Agent Modeling in Python

J. Adam Fidler

April 2020

Abstract

This paper provides an introductory overview of a simulation framework for Python called Snakes on a Robot, and its use in modeling a simple cognitive agent. The source code for this project can be found at https://github.com/fidlerja/soar.

1 Snakes on a Robot

Snakes on a Robot, referred to by the developers as Soar (not to be confused with the cognitive modeling architecture Soar, developed at Carnegie Mellon University [1]), is a framework for simulating and interacting with robots in Python [2]. Soar is an object oriented Python package that defines robots, brains, and worlds. A user can load a robot brain, the simulated world in which the robot resides, and visualize the world in a simple graphical user interface (see Figure 1). Soar also supports interfacing with physical robots via hardware input/output devices and network connectivity, however these are beyond the scope of this project.

1.1 BaseRobot

Every Soar robot object has a number of basic attributes and methods. The core attributes include the robot's pose, given as ordered triples in the form of (x, y, θ), where x and y are the coordinates of the robot in the world, and θ is the angle it faces, measured in radians. The robot also has attributes that store its forward and rotational velocities, measured in meters per second and radians per second, respectively. Each robot has basic methods that allow it to move, detect collisions, and process commands on every startup, pause, and time step.

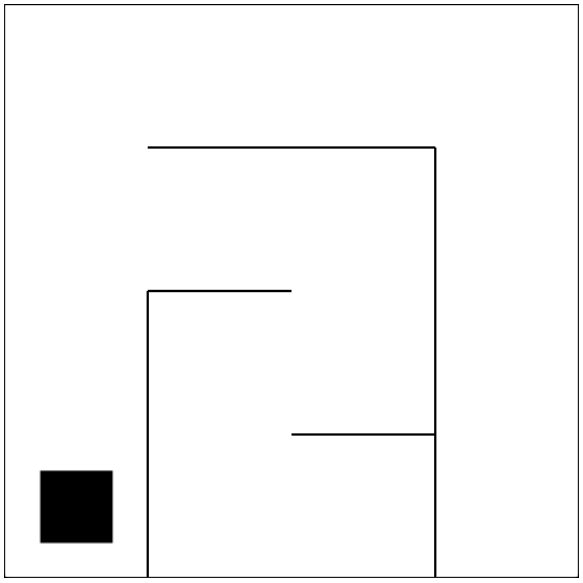


Figure 1: A BaseRobot in bigFrustrationWorld

1.2 MyRobot

The goal of this project was to create an agent capable of navigating an environment and fulfill a specific objective. The `MyRobot` agent is designed to search its world, avoid obstacles, and reach the target block (designated by the color purple). The `MyRobot` class is a custom adaptation of the `PioneerRobot` class, which is designed to interface with the Pioneer 3 P3-DX robot, a small lightweight robot equipped with sonar, developed by Adept MobileRobots [3]. In addition to the attributes and methods inherited from the `BaseRobot` class, `PioneerRobot` use a `sonars` attribute to take advantage of the Pioneer 3's eight sonar relays positioned on the forward section of the chassis at varying angles. This attribute lists the distance of each sonar (modeled as a simple beam in the Soar simulator) to the nearest obstruction, where the indices enumerate the robot's sonar relays.

Because `PioneerRobot` has no sensor facing directly forward, a ninth central sonar was added to `MyRobot` in order to better track the target block. Moreover, `MyRobot` has additional attributes and an additional method called `is_color` that checks whether a particular object in sonar range is the target color (purple in this case). Some additional attributes added to `MyRobot` in order to facilitate objective tracking include `target_found`, `target_heading`, `target_headings`, and `success`. The robot's `target_found`

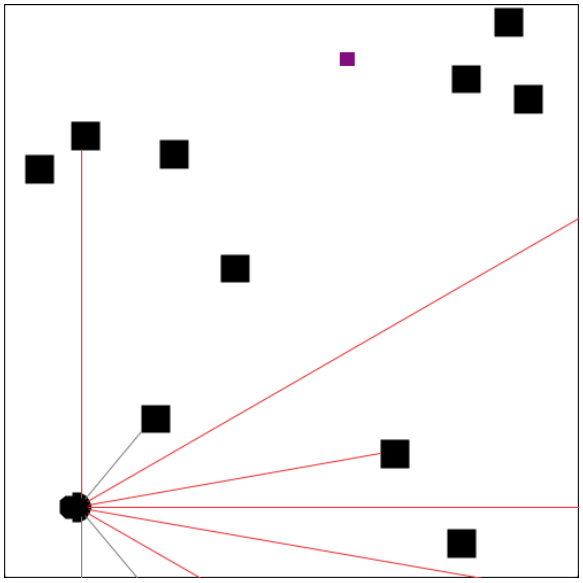


Figure 2: MyRobot in bigRandomWorld (random seed = 20)

boolean notes whether the robot as located the target block at any point during the simulation. The `target_heading` and `target_headings` attributes are an integer and an unordered set of integers, respectively, that record which sonars intersect with the at the current time step. The robot then uses the `success` attribute to determine whether to stop moving and end the simulation.

2 The Model

The Soar package comes with only a `blankBrain` class that initializes a `BaseRobot`. This was used as a template to create the `my_brain` agent used to locate the target block. `my_brain` starts by initializing an instance of the `MyRobot` class. For each time step, the robot analyzes its surroundings and makes a decision that ideally puts itself closer to the target block.

At every iteration, the agent first determines whether it has collided with an object via the robot's `collided` attribute. If so, it sets the forward velocity to -0.5 meters per second in order to reverse and sets `collided` to `False`. Otherwise, it continues forward at 0.5 meters per second.

The robot then checks for obstacles within 0.3 meters of Sonars 2 through 6, the sonars that form a forward-facing arc of $\frac{\pi}{3}$ radians. If there are any ob-

jects aside from the target (see Listing 1), as indicated by the `target_headings` attribute, then it turns away from the object at a rotational velocity of $\frac{\pi}{3}$, $\frac{\pi}{2}$, or π radians per second, drawn randomly from a uniform distribution (to avoid steady states), but continues forward if there are no obstructions detected.

```
# produce a boolean list of sonars detecting nearby obstacles
proximity = robot._sonars[2:7] <= .3

# shift indices (since 'proximity' indexes from 2 to 7)
target_headings = robot._target_headings - 2

# tell sonars detecting the target block to ignore it
proximity[target_headings] = False
```

Listing 1: Ignore target block when calculating nearby obstacles (otherwise the robot would redirect away from the target as it approached)

However, if `target_found` is `True`, then the robot centers the heading on the target block. This is accomplished by turning the robot until the `target_heading` attribute changes to `4`, at which point the robot increases its speed to 1 meter per second.

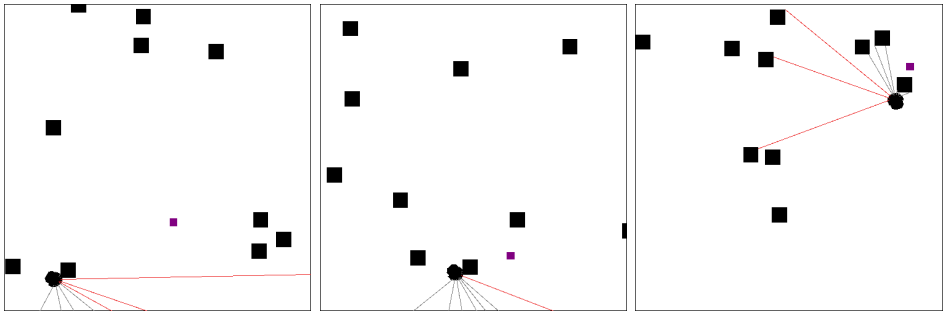
```
if robot._target_found:
    # get the index of the sonar that detected the target
    heading = robot._target_heading
    # turn robot until the target is directly ahead
    if heading < 4:
        new_rv = pi/5
    elif heading > 4:
        new_rv = -pi/5
    else: # target is directly ahead
        new_rv = 0
        robot._fv = 1
    # update rotational velocity
    robot._rv = new_rv
```

Listing 2: Algorithm to set robot heading for target block

3 Results

The cognitive agent as described above, while far from optimal, performed quite well under most moderate conditions. It was able to navigate to the target block quickly and consistently with little difficulty. Some situations

proved too difficult for `MyRobot` to navigate, particularly when the target block was generated immediately behind an obstacle. In these cases the robot was usually not maneuverable or smart enough to go circle around the obstacle, causing it to become stuck indefinitely (see Figure 3). Solving this would require a better avoidance heuristic.



(a) Random seed = 100 (b) Random seed = 111 (c) Random seed = 92

Figure 3: Robot is unable to navigate around some configurations

Another issue is that the robot is able to detect the target block even when the sonar is obscured by an obstacle. This causes the robot to move directly toward the target block, despite the lack of visual, which often results in a similar situation as in Figure 3. This "x-ray vision" is a byproduct of the Soar package's collision detection functions. Even though the sonar beams are drawn up until they collide with an object, under the hood they still propagate through the object, meaning sonar-target collisions still register through obstructions.

4 Conclusions and Future Work

The described agent works surprisingly well, considering it's relatively simple navigation heuristic. The Soar Python package has its shortcomings, however it is intuitive to implement, simple to modify, and easy to generalize.

`MyRobot` is nowhere near optimal, and no attempt to produce an optimal agent was attempted in this project. However, much work has been done in path optimization, especially in navigating surfaces [4] and obstacles [5]. Future iterations of `MyRobot` may include such optimization methods. Experimentation of different proximity parameters, velocities, and collision handling are also likely to produce better results.

References

[1] J. E. Laird, *The Soar Cognitive Architecture*. The MIT Press, 2012.

[2] A. Antonitis, "Soar: Snakes on a robot," 2019.

[3] A. T. Inc. http://www.mobilerobots.com, 2011.

[4] Z. Chase, R. S. Collings, J. A. Fidler, and E. Potokar, "Optimal controls of a vehicle on a surface," 2020.

[5] T. Mansfield and E. Todd, "Optimal path planning," 2020.