Report AI II

Alessi Roberta S4849209
Azzini Matteo S4475165
Deshini Suraj S5066029
Fiasché Enrico S4482512

July 2021

1 Abstract

The aim of the assignment was to make us familiarize wih modeling integrated task and motion planning problems. The assignment was based on the coffee shop scenario of Assignment 1 in which the waiter robot had to serve orders in an optimal way.
In order to perform this assignment, we needed to install the popf-tif planner. This planner's goal is to implement a domain dealing with movements of a mobile robot in a certain environment using PDDL 2.1. The domain folder contains a PDDL domain file consisting of a single action and a corresponding problem file where a robot has to visit four regions.

2 Introduction

As a requirement of the assignment we are asked to write an additional action in the PDDL domain file that computes the Euclidean distance travelled between 2 regions. The additional action that needs to be defined should compute the action cost, which is the distance traversed by the robot.
On computing the distance, additionally the cost due to uncertainty should be computed.
Assuming any motion model for the robot, the final robot state using Extended Kalman Filter (EKF) or any other estimation technique is estimated.

3 Domain file

Predicates:

We defined only the basic predicates related to the known information, such as "robot_in" to localize the robot among the different regions, "visited" to identify whether or not a certain region is visited and "last_path" to have a track of the path traveled.

Functions:

- canGo
  It returns True if its value is greater than 0, or False if its values is lesser than 1.
- act-cost
  It is the actual cost of the total path performed by the robot.
- triggered
  It is used to trigger the path performed by the robot changing its value to 1 while the robot is moving.

- dummy
  It is a variable used to update the actual cost, adding the partial cost, each time a motion is completed from one region to another.

  The last three functions are called in the VisitSolver.cpp file and are required to compute the distance traversed by the robot between two regions.

Actions:

- goto_region
  It's a durative action. The duration was provided by the assignment. As condition, for the action to happen, we firstly control the condition CanGo being false. Once the robot is moving to a different location, the function CanGo becomes true. The action ends when the robot reaches a certain location, that is now marked as visited.

- compute_cost
  We set it as a durative action. The action take place once the previous actions is ended, in fact it start when the robot is still in the last location he visited. The function canGo, set as condition, is true in this case. As effect we have that the value of the function triggered is increased by one. The action ends when the robot is not still in the last location anymore and this means that the previous action has started again. At the end, the value of the function triggered is initialized to zero.
  When the function ends as effect, both the function act-cost and dummy increases.

4 Motion Planning

The Motion Planning of the robot was coded in the VisitSolver.cpp. The robot in this exercise is created in such a way that it can plan paths for paths between the starting point and the goal. As a requirement of the assignment since we do have to consider any motion model for the robot, that is, the odometry model or velocity model, to estimate the final robot state, we have used the estimation technique that uses the Extended Kalman Filter.
The particular cpp file contains the code for localization which allows more complex robot models in order to move from one point to another point.
The first requirement was to update the evaluation of the actual cost computing the Euclidean distance between the starting and final region of robot motion, so we added the function localize(from,to) to compute that distance. This value is assigned to "dummy" variable, which update the actual cost each time a motion is completed.

EKF (Extended Kalman Filter)

Extended Kalman Filter is a part of the estimation theory wherein it is the non-linear version of Kalman filter which linearizes about an estimate of the current mean and the covariance. In transition models, EKF is considered the de facto standard in the theory of nonlinear state estimation, navigation systems and GPS. While solving the problem related to the EKF, we considered our problem solving similar to that of the principle used in GPS.
In the EKF, the state transition and the observation models need not be linear functions of the state but may be differentiable functions.
The Prediction equations include the predicted state estimate and the predicted covariance estimate which are give by the equations:

$$\begin{cases} x(k) = x(k-1) + steps(k-1) * cos(\theta)(k-1) + Q \\ y(k) = y(k-1) + steps(k-1) * sin(\theta)(k-1) + Q \\ \theta(k) = \theta(k) + \delta\theta(k-1) + Q \end{cases}$$

$$P_K^- = A * P_{init} * A^T + Q_{alpha};$$

The prediction steps are exactly the same as that of a Kalman Filter. It does not matter where the data is coming from.

For the localizing section, we initialized two sets of coordinates in total defining the *from* and the *to* positions. The distance travelled is calculated using the distance formula and the orientation of the robot prior to movement is defined as *dth* which is the defined by the inverse trigonometric of the difference between the y-coordinates of the to and from positions over the difference between the x-coordinates of the to and from positions.
The odometry equations are set are set for the various coordinates in order to calculate the predictions for x and y coordinates which move across a minimal value of steps (which are the segments into which the path has been split). The step values that we took into consideration here was 0.05. After measuring the prediction covariance, a matrix was established which determined the position of the robot and based on the state derivative and gain matrix a final prediction covariance was obtained.

Assumptions

In the prediction phase the angle thodom is taken as the pre-orientation value.
The Q_{alpha} and the Q_{gamma} which are the noises matrices in the function startEKF were the diagonal values were very considered very small.
These assumption was taken by us due to the lack of information and uncertainty.

5 Building and running

As mentioned, the planning used is popf-tif, which needs to be installed together with this package. This package uses Armadillo tools to optimize the matrix computations, please install it as well and pay attention to all the dependencies it needs. Lastly, the semantic attachment library can be built going into the directory:

visits/visits_module/src

Then should be executed the following command:

./buildInstructions.txt

Once the building process completes, the planner can be executed moving the executable file *popf3-clp* in the *visits/visits_domain* and executing the command:

./popf3-clp -x -n -t10 dom.pddl prob.pddl ../visits_module/build/libVisits.so region_poses

The -x flag is used to inform the planner that we are going to pass it a library with the external solver description; the -n -t10 flags are used instead to enable the anytime planning modality, making the system run for up to t seconds (here 10), which allows to expand the first solution found, hoping to discover

5.1 Link to repository

https://github.com/Mattazzo/AIRO2_second_assignment

6 Testing and Results

landmark.txt, waypoint.txt and the region_poses are files that describe the scenario and show the impact of the covariance matrix on the chosen path. They can be modified in order to see the different scenarios.

In the picture below is reported the optimal plan estimated using the Kalman filter.
At first, the robot moves from region_0 to region_4. By knowing the odometry of the system we could compute the Euclidean distance and the EKF matrices related to the travelled path. The result of the trace of the covariance matrix provides us the cost due to the uncertainty for each locations reached by the robot. In this way we can keep track of the covariance matrices for every chosen path. As previously stated, in the dummy variable are stored the computations made in the external module part, in particular this variable is assigned to action cost function.

Its values is continuously upgraded for each visited location and it's clearly noticeable by looking at the increasing values of the action duration.
Once the robot has finished its motion among the various regions, the plan gives us information also about the final cost, which is given by the sum between the cost due to the distance and the cost due to the uncertainty.

```
Initial heuristic = 8.000
b (7.000 | 100.000)
Euclidian distance = 2.000
Trace of covariance matrix = 0.492
b (6.000 | 100.002)b (5.000 | 200.003)
Euclidian distance = 2.828
Trace of covariance matrix = 0.564
b (4.000 | 200.005)b (3.000 | 300.006)
Euclidian distance = 2.828
Trace of covariance matrix = 0.565
b (2.000 | 300.008)b (1.000 | 400.009)
Euclidian distance = 2.828
Trace of covariance matrix = 0.565
;;;; Solution Found
; States evaluated: 17
; Cost: 12.671
; External Solver: 0.000
; Time 0.01
0.000: (goto_region r2d2 r0 r4)  [100.000]
100.001: (compute_cost r2d2 r0 r4)  [0.001]
100.003: (goto_region r2d2 r4 r3)  [100.000]
200.004: (compute_cost r2d2 r4 r3)  [0.001]
200.006: (goto_region r2d2 r3 r2)  [100.000]
300.007: (compute_cost r2d2 r3 r2)  [0.001]
300.009: (goto_region r2d2 r2 r1)  [100.000]
400.010: (compute_cost r2d2 r2 r1)  [0.001]
```