CS747 Assignment 3

Sarthak Mittal

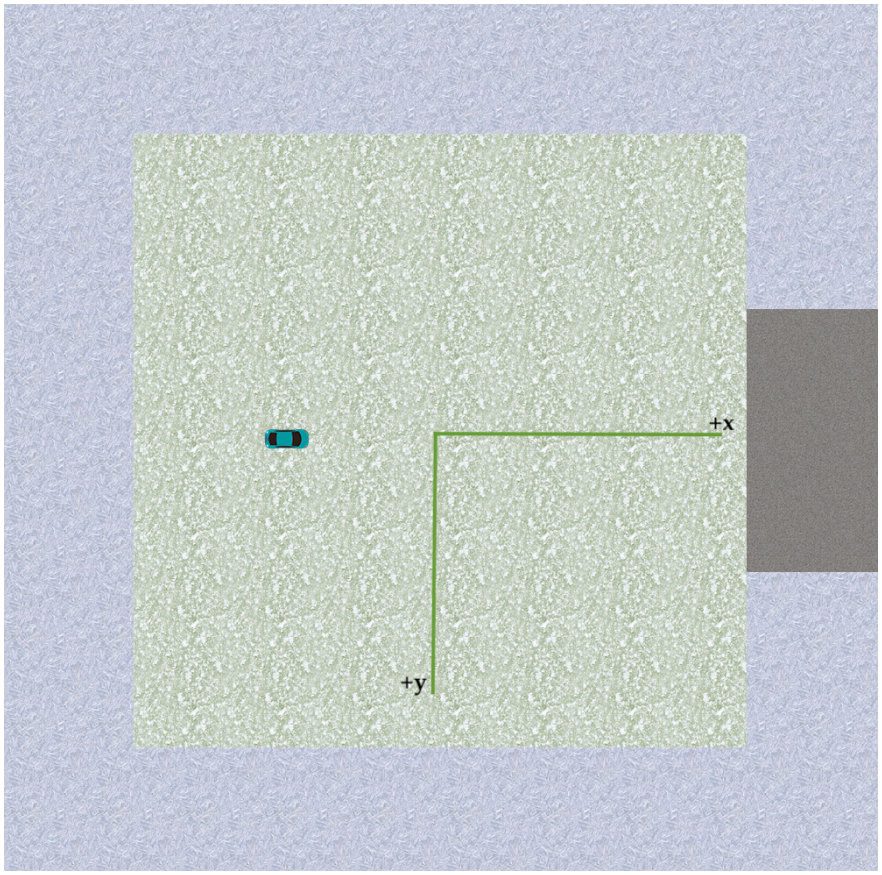
November 5, 2022

Contents

1 Task 1: Parking Lot Problem

1.1 Setup

In this task we simulate a car placed in a parking lot. We need to navigate it by controlling steering and acceleration and take it out on the road. The figure below shows a sample of the driving/simulation environment generated using `envs/environment.py` and `envs/driving_env.py`:



Task 1: A car in a parking lot

1.2 Algorithm

Since the field is empty (no obstacles) and the center of the road is a fixed point, we can proceed to make the car **"chase" the end point** for deciding the next action (steering and acceleration). For chasing the end point, we first shift the coordinate frame such that center of the road is the origin for ease of calculation. Then we first compare the orientation of the car with the angular position, and determine the direction and amount of **steering required to align it** with the straight line path to the end point (with some small angular threshold which equals the maximum steering in one time step). Once aligned, we stop steering and start **accelerating**. Thus, the car moves at least a bit closer to the end point. Now the calculations repeat at each turn, and the car essentially **traces the path from its start point to the center of the road**. The function `chase` uses the `state` and the end point $(\texttt{x}_0, \texttt{y}_0)$, and determines the next steering (clockwise or anti-clockwise) and acceleration (slow down, 0 or go faster) to move towards it.

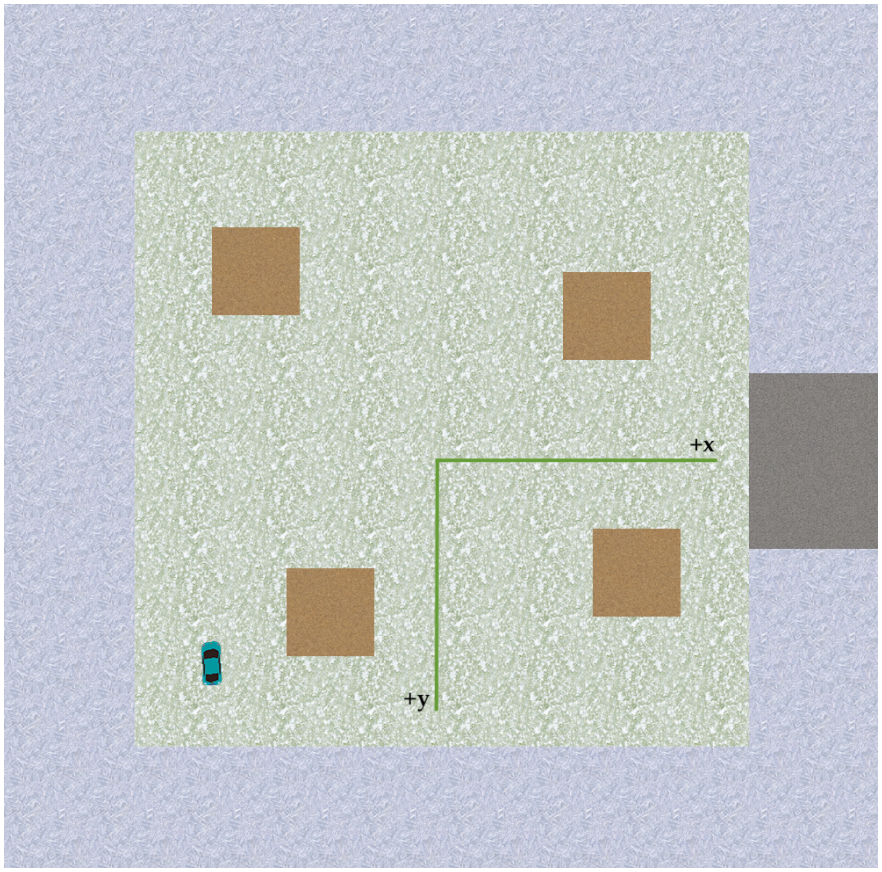
1.3 Simulation

The recording of my simulation for this task can be found here (Google Drive).

2 Task 2: Parking Lot with Puddles

2.1 Setup

In this task the setup is similar to previous task, but we also have puddles in the parking lot that we cannot let the car step into. The figure below shows a sample of the parking lot:



Task 2: A car in a parking lot with puddles

2.2 Parameters

The common parameters are as follows:

2.2.1 pits

Stores the coordinates of the randomly generated pits.

2.2.2 thres

The threshold distance to keep between the pit edges and the car.

2.2.3 steps

The number of steps to move while steering away from a pit.

2.2.4 count

The counter for the number of steps moved while steering away.

2.2.5 safe

The safe point to chase while steering away.

2.3 Utilities

Some utility functions to assist in the algorithm are implemented as follows:

2.3.1 rotate

Rotates the point $(\texttt{x}_1, \texttt{y}_1)$ about the point $(\texttt{x}_0, \texttt{y}_0)$ by angle `th`.

2.3.2 inPit

Checks if the position $(\texttt{x}, \texttt{y})$ is close to a pit, and returns the coordinates of the pit also if so.

2.3.3 path

Gives equally-spaced points on the line segment between the current position `state` and given point $(\texttt{x}_1, \texttt{y}_2)$.

2.3.4 nextValid

Finds the farthest safe point on the path decided to take.

2.3.5 steerAway

Diverts the car to a safe point so that pit can be avoided.

2.4 Algorithm

Since the field has obstacles, we proceed to make the car **"chase" the farthest safe point** for deciding the next action (steering and acceleration), **evading pits** on the way.

2.4.1 Chasing

We first determine the **farthest safe point** on the path from the current position to the end point using the utility functions. The similar to task 1, we **chase** that point using the `chase` function. If there is a pit encountered, we move on to **evading** it.

2.4.2 Evading

When we encounter a pit, we use the utility functions to **steer away** from it. Essentially, we first locate the center of the car and the pit, and then choose the point that is in a direction at **right angles** (clockwise or anti-clockwise depends on position on the board) to the line segment joining these centers. We then rotate the pit center about the car center and move to a point in that direction for **some steps**. After we reach this point, we go **back to chasing the end point**.

2.5 Simulation

The recording of my simulation for this task can be found here (Google Drive).