Viewpoint Optimization for grasping unknown objects using Principal Component Analysis

Directed Research Report - Fall'19 & Spring'20

Vamshi Krishna Uppununthala

***Abstract*—Grasping unknown objects is one of the crucial problems in the domains of waste segregation and recycling. Due to the unknown features of the object, the problem's complexity increases with the cluttering in the environment. This missing object's information is directly related to the vision sensor viewpoint. As there are many research works which have shown different grasp synthesis algorithms to synthesize a grasp just by using single image of the target object and making assumption about the object's model, this research concentrates on finding the best viewpoint as quickly as possible. Here, in this report, as a part of Directed Research in Fall'19, I propose and test a heuristic based approach to find the best view point with no prior information about the target object. The approach makes the robot move on the view sphere in the direction pointed by the least principal axis to explore in a sequence of steps. A finite state machine is implemented in this work to test the functionality of the approach. This heuristic based approach aims to remove the dependence on training data of Reinforcement Learning(RL) based algorithms. As a part of Directed Reserach in Spring'20, I continued the work to compared the PCA based viewpoint optimization to normal exploration from various orthogonal views of the object which is spawned in different poses.**

I. Introduction

Grasp synthesis algorithms help in grasping objects without the prior information of the target object. One of the options is to reconstruct the total object using multiple viewpoints[1]. But this approach is not efficient as it's not practical to regenerate the target object's shape and using this information to find grasping regions. There are different methods developed in the literature like feature-based techniques[2], symmetrical objects[3]' based etc. to solve the grasp synthesis problem with partial shape information. Most of these works make assumptions on shape based on the single image of the target object.

The significance of find the best viewpoint is clear from the above mentioned works. In this work, a heuristic based approach using Principal Component Analysis(PCA). Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. The robot is made to move in the direction as the principal axis with least principal component. The idea is to make the robot explore in the direction with least data variance and hence helping to better find the grasping region. The approach's functionality is tested in Gazebo Simulator with Franka Emika Panda robot

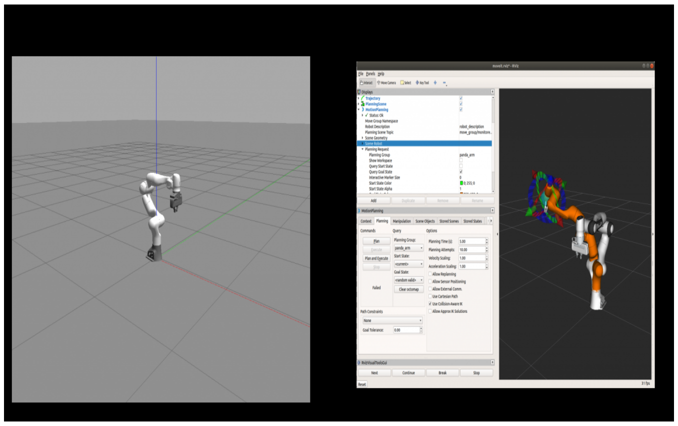


Fig. 1. Panda robot in Gazebo and RViz

model to find the grasping region of bowl object. The panda robot model is modified to have depth sensor at the end effector with identity transformation between them.

Subsequently, a state machine is implemented to carry out tasks including segmenting the object, pointcloud processing to find the grasp regions and commanding the robot to next state based on the proposed heuristic approach. The state machine and the flow of steps is explained in section III.

II. Related Work

Paper [1] uses the depth sensor of the robot to optimize the viewpoint of the sensor for increasing the quality of the synthesized grasp. A two manifold approach is used. Firstly, to generate the training data, a random search algorithm is implemented. The second step involves application of a reinforcement learning policy over this training data to synthesize a grasp with a better quality and the least amount of resources. The methodology is based on a simple force moment based grasp synthesis algorithm. The algorithm was successful in achieving a grasp in 94% of the cases in which the initial viewpoint was insufficient to synthesize a grasp.

Paper [2] by Calli et al. introduces a novel algorithm for grasping of unknown objects using curvature optimization achieved through active vision. The key features of their algorithm are that it does not require any 3D model, and also does not assume any models of the object. The observed object edge curvature is modeled using Elliptical Fourier Descriptors. Once, curvature maximizing points are found, visual servoing is used to reach those points. An important property of their algorithm to be noted is that as it might not find the optimal curvature maximizing points with initial view, it looks for better grasp points in the nearby region while executing and updates the same.

The work presented in the paper [1] closely relates to this work. The single view partial point cloud is processed to get the pointcloud of the target object. After principal component analysis on the target object, the grasping algorithm checks for the grasping region by optimizing total force on the points of consideration in the object frame. The solution proposed for the exceptional case provides information about dealing with the cases where no grasping region is found with the first view-point. The algorithm suggests the robot to move in the direction parallel to the main plane (plane with most data points on the target object from a particular view) if the width of the main plane is greater than the graspable range of the gripper else, the robot would move perpendicular to the main plane. The work presented did not clearly mention the case if the main plane does not actually represent the main plane of the target object. There was no discussion about exploration as this was viewed as an exception but this case would be most common for complex objects.

Paper [2] discussed about combining vision and robot hand real-time grasp control action to achieve reliable and accurate object grasping in a cluttered scene. An efficient online algorithm for collision-free grasping pose (data structure to represent position and orientation) generation according to a bounding box is proposed, and the grasp pose will be further checked for grasp quality. Finally, by fusing all available sensor data appropriately, an intelligent real-time grasp system was achieved that is reliable enough to handle various objects with unknown weights, friction, and stiffness. The work assumes that the bounding box generated using Principal Component Analysis can be graspable if the bounding box fits the gripper. This contradicts with most of the daily used objects like pitcher, mug and in fact all the objects with handles as the objects as a whole do not fit the gripper but the handle does.

III. Methodology

A. Grasping region detection:

This section implements the force balance based algorithm mentioned in [3]. Towards achieving the grasp quality, initially, the pointcloud of the target object is down-sampled in the first stage to minimized the number of points and hence the computation time. The pointcloud is convoluted with a voxel grid of leaf size 0.01^3. The surface normals of the down-sampled pointcloud are found. The work by [4] shows that the grasping region can be identified as a region using force balance optimization where the surface normals of two points are with an angle of 0 or 180 degrees. Consequently, the distance vector for the two points of consideration is also found to check if that fits the gripper width. Finally, the angle between distance vector and one of the normals is found. If this also is closer to 0 or 180 degrees, we can say that the two points constitute to a grasping region.

B. PCA for calculating next action:

1) Principal Component Analysis: PCA is often used in classification and compression techniques to project data on a new orthonormal basis in the direction of the largest variance. The direction of the largest variance corresponds to the largest eigenvector of the covariance matrix of the data, whereas the magnitude of this variance is defined by the corresponding eigenvalue.

Therefore, if the covariance matrix of two point clouds differs from the identity matrix, a rough registration can be obtained by simply aligning the eigenvectors of their covariance matrices.

In the following, let A be the covariance matrix, let v be an eigenvector of this matrix, and let λ be the corresponding eigenvalue. The eigenvalues decomposition problem is then defined as:

$$Ax = \lambda x \tag{1}$$

and further reduces to:

$$x(A - \lambda I) = 0 \tag{2}$$

It is clear that 2 only has a non-zero solution if $A - \lambda I$ is singular, and consequently if its determinant equals zero:

$$det(A - \lambda I) = 0 \tag{3}$$

The eigenvalues can simply be obtained by solving Eq. 3, whereas the corresponding eigenvectors are obtained by substituting the eigenvalues into Eq. 1.

PCA was used to find the approach vector when grasping objects as mentioned in the work of [5]. In this work, PCA is used to find the data variance along the principal axes. The axis with least data variance will be chosen as the direction in which the end effector would be moving.

Figure 2 demonstrates how the next target point of the end effector is being calculated. $\vec{m}$ is the vector with least principal component value. $\vec{v}$ is the vector joining the centroid of the object O and end effector point E. The cross product of $\vec{v}$ and $\vec{m}$ gives an axis vector with which E has to be rotated with a constant step angle θ in each step of the exploration.

The resultant point after the rotation $\tilde{E}$ is normalized to make the point reside on a view sphere of unit radius.

2) State machine - Overview: The process of acquiring pointcloud data to checking if the pointcloud has a grasping region can be represented as a state machine as shown in Figure 3

In the first stage, all the gazebo and RViz simulation environment is setup. From the depth sensor placed at the end effector, pointcloud P is captured. P is fused or stitched to the existing point cloud of the target object (if any) to get new pointcloud F. Grasping regions are detected using F. If there exists a grasping region, we quite the state machine. Else, we go on to find the next target pose of the end effector using the approach mentioned in III-B. The robot is then commanded to move to the target pose using MoveIt.

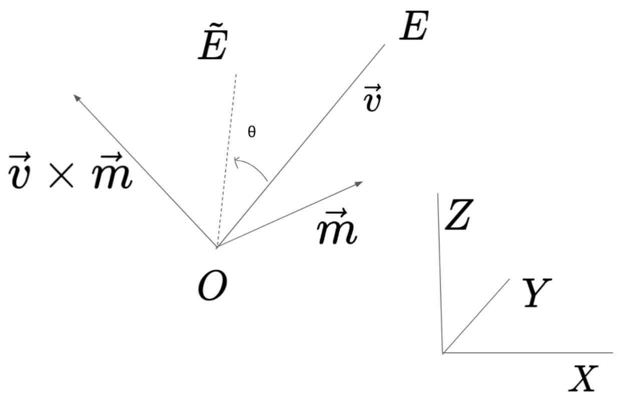


Fig. 2. PCA based approach for calculating next action

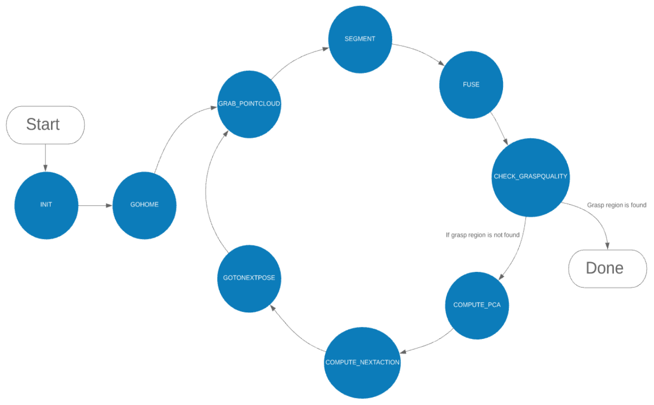


Fig. 3. State machine of the implementation

C. *Fixed path exploration*

A generic exploration schema is used to compare the PCA based heuristic's performance. The exploration path consists of four view points through with the object is viewed. The object is assumed to be enclosed in an imaginary rectangular prism. The vectors from center of base to corners on the top of rectangular prism are used as directional vectors. The length of the vector need to be adjusted as per the object. A state machine is implemented to make the camera on the end effector move through all of these points in order and to check if the grasping region is found from any of these points. The idea is that these points should almost build the object's model and hence it would be the best baseline to be used against the PCA based heuristic.

1) State machine-Overview: The state machine similar to the one mentioned in section III-B2 with the approach mentioned in III-B is replaced with the hard-coded poses calculated as shown in Figure 4. After grasp region detection or if all the four orthogonal views are visited, the model pose is automatically changed. The 10 quaternion poses as shown in Figure 5 are spawned sequentially. The steps taken to find a grasping region for each of the model poses is logged for later analysis.

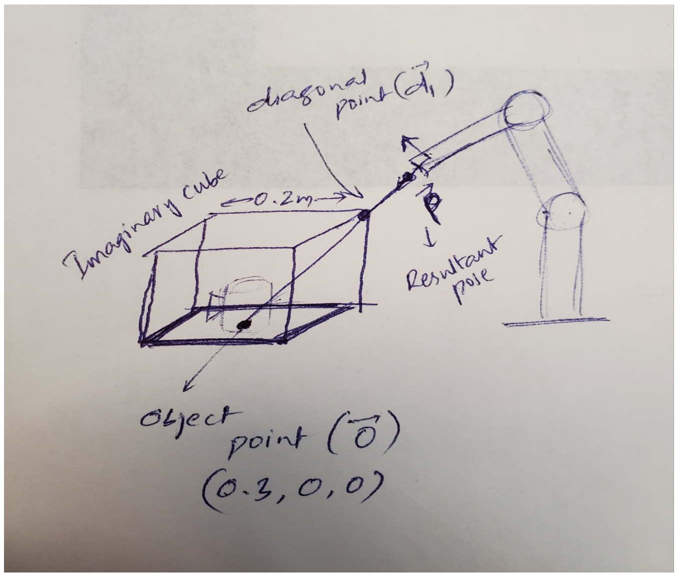
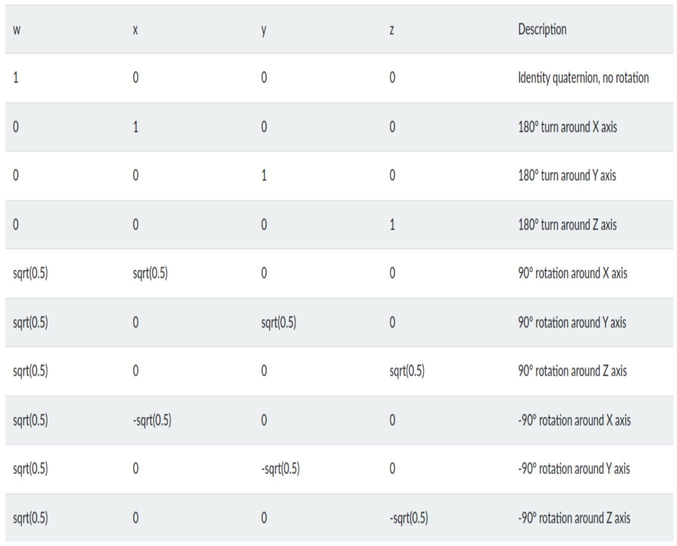


Fig. 4. Orthogonal view calculation



w	x	y	z	Description
1	0	0	0	Identity quaternion, no rotation
0	1	0	0	180° turn around X axis
0	0	1	0	180° turn around Y axis
0	0	0	1	180° turn around Z axis
sqrt(0.5)	sqrt(0.5)	0	0	90° rotation around X axis
sqrt(0.5)	0	sqrt(0.5)	0	90° rotation around Y axis
sqrt(0.5)	0	0	sqrt(0.5)	90° rotation around Z axis
sqrt(0.5)	-sqrt(0.5)	0	0	-90° rotation around X axis
sqrt(0.5)	0	-sqrt(0.5)	0	-90° rotation around Y axis
sqrt(0.5)	0	0	-sqrt(0.5)	-90° rotation around Z axis

Fig. 5. Quaternion poses for spawning object

IV. Implementation

This section covers the implementation details being put into each state of the state machine. Robotic Operating System(ROS) is used as a platform with Gazebo simulator as physics engine and RViz to visualize the robot motion and point clouds. The major chunk of implementation is done using ros services and then eventually calling these service from the agent application. In fixed path exploration, after each iteration, the model's pose is changed based on the predefined quaternion poses (Figure 5).

A. *INIT:*

In this state, the environment is set up. A robot is spawned in the Gazebo simulator environment with an object (e.g. bowl)

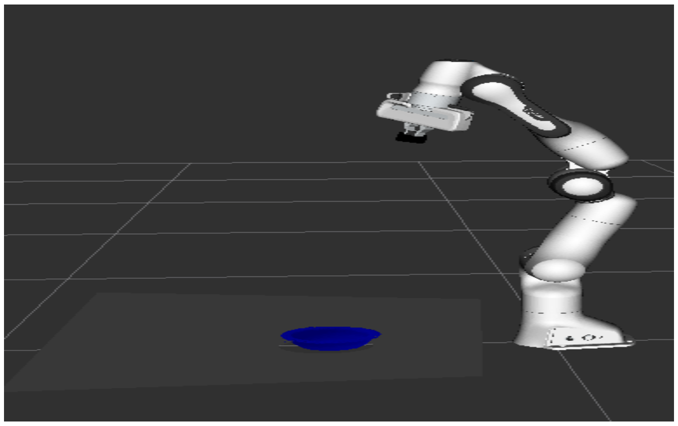


Fig. 6. Home position during viewpoint optimization

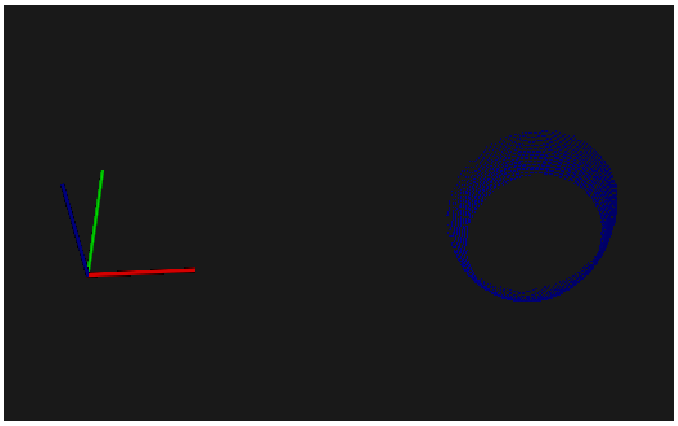


Fig. 7. Target object's pointcloud after segmentation

is place at a distance of 0.3m from the base of the robot manipulator (Franka Panda robot here) to be able to make the object viewable from all the points on the view sphere of radius 0.5 with the center as the centroid of the target object.

Also, the depth camera in the end of the robot is initialized. The tf tree [6] handle is also started to be able to access the transformation tree between all possible co-ordinate links.

B. GOHOME:

The robot is commanded to move to home position in this state. The home position is defined to be having co-ordinates of (0.14, 0, 0.94) metres in the world frame. If $\vec{v}$ is the vector joining the centroid of the object O and End effector point E, the objective is to find the orientation of the end effector frame such that the camera directly faces towards O. O in this state is considered to be the place of spawning the object but in later sections, O is considered to be the centroid of the acquired pointcloud of the target object from different viewpoints.

In this regard, $\vec{v}$ is considered to be the R_z in calculating rotation matrix R for the end effector frame as mentioned in Equation. 4. The calculation of R_z, R_x and R_y follows equations 5, 6 and 7.

$$R = \begin{bmatrix} R_x & R_y & R_z \end{bmatrix} \tag{4}$$
$$R_z = \vec{v} \tag{5}$$
$$R_x = R_z \times \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{6}$$
$$R_y = R_z \times R_x; \tag{7}$$

For path and motion planning of the robot manipulator, ROS MoveIt [7] is used throughout the project. The resultant home position is shown in 6.

C. GRAB_POINTCLOUD:

This state grabs the current pointcloud and be stored in a buffer to be processed. This instance of the pointcloud data will be used in the next stages for identifying grasping regions. The rest of the stream of pointcloud data will be ignored when the agent is in other states.

D. SEGMENT:

The raw pointcloud which might have unwanted points along with the target object. This state deals with segmenting the target object. The raw data of a point cloud contains numerous useless points. Therefore, a pass-through filter is used to set a range over the 3D space so that points within that range are kept unchanged, and points outside of the range are removed.

A point cloud might contain points that provide no additional information due to noise or inaccuracy of the capture. On the other hand, when there are too many points in a point cloud, processing can be computationally expensive. Hence, the points are reduced by the process of down-sampling using a voxel grid of size $0.02x0.02x0.02$ after transforming the pointcloud to world frame.

The target object is placed on the flat surface. In a point cloud representation of a scene, a random sample consensus [2] can help to distinguish flat surfaces. Isolating a point cloud cluster that represents the unknown object is done by removing the entire points on the flat surface. After the planar model has been found, we can easily remove the ground and keep those objects on the ground.

Figure 7 shows the segmented bowl object.

E. FUSE:

In each step, the target object's pointcloud has to be combined/stitched with the already seen pointcloud of the target object. To fuse the pointcloud, the pointcloud is transformed to the common or static frame(world frame here). With the previously fused pointcloud and current pointcloud in the same co-ordinate frame, the points of the current pointcloud can be directly appended to the fused pointcloud. The newly formed pointcloud can be used as target object's fused cloud in the subsequent steps. In figure 8, the pointcloud of bowl object is got after two steps of fusion.

F. CHECK_GRASPQUALITY:

To be able to find the quality of grasp, several things have to be done in our grasp synthesis algorithm. The first thing

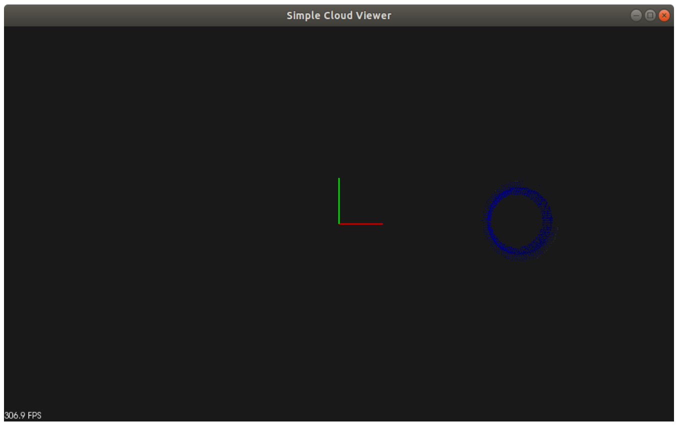


Fig. 8. Fused target object's pointcloud in Step 2

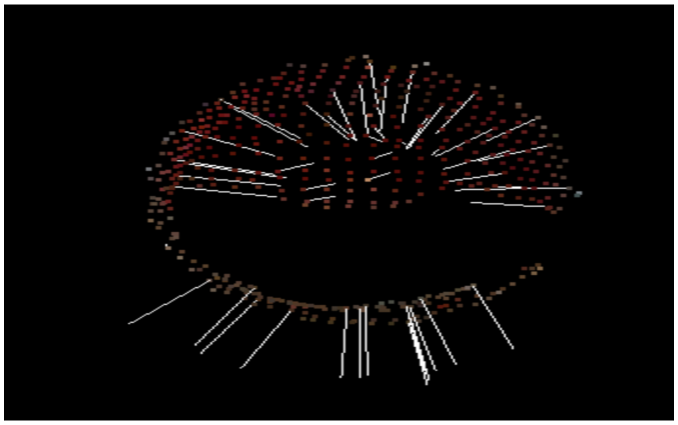


Fig. 9. Object's pointcloud with surface normals

that that our algorithm accomplishes is to calculate the surface normals of the given point cloud. To do this, we use the Point Cloud Library one of their built in functions that compute the surface normals. Afterward, we use the calculated surface normals to determine a grasping point. The grasping points are determined by calculating the angle between two particular normals. The closer the angle between the normals is to zero or one hundred-eighty the more aligned the two potential grasping points are. A tolerance angle is set to determine if we should look at those two points more thoroughly. The tolerance angle is set at such a value that we might want to consider the grasps due to the fact we might not get a perfect zero degrees. Once we determine that we want to look at the point we need to look to see if we can reach the point and if there would be any collisions. 9 shows some of the surface normals that are calculated by the algorithm.

The final measure of grasp quality is measured as the sum of angle between normals and angle between one of the normals and the distance vector between the two points of consideration. A threshold of 300 degrees is used to check if the grasp quality is good enough to categorize the points as the grasp region. If the grasp region is found, the state moves to Done. Else, the robot continues to find the grasp region as explained in Fig. 3.

G. COMPUTE_PCA:

To follow the approach mentioned in III-B, the principal component analysis is done here in this state on the fused pointcloud to find the eigen vectors in 3 directions. To efficiently calculate this time consuming operation, the PointCloud Library(PCL)'s MomentofIntertiaEstimation module is used. The eigen vector with least eigen value will be termed as minor vector in subsequent sections. The normalized minor vector would be returned from this state to be used in further sections.

H. COMPUTE_NEXTACTION:

For computing the next target position based on the minor vector, the approach mentioned in III-B is used to transform current camera co-ordinates to new target by applying a rotation. The axis angle rotation is done with a step angle of 10 degrees in each step as shown in Figure 2. The computed target position is normalized and multiplied with 0.5 to make the point reside on the virtual view-sphere of 0.5 radius with center at O. To acquire the current co-ordinates of the end effector, forward kinematics is used from the tf tree information.

To calculate the orientation to be maintained by the end effector, the approach of finding the orientation as mentioned in Section IV-B.

In fixed path exploration, the orthogonal viewpoints are used sequentially to be given as next pose.

I. GOTONEXTPOSE

With the new target pose determined, the robot is commanded to move to the target pose. A planning time of 10 seconds is allowed for the MoveIt software package and a maximum of 3 attempts is allowed for planning. If the plan couldn't be found, it might be mostly because the target pose falls out of the workspace. To handle this situation, the virtual view sphere is made to be 0.5m.

J. DONE

The state machine ends at Done if the suitable grasp region has been found out during the process. During Done, all the ROS services are closed and ROS is shutdown.

V. RESULTS & ANALYSIS

The approach is implemented using the state machine in Gazebo Simulator. The bowl object finds the grasping region from end effector's home position itself most of the times. In some cases, the bowl object's grasping region takes two steps. The bowl object in step2 looks as shown in figure 8.

The experiment on Pitcher object is also being done. The approach takes two steps to find a grasping region. Figure 10 shows the fused pointclouds of pitcher object in both the steps.

The fixed path exploration is implemented with model poses changed at each iteration as shown in Figure **??**. The number of steps taken to detect a grasping region against each of the quaternion pose of the model can be seen in Figure **??**

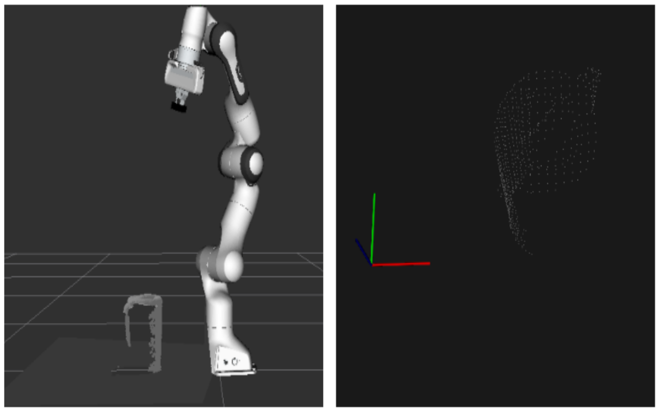


Fig. 10. Picture showing results with pitcher object. left: Home position with pitcher object. right: pointcloud of pitcher object from home position.

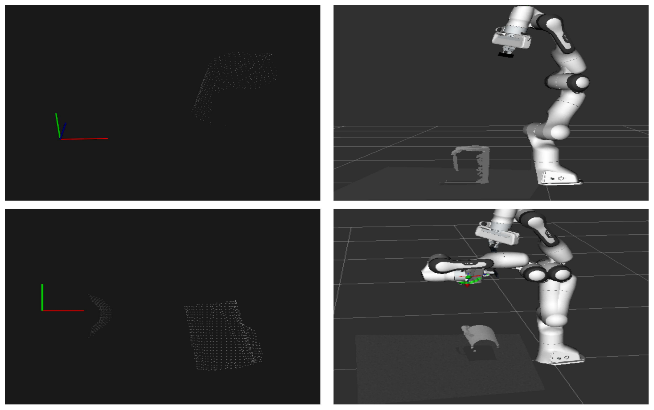


Fig. 11. Picture showing results with pitcher object. Top left: Home position with pitcher object. Top right: pointcloud of pitcher object from home position.Botton left: Home position with pitcher spawned at different pose. Bootm right: pointcloud of pitcher object in botton left's view

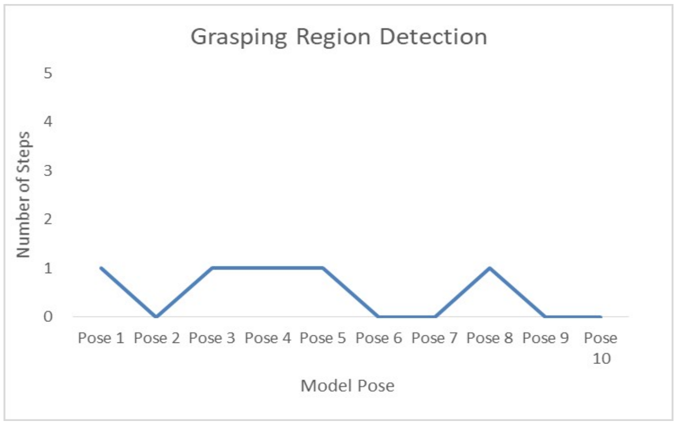


Fig. 12. Picture showing the number of step taken to detect the grasping region with various model poses

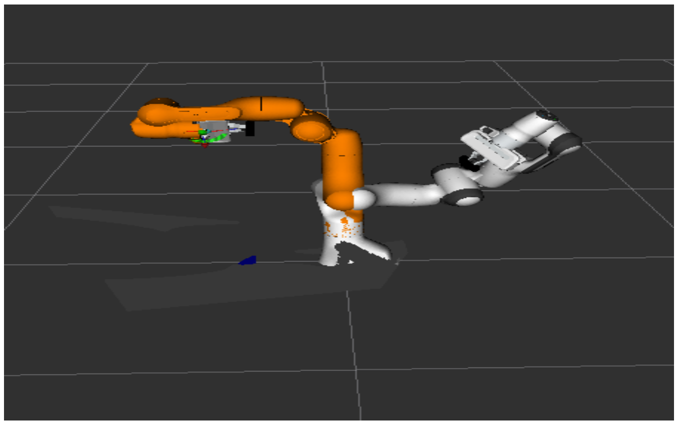


Fig. 13. Scenario where manipulator obstructing the view of target object

VI. Challenges

During segmentation, the pointclouds of smaller objects (like knife, sugar bowl etc.) are not properly getting registered and hence, the detection not being done. With varying leaf sizes and time complexities of handling heavy pointclouds needs to be yet considered to process these kind of objects. During the viewpoint optimization, the robot might reach a state where it's link obstruct the view of the target object as shown in Fig. 13 There is no solution yet when these cases are encountered except to restart the process.

One of the major challenges when implementing fixed path exploration is that the object gets disappeared and surprisingly, leaving a shadow after multiple model pose changes as shown in Figure 14. The problem got fixed by changing the depth camera plugin to realsense camera plugin. The segmentation fault runtime error is solved by reducing the extra visualisations of pointcloud viewer.

VII. Conclusion & Future Work

The project involved implementing the states mentioned in the state machine. The proposed heuristic approach of viewpoint optimization is being implemented and tested on objects like bowl and pitcher. The project extensively uses PCL libraries along with ROS MoveIt for robot manipulation.

The fixed path exploration has got the features of automatic model pose changes with logging. The same needs to applied to PCA base heuristic on a large dataset for one on one comparison. Further this can be compared against RL based approaches.

References

[1] Q. Lei, G. Chen, and M. Wisse, "Fast grasping of unknown objects using principal component analysis," *Aip Advances*, vol. 7, no. 9, p. 095126, 2017.

[2] S.-Q. Ji, M.-B. Huang, and H.-P. Huang, "Robot intelligent grasp of unknown objects based on multi-sensor information," *Sensors*, vol. 19, no. 7, p. 1595, 2019.

[3] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5062–5069.

[4] Q. Lei and M. Wisse, "Fast grasping of unknown objects using force balance optimization," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2454–2460.

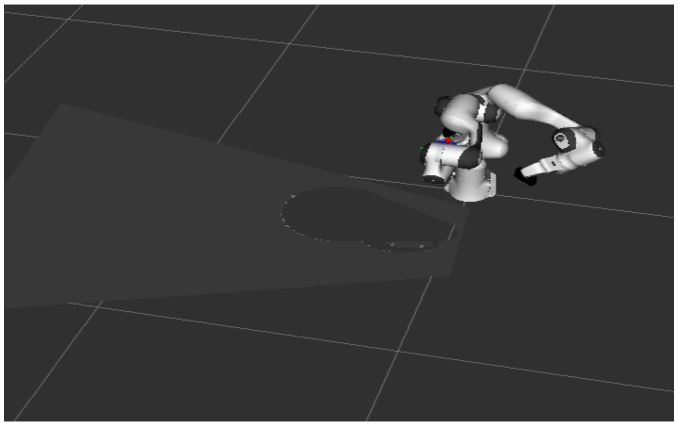


Fig. 14. Scenario where invisible object with shadow

[5] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales, “Mind the gap-robotic grasping under incomplete observation,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 686–693.

[6] T. Foote, “tf: The transform library,” in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–6.

[7] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ros topics],” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.