Jordan Ramsdell **jsc57** **March 5, 2017** **Primitive Specs**

Note: skip to sections 4 and 5 if you just want to use my Python functions instead of making your own. You just have to know where the primitives file and primutils.py are.

1. Discretization

One of the goals of generating these action primitives is to precalculate applying them to a state. Because there are potentially an infinite number of states and controls that we could apply, we must discretize the states and primitives. To use the following action primitives, you should know the following about the current state: heading (in radians), linear velocity (in m/s), and angular velocity (in radians).

Each motion primitive refers to a particular linear and angular acceleration that is applied to the current state over 250 milliseconds. What is calculated is the relative coordinates (in map units) of the robot after the motion primitive is applied, along with a new heading and a set of cells that the robot passes through during the action.

Again, we can't calculate this for each and every possible state, and so we must discretize them:

State Velocities: $0.0, 0.3, 0.6, 0.9, 1.2$

State Headings: $0, \frac{\pi}{8}, \frac{2\pi}{8}, \frac{3\pi}{8}, ..., \frac{15\pi}{8}$

State Angular Velocities: $-\frac{2\pi}{4}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{2\pi}{4}$

Primitive Accelerations: $-0.3, 0, 0.3$

Primitive Angular Accelerations: $-\frac{2\pi}{4}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{2\pi}{4}$

Note that in this current version negative linear velocities are not considered.

2. File Format

The primitives are stored in the file primitives.txt. The lines are formatted as follows:

Line 1: Action duration expressed as fractions of a second.
Line 2: Linear Velocity Divisor (see below)
Line 3: Angular Velocity Divisor (see below)
Line 4: Heading Divisor (see below)
Line 5+: Action Primitives

Action primitives have tab-separated columns that are formatted as follows:

Column 0: Name of the primitive (Example: "a0")
Column 1: The primitive's linear acceleration (Example: 0.3)
Column 2: The primitive's angular acceleration (Example: $\frac{\pi}{4}$)
Column 3: State's initial linear velocity key (Example: 1)
Column 4: State's initial angular velocity key (Example: -1)
Column 5: State's initial heading key (Example: 10)
Column 6: Relative x-location (in map units) of robot after action is finished

Column 7: Relative y-location (in map units) of robot after action is finished
Column 8: Heading (in radians) of robot after action is finished
Column 9: Relative coordinates of cells that robot passes through, space separated. (Example: (0,1) (0,2))

3. Using the Primitives

If you're just interested in the primitive names and their associated acceleration controls, just take the first three columns (filter by unique, as there will be duplicates). Otherwise, if you want to take advantage of the precalculated heading, coordinates, and cells to check for collision, you should create a dictionary.

The keys of dictionary are whatever you prefer, but I suggest a tuple of columns 0, 3, 4, and 5. To use this dictionary, you would need to convert your state's linear velocity, angular velocity, and heading into keys. This is where the divisors (lines 2, 3, and 4 of the file) come into play.

The first key would simply be the name of the motion primitive you are applying.

The second key would be: int(state linear velocity / linear velocity divisor)

The third key would be: int(state angular velocity / angular velocity divisor)

The fourth key would be: int(state heading / heading divisor)

There may be cases where you will not find an entry in your dictionary using the above keys (for instance, no entry exists for a state with a velocity of 1.2 applying a motion primitive with an acceleration of 0.3, because this exceeds the robot's max speed of 1.2).

4. primutils.py

To facilitate using the primitives file, I've create some utility functions. You can import the Primitive class and the read_primitives function from this script.

read_primitives(filename):

As an argument, you pass the filepath pointing to the location of the primitives file. It will parse the file and return a list of Primitive objects.

Primitive:

The linear acceleration and angular acceleration of each primitive are stored as attributes (va and wa respectively). There are two useful methods you can call:

get_entry(state_v, state_w, state_h)

You pass the state's velocity (in m/s), angular velocity (in radians/s) and heading (in radians) as arguments. The primitive then applies its accelerations to the state and tries to retrieve a precalculated entry if it exists (otherwise it returns None).

The entry is a 4-tuple containing:
Index 0: The relative x-coordinate (in map units) of the robot after the action is finished
Index 1: The relative y-coordinate (in map units) of the robot after the action is finished

Index 2: The heading (in radians) of the robot after the action is finished
Index 3: A list of (x,y) relative coordinates of the cells that the robot passes through during the action.

apply(x, y, state_v, state_w, state_h)

You pass the state's x, y, velocity (in m/s), angular velocity (in radians/s) and heading (in radians) as arguments. The primitive then applies its acceleration to the state and tries to calculate the robot's new position and heading.

It returns a 3-tuple containing:
Index 0: x-coordinate (in map units) of the robot after the action is finished
Index 1: y-coordinate (in map units) of the robot after the action is finished
Index 2: heading (in map units) of the robot after the action is finished

5. prim_demo.py

This script is a quick demonstration of how to use primtutils.py You can run it in the primitives directory as follows: ptyhon3 prim_demo.py primitives.txt Look at the code to see how things are called.

6. create_primitives.py

This is the script that is used to generate the primitives. Currently I didn't make it configurable (work in progress), but you can edit the global values to change how it generates them. The generated primitives are printed to standard out, and I went ahead and wrote them to primitives.txt.