose of an object, that object must first be recognized by the algorithm. The algorithm is able to identify an object by quantifying its geometric properties such as the surface normal directions, surface curvature, point distribution, etc. This quantification is known as a descriptor and has two distinct classifications, local and global, based on how they quantify the geometric properties. The choice of descriptor is dependent on the requirements of the system in terms of type of operating environment, accuracy, computation speed or personal preference.

The first class of descriptors are the local descriptors and these describe the geometry at localized areas in the point cloud. These descriptors look for repeatable and distinct points in the point cloud. As a result of only describing individual points, this type of descriptor must be calculated at many different points in the point cloud to effectively describe an object. This requires many descriptor calculations causing the descriptor computation to take a while and the data can consume a large amount of memory. To address these issues, the descriptor is generally only computed at points considered keypoints for the specific

descriptor used. A key advantage of local descriptors is their ability to handle occlusion well as they do not require complete views to provide a description that is sufficient for object recognition.

The second class of descriptors are the global descriptors which describes the overall shape of the point cloud by using all the points present. As a result of using all the points, typically only one descriptor needs to be calculated to provide a sufficient description. This yields a lower computation time and lower memory requirement to store the data than local descriptors. Global descriptors generally do not handle occlusion well as the description of an object in full view is different than an object that is partially obstructed. Global descriptors typically handle noise better than local descriptors. Unlike local descriptors, to describe an object using global descriptors that object's point cluster must first be isolated from the point cloud by a process known as segmentation.

2.3 – The Viewpoint Feature Histogram

Given the operating conditions for this project the View Point Feature Histogram (VFH), a global descriptor, is used. This descriptor has been shown to have a high object recognition accuracy, accurate in the presence of noise and can be computed in a small amount of time. The VFH descriptor was developed by [2] and originally consisted of two components, the viewpoint direction component and the extended Fast Point Feature Histogram (FPFH) component. To make the descriptor more robust, the Shape Distribution Component (SDC) was added in the PCL implementation.

The viewpoint component characterizes the surface normal direction relative to the camera's central viewpoint. This is done by first drawing a vector down the z-axis of the camera. The angle between this viewpoint vector and the computed normal at each point cluster is measured. The measured angles are stored in a 128 bin histogram.

The extended FPFH characterizes the relationship of the surface normals and the centroid of the point cluster. This involves first computing the centroid by finding the average of all points in the cluster. As the centroid does not have a normal or any other vector associated with it, one is drawn from the camera's viewpoint to the centroid. Using this vector as a normal for the centroid, a Darboux coordinate frame is constructed as described in [3]. The normals at each point in the cluster has its pan, tilt and yaw angles measured off the computed Darboux frame. The angles are stored in three histograms with 45 bins each.

Originally the SDC was not part of the VFH descriptor, but is taken from the Cluster View Point Histogram (CVFH) descriptor in [4] which was developed as an extension of the VFH descriptor. The SDC looks at the distribution of points around the centroid by measuring their distance from the centroid. This enables the descriptor to differentiate objects with a similar size and point normal distributions but with different point distributions. These distances are stored in a 45 bin histogram.

Concatenating the histograms together, the extended FPFH pan, tilt and yaw (45 bins each), the SDC (45 bins) and the viewpoint component (128 bins) form the full VFH consisting of 308 bins. An example VFH is show in Figure 4.

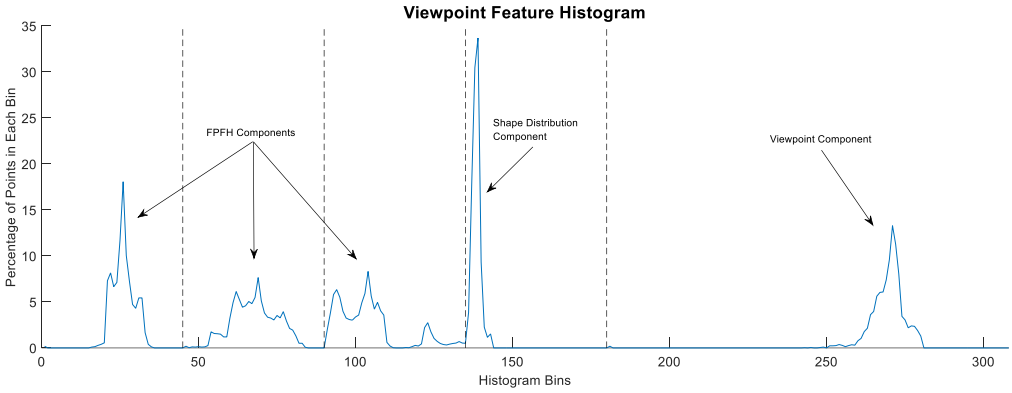

Figure 4: Sample Viewpoint Feature Histogram

2.4 – The Database

For the system to recognize an object in an environment it must compare the description of that object with a description in a database. By comparing the descriptor information of the object in the environment with one in its database it can draw a conclusion as to what that object is based on the similarities of their histograms. As the descriptors only provide information about the object view that is visible to the camera, the database description must contain images from many different views. This requires the database to contain point cloud images from angles all around an object for the system to recognize it at any possible orientation. The database made for this project consisted of 80 views per object.

There are two ways of filling the database with object views, one is to use a program to virtually scan CAD models and the other is to manually scan in real objects using a camera. For this project CAD model scans are used to fill the database and is chosen for its simplicity and increased accuracy over manually scanning objects. This method has several advantages over manually scanning. One being that the point clouds generated this way do not contain sensor noise or measurement errors. Another is that the simulated images do not require segmentation to isolate the object which can be difficult and introduce errors into the point cloud. Finally, this method can be accomplished in a very small amount of time when compared to manually scanning an object.

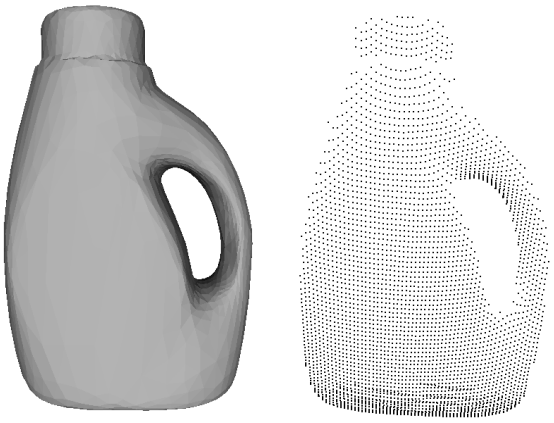

Figure 5: CAD model and a synthetically generated point cloud

3 – Method

This section details the algorithm's process to recognize an object and estimate its pose from a point cloud image. When using a global descriptor like the VFH, this process can be broken into three main steps. The first phase is the preprocessing and segmentation stage to isolate the object clusters and prepare them for the recognition stage. Second is the object recognition phase where the descriptors are calculated and matched against the database. The final phase is estimating the object's pose and verifying that the pose is a reasonable estimate.

3.1 – Preprocessing and Segmentation

The preprocessing phase is an important step in preparing the raw point cloud data for the object recognition phase. The only preprocessing done before segmentation is downsampling the point cloud to reduce computation time of the segmentation and later steps. After the downsampling is complete, the objects are segmented from the scene and each cluster has its surface normals computed. These clusters are then sent to the object recognition phase.

Figure 6: Preprocessing and Segmentation Flowchart

3.1.1 – Downsampling

The first phase of the algorithm begins with receiving a raw point cloud image from the camera. The raw image from the Xtion contains 307200 points of data and using all these points for computations can be a time consuming process. To ease this computational load the point cloud is first downsampled. This is done by breaking the pointcloud into cube-shaped regions called voxels that are of a specified dimension or resolution. All the points that lie in a cube are replaced with a single point that is the average XYZ component of the all the points that lie in the cube.

Choosing an appropriate voxel resolution is dependent on the accuracy of the raw point cloud. The more accurate the point cloud the more it can be downsampled. Based on experimental data from the Xtion camera, the average resolution of the pointcloud was found to be around 3mm. Points closer to the camera had a smaller resolution, around 1.5mm at a distance of 0.75m from the camera, while points farther had a larger resolution, around 4mm at 2m from the camera. The voxel resolution was chosen to be the average pointcloud resolution of 3mm, which reduced the number of points by around 30% for most images. Shown in Figure 7 are the effects of downsampling on the three most time consuming methods in the algorithm, Euclidean Cluster Extraction segmentation, the surface normal estimation and the Iterative Closest Point (ICP) algorithm. Downsampling has a large effect on the segmentation and surface normal estimation, reducing the computation time of both by around 55% when using a voxel resolution of 3mm. This reduction in points had little effect on the accuracy when compared to the results without downsampling.

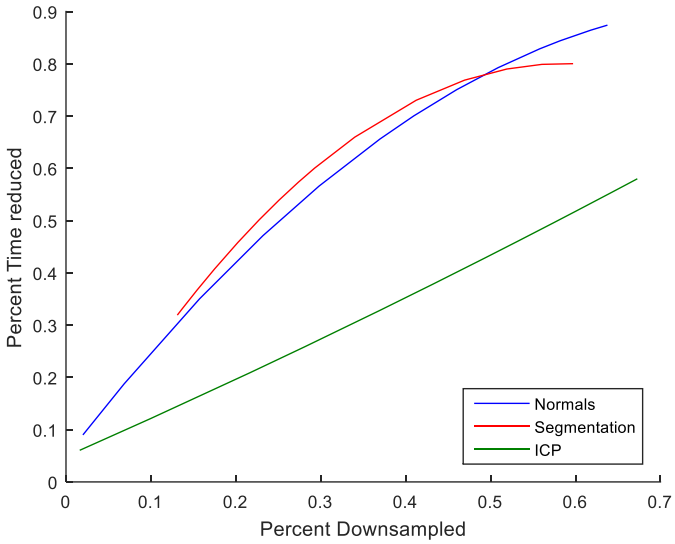

Figure 7: Effect of downsampling on surface normal estimate, segmentation and iterative closest point

3.1.2 – Segmentation

When using global descriptors it is necessary to first segment out the object clusters else the descriptor will describe the entire scene and not the object in the scene. A simple and effective method of segmentation known as Euclidean Cluster Extraction (ECE) is utilized to locate and isolate these object clusters. The ECE does well at segmenting the object clusters for simple scenes with low clutter and when the objects are not occluded. When done, the point clusters that remain are considered potential objects and will be passed on for further processing.

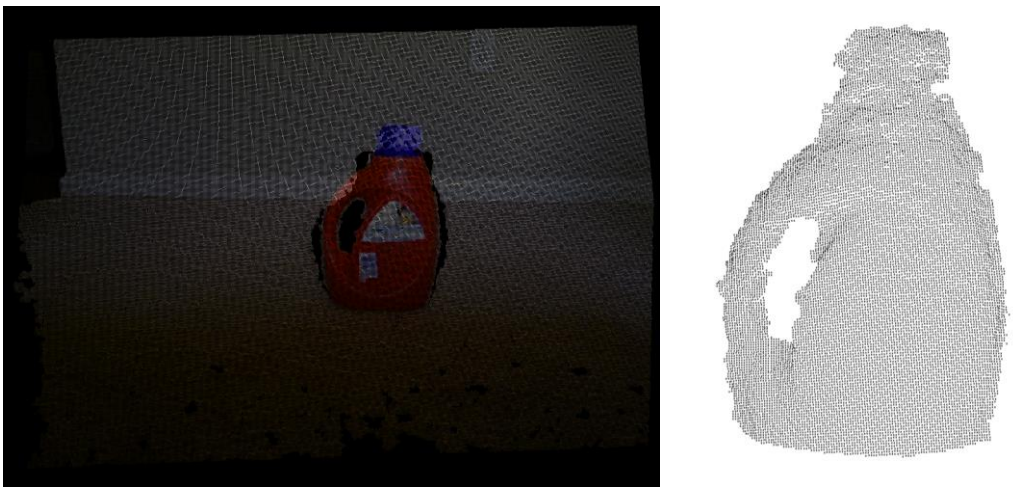

Figure 8: Before and After Segmentation

The implementation for ECE is to locate and remove all large planar surfaces that could be walls, tables, floors, etc. from the point cloud. This is done by first using the RANSAC algorithm to find point clusters that fit a planar model. Points are considered inliers of this model if they are within a specified Euclidean

distance from the plane. The point clusters that fit this planar model are then removed from the scene. As the objects could contain a planar surface, only planar surfaces with a number of points greater than a specified value are removed. This requires some knowledge of the size and shape of the object to ensure that it or parts of it are not removed by this process. Further information on the ECE method can be found in [5].

Before the VFH descriptor can be calculated the surface normals at each point of the object clusters needs to be calculated. The technique used is to approximate a plane that is tangent to the point by using all the points in a radius around the point for which the normal is being estimated. The best estimate of the plane is done by solving a least squares problem described in [6]. After all the surface normals for the object clusters have be computed they are ready for the object recognition phase.

3.2 – Object Descriptors and Matching

This phase deals with describing the object clusters found by the earlier segmentation step. The descriptor is calculated for these clusters and then matched against the database to determine what the object is.

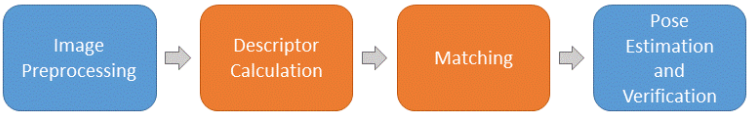

Figure 9: Object Recognition Flowchart

3.2.1 – Descriptor Calculation

After the segmentation process has found potential object clusters in the scene, the system needs to identify what those object clusters are. To this the algorithm must first compute the VFH descriptor for each object cluster that the segmentation step located. For this implementation the bins of the histogram are normalized so the voxel resolution of the object clusters and database clusters do not need to be similar. This does make the VFH scale invariant but this issue is addressed in the pose estimation and verification phase. Once the algorithm has a description of the object it can then identify what the object is.

3.2.2 – Matching

The object is identified by finding the most similar histogram in the database to match the scene cluster histogram. The best possible candidate is determined by the smallest chi squared distance between the scene VFH and one from the database. Ideally, the closest match is the correct view and will be able to provide the correct pose estimation. However, due to noise and measurement errors with the sensor, the nearest match will not always produce a correct pose estimate. To account for this the algorithm searches for the nearest 20 matches in the database. Finding these extra matches does little to effect the time it takes to find the matches as it only takes around 2ms to find 20 matches per object.

3.3 – Pose Estimation and Verification

After all the matches have been found, the poses for each object in the scene can be estimated. This consists of three steps. First, a rough 6DOF pose estimate is obtained through a centroid alignment coupled with the Camera Roll Histogram (CRH) [4] alignment. This is followed by a refinement of the initial pose using the Iterative Closest Point (ICP) algorithm. Finally, the pose is verified for correctness through hypothesis verification.

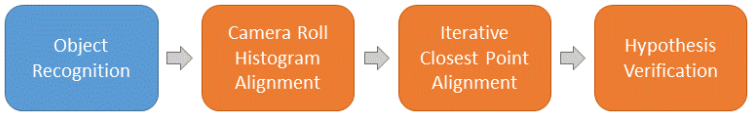

Figure 10: Pose Estimation and Verification Flowchart

3.3.1 – Camera Roll Histogram Alignment

A rough pose estimate is found by computing the centroid of the scene cluster and database cluster by averaging the XYZ components of all the points in the cluster. The database cluster's centroid is then transformed to the scene cluster's centroid. With this transformation and knowledge of the initial pose of the database cluster a 5DOF pose estimate is found. Due to the VFH being invariant to camera roll, a complete 6DOF pose cannot be determined until this roll angle is calculated.

To obtain this roll angle and a complete 6DOF pose for the object, the CRH is computed to determine the roll angle between the scene cluster and the database cluster. Aligning these two histograms will provide the necessary roll angle to complete the 6DOF pose estimate. This is usually a decent estimate of where the object is located and how it is oriented but for most tasks this will not be good enough. Figure 11 shows the results of a CRH alignment that is a decent alignment but not a reasonable pose estimate.

Figure 11: Pose estimate after camera roll histogram

3.3.2 – Iterative Closest Point Alignment

Refining the pose estimation is done by using the ICP algorithm. The ICP will iteratively align the two point clusters as closely as possible and will terminate after an error threshold has been met or the max iteration limit is reached. ICP can be a time consuming task and therefore both the scene cluster and the database cluster are downsampled to a voxel resolution of 5mm to reduce this computation time. Figure 12 shows the alignment of the same point clouds from Figure 11 after the ICP alignment.

Figure 12: Pose estimate after ICP alignment

3.3.3 – Hypothesis Verification

The final stage of pose estimation is to verify that the correct database cluster was matched and that the alignment of the two point clusters are correct. This is done by using the Global Hypothesis Verification method developed by [7] to validate that the two point clouds are within a specified tolerance. If this tolerance is not met, the algorithm moves on to the next nearest match in the database and repeats the pose estimation process. If the tolerance is met the estimated pose is considered correct and the algorithm moves on to the next scene cluster and does the pose estimation process for that cluster. Figure 13 shows a rejected pose estimate on the left and an accepted pose on the right.

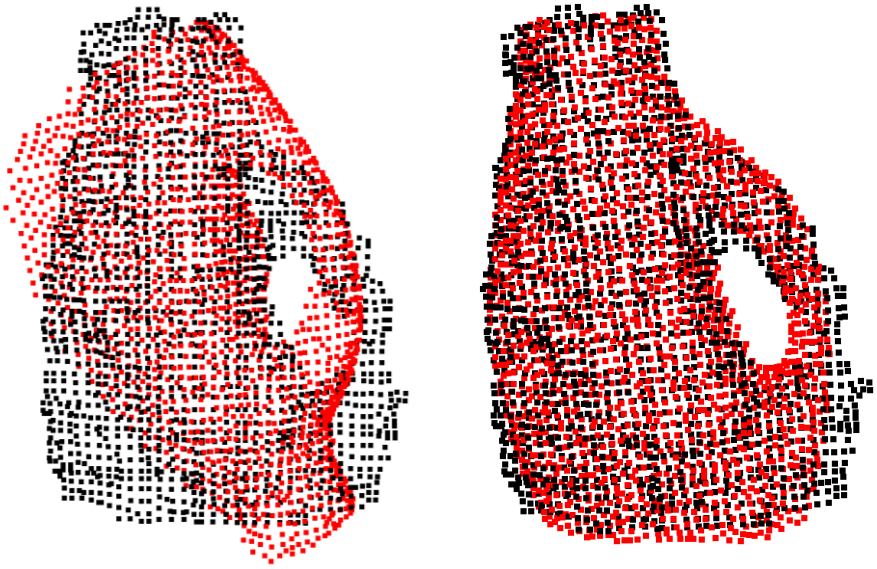

Figure 13: Reject and accepted pose estimate

4 – Results

The algorithm was tested using real data acquired with the Xtion camera and evaluated based on two criteria. First, the algorithm was tested to determine the success rate of the pose estimation at varying distances from the camera. Second, the algorithm was tested to determine how quickly a successful pose was obtained. A focus was put on determining the operating distance at which the algorithm was capable of providing a reasonably high success rate for the pose estimation.

4.1 – Experimental Setup

The setup used consisted of Xtion camera placed in a stationary position and an object, in this case a bottle of Tide laundry detergent, was placed at distances of 0.75m, 1m, 1.25m and 1.5m from the camera. This distance was measured from the camera's lens to the nearest face on the object. At each distance the object was placed in 20 different orientations and a snapshot of the scene was taken with the camera. At each of the four distances the same 20 orientations were used.

Figure 14: Experimental Setup

The bottle of Tide has the dimension shown in Figure 15 and was chosen for its size being similar to what a robot might be interacting with and varying distinctness of features when viewed at different angles. When viewed from the front or back the features are more distinct than those being viewed from the sides.

Figure 15: Dimensions of Tide

4.2 – Pose Success Rate

The first evaluation was concerned with determining how well the algorithm was able to estimate the pose of an object at varying distances from the camera. The pose estimation rate was based on how many of the 20 snapshots the system was able to correctly estimate the object's pose. A successful estimate was determined by a pass or fail as given by the hypothesis verification. The results of the trials are displayed in Table 1. Each pass/fail was visually inspected to insure there were no false positives. For all the trials run there were no false positives but there were a few false negatives at greater distances, one at 1.25m and four at 1.5m.

Table 1: Correct pose estimation rate at varying distances

Distance from camera	Success Rate
0.75m	90%
1.00m	95%
1.25m	90%
1.50m	60%

The algorithm at the 0.75m, 1m and 1.25m distances was able to estimate the pose at all orientations except for the one shown in Figure 16. From this camera angle the object's point cloud has many incomplete sections making it difficult for the VFH to effectively describe and match the object.

Figure 16: Failed pose estimation due to missing sections

At a distance of 1.5m the success rate drops of considerably. This can mainly be attributed to the sensor noise and measurement errors of the sensor. Of the 20 views four were deemed unsuccessful by the algorithm but were in fact correct when visually inspected. These false negatives, similar to the one in

Figure 17, show that sensor artifacts around the edge of the object became a large problem by producing many erroneous points leading the hypothesis verification to deem the pose estimate incorrect.

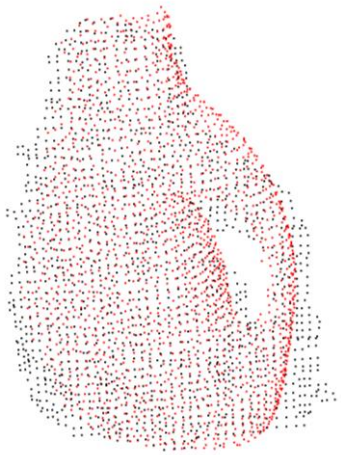

Figure 17: False negative at 1.5m

4.3 – Speed

Two factors were looked at when evaluating the speed of the algorithm. The first was how long each stage of the algorithm took and the second is how many database matches were required before a successful pose estimate was found.

Each stage was timed and averaged out over all the trials with the results shown in Table 2. The ICP alignment and hypothesis verification time in the table only represent a single run and does not take into account if multiple matches are required to estimate the pose. By a large margin the segmentation and alignment and verification stages take the most amount of time. Assuming the system determines the pose on the first match, the algorithm takes 3.21s to estimate the pose. This is far too slow for real-time applications but may be acceptable in some applications.

Table 2: Algorithm stage times

Stage	Time (ms)
Segmentation	1003 ± 106
Surface Normal Estimation	223 ± 25
VFH Computation	6 ± 1
Descriptor Matching	3 ± 1
CRH Alignment	5 ± 1
ICP Alignment and Hypothesis Verification	971 ± 79

The second factor determining the runtime of the algorithm was how many database matches were required for the pose estimation. For every match that the algorithm has to check, the alignment and verification stage must be run again. Figure 18 Shows that the farther the objects is from the camera the more matches area required for a successful pose estimate. Even at a close range the algorithm cannot reliably produce a pose estimate with the first database match.

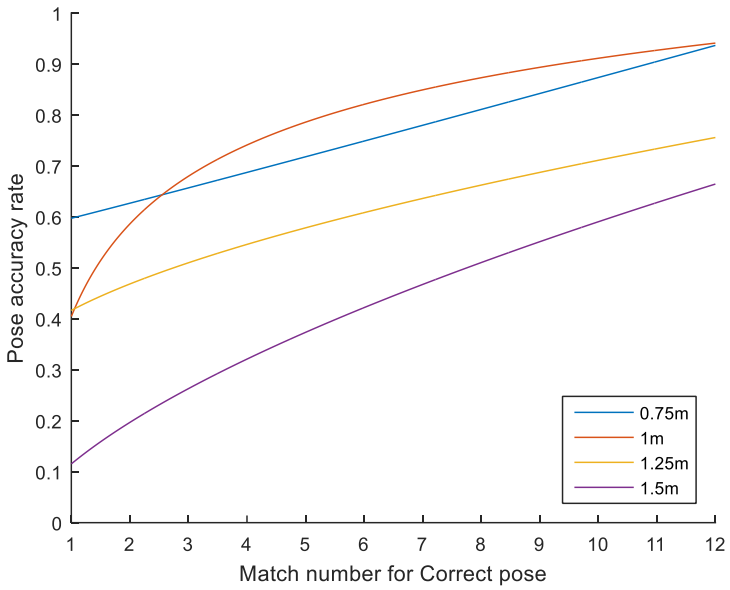

Figure 18: Rate of pose accuracy at increasing database matches

5 – Conclusion

The algorithm presented has been shown to be an effective way of determining the pose of an object when using the ASUS Xtion camera. However, the Xtion camera does impose limitations on the operational ranges due to the noise and measurement errors from the camera. Improvements could be made that would increase the accuracy of the VFH which should increase the operational range as well as reduce the number of matches needed to achieve a successful pose estimation. While modifications and optimizations to the algorithm could improve computation time, it is unlikely to be enough for real-time applications but could be considered adequate for some.

References

[1] S. C. Radu Bogdan Rusu, "3D is here: Point Cloud Library (PCL)," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Shanghai, 2011.

[2] G. B. R. T. J. H. Radu Bogdan Rusu, "Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, 2010.

[3] N. B. M. B. Radu Bogdan Rusu, "Fast Point Feature Histograms (FPFH) for 3D Registration," in *ICRA*, 2009.

[4] M. V. Aitor Aldoma, "CAD-Model Recognition and 6DOF Pose Estimation Using 3D Cues," in *IEEE International Conference on Computer Vision Workshops*, Barcelona, 2011.

[5] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," Unpublished, München, Germany, 2009.

[6] T. C. Jens Berkmann, "Computation of Surface Geometry and Segmentation Using Covariance Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 16, no. 11, pp. 1114-1116, 1994.

[7] F. T. L. D. S. a. M. V. Aitor Aldoma, A Global Hypotheses Verification Method, Springer Berlin Heidelberg, 2012.