Technion – Israel Institute of Technology

HW5

Autonomous Navigation and Perception

086762

Alon Spinner	305184335	alonspinner@gmail.com
Dan Hazzan	308553601	danhazzan@campus.technion.ac.il

May 21, 2022

Theoretical Questions – Q1

Monte Carlo planning is a search method based on sampling. It works well when we can represent the search space well enough with a small, discretized set where the combination numbers are finite. When the search space cannot be granulated efficiently, as in continuous state spaces, we cannot guarantee that the sampling combinations will ever lead to a proper solution to the search problem in finite time.

Hands-on – Q1+2

We were asked to simulate a robot autonomously navigating using the following framework at each time step:

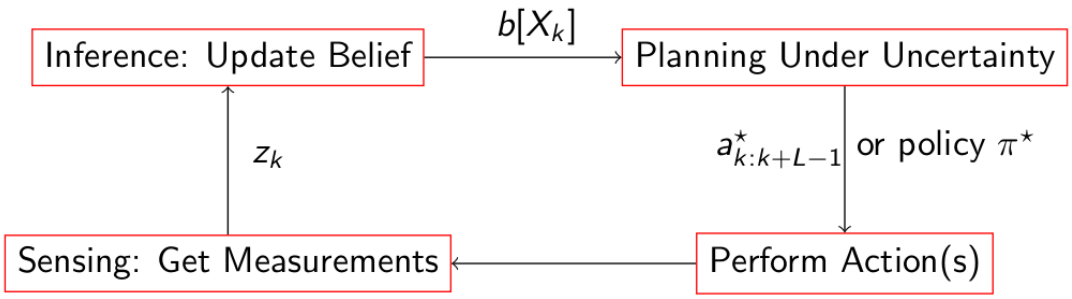


Figure 1: framework for each timestep

Planning was done with a ***modified Sparse Sampling Algorithm*** where sigma points are chosen and weighed from an observation distribution generated using a propagated belief instead of averaging over random samples.
for each sigma point (z_i, w_i) we compute the belief transition via Kalman $b_{k+1} = \Psi(b_k, a_k, z_{k+1})$ and the partial cost $J_i = w_i \cdot cost(b_k, a_k)$.

The code can be viewed in *02_plan.jl.*
Our implementation for choosing the points and weights follows *[Kalman and Bayesian Filters in Python, Roger R Labbe Jr, 2020]* and can be viewed in function.

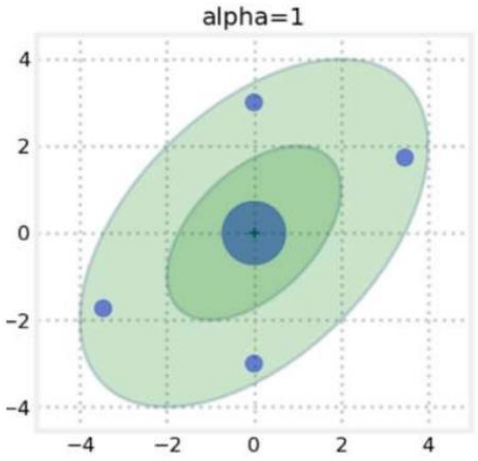


Figure 2 Weighted Sigma Points of a 2D gaussian distribution

We ran the simulation with $T = 10$ time steps, and a rather low horizon of $L = 3$ to allow a reasonable computation time.

Two Scenarios were tested:

a. The goal point lies outisde the beacon grid $x_{goal} = [5, -5]$
b. The goal point lies inside the beacon grid $x_{goal} = [8, 5]$

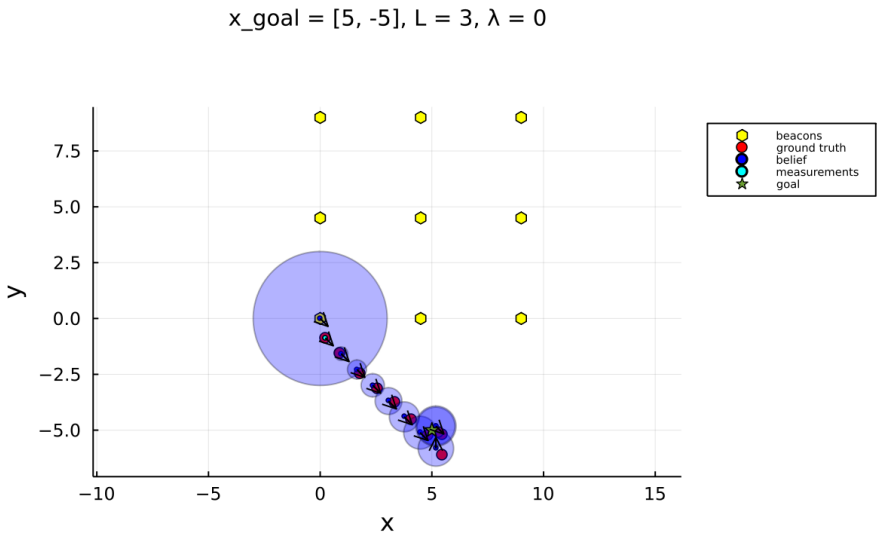
In both simulations we tested for three cases – three λ values:

a. The robot goes straight towards the goal ignoring the beacons
b. The robot attempts to each the goal with low uncertainty.
c. The robot won't step away from the beacon as to ensure low uncertainty.

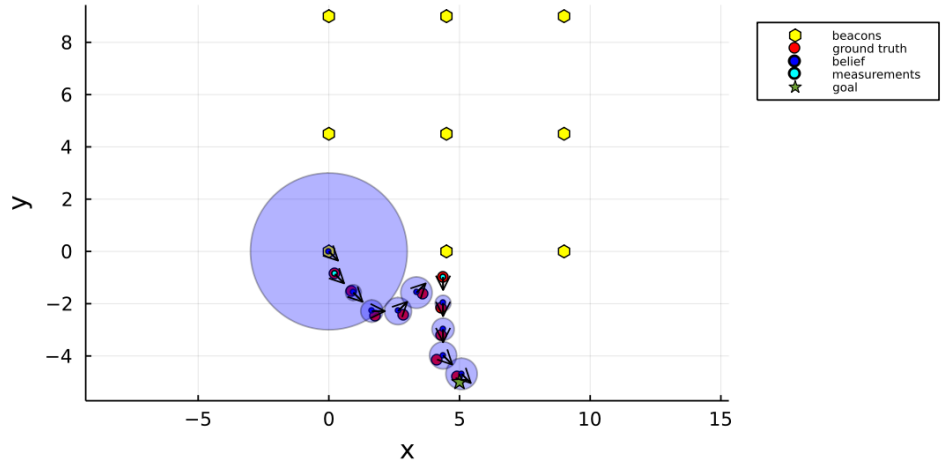
Given the low horizon L, we found that $\lambda \in [1500,2000]$ works best.
Many things can be said about the chosen cost function, but we would argue, without showing here, that it is too sensitive to the initial distance to the goal, having us tune λ per scenario. We would suggest using max, min functions in the cost function to allow for a better 'local' or 'immediate' decision making.

We provide the figures below for each scenario.

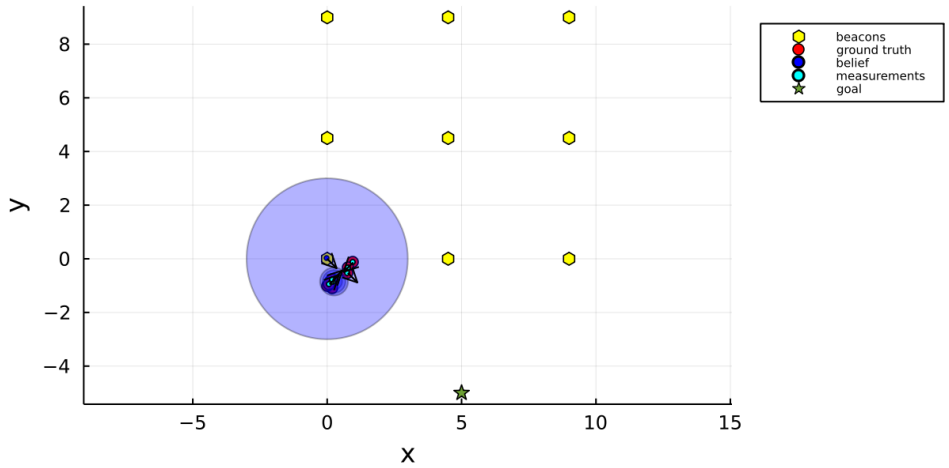
Scenario One - $x_{goal} = [5, -5]$:



x_goal = [5, -5], L = 3, λ = 2000



x_goal = [5, -5], L = 3, λ = 10000



Scenario Two - $x_{goal} = [8,5]$: