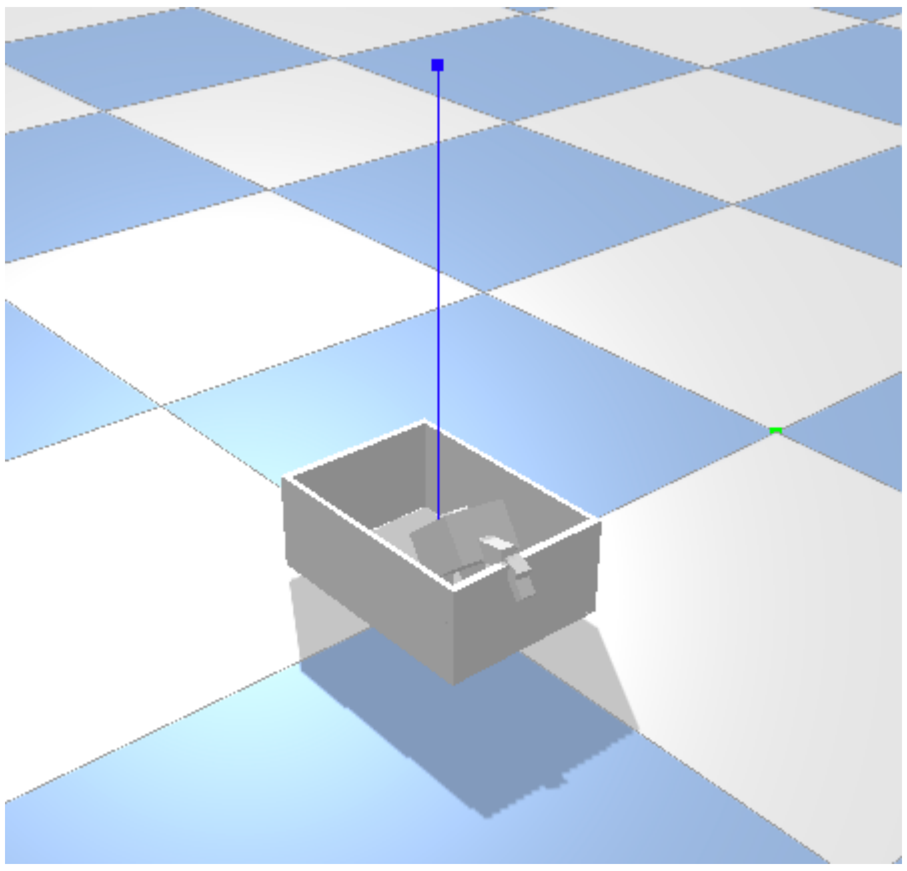
CSCI 4302 - LAB 4 - COLLISION AVOIDANCE



The goal of this lab is to populate the pybullet server with real-time information from the Webots world and check whether there is a collision with projected motions of the robot in order to determine suitable grasps. At the same time, you will need to compute suitable grasp poses given an object-of-interest's pose.

You are provided with an URDF files of the crate and the Robotiq gripper. You can use the gripper coordinate system to locate the crate and then perform relative motions with the gripper. You are also provided with a perfect object detection pipeline that segments all objects in the crate.

Steps:

- Gain an understanding of how to load URDF and basic geometric objects into the PyBullet physics environment
- Understand the two ways that you can compute collisions: computing the closest points between two objects (does not require simulation step) and computing the actual contact points.
- Compute an Open3D point cloud, down sample it and approximate it with cubic or cyldrical objects. Also populate the box URDF into the same coordinate system.
- Pick a single can, obtain its bounding object, and compute the position and orientation for grasping it. For this lab, you can limit yourself to grasping the cans at the center along its shorter width. Start with a simple world in which there is only one object to grasp and possible collisions are limited.
- Test whether the grasp is feasible or not. If yes, execute it. If not, pick a new grasp. If no grasp is feasible, you can agitate the crate.

Deliverable: A Python script that cycles through possible grasps and select a can it can grasps.

OVERVIEW

In this lab, soda cans are presented randomly in a crate. The cans can be identified by Webot's buil-in **object recognition** pipeline that augments the "camera" node by a **"segmentation image"**.

You will be furnished with relevant snippets from the **PyBullet tutorial** as well as URDF files for the robot gripper and the crate.

You will then use Open3D to identify the crate's and other object locations, populate the PyBullet simulation environment, compute possible grasps, test whether they are feasible in PyBullet and execute them otherwise.
This lab consists of two worlds: an environment with a crate full of randomly placed cans and another environment with just one can on the table.

REQUIRED FILES

- Download files from **Github**
- You will need to install the following libraries: open3d, matplotlib, PyBullet.

PRELIMINARIES

- Install the required libraries and work through the PyBullet examples.
- You might need to skim through the PyBullet tutorial to better understand what the library does.
- Work through the example that helps you in detecting the cans, extract the point cloude for a can segemented in a specific color, and compute the

grasp pose.

- Pay close attention to the way the centroid locations are translated into the robot's base frame. You will need to complete this computation.

  PyBullet does not have an obvious way to translate an object once it has been spawned in the simulator. Think about why this is.

FINDING THE OBJECTS

- Write code that extracts the centroid and orientation of the point cloud as a whole and use it to compute the relative orientation of the crate. Use this information to populate the PyBullet world. Assume that the robot's end-effector is at (0,0,0) of this world. You might need to disable gravity.
- Downsample the point cloud to a reasonable resolution, e.g. 5cm, and use it to populate the collision avoidance world with the crate's content.

COMPUTE A GRASP POSE

- Use the existing code snippets to determine a suitable grasp pose based off an object of interest.
- Compute grasp poses for the different cans based on the example given. You might need to experiment in a less complex environment using just one can on the table. Use transforms to compute the grasp pose from the object's centroid and orientation. This will require a translation, to reach the object from above, and a rotation, to make sure that the gripper is aligned correctly.
- Test your grasp pose by moving the robot there. The sample that you are provided already aligns the gripper's orientation with that of the object, but is facing upwards. You will need to add one more transformation to find the appropriate pose.

COLLISION AVOIDANCE

- Place the gripper at (0,0,0) of the collision avoidance world and mimick its orientation. Write code that tells you whether the gripper is in collision.
- Respawn the gripper at a potential grasp pose to test whether it is feasible.

PUTTING IT TOGETHER

- Combine object recognition, populating the collision world, computing grasps, and testing them.
- Loop through all of the possible objects (using the colors from the unique_colors array) and grasp the first feasible object.

DELIVERABLES

- A single Python file that implements your controller in the provided Webots world. You will be graded on collision avoidance and grasping implementation separately. This lab can be submitted as a team effort. Please indicate all team members in the source code. You are encouraged to create a git repository to facilitate collaboration and track contributions.