

Development of grasp localization system based on deep learning aided vision system

Jemuel Stanley Premkumar
Department of Mechatronics Engineering
SRM Institute of Science and Technology, KTR
India
jemprem@umich.edu

Abstract

This B.Tech final project focuses on the development of a grasp localization system, employing state-of-the-art methods namely GQ-CNN and ContactGraspNet for executing grasps on novel objects. The project centers on the ABB YuMi Dual-arm Cobot, requiring the development of the entire system from its conceptualization.

Primary activities encompassed the establishment of seamless communication among the robot controller, server, and the vision machine client. A critical aspect of the project involved configuring the vision system, incorporating an overhead Kinect RGB-D camera, and addressing camera calibration and depth estimation challenges.

The undertaken responsibilities predominantly revolved around robot control, vision system setup, and the integration of all subsystems. This document serves as a concise report highlighting key project activities and outcomes.

Keywords: manipulation; grasp localization; robot control; computer vision

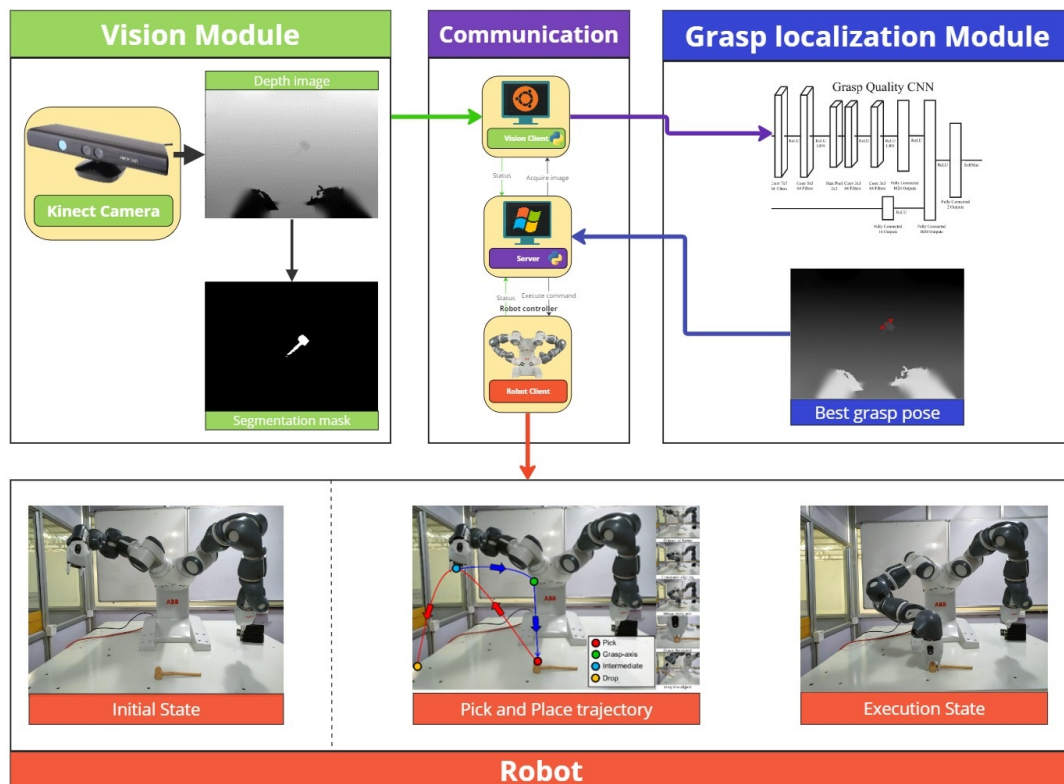


Figure 1: Overview of the developed grasp localization system

1 Project Highlights and Approach

1.1 Key contributions



1. **Establishment of Python-Based wrapper for the Robot:** We developed a Python wrapper for the RAPID programming language, encompassing common motion, sensing, and feedback commands. This wrapper facilitates a more Pythonic coding approach, offering seamless integration and simplifying the utilization of RAPID functionalities within the Python environment.
2. **Provision of Two Readily Available Solutions for Grasp Localization:** Our project introduces three readily available solutions (GQ-CNN, ContactGraspNet, Planar-Antipodal grasping) for grasp localization, offering flexibility and choice in implementation.
3. **Modification of Models for Live Inference Grasp Execution:** The models were adapted to facilitate live inference for grasp execution, enhancing the real-time applicability of the system.
4. **Development of a Primer for Future Collaboration:** We devised a comprehensive primer to ease future collaborations, providing a structured foundation for ongoing and future endeavors.
5. **Provision of First-Hand Guidelines and References for Robot-Related Troubleshooting:** Our contribution includes detailed guidelines and references for troubleshooting, serving as a valuable resource for addressing challenges in the context of robotic operations.
6. **Identification of Optimal Robot Configuration Values:** We determined optimal configuration values for the robot through thorough simulations on MATLAB and testing on the physical robot, optimizing its performance and ensuring efficient operation.

By establishing a robust foundation for future projects and research in grasp localization, our contributions are poised to facilitate the work of future collaborators in the department. The insights provided, ranging from communication frameworks to troubleshooting guides, collectively contribute to advancing the field and fostering continued work in robotic manipulation.

1.2 Methodology

- **Pre-trained Deep Learning Networks:**
 - Utilized due to project constraints, including limited duration and unavailability of GPU for full network training.
 - Note: The deep learning model chosen does not track object dynamics, emphasizing a preference for static objects.
- **Choice of Image Acquisition Hardware:**
 - Camera Placement: At camera's optimized working distance
 - Sensor Resolution: Aligned with deep learning model specifications.
 - Environmental Conditions:
 - * Structured infrared light projection necessitates avoiding reflective objects in the camera's FOV.
 - * System recommended for indoor use due to reliance on infrared technology.
 - Object Attributes:
 - * Maximum object height limited by camera-table positioning.
 - * Objects must be opaque, non-black, and non-reflective due to infrared projector characteristics and within 0.5 kg
- **Gripper Type and Payload Constraints:**
 - Parallel jaw gripper chosen for alignment with the deep learning model's training.
 - Gripper has a maximum extendability of 50 mm, crucial for object segregation based on specified constraints.
- **Choice of Software Platform:**
 - RAPID (high-level language for ABB industrial robots) used for robot motion control.
 - Majority of software development, including vision and deep learning systems, implemented in Python including wrapper for RAPID commands
 - Ensures a cohesive framework, accommodating library dependencies and enabling seamless integration of components like Libfreenect, which relies on Python functionality.

1.3 Softwares and frameworks used

Programming Languages	Frameworks	OS	Hardware
Python, RAPID, MATLAB	Tensorflow, OpenCV, numpy, Scipy, PyKinect, OpenKinect, Socket	 , 	Kinect RGB-D camera, Intel Realsense D435i, Zed Mini, ABB IRC5 controller

2 Vision

2.1 Hardware setup

2.1.1 Single-view setup

The vision hardware setup was initially designed as a single-camera configuration, taking into account scene constraints and various limitations. Depth images obtained directly from the Kinect camera were tested in the GQ-CNN network. Efforts were made to enhance the quality of depth images acquired through a single-camera setup, exploring various deep learning methods for denoising and resolution enhancement.

2.1.2 Multi-view setup

Considering the insufficiency of a single Kinect sensor to generate a point cloud with the desired resolution, an exploration of a multiple-camera setup ensued. Two Kinect sensors were employed to acquire separate point clouds, mitigating interference issues caused by simultaneous IR projection. Although initial attempts involved aligning these point clouds using ICP (See Figure 2) to restore the object's original pose in the world, the increased complexity and execution time prompted a re-evaluation. Consequently, a decision was made to revert to a single-camera setup.

Alternative hardware options, such as the Intel Realsense Depth camera and Zed Mini, were also experimented with. However, the Kinect proved superior within an 800mm distance, leading to its selection and overhead placement for optimal performance.

2.2 Computer vision

2.2.1 Camera calibration and transformations

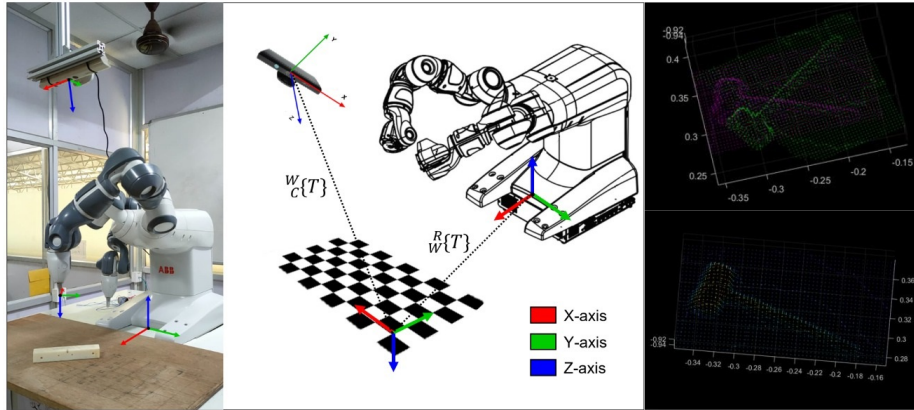


Figure 2: Co-ordinate frames (left) ; Camera calibration setup (middle) ; ICP for aligning multiple views (right)

2.2.2 Depth estimation

The GQ-CNN model necessitates a depth image with authentic depth values, requiring the calculation of actual depth values from the depth intensity image. To achieve this, a Blackbox model approach was employed to establish a relationship between depth intensity and actual depth measurements. A depth intensity image of a flat object (calibration board) was captured at measured distances from the camera, ranging from 60cm to 100cm in 5mm intervals (see Figure 3). Utilizing this mapping data, a regression model determined the coefficients of the polynomial describing the correlation between depth intensity and actual depth values. The derived polynomial was then applied to convert the depth intensity image into the requisite actual depth value image for the network (see Figure 3).

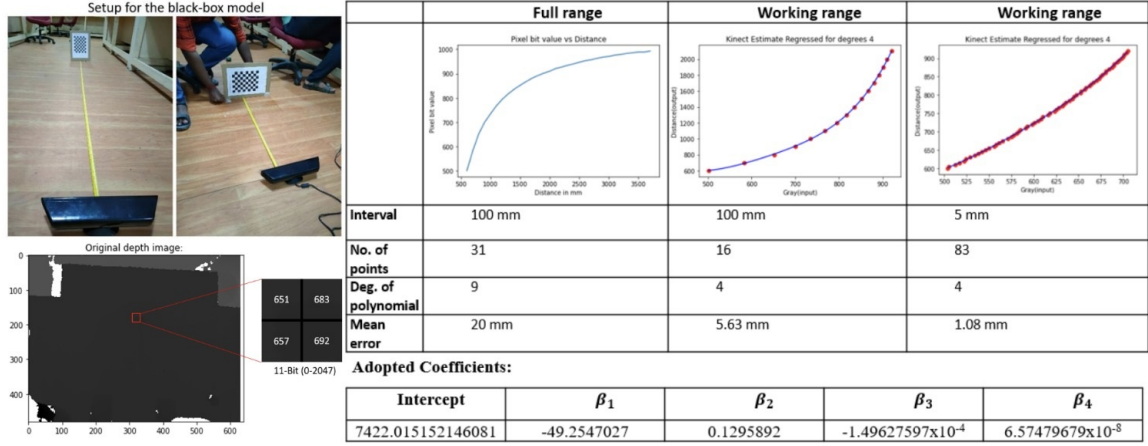


Figure 3: Depth estimation setup (top-left) ; 11-bit Depth intensity image (bottom-left) ; Black box model for Kinect camera (right)

2.2.3 Binary segmentation mask

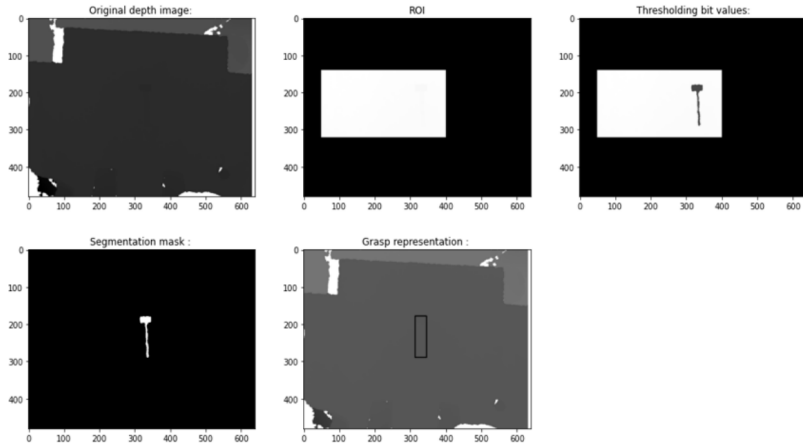


Figure 4: Depth based binary segmentation mask and classical planar grasp pose generation

The deep learning model necessitates both a segmentation mask and an actual depth value image to precisely localize the object of interest within the workspace. Achieving this involves bitwise thresholding on the depth intensity image, where pixel values above a specified threshold are set to 0 (black), and those below the threshold are set to 2047 (white) in the 11-bit image. The threshold value is determined through iterative trials to ensure effective segmentation above the workspace table. A boundary is applied to the depth image to eliminate noise or detect objects outside the defined workspace (Figure 4)

To benchmark the deep learning model's performance, a computer vision solution for grasp localization was implemented (Figure 4) and tested. In this approach, the depth image is acquired, and the segmentation mask is generated initially. Subsequently, the object's edges are detected using the mask, and the contour is determined.

A rectangle is fitted to the contour with the minimum possible area. The orientation of this oriented rectangle on the plane with respect to the camera is computed using intrinsic parameters. Depth is calculated using the acquired depth image and the polynomial generated by the Blackbox model, relating depth intensity to actual depth measure. If the grasp is deemed suitable for execution (based on gripper width), the grasp pose with respect to the camera is computed and communicated to the Robot Control Server. The proposed solution's output is illustrated in Figure 4.

However, a limitation of this solution is its qualitative performance for objects placed in an out-of-plane pose. The system assumes the object to be in-plane and oriented within the plane. Qualitatively, the grasp approach must align with the normal of the object's top plane, which would be oriented differently if the object is placed out of plane.

3 Robot control and communication

3.1 Motion primitive

The Pick-and-place block in Figure 1 illustrates the pre-defined grasp planning trajectory.

3.2 Socket communication

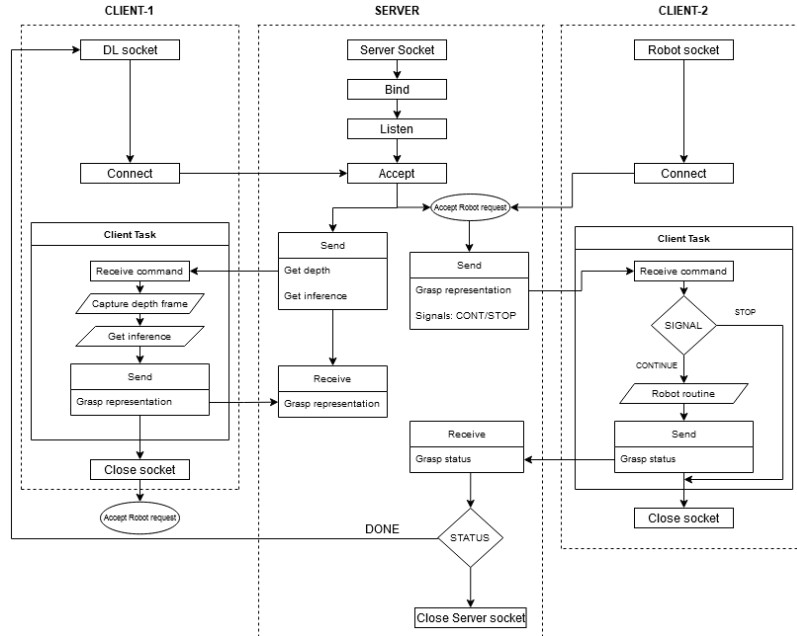


Figure 5: Flowchart of communication logic

The communication interface involves three primary systems: the Python server, Vision-DL client, and the Robot client. Initially, the Vision-DL establishes socket communication with the server initiating the live inference process. Post grasp inference in the form of translations and quaternion, the resulting grasp information is encoded into a byte string and transmitted back to the Python server.

Upon receiving the grasp information, the socket communication between the Vision-DL client and the Python server is closed, and a new communication link is established between the Python server and the Robot client. Subsequently, the grasp string representation is sent to the Robot Studio, where the RAPID code interprets the information for a pick-and-place routine. Once the routine is completed, a message is relayed to the Python server, indicating whether the grasp was executed or not. Upon receiving a positive command, the socket communication is closed, marking the end of the pipeline. Figure 5 provides a systematic schematic of the communication and information flow between the Python server and the two clients.

Socket programming was chosen for its high reliability and in-order timely data delivery. The communication protocol employed is TCP/IP (Transmission Control Protocol), and the chosen socket type is Stream socket, available within the Python framework.

4 Result

The results presented in this section offer a glimpse into the project's outcomes, acknowledging that several activities are omitted in this report for brevity. Figure 6 provides a visual representation of a sample inference, demonstrating the grasp localization and segmentation achieved by the system.

Furthermore, Figure 7 showcases additional inference samples for all the objects used. These samples underscore the effectiveness of the proposed approach in accurately localizing and grasping objects within the defined workspace.

While the detailed analysis and discussion of these results extend beyond the scope of this report, the provided samples serve as illustrative examples of the project's achievements in the realm of grasp localization using computer vision and deep learning techniques.



Figure 6: Sample of the grasp localization output

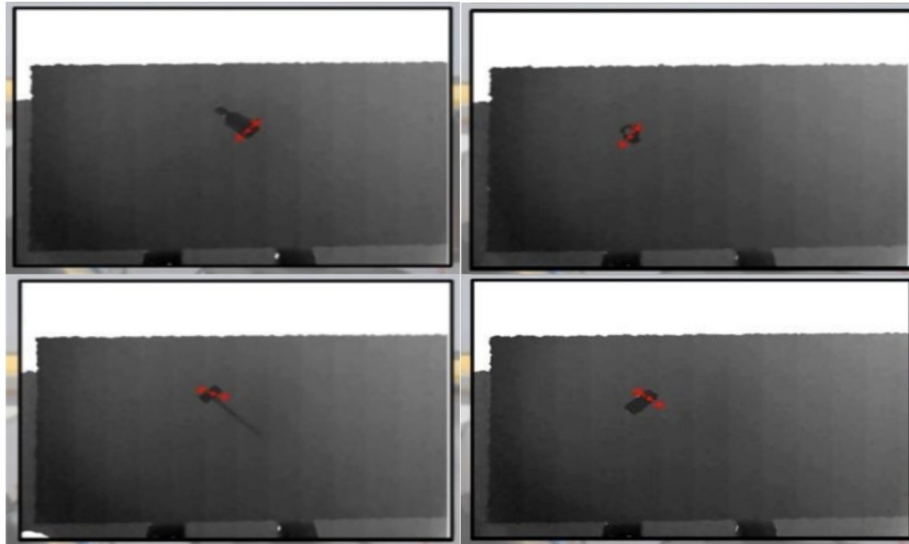


Figure 7: Samples of grasp inference on bottle (top-left); tape-holder (top-right); mallet (bottom-left); block (bottom-right)

APPENDIX

Github Repository: http://www.github.com/JemuelStanley47/GL_2022.