CSE 291 HW 5

Thomas Liu

Architecture

Subsumption

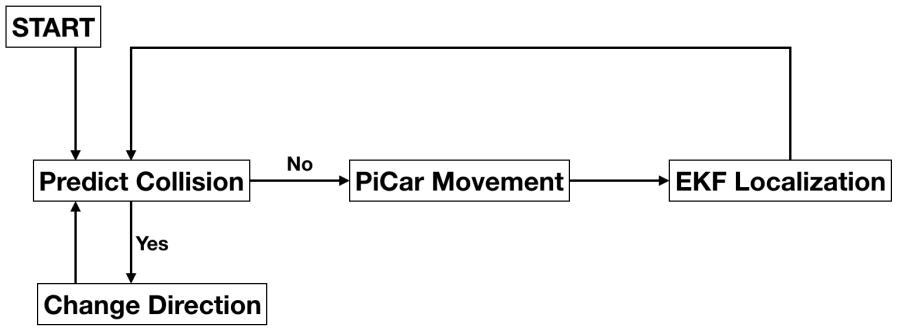
The basic idea was to just use random movement while avoiding crashes. When the PiCar is about to crash, it instead changes direction and then randomly chooses whether to turn its wheels left or right and for how many time steps. After the randomly chosen number of time steps has passed, the front wheels of the PiCar are returned to a straight position.

EKF Localization

The extended Kalman filter was used to improve localization of the PiCar, which was crucial in avoiding obstacles and boundaries.

Diagram

A simple diagram of the basic steps involved is shown below. At each time step, the PiCar makes a prediction of where it is going to end up in the next time step and whether that would lead to a collision or not.



Behavior Description

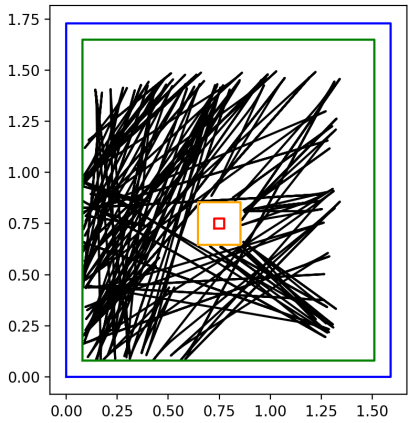
The behavior resulting from the subsumption architecture that I implemented was semi-random movement in mostly straight lines. The PiCar also would travel back and forth in a single area a lot, which would lead to staying in one area for a long time before moving on to a different area. This was mostly due to the PiCar not changing orientation very much upon changing direction, if at all (choosing no change in orientation was purposefully put in to avoid the possibility of getting stuck in a corner).

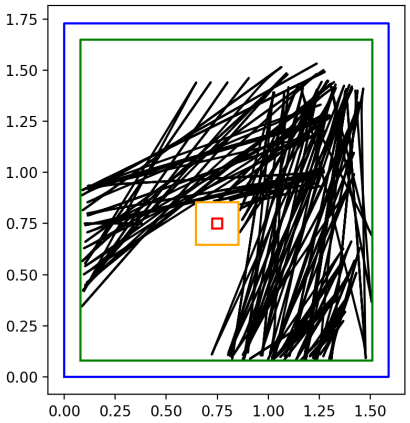
Performance

Simulated Trajectories

For trajectory simulation, I tried setting different amounts of possible orientation changes. A single orientation change is equivalent to letting the front wheels of the PiCar go left or right instead of straight and moving forwards or backwards for 0.1 seconds. All of the runs were done over 5000 time steps. Each time step in the real world would take about 1 second, so in an actual physical run, they would take approximately 1 and a half hours.

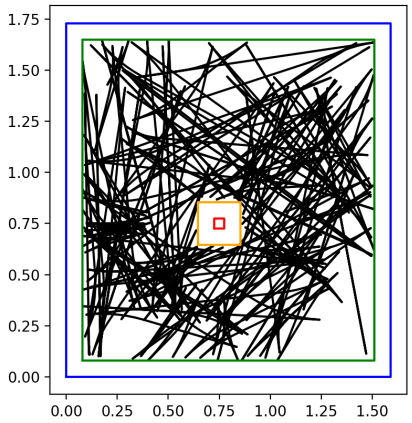
The image on the left below shows the trajectory for a run with 6 possible orientation changes (anywhere from 0 to 5 turns before going straight). The obstacle is shown in red, and the boundaries are shown in blue. The orange and green lines are the padded obstacle and padded boundaries, which are used to help prevent collision. It covers most of the area (keep in mind that the plot is of the back wheel so the areas at the top are actually covered by the front of the PiCar), but misses some spots like towards the bottom middle.



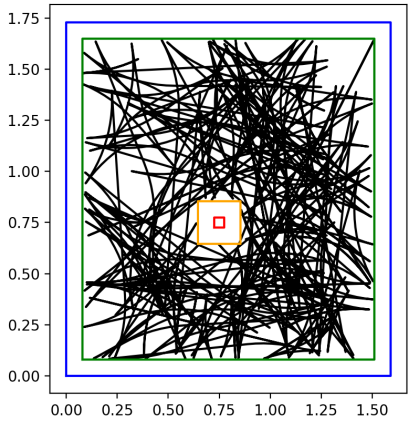


There were also runs where the trajectory tended to mostly stay in a few particular places while completely missing other places like the image shown on the right above.

The image below shows the trajectory for a run with 11 possible orientation changes (anywhere from 0 to 10 turns before going straight). It does a better job at covering the area, but more importantly it changes places much more often, so it would probably take a lot less time to cover most of the area.



The image below shows the trajectory for a run with 26 possible orientation changes (anywhere from 0 to 25 turns before going straight). It does a good job covering the area, and is more evenly distributed, which would probably mean an even lower amount of time needed to cover most of the area.



The problem with using a higher number of possible orientation changes is that it requires the PiCar to have a very accurate localization. Since the PiCar's localization isn't that reliable, I did an implementation with only 6 possible orientation changes.

Video Demonstration

The video (run.mp4) shows a successful run for 5 minutes of the architecture I implemented (using 6 possible orientation changes). The PiCar was able to avoid boundaries and an obstacle for the entire duration. A plot of its EKF-corrected trajectory is shown below.