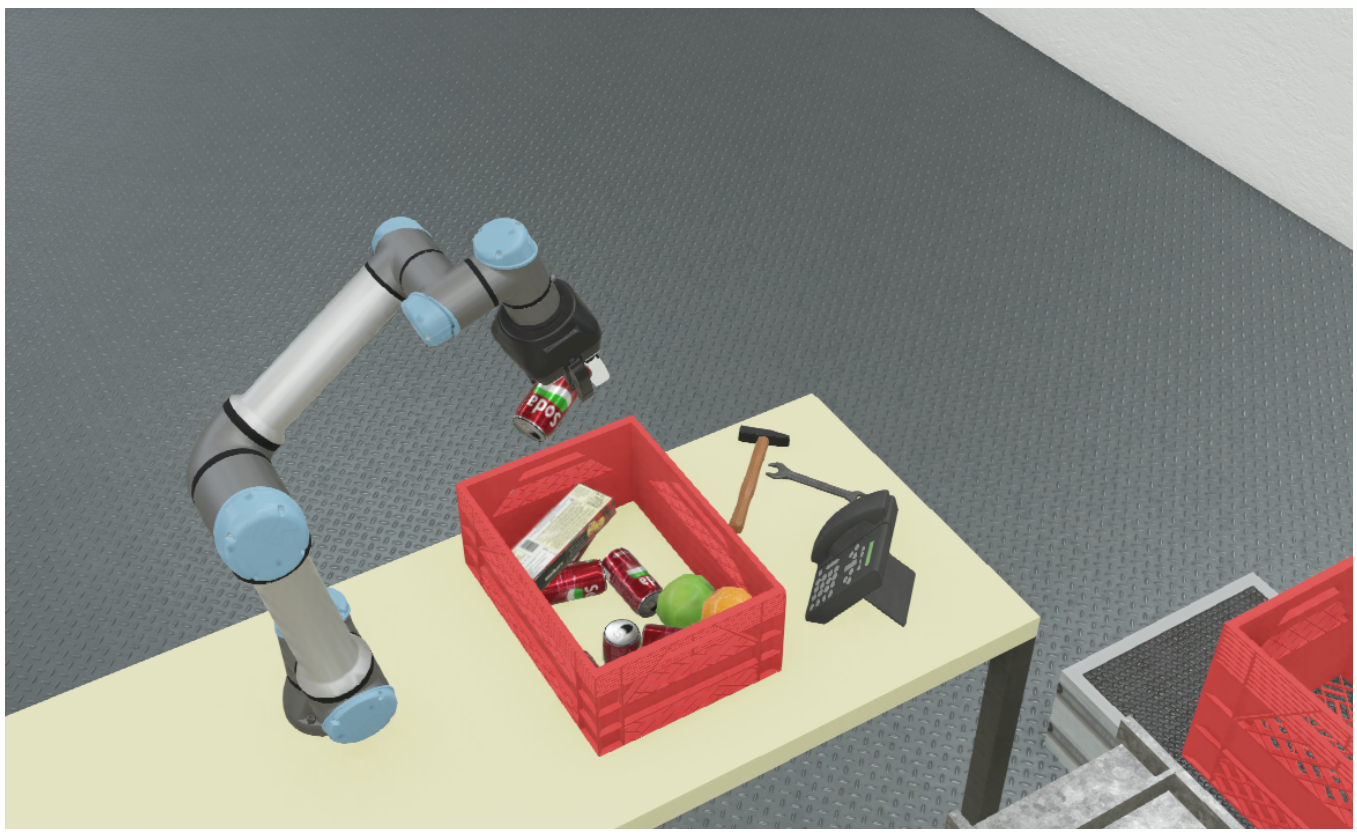
CSCI 4302 - LAB 6 - BIN PICKING



The goal of this lab is to combine inverse kinematics, object recognition, grasping and collision avoidance into a functional bin picking pipeline using “Behavior Trees”. In order to facilitate this process, you are provided with three iPython notebooks:

- jacobian_ik.ipynb: A demo of an inverse Jacobian solver by Michael Lauria. Examples include moving to a specific pose, moving along a coordinate axis and moving into a desired orientation.
- BT_demo.ipynb: A demo of a simple Behavior Tree that implements basic grasping using the conveyor belt world from **Lab 0**.
- binpicking.ipynb: A skeleton implementation for this lab consisting of a Behavior Tree that randomly grasps from the bin until an object is found. This implementation also serves as an example for “Sequence”, “Selector” and “Decorator” nodes.

Steps:

- Familiarize yourself with the inversion Jacobian method. Using inverse Jacobian is highly recommended to create smooth grasping motions.
- Understand how py_trees implements Behavior Trees and test them in a basic grasping setting.
- Further your understanding of BTs by inspecting the lab skeleton
- Integrate your solution from **Oriented grasping**, and optionally, your results from **Collision avoidance** and Object Recognition into the skeleton framework.

Deliverable: A Python script that cycles through possible grasps and grasps all items in the bin. If you chose not to use the provided libraries, please include all required methods into a single file.

REQUIRED FILES

- Download files from **Github**
- You will need to install the following additional libraries: py_trees.

PRELIMINARIES

- Install the required libraries
- Work through the provided examples to understand the inverse Jacobian solver and the Behavior Tree example
- Inspect and run the code in the binpicking notebook

BEHAVIOR TREES

The provided example runs until the “Object in Hand?” behavior returns “Success”. In order for preventing other behaviors in the sequence from triggering a “Success” status, they are wrapped in a “Success-is-Running” decorator. Note that behaviours do not accept parameters. Instead, parameters are implicitely passed via a “blackboard” mechanism is used to pass parameters between different behaviors.

COMPUTE A GRASP POSE

Use your code base from previous lab to create a new behavior “Find object” that returns a grasp pose. You can use the blackboard mechanism to store candidate grasp pose(s).

COLLISION AVOIDANCE

Optional: cycle through all possible grasp poses to find a collision-free pose. Otherwise, simply rely on the simulated F/T sensor to limit the impact of collisions with the bin and its content.

PUTTING IT TOGETHER

Extend the behavior tree to empty the entire bin and finish only when no object remains. Optional: use the arm to shake the bin to randomize its content and making grasping easier.

DELIVERABLES

A single Python file that implements your controller in the provided Webots world. 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.