e-YSIP 2023

Exploring Algorithm for Grasping Unknown Objects using Two Finger Gripper

Gokul M K
Archit Jain
Mentors: Jaison Jose, Ravikumar Chaurasia
Duration of Internship: 22/05/2023 – 09/06/2023

2023, e-Yantra Publication

Exploring Algorithm for Grasping Unknown Objects using Two Finger Gripper

Abstract

The objective of this research is to significantly enhance the grasping capabilities of the two-finger gripper robotic arm (such as the UR5 arm) by leveraging a combination of learning and analytical-based algorithms. The focus is on enabling the arm to effectively grasp and manipulate unknown objects with minimal prior knowledge or specific object information. To achieve this, three key algorithms are employed: GrapsNet, Height Assistive Feature (HAF), and Elliptical Centroid Grasp (ECG).

Introduction

- **GraspNet:** GrapsNet is a learning-based algorithm that aims to improve the arm's grasping performance by leveraging large-scale data. By training on a diverse set of grasping examples, GrapsNet learns to generalize and adapt its grasping strategies to different object shapes, sizes, and orientations. This algorithm enables the UR5 to grasp objects that were previously unknown to the system, thereby enhancing its ability to handle a wide range of objects autonomously.

- **Height Assistive Feature (HAF):** The Height Assistive Feature (HAF) algorithm plays a crucial role in improving the arm's grasping precision. It utilizes depth-sensing techniques to estimate the height of an object and provide feedback to the arm's control system. By incorporating this feature, the UR5 can adjust its gripping strategy based

on the object's height, ensuring a secure and stable grasp. This assists in mitigating potential issues such as slippage or dropping objects due to incorrect grip force or inadequate finger placement.

- **Elliptical Centroid Grasp (ECG):** The Elliptical Centroid Grasp (ECG) algorithm combines the analytical approach with the benefits of GraspNet and HAF. ECG starts by segmenting the object from the scene and then estimates the grasp poses. It achieves this by constructing an approximate ellipse around the segmented object and utilizing the center and angle of the minor axis as the position and orientation of the gripper.ECG demonstrates high accuracy when dealing with common objects such as cuboids and cylinders. However, it may encounter challenges when applied to complex objects that possess intricate shapes or characteristics. Despite this limitation, ECG offers a valuable grasp estimation technique that leverages the strengths of existing methods to provide reliable results.

- **Grasp Pose Detection (GPD):** Grasp Pose Detection is an algorithm that uses point cloud data to estimate grasp poses. We employed the integrated version of Grasp Pose Detection (GPD) with MoveIt Task Constructor to gain fine-grained control over the arm's planning process until the grasp pose is achieved. This integration holds significant value, especially in industrial settings and factories where precise estimation and planning play a vital role.The combined power of GPD and MoveIt Task Constructor enhances the overall efficiency and accuracy of grasp pose detection, making it a valuable solution for various applications that require precise manipulation.

- **Point Cloud Grasp (PCD):** Point Cloud Grasp is a crucial algorithm utilized in this research project. Its primary objective is to efficiently segment the point cloud data acquired from sensors, extracting meaningful representations of objects. This algorithm employs surface normals in proximity to the centroid of the object's point cloud to estimate grasp candidates. However, it is important to note that this algorithm is currently undergoing research and development. There are still additional steps to be taken before it can accurately estimate the optimal grasp poses based on various factors.

Overall, this research aims to advance robotic manipulation systems by empowering robotic arms to autonomously pick and handle a wide variety of unknown objects using a two-finger gripper. By the capabilities of GrapsNet, HAF, and Point Cloud Grasp, the arm becomes more adaptable, robust, and efficient in its grasping operations. This research contributes to the realization of versatile and autonomous robotic agents capable of interacting effectively with real-world environments and objects.

Completion status (with brief details of work)

We have successfully implemented four different algorithms, namely GraspNet, Height Assisted Feature (HAF), Elliptical Centroid Grasp (ECG), and Grasp Pose Detection (GPD). These algorithms differ in terms of their computation, architecture, and methodology, showcasing the diversity of approaches we have implemented.

Also, we have created a quality metrics for the GraspNet, HAF and ECG to measure some properties such as accuracy, computational resources, planning time, precision, stability, success rate, etc.

We have created a grasping pipeline for such algorithms in ROS which enables developer to execute required algorithm in one flow. This pipeline has a config file which user needs to update as per it's hardware and file structure setup and utilise this package to execute different services available such as initialise robot arm for capturing data, planning grasp poses based on best planning time, execute grasping using moveit commander, etc.

Research

At the outset of our exploration into grasping algorithms, we conducted thorough research on various open-source packages. Our initial list included 8 algorithms that were meticulously coded and accompanied by comprehensive documentation. These algorithms are as follows:

- GraspNet
- Height Assisted Feature (HAF)
- Elliptical Centroid Grasp (Developed)

- Grasp Pose Detection (GPD)
- Dexnet
- GraspIt
- PointNetGPD
- Antipodal Grasp Identification and Learning (AGILE)

Out of the aforementioned list, we **successfully installed 6 out of 8** grasping algorithms with relative ease. Below is a table that provides a comparison of the 6 algorithms based on specific features.

Table 1.1: Comparison of Grasping Algorithms

Algorithms	**GraspNet**	**Dexnet**	**HAF**	**GraspIt**	**GPD**	**ECG**
Type	Learning	Learning	Analytical	Analytical	Analytical	Analytical
Input	RGB-D Image	3D-Meshs	Pointcloud	3D-Meshs	Pointcloud	RGB-D Image
Architecture	Deep Neural Network	Deep Neural Network	Height Information	Friction Cones	Friction Cones	Math Formulations
Final Grasp	Network extracts features	Samples grasps	Constructs height map	Collision Detection	Force Closure	Center and Minor Axis
Stability	Does not ensure	Does not ensure	Does not ensure	By force closure	By contact points analysis	Grasp around object center
Pros	Eliminates need for mesh	Predicts grasp qualities	Incorporates Depth Cues	Friction Cone analysis for stability	Generalize to novel objects	Easy to implement
Cons	Relies on dataset	Requires large dataset	Struggle with complex objects	Fails with deformable objects	Acucurate depth sensing	Not detailed approach

However, when it came to selecting algorithms for implementation, we decided to focus on four specific ones: **GraspNet, Height Assisted Feature (HAF), Elliptical Centroid Grasp (ECG), and Grasp Pose Detection (GPD)**. These four algorithms showed promising results and aligned

well with our project goals. On the other hand, the remaining two algorithms presented significant challenges during integration, which led us to exclude them from our final selection.

Reasons for Discarding the other 4 Algorithms

- **Dexnet:** (Installed Successfully, Issues while Integration)
  Dexnet is designed to operate with known objects, requiring the model to be trained on our specific object dataset to accurately estimate grasp poses. Since our focus is on evaluating the quality of grasping algorithms of unknown objects, Dexnet is not suitable for our purposes and is therefore excluded from our evaluation.

- **GraspIt:** (Installed Successfully, Issues while Integration)
  GraspIt offers a simulator that takes robot hand and object meshes as input to estimate the grasp pose. However, when it comes to running the algorithm seamlessly from input object to executing the plan with the robot arm, it becomes challenging with GraspIt. This difficulty in achieving an end-to-end execution is one of the reasons why we decided to exclude GraspIt rom our evaluation.

- **PointNetGPD:** (Issues while Installing)
  PointnetGPD utilizes the PointNet architecture, a neural network, to estimate grasp poses. However, there are no pre-trained models available, which meant that we had to train the algorithm ourselves. This training process was time-consuming, especially when considering the integration with the MoveIt planning framework. As a result, we decided to exclude PointnetGPD rom our evaluation.

- **AGILE (Antipodal Grasp Identification and Learning):** (Issues while Installing)
  AGILE presented significant challenges during the package installation process. We encountered several dependency mismatches, and resolving them resulted in conflicts with other packages that were functioning correctly. Despite our numerous attempts, we were unable to overcome these issues. As a result, we made the decision to exclude AGILE from our evaluation.

Selected Algorithms and Descriptions

Given Below is the Detailed Description of Each Selected Algorithm

- **Algorithm: GraspNet**
  - **Type:** Learning based.
  - **Architecture:** It employs a deep neural network, such as a (CNN) or a combination of CNN and (RNN), for grasp pose estimation.
  - **Description:** The network learns to extract relevant features from the image and maps them to grasp poses in the 3D space. The network is trained on a large dataset of annotated grasps, where the ground truth includes the 3D position and orientation of successful grasps.
  - **Input Data Required:** RGB and Depth Image.
  - **Output Received:** The output of the network is a set of predicted grasp poses represented by their 3D positions and orientations. It provides with top 50 grasp pose with score based on grasp evaluation by itself.
  - **Stability of Grasping:** Does not ensure the stability of the grasp, just estimates grasp poses.
  - **Gripper Dependency:** Designed to be agnostic to the type of gripper.
  - **Pros:**
    - It can estimate grasp poses directly from RGB-D images, eliminating the need for explicit 3D models of objects.
    - It can generalize well to novel objects and scenarios due to its learning-based approach.
    - It can be used for assistive applications to pick unknown objects with complex geometries and orientations.
  - **Cons:**
    - It's performance heavily relies on the quality and diversity of the training dataset.
    - It may struggle with object occlusions, cluttered scenes, or cases where the visual appearance of the object is not distinct.
- **Algorithm: Height Assistive Feature (HAF)**

- **Type:** Analytical based.
- **Architecture:** Analytical approach to incorporate depth cues and geometric characteristics of the object from the image.
- **Description:** Accumulates the height information from the depth data, constructs a height map. The pre-processed point cloud is used to extract visual features (object's appearance, texture, and shape). Both are fused and allows the algorithm to incorporate depth cues and geometric characteristics of the object defining the position vector and orientation for grasp.
- **Input Data Required:** Point Cloud Data file format (PCD file).
- **Output Received:** The final output is the estimated grasp pose, including the grasp position (grasping direction vector) and the orientation of the gripper relative to the object.
- **Stability of Grasping:** Doesn't ensure stability, instead estimates grasp poses.
- **Gripper Dependency:** Designed to be adaptable to different gripper sizes.
- **Pros:**
  - It does not rely on machine learning or training data, making it computationally efficient and adaptable to various objects.
  - It can provide reasonable grasp pose estimations even in cluttered or unknown environments.
- **Cons:**
  - It may struggle with objects that have complex or irregular shapes where geometric features are not well-defined.
  - It may not generalise well for all different applications as this is restricted towards top down approach (Camera facing down towards the object).

- **Algorithm: Elliptical Centroid Grasp (ECG)**
  - **Type:** Analytical based.
  - **Architecture:** It estimates the pose and orientation by considering the centroid and angle of minor axis from an ellipse constructed surrounding the object.
  - **Description:** The segmentation process involves examining the double derivative of pixels, enabling the identification of object

boundaries. The contour construction technique is then employed to estimate an ellipse that encloses the segmented object.

- **Input Data Required:** RGB and Depth Camera Message
- **Output Received:** Filtered grasp pose representing the translation and orientation for the gripper to grasp object.
- **Stability of Grasping:** Stability of the grasp is ensured by considering factors such as the object's mass distribution, center of gravity, and the gripper's design.
- **Gripper Dependency:** Designed to be adaptable to different gripper sizes.
- **Pros:**
  - Lesser planning time and computational resource to operate.
  - Can be used for industrial pick and place applications with simpler objects for better reliability.
- **Cons:**
  - It may misbehave for complex geometries with unstructured spline surfaces.

Camera Configuration

We planned to implement two different camera configurations, eye-on-hand (EOH) and eye-to-hand (ETH), to test our grasping algorithm.

- **Eye-on-Hand (EOH):**
  Eye-in-hand configuration is the most common camera configuration for manipulation tasks. In this configuration, the camera is mounted on the end-effector of the robotic arm, so it moves along with the gripper. This gives the robot a close-up and detailed view of the objects being grasped, which allows for better hand-eye coordination, precise positioning, and feedback-based control during grasping tasks. We tested **GraspNet, HAF, and ECG** algorithms with this configuration.

- **Eye-to-Hand (ETH):**
  Eye-to-hand configuration involves positioning the camera separately from the robotic arm, typically fixed in a static location within the workspace. It captures the scene from a stationary viewpoint while the robotic arm performs grasping tasks. This configuration is commonly used in situations where the camera's position remains constant, and the robot needs to perceive and localize objects in the environment

relative to the fixed camera. We tested **ECG, GPD** algorithms with this configuration.

It is important to note that, when testing the HAF algorithm, we had to configure the robotic arm's home position so that the **camera was pointing directly downwards**. This is because the HAF algorithm only works on heights, and it needs to be able to see the object from above in order to calculate the best grasp (refer Demo page).

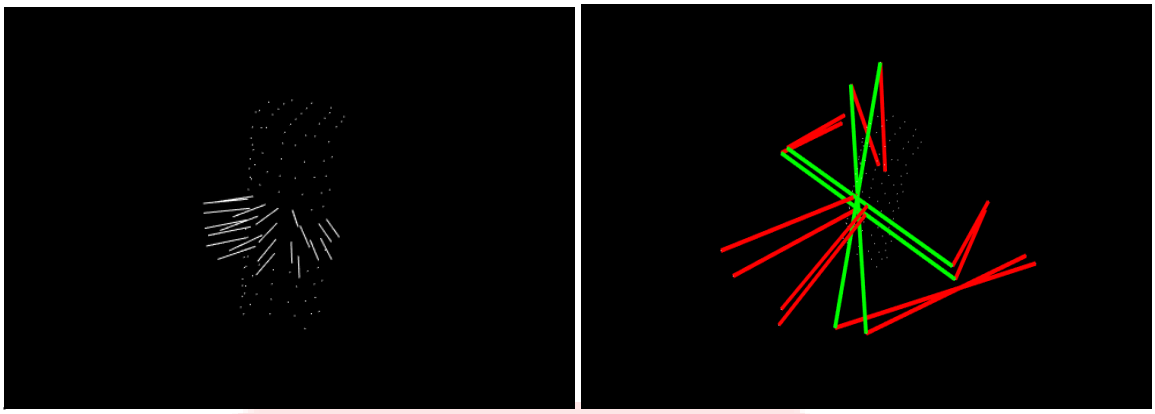
Custom Grasping Algorithm

After experimenting with various Grasping Algorithms, we had this idea of using Poincloud library to come up with a very efficient and intuitive Grasping Approach, thats where we came up with **Pointcloud Grasp (pcd-grasp)**.

Algorithm: Pointcloud Grasp (pcd-grasp):

- **Type:** Analytical based.
- **Architecture:** It estimates the pose and orientation by considering surface normals of the segmented object from the scene, that are close to the object's centroid
- **Description:** The objects are initially segmented using Euclidean clustering with the assistance of a K-D Tree. An oriented bounding box is then estimated around each segmented object. The grasp candidates are determined by considering the surface normals in proximity to the centroid of the object's point cloud.
- **Input Data Required:** Pointcloud Data from Camera
- **Output Received:** Raw grasp candidates
- **Gripper Dependency:** Designed to be work with parallel-jaw grippers

Please note that this algorithm is currently undergoing research and development, and it is not yet fully optimized for accurate grasp estimation.



(a) Surface Normals (b) Grasp Candidates

Quality Metrics

We conducted extensive experiments using different novel objects to evaluate the performance of the three algorithms (GraspNet, HAF, ECG). In addition to the existing grasp quality metrics, we developed our own set of metrics after reviewing the work of Máximo A. Roa and Raúl Suárez (2015) and Kim, Iwamoto, Kuffner, Ota, and Pollard (2011). Our metrics are intuitive and straightforward, and they are designed to assess the grasping results in a way that is both meaningful and informative.

The Features of the Quality Metrics are:

- **Success Rate:** (Score: 0/1) Success in Grasping
- **Precision:** (Score: 0-1) Gripper Close to estimated Grasp Location
- **Accuracy:** (Score: 0-1) Minimal Deviation between shown Grasp and Actual
- **Planning Time:** (Time in minutes) Time taken by the Algorithm to estimate Grasp Pose
- **Computational Resources:** (0.1/0.5/1) Processing Power of the Algorithm
- **Stable:** Stability of the Grasp
- **Secure:** Secure and Stiff Grasp

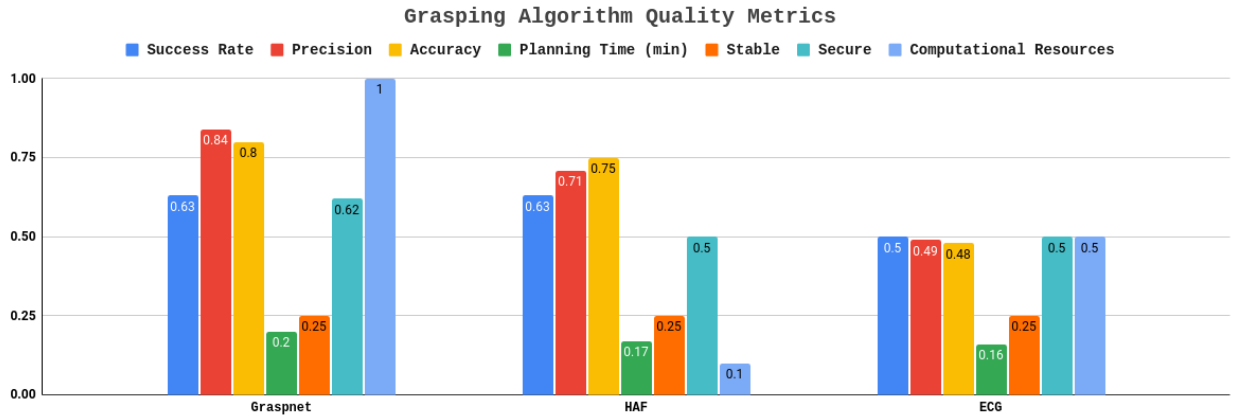


Chart1.1: Grasping Algorithm Quality Metrics

1.1 Hardware parts

- **List of hardware**
  - Universal Robotic 5 Arm (UR5 Arm Kit including power supply and Ethernet cable for communication)
  - OnRobot - RG2 - Flexible Gripper (2kg Payload)
  - Intel D435i RealSense Depth Camera (including C-type cable)
- **Detail of each hardware**
  - UR5: The Universal Robots UR5, is a highly flexible robotic arm that enables safe automation of repetitive, risky tasks. With a carrying capacity of 5 KG and a radius of 850 mm, it is the perfect cobot for performing light tasks such as packing, assembly, or testing.

    Here is the technical specification and other details of the arm: UR5 Technical Specification
  - OnRobot Gripper: The RG2 - 2kg payload robot arm gripper is a flexible collaborative gripper with a built-in Quick Changer up to 110mm stroke. It provides intelligence, fast deployment, easy customization, and programming.

    Here is the data-sheet of the OnRobot RG2 Gripper: OnRobot RG2 Gripper data-sheet
  - RealSense Camera: The Intel RealSense D435i places an IMU into our cutting-edge stereo depth camera. With an Intel module and

vision processor in a small form factor, the D435i is a powerful complete package that can be paired with customized software for a depth camera capable of understanding its own movement.

Here is the data-sheet of the RealSense Camera: Intel RealSense D400-Series data-sheet

1.2 Software used

For the implementation of our grasping algorithms, we employed a range of technology stacks. Specifically, we utilized the Ubuntu 20.04 operating system as the platform to run and test our algorithms in conjunction with the hardware components.

- **Robot Operating System (ROS) Noetic :** ROS Noetic
  We used the ROS Framework to to validate our grasping algorithms, grasp planning scripts, and control the UR5 hardware arm.
- **Pointcloud Library (PCL):** PCL
  PCL played a pivotal role in our project, serving as a versatile tool for point cloud visualization, preprocessing, and the development of our custom grasping algorithm.
- **OpenCV:** OpenCV
  OpenCV played a tremendous role while using ECG to estimate Grasp Poses, it also stays as a prerequisite to other grasping algorithms. In this project, we builded it from source to avoid any version conflicts.
- **Intel Realsense:** Realsense SDK; Realsense ROS Wrapper
  The Intel Realsense d435i Depth Camera was utilized as an input interface for our grasping algorithms. It allowed us to capture RGB, depth, and point cloud information of the environment, which served as crucial inputs for estimating grasp poses

It's important to note that there are additional software applications that contributed to the this project, although they were not mentioned above due to their automatic installation or being prerequisites for the mentioned tools. For instance, Gazebo, a powerful physics engine, and RViz, a robot visualization tool, are automatically installed when ROS is installed, and they played integral roles in our project's development and simulation.

1.3 Software and Code

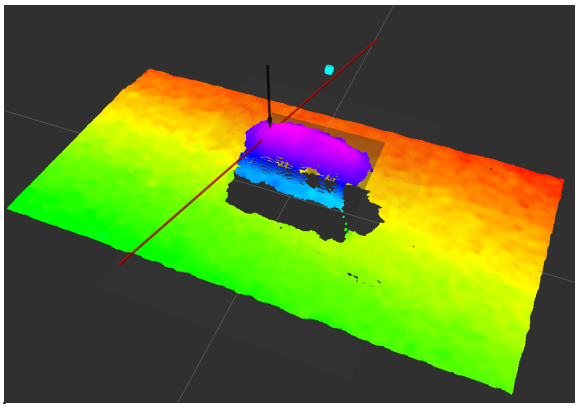
GitHub served as our platform for source code sharing and collaborative contribution. Link for the repository of code.

Here is a concise overview of all the branches available in the repository:

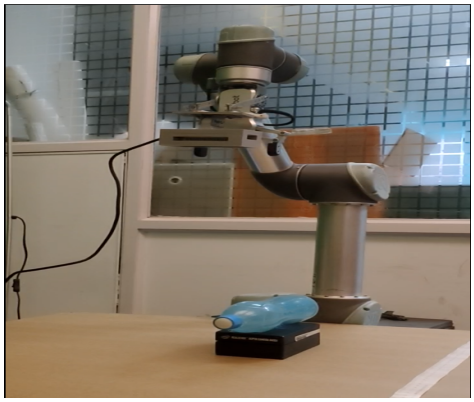
- **'main' branch:** This branch serves as a summary and provides an overview of the entire project.
- **'haf pipeline' branch:** The code to implement the HAF pipeline is located in this branch, organized into three bash scripts.
- **'graspnet pipeline' branch:** The code to implement the Graspnet pipeline is located in this branch, organized into ROS service calls.
- **'ecg pipeline' branch:** This branch includes all the essential files and scripts required to run ECG with simulation setup.
- **'pcd grasp' branch:** The code to implement the Pointcloud Grasp Algorithm , which works with PCD files, is located in this branch.
- **'grasp sim' branch:** This branch includes all the essential files and scripts required to launch the simulation of the UR5 arm in Gazebo and visualize the sensors and motion planning using RViz.

1.4 Demo

1) Height Assisted Feature (HAF) Hardware Setup

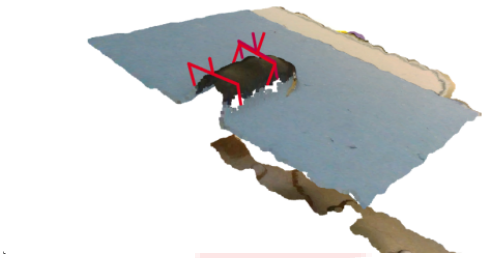


(a) HAF Grasp Estimate



(b) UR5 Hardware Setup for HAF

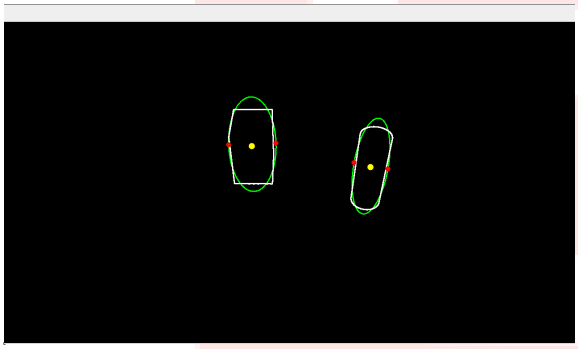
2) GraspNet Hardware Setup



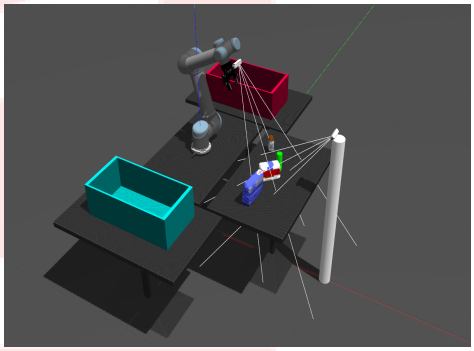
(a) Graspnet Grasp Estimate

(b) UR5 Hardware Setup for Graspnet

3) ECG Simulation Setup

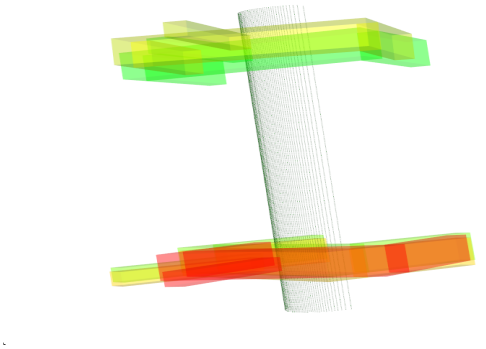


(a) ECG Grasp Estimate

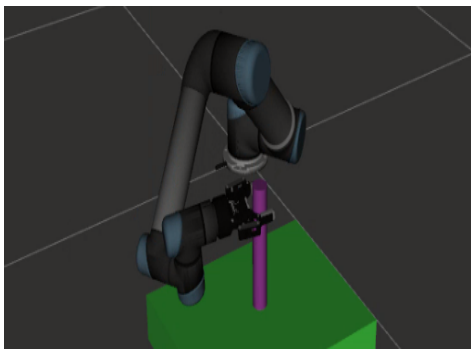


(b) UR5 Simulation Setup for ECG

4) GPD Simulation Setup



(a) GPD Grasp Estimate



(b) UR5 Simulation Pick for GPD

Demonstration Video
We have documented our grasping experiments for all four algorithms and uploaded them to YouTube for reference. Playlist Link of demonstration video

1.5 Future Work

- Explore computationally efficient and accurate learning-based and analytical algorithms beyond our current implementations.
- Conduct extensive experiments in cluttered scene and factory settings to evaluate the performance of these algorithms.
- Develop a flexible pipeline or framework that integrates the most effective algorithms based on our quality metrics and can accommodate future algorithms and techniques, ensuring continuous improvement of the solution.

1.6 Bug report and Challenges

- **Orientation Issue:** A considerable portion of our time was devoted to resolving the disparity in orientation introduced by GraspNet and integrating it into our motion planning framework, MoveIt. The camera orientation employed by GraspNet differed from that of the UR5 robot, necessitating our attention to rectify this mismatch. To address this issue, we incorporated supplementary transforms to accommodate the orientation variation. By implementing these transforms, we were able to effectively convert between the two coordinate frames, leading to improved grasp accuracy and dependability.
- **Version Conflicts:** During the installation process of the required packages for the grasping algorithms, we encountered multiple challenges and encountered version conflicts along the way.We unknowingly deleted certain packages while installing GraspIt, such as yaml loader (ruamel), meshpy, and scipy, which were necessary for GraspNet. We spent time manually fixing the issue.
- **Weird Motion Planning:** We encountered a challenge with the motion planner in MoveIt, as it initially produced unexpected and unconventional plans. It required considerable effort and time to gain

control over the planner and introduce constraints to restrict its motions to meet our requirements.

- **Ideal Environment Setup:** Achieving accurate grasp estimates and planning required our hardware setup to be carefully configured. In some instances, we had to rely on having a proper base plane for the objects to ensure reliable grasp pose estimation. This ensured that our grasp planning algorithms could work optimally and deliver precise results.

- **MoveIt-Grasps:** Within the MoveIt Framework, there is a valuable package known as Moveit-Grasps that serves the purpose of estimating grasps and determining optimal waypoints to reach them. Unfortunately, we faced challenges when it came to installing these packages and establishing a connection with the UR5 robot. The documentation proved to be confusing, specifically concerning the compatibility of ROS versions. While certain packages were accessible on the Melodic distribution, others were exclusively available on the Noetic distribution.

- **Pre-trained models:** At times, we faced difficulties in determining the correct location to obtain pretrained models. This challenge was particularly prominent during the installation of Dex-Net, as it required extensive effort to locate and download the appropriate pretrained models. Additionally, when working with PointnetGPD, we encountered a lack of available pretrained models. Consequently, our only option was to train the model ourselves, which proved to be challenging due to time constraints.

Bibliography

[1] Fang, Hao-Shu and Wang, Chenxi and Gou, Minghao and Lu, Cewu, *GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping*, 2020

[2] Hao-Shu Fang, Chenxi Wang, Minghao Gou, Cewu Lu, *GraspNet: A Large-Scale Clustered and Densely Annotated Dataset for Object Grasping*, 2020

[3] David Fischinger and Astrid Weiss and Markus Vincze, *Learning Grasps with Topographic Features*, 2015

[4] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt, *Grasp Pose Detection in Point Clouds*, 2017

[5] Andreas ten Pas and Robert Platt, *Using Geometry to Detect Grasp Poses in 3D Point Clouds*, 2015

[6] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, Ken Goldberg, *Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics*, 2017

[7] Andrew Miller and Peter K. Allen, *Graspit!: A Versatile Simulator for Robotic Grasping*, 2004

[8] Olyvia Kundu and Swagat Kuma, *Graspit!: A Versatile Simulator for Robotic Grasping*, 2004

[9] Chen CS, Hu NT, *Eye-in-Hand Robotic Arm Gripping System Based on Machine Learning and State Delay Optimization*, 2023

[10] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, Jianwei Zhang, *PointNetGPD: Detecting Grasp Configurations from Point Sets*, 2019

[11] Roa, M.A., Suárez, R., *Grasp quality measures: review and performance*, 2015

[12] Junggon Kim, Kunihiro Iwamoto, James J. Kuffner, Yasuhiro Ota and Nancy S. Pollard, *Physically-based Grasp Quality Evaluation under Pose Uncertainty*, 2011