Tech GC 2024

Interactive Robotics

Team Report

Problem statement:

Basic Overview:

The problem statement required us to develop a fully functional control system for a robotic arm in CoppeliaSim. The arm must have features which allow it to control its movements and interact with the environment through external tools. We were also required to create an interface which allows others to interact with and control the robot at different levels of abstraction.

Therefore, we divided the problem statement into 2 Major Parts:

1. Robot Simulation:

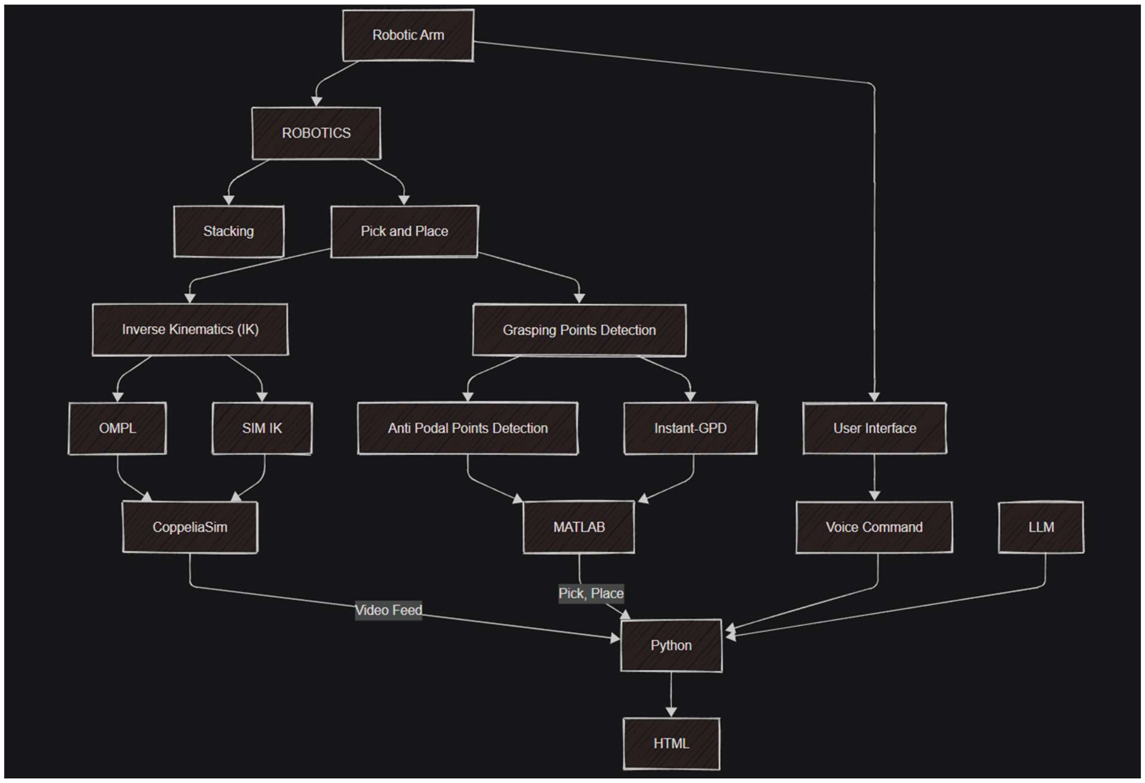
1. Choosing of a pre-build arm available in CoppeliaSim dataset. (UR5)
2. Joint Control
3. End Effector Control
4. Interaction with the environment
   1) Picking of Objects
   2) Stacking of Objects

2. Developing the User Interface:

1. Terminal Based Control
2. Graphic User Interface
3. Voice – Controlled Interface

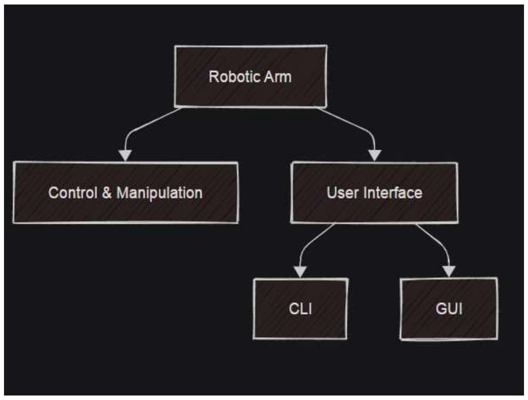
Methodology

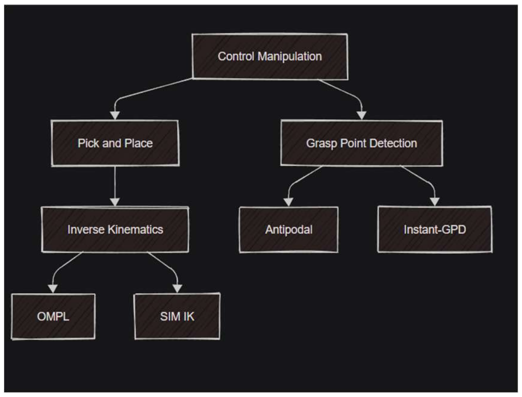
System Architecture:



Caption: Overall system architecture

Movement of the Arm:





Caption: Explanation of working of Robotic Arm

We are using Inverse Kinematics (IK) for movement of the joints of the arm. It enables us to control the movement of the joints and end effector ensuring that our arm reaches its desired position.

We are integrating the concept of IK with CoppeliaSim using two modules: SIM IK and OMPL.

SIM IK (Subspace Inverse Kinematics): -

It is a machine learning based approach to solving inverse kinematics problems. Rather than solving IK in a full joint space, SIM IK operates in a reduced dimensional subspace which makes it faster and more natural looking than other traditional IK solvers.

By leveraging a data-driven approach, SIM IK captures realistic human movement patterns, avoiding unnatural and physically implausible poses. This allows the robot to reach its target position smoothly and efficiently.

The major issue with SIM IK was that it did take object collisions and alternative approaches into consideration, thus it did the most optimal solution, but because of its reduced dimensional subspace, the solution often violated constraints of the environment.

OMPL (Open Motion Planning Library): -

It is an Open-source Motion Library used for robotic motion planning. It provides a collection of sampling-based planning algorithms such as **RRT (Rapidly exploring Random Tree)** and **PRM (Probabilistic Roadmap)** that help robots navigate complex environments while avoiding obstacles. It supports the gripper's joint-space motion allowing us to control our end-effector and provides a support system for our 6-DOF robot (UR-5).

Our Grasping Points Detection Algorithms identify the **optimal grasping edge**, after which the OMPL library takes over by orienting the end effector into the correct grasping position. Once aligned, the end effector picks up the object efficiently.

To show efficient control of end-effector and robot we have demonstrated 4 predefined paths using a combination of these paths and other complex motions our robot will move in different ways to reach to our destination position. The 4 predefined paths are as follows:

- Circle
- Square
- Triangle
- T-shape

These two modules ensure efficient picking of the object and then we use OMPL to define the path of the robotic arm from initial to final location. As end-effector control is not very high-dimensional and final state of the robotic arm assures no collision, we choose to use SIM IK for end effector control. The two libraries are linked with CoppeliaSim using Lua.

Grasping Points Detection:

Well with the help of SIM IK and OMPL library we are moving our robot and simulating our end effector to pick up the object. But to orient our end effector in such a way that it finds the best grasping points for picking up the object we developed the following two algorithms.

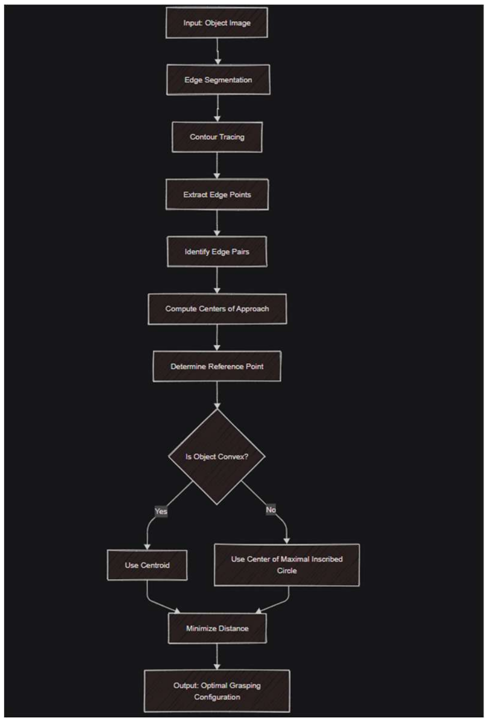
Anti Podal Points Detection: -

Inspired from “Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network” a research paper by Sulabh Kumra, Shirin Joshi and Ferat Sahin from Cornell University [10 June 2021] [here].

It utilizes a novel Generative Residual Convolutional Neural Network (GR-ConvNet) model, capable of generating robust antipodal grasps from an n-channel input in real time (~20ms). The model is pre-trained on a fine-tuned dataset from Cornell University, allowing it to accurately detect grasping points for new objects with high reliability.

This has two major drawbacks. It does not handle skewed angles of the images from the vision sensor, and the grasp point inference is computationally expensive, so we decided to switch to our own novel algorithm “Instant-GPD”.

Instant-GPD: **-**



Caption: Flowchart explaining Instant GPD Algorithm

The Instantaneous Grasping Point Detection (Instant-GPD) algorithm provides a robust solution for determining optimal grasping points across diverse object geometries, encompassing both regular and irregular morphologies.

The methodology follows a systematic approach:

1. Initial edge segmentation is performed on the object's shape
2. Edge points are extracted via contour tracing
3. Edge pairs are identified through an angle threshold criterion, where pairs consist of antiparallel edges
4. For each valid edge pair, a center of approach is computed as the midpoint between the constituent edges
5. The optimal grasping configuration is determined by minimizing the distance between each center of approach and a reference point (designated as **CENTER**) The reference point is defined as either:
    - The centroid for convex objects
    - The center of the maximal inscribed circle for concave geometries

This approach ensures a deterministic solution for grasping point identification regardless of object complexity.

We code these two algorithms in our MATLAB file and then using MATLAB’s remote API file we link the MATLAB and CoppeliaSim file to ensure proper movement and grasping of objects.

Alternative Solution (Grabbing using suction)

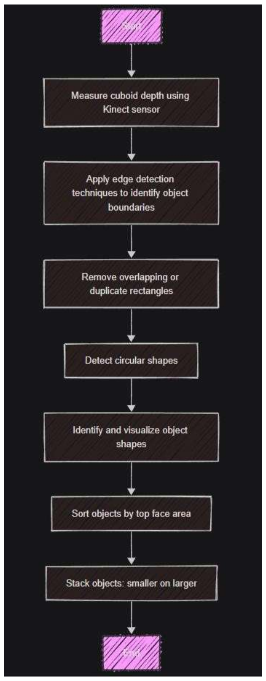
Suction-based gripping is one of the most fundamental methods for object manipulation. It operates by creating an airtight seal using suction cups and employing a vacuum system to securely hold objects in any orientation without the risk of detachment from the end effector.

However, this approach presented significant challenges, particularly in achieving an airtight seal on curved or irregular surfaces, which hindered the effectiveness of the gripping mechanism. Even in cases where the suction force was sufficient to grasp an object, there remained a high probability of the object dislodging during transit, compromising the overall efficiency of the system.

Additionally, ensuring a consistently strong grip required precise alignment between the robotic arm and the object's surface. This necessitated highly accurate control of the end effector’s positioning, adding further complexity to the implementation. Given these limitations, the inherent drawbacks of suction-based gripping led to the decision to abandon this approach in favour of more reliable alternatives despite their complexity

Stacking

- ### Regular Stacking


Start
Measure cuboid depth using Kinect sensor
Apply edge detection techniques to identify object boundaries
Remove overlapping or duplicate rectangles
Detect circular shapes
Identify and visualize object shapes
Sort objects by top face area
Stack objects: smaller on larger
End

Caption: Regular stacking explained via Flowchart

Stacking regular objects are a straightforward task.

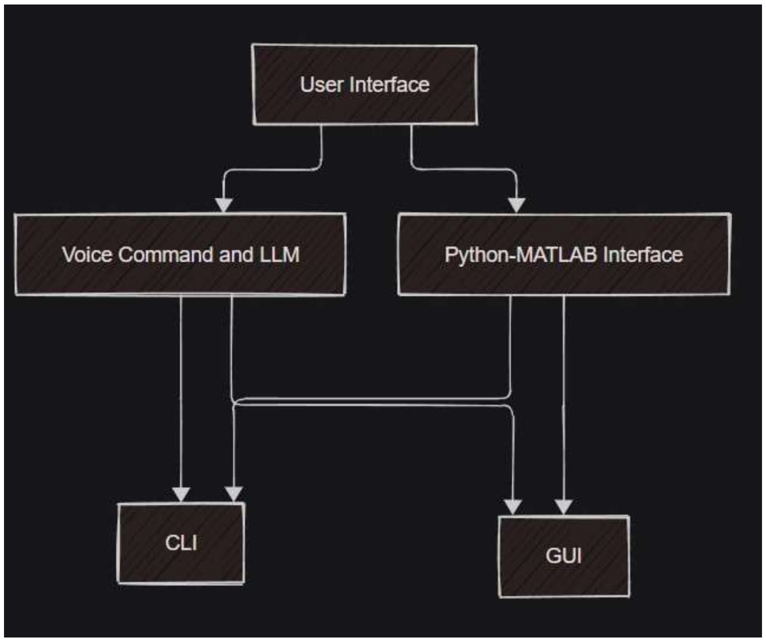
1. Initially, the depth of the cuboid is measured using a **Kinect sensor**.
2. **Edge detection techniques**, as previously described, are applied to identify the object boundaries.
3. To ensure accurate classification, overlapping or **duplicate rectangles are removed**. Additionally, circular shapes are detected.
4. Once the object shapes are identified, they are visualized, and stacking is performed by placing objects with smaller top face areas on those with larger top face areas, ensuring stability and optimal arrangement.

• Irregular Stacking

Irregular object stacking may appear complex, but we have devised the following systematic approach:

1. **Stackability Assessment**: Determine whether an object is stackable by identifying two parallel or opposite faces that are planar. Objects failing this criteria are classified as toppleable.
2. **Antipodal Point Detection**: Identify a set of antipodal points within a group to effectively verify the presence of opposite parallel faces.
3. **Object Placement**: Sequentially pick and position the stackable objects on top of one another.
4. **Handling Non-Stackable Objects**: Objects that do not meet the stackability criteria are classified as "toppleable."
5. **Final Arrangement**: "Toppleable” objects are placed at the top of the stack once all stackable objects have been positioned.

User Interface: -



Caption: Flowchart explaining the working of UI

We have integrated MATLAB and Python to create a backend for our user interface, with a custom buffer module facilitating seamless communication between the two technologies.

The system captures voice input through Google Speech Recognition, which converts speech into text. This text is then processed by custom Python modules that interact with Large Language Models (LLMs). We utilize the Mistral model, accessed via APIs from Groq, to interpret the text and convert it into executable commands.

The user interface is divided into two main parts: a terminal-based control system and a

graphical user interface (GUI). Both interfaces provide intuitive ways for users to interact with the system using voice commands.

Terminal based control: -

Terminal supports different commands like:

- **pick_up(obj)**: Picks up a specified object

- **stack()**: Drop the object to the stack
- **take_from_speech()**: Listen for voice commands till some valid command is recognized after which it will get terminated automatically.
- **get_list()**: Get list of available objects.
- **help()**: Show help message to assist new users.
- **Q or q or exit**: terminate the complete program.

GUI based control: -

GUI supports commands like:

- Picking up object which is discoverable to sensors attached.
- Dropping the object at any random position the manipulator is on.
- Stacking objects kept in view of sensors.

Voice – enabled GUI has been installed.

Difficulties faced:

SIM IK- Fails to find a path depending on the environment:

SIM IK plays a crucial role in our project, providing superior performance compared to traditional inverse kinematics solvers. However, it does not consistently ensure a collision-free path. Its tendency to prioritize optimal path computation can result in unintended collisions with objects in the environment. Due to these limitations, we opted to utilize the OMPL library for improved motion planning and collision avoidance.

External Path Planning- by integrating ROS MoveIt:

External path planning in robotics is inherently complex. Developing an efficient controller to interface with CoppeliaSim posed significant challenges. Initially, we attempted to integrate ROS MoveIt with CoppeliaSim for external path planning. However, after careful evaluation, we determined that foregoing external path planning was difficult due to time lags and a feedback requirement from Coppeliacontroller to jointstatecontroller in ROS that we dropped this idea.

Irregular Objects – A grasping challenge:

It was really a good challenge for us to encounter the irregular objects. Their unorthodox shape presented a big task in front of us to pick the objects using our end effector. The major difficulty in this was trying to find valid grasping points for the object; and even when they were found, they went out of range for our arm; but we have integrated OMPL and SIM IK, helping us manipulate our end-effector more effectively; and Instant-GPD, helping us identify ‘better’ grasping points for our end-effector; in such a way that they are picking up objects.

Results: -

We have successfully completed our assigned problem statement and developed a functional robotic arm capable of picking up objects based on voice commands and stacking them as required. The robotic arm can implement the following key capabilities:

- Voice – Activated control interface
- Precise manipulation of regular and irregular objects
- Customizable stacking functionality

To substantiate our achievements, we have compiled a comprehensive set of evidence, which has been submitted as a zip file. This package includes:

- Video demonstrations showcasing the arm's functionality
- Photographic documentation of successful operations
- MATLAB source code detailing our implemented algorithms
- CoppeliaSim simulation files for virtual replication and testing
- Python source code files for GUI and Voice Commands