CSE591 Perception in Robotics
3D Object Reconstruction and Grasp Pose Detection

Chirav Dave, Diptanshu Purwar, Kalyan Ainala, Natasha Mittal
{cdave1, dpurwar, kainala, nmittal4}@asu.edu

Abstract

In the project, we implemented an end-to-end framework for 3D object reconstruction and grasp pose detection. For 3D object reconstruction, 3D-R2N2[2] is used where the neural network takes in one or more images of an object instance from arbitrary viewpoints and outputs a reconstruction of the object in the form of a 3D occupancy grid. This 3D occupancy grid is then visualized in RVIZ (ROS[7] Visualization) by converting the occupancy grid to point cloud data using Point Cloud Library(PCL)[8] in ROS[7]. Grasp Pose Detection (GPD)[6, 3] framework is used to predict a large set of 6-DOF grasp candidates and then each of them is classified as a good or a bad grasp. We conducted the experiments by taking real-life object images using Microsoft Kinect from arbitrary viewpoints, then executed 3D-R2N2[2] network on these images to obtain a 3D voxel space for visualization and generated candidate grasp poses for the same.

1 Introduction

Our project integrates both 3D Object Reconstruction and Grasp Pose Detection which are challenging problems involving perception, planning, and control. Whether it is object reconstruction or grasp planning, both require a tremendous amount of feature engineering.

Most of the state-of-the-art methods for 3D object reconstruction are subject to a number of restrictions, such as objects must be observed from a dense number of views and views must have a relatively small baseline. This is an issue when users wish to reconstruct the object from just a handful of views or ideally just one view.

In grasp planning, humans can grasp and manipulate an object with great ease, but asking a robot with multi-fingered hand to perform a simple task is not a trivial work because it relates to kinematics, motion planning, force-closure grasp, optimization of grasp forces and finger gaits etc. Also, the number of degrees of freedom (DOF) of a robotic hand creates a large number of hand configurations.

But recently, Deep Learning has demonstrated state-of-the-art performance in a wide variety of tasks, including visual recognition, audio recognition, and natural language processing. These techniques are especially powerful because they are capable of learning useful features directly from both unlabeled and labeled data, avoiding the need for hand-engineering.

3D-R2N2[2] network outperforms the state-of-the-art methods for single view reconstruction, and enables the 3D reconstruction of objects in situations when traditional SFM/SLAM methods fail.

GPD[6, 3] framework infers an optimal grasp for a given object which maximizes the chance of successfully grasping the object.

Our goal in this project is to develop an end-to-end system for 3D object reconstruction and grasp pose detection. Our framework is divided into two stages. In the first stage, we collect live images of an object from multiple views using Kinect sensor for which we want to detect grasping positions.

After this we feed these images as input to the 3D-R2N2[2] network which then generates a 3D reconstructed shape of the object in the form of a 3D occupancy grid. In the final stage, we convert this occupancy grid into a point cloud format to visualize the same in RVIZ and then run a ROS[7] library called as Grasp Pose Detection(GPD)[6, 3] on this point cloud to get the grasping positions.

2 Approach

Our project is divided in two main modules: Perception Module where 3D-R2N2[2] is used for object reconstruction and Control Module where GPD[6, 3] is used for pose detection. In section 2.1, an overview of 3D-R2N2[2] network architecture is provided along with the steps followed for obtaining 3D occupancy grid of the objects. In section 2.2, an overview of GPD[6, 3] is provided along with the steps followed for generating candidate grasp poses.

2.1 Perception Module

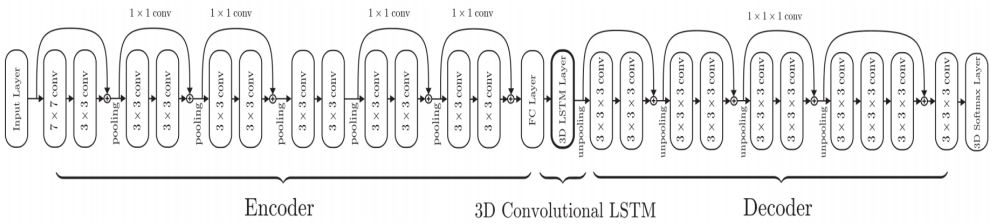


Figure 1: 3D-R2N2[2] Network Architecture

3D-R2N2[2] network is made up of "three components: a 2D Convolutional Neural Network (2D-CNN), a novel architecture named 3D Convolutional LSTM[4] (3DLSTM), and a 3D Deconvolutional Neural Network (3D-DCNN) as shown in Figure 1. Given one or more images of an object from arbitrary viewpoints, the 2D-CNN first encodes each input image x into low dimensional features T(x). Then, given the encoded input, a set of newly proposed 3D Convolutional LSTM[4] (3D-LSTM) units either selectively update their cell states or retain the states by closing the input gate. Finally, the 3D-DCNN decodes the hidden states of the LSTM[4] units and generates a 3D probabilistic voxel reconstruction."[2]

Figure 2 presents an overview of the first stage of our framework where we used 3D-R2N2[2] to generate 3D object reconstruction.

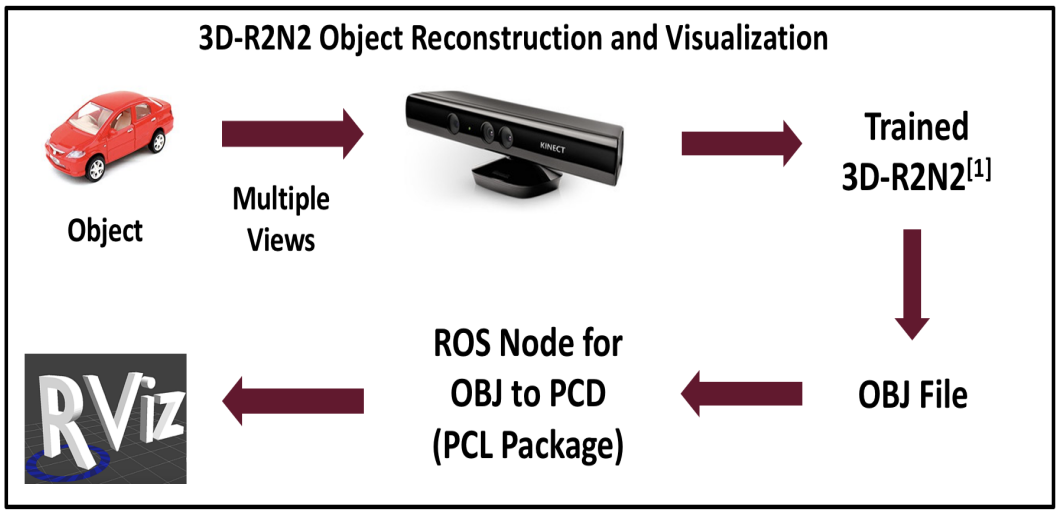
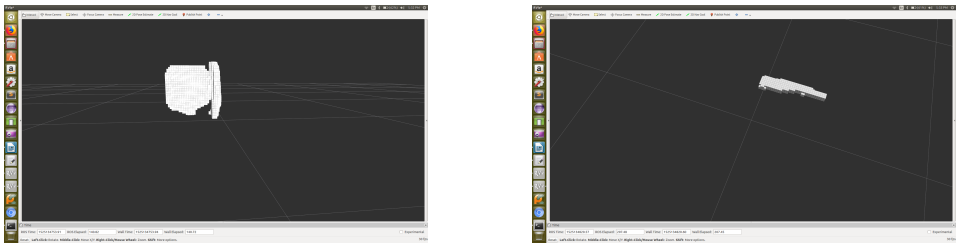


Figure 2: Perception Module

The steps followed for 3D Object Reconstruction and Visualization are as follows:

1. Microsoft Kinect sensor is used to capture images of objects like speaker, laptop, toy-car, etc. from arbitrary viewpoints.
2. These images are then fed to trained 3D-R2N2[2] network to generate a 3D voxel grid file in .obj format.
3. The .obj file is converted to .pcd (point cloud data) using Point Cloud Library (PCL)[8] in ROS.
4. The .pcd file generated above is then visualized in RVIZ (ROS[7] Visualization).



(a) Visualization of Laptop in RVIZ (b) Visualization of Rifle in RVIZ

Figure 3: Visualization in RVIZ

2.2 Control Module

Algorithm 1 Grasp Pose Detection

Input: a viewpoint cloud, $\mathbb{C}$; a region of interest, $\mathcal{R}$; a hand, Θ; a positive integer, N

Output: a set of 6-DOF grasp candidates, $H \subset \mathcal{R}$

1: $\mathbb{C}' = \text{PreprocessCloud}(\mathbb{C})$
2: $\mathcal{R} = \text{GetROI}(\mathbb{C}')$
3: $S = \text{Sample}(\mathbb{C}', \mathcal{R}, \Theta, N)$
4: $I = \text{Encode}(S, \mathbb{C}', \mathcal{R}, \Theta)$
5: $H = \text{Score}(I)$
6: $g = \text{SelectGrasp}(S, H)$

Figure 4: Overview of the Grasp Pose Detection Algorithm

GPD[6, 3] algorithm follows the steps shown in Algorithm 1. "Step 1 preprocesses the viewpoint cloud. It compresses and denoises the point cloud by voxelizing, removing outliers, and performing other standard steps. Anything that can be done to reduce noise or errors in the point cloud should be performed in this step. Step 2 identifies a region of interest (ROI), R, where the grasp will occur. It is important to note that this does not necessarily mean segmenting the object from the background. The ROI could include a set of multiple objects or all objects in a scene. Step 3 samples N (several thousand) grasp candidates from the ROI where each candidate is a 6-DOF hand pose. Step 4 encodes each grasp candidate as a stacked multi-channel image. Step 5 assigns each candidate a score using a four-layer convolutional neural network that indicates how likely the candidate is to be a grasp. Step 6 selects a grasp for execution based on the score evaluated in Step 5 and other considerations related to the suitability of the grasp."[6, 3]

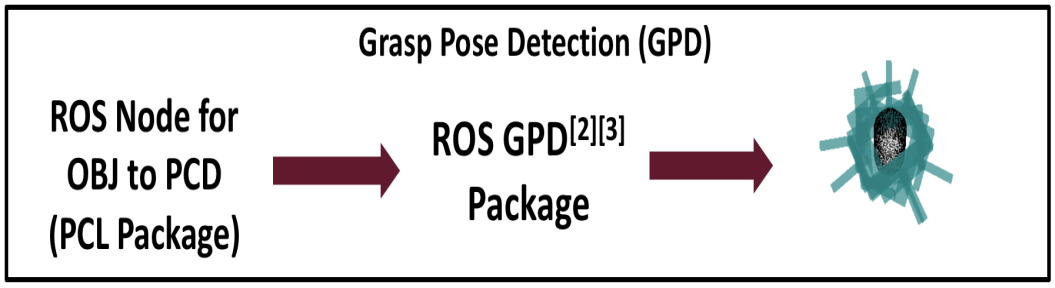


Figure 5: Control Module

The steps followed for grasp pose detection are as follows:

1. The .pcd file generated above is fed as input to Grasp Pose Detection (GPD)[6, 3] in ROS[7].

2. The candidate grasp poses for the given object are visualized in default viewer used by GPD[6, 3] framework.

3 Experiments

3.1 Data Collection

For training, the 3D-R2N2[2] network has been trained on the ShapeNet[1] dataset which is a collection of 3D CAD models that are organized according to the WordNet[5] hierarchy. A subset of the ShapeNet[1] dataset is used consisting of 50,000 models and 13 major categories which includes plane, bench, cabinet, car, chair, monitor, lamp, speaker, firearm, couch, table, cellphone and watercraft. The dataset was split into training and testing sets, with 4/5 for training and the remaining 1/5 for testing. For final evaluation of the entire model, we took live images of some objects like laptop, lamp, toy car, chair and speaker using Kinect.

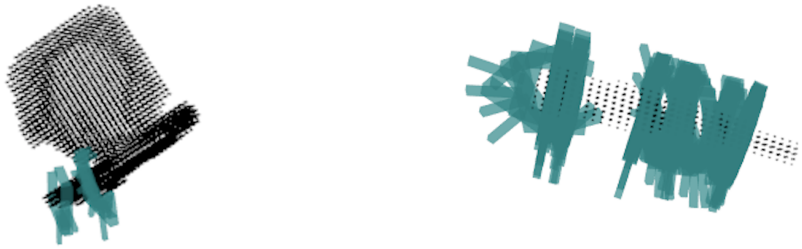
3.2 ROS[7] System Setup

For the ROS[7] system, we installed ROS[7] Kinetic Kame which is the tenth ROS distribution release. It is primarily targeted at the Ubuntu 16.04 (Xenial) release, though other Linux systems as well as Mac OS X, Android, and Windows are supported to varying degrees. While installation we selected full desktop versions which includes most of the modules that were needed for our project. For connecting kinect sensor with ROS we had to install few other packages like libfreenect-dev and ros-indigo-freenect-launch. For GPD[6, 3], we installed the library called as Grasp Pose Detection.

3.3 Experimental Results

We used the pre-trained [Res3D-GRU-3][2] network in this experiment. We evaluated the network with the live images taken from the kinect sensor. We rendered six to nine random views for each model like laptop, toy car, lamp, speaker and chair. We took the images in such a way that they had a uniform white colored background to the images and the reason for doing this was because the network was trained on 3D CAD models which has every image in a white background.

After getting the output form the network which is a 3D reconstructed shape of the object in the form of 3D occupancy grid, we convert it into a point cloud format so that it can be viewed in RVIZ. Finally, we apply grasp pose detector on this point cloud data of the model using the Grasp Pose Detection[6, 3] library available in ROS[7].



(a) Grasp Poses for Laptop

(b) Grasp Poses for Rifle

Figure 6: Grasp Poses detected using GPD[6, 3]

4 Discussion and Conclusion

In this work, we developed an end-to-end system that unifies single and multi-view 3D reconstruction along with grasp pose detection. Although we considered only RGB images from the kinect sensor along with white background limitation, so working with RGB-D images and with no background limitations could be the possible future work. In summary, the framework comprises of 3D-R2N2[2] which is state-of-the-art network architecture that does not require a minimum number of input images in order to produce a plausible reconstruction and is able to overcome past challenges of dealing with images which have insufficient texture or wide baseline viewpoints and GPD[6, 3] framework for grasp pose detection which gives promising results for the 3D objects.

5 Acknowledgement

We acknowledge the support of Professor Yezhou Yang and TA Varun for guiding us and providing the hardware and software support for our project.

References

[1] Angel X. Chang et al. "ShapeNet: An Information-Rich 3D Model Repository". In: *CoRR* abs/1512.03012 (2015). arXiv: 1512.03012. URL: http://arxiv.org/abs/1512.03012.

[2] Christopher Bongsoo Choy et al. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *ECCV*. 2016.

[3] Marcus Gualtieri et al. "High precision grasp pose detection in dense clutter". In: *CoRR* abs/1603.01564 (2016). arXiv: 1603.01564. URL: http://arxiv.org/abs/1603.01564.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[5] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

[6] Andreas ten Pas et al. "Grasp Pose Detection in Point Clouds". In: *CoRR* abs/1706.09911 (2017). arXiv: 1706.09911. URL: http://arxiv.org/abs/1706.09911.

[7] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[8] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011.

[2] [6] [3] [4] [1] [5] [7] [8]