RBE-550 Motion Planning
Daniel Montrallo Flickinger, PhD

Assignment 6
Theft

DUE: 2021-12-06 @ 12:00 UTC

Abstract

You find yourself at an abandoned gas station in the desert. Thirsty, yet bereft of any spare change, you spot a battered old vending machine. Can the magic of RRT conjure up a free soda from deep within the bowels of the machine?

1 Introduction

The soda starts trapped inside a chamber in the vending machine, and must be maneuvered past two barriers to get outside. Create an RRT type motion planner to steal the soda.

2 Environment

The environment consists of the vending machine interior, with the main chamber and two barriers connected. Each barrier has a window, just barely large enough to fit a soda can. These windows are misaligned, as an anti-theft measure that must be defeated.

The soda can is a standard 12 oz. aluminum beverage can, modeled as a perfect cylinder with a diameter of 52 mm and height of 122 mm. The main chamber of the vending machine is a cube 1 meter on a side. The first barrier is on the far wall, with an opening with its center 200 mm from the top. The second barrier is offset 250 mm from the first, with an opening with its center 200 mm from the bottom. Both windows are squares, 150 mm on a side, and aligned with the centerline of the inner chamber, along the axis normal to the barriers. Refer to the diagrams in Figure 1 and Figure 2.

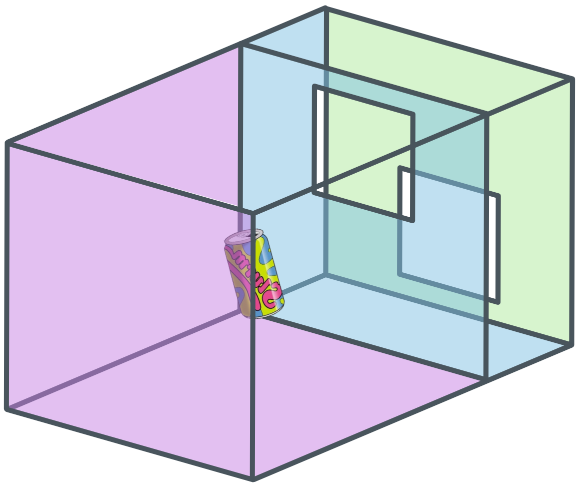

Figure 1: The vending machine workspace.

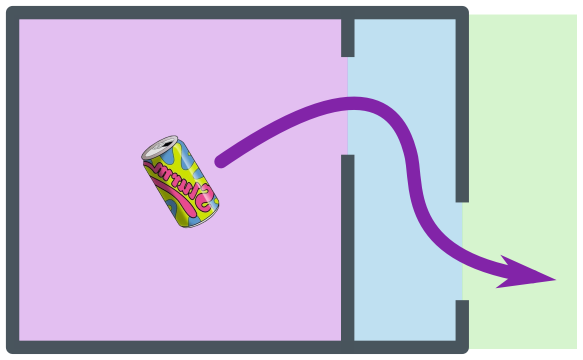

Figure 2: Side view of vending machine workspace.

The initial position of the can is offset 200 mm from the wall opposite the inner barrier, laying horizontal on the floor, with its axis of radial symmetry parallel to the barrier planes, and on the centerline of the interior chamber. The final position of the can is 500 mm offset from the outer barrier, outside the vending machine, also along the centerline. The final orientation has the can standing up, with its axis of radial symmetry aligned with the *up* vector.

3 Methods

Use an RRT algorithm, or a variant, to design a motion planner to transport a soda can in three dimensional space. The single rigid body has a full six degrees of freedom, and no constraints aside from those imposed by collision. In creating a tree, note that the configuration space is a full six dimensions, including both position and orientation of the can.

Use any software or programming language in your implementation, which should include the following components as detailed in Sections 3.1 through 3.3. Python, C++, MATLAB, ROS, or any other tools are acceptable. Third party libraries such as CGAL, OMPL, or others may be used for any component, including the planner implementation. Be clear in your report and delineate what you wrote and what you integrated from external sources.

Refer to Chapter 5 in LaValle [2006], and the course lectures for details on RRT implementation.

3.1 Planner

Implement or integrate an RRT-based planner that takes the initial and goal states as input. It should create a tree in full configuration space, and find a solution path. Note that a bidirectional RRT might be needed for this problem. With this planner implementation, the following sub-components are required:

- graph data structure (store vertices and edges)

- nearest neighbor search
- local controller (move the can in 3D space)
- sampling method

3.2 Collision Checker

The collision checker should check the current configuration of the can against the walls in the environment. Generally check for cylinder overlap with a bounded plane. Take careful consideration in modeling the passages through the barriers. Assume that collision checking is absolute, and that the barriers are perfectly rigid. Also assume that damaging the can through deformation contravenes the entire purpose of stealing a free drink.

3.3 Visualization

Visualization for this assignment is straightforward. Only basic 3D shapes need to be represented, plus lines to illustrate graph edges and path segments. Use tools like MATLAB figures, or plotting with Matplotlib in Python to produce figures and graphics. If you want to include fancy stuff like textures, shading, and lighting, more power to you. Writing this entire thing in a shading language would be time consuming, yet awesome.

4 Results

Produce two plots of the resulting path. Create one figure of the resulting path (position component only) as a 3D line. Choose any view orientation that clearly illustrates the characteristics of the path. Additionally, create an animation of the can traveling along the path (which should include the orientation component). A static plot showing the can at multiple interesting configurations along the path is acceptable in lieu of a full animation. Note that for this and the following plot(s), showing the barriers is helpful to add context.

In addition to the solved path, produce a figure depicting the RRT graph structure projected into 3D (and projected into a 2D figure). Choose a view orientation that makes sense to illustrate the structure of the tree. Include multiple figures if necessary.

Include full source code with your submission, along with a brief writeup about the development, implementation, and experimentation. Also include a photo of a popular soda/pop/coke/soft drink from your geographic region. (e.g. Polar)

5 Grading and Submission

This assignment is due 2021-12-06 @ 12:00 UTC. *Late submissions are not accepted.* Upload completed assignment components (as individual files, not a single ZIP or TAR file) to the course site on Canvas.

Weight	Type	Component
10%	PDF	path figure
10%	video,PDF	path animation
10%	PDF	RRT figure
20%	source code	complete source code
30%	source code	RRT planner implementation
20%	source code	collision checker implementation

6 References

Steven M. LaValle. *Planning Algorithms.* Cambridge University Press, May 2006. ISBN 9780521862059. URL `http://lavalle.pl/planning/`.

7 List of URLs

Last update: November 8, 2021