97

Technion – Israel Institute of Technology

HW2

Autonomous Navigation and Perception

086762

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

April 14, 2022

Question 1 – Theoretical questions

Given:

$$A \doteq \{a_{0:L-1}\}$$

And assume motion and observation models are:

$$p(x_k \,|x_{k-1}\,, a_{k-1}\,)\,, p(z\,|x\,)$$

$\{denoting\; r(a,b) = \int r(x,a)b[x]\,dx$ where $b[x_k] = p(x_k|a_{0:k-1}, z_{1:k}))\}$

a. Write the objective function explicitly for the above considered setting:

$$J(b[x_k], a_{k:k+L-1}) = \underbrace{\mathbb{E}}_{z_{k+1:k+L}} \{\sum_{l=0}^{L-1} r(b[X_{k+l}\,], a_{k+l}) + r(b[x_{k+L}])\}$$

$\{denoting\;\; b[x_k] = b_k = p(x_k|a_{0:k-1}, z_{1:k})$

b. Derive a recursive formulation that expresses $J(b[x_0], a_{0:0+L-1})$:

Given: $a_{0:L-1} \in \mathcal{A}$

$$J(b[x_0], a_{0:L-1}) = \underbrace{\mathbb{E}}_{z_{1:L}} \{\sum_{l=0}^{L-1} r(b[X_l\,], a_l) + r(b[x_L])\} \underbrace{=}_{Linearity}$$

$$\underbrace{=}_{1.\; Linearity} \mathbb{E}\{r(b[x_0], a_0)\} + \underbrace{\mathbb{E}}_{z_{1:L}} \{\sum_{l=1}^{L-1} r(b[X_l\,], a_l) + r(b[x_L])\} =$$

$$\underbrace{=}_{sufficient\; statistic} r(b_0, a_0) + \int p(z_{1:L}|b_0, a_{0:L-1}\,) \left[\sum_{l=1}^{L-1} r(b_l, a_l) + (b_L)\right] dz_{1:L} =$$

$$\underbrace{=}_{Chain\; rule+indep.} r(b_0, a_0) + \int p(z_1|b_0, a_0\,) p(z_{2:L}|b_0, a_{1:L-1}, z_1) \left[\sum_{l=1}^{L-1} r(b_l, a_l) + r(b_L)\right] dz_{1:L} =$$

$$= r(b_0, a_0) + \int\int p(z_1|b_0, a_0\,) p(z_{2:L}|b_0, a_{1:L-1}, z_1) \left[\sum_{l=1}^{L-1} r(b_l, a_l) + r(b_L)\right] dz_{2:L} dz_1 =$$

$$\underbrace{=}_{indep.} r(b_0, a_0) + \int p(z_1|b_0, a_0\,) \int p(z_{2:L}|b_0, a_{1:L-1}, z_1) \left[\sum_{l=1}^{L-1} r(b_l, a_l) + r(b_L)\right] dz_{2:L} dz_1 =$$

$$= r(b_0, a_0) + \int \underbrace{p(z_1|b_0, a_0\,)}_{Probability\; of\; objective\; function} \cdot \underbrace{J(b_1, a_{0:L-1}) dz_1}_{Value\; of\; objective\; function}$$

c. Provide similar derivations to the value function $V_0^\pi(b_0)$ and discuss the differences

Value function definition

$$V_0^\pi(b_0) = \underbrace{\mathbb{E}}_{z_{1:L}} \{\sum_{l=0}^{L-1} r(b[X_l], a_l) + r(b[x_L]) | a_l = \pi(b_l)\}$$

Recursive formulation

1. $V_0^\pi(b_0) = r\big(b_0, \pi(b_0)\big) + \underbrace{\mathbb{E}}_{z_{1:L}} \{\sum_{l=1}^{L-1} r(b[X_l], a_l) + r(b[x_L]) | a_l = \pi(b_l)\} =$

$$= r\big(b_0, \pi(b_0)\big) + \int p\big(z_{1:L} | b_0, \pi(b_{0:L-1})\big) \left[\sum_{l=1}^{L-1} r\big(b_l, \pi(b_l)\big) + r(b_L)\right] dz_{1:L} =$$

$$= r\big(b_0, \pi(b_0)\big) + \int p\big(z_1 | b_0, \pi(b_0)\big)\, p\big(z_{2:L} | b_0, \pi(b_{0:L-1}), z_1\big) \left[\sum_{l=1}^{L-1} r\big(b_l, \pi(b_l)\big) + r(b_L)\right] dz_{1:L} =$$

$$= r\big(b_0, \pi(b_0)\big) + \int p\big(z_1 | b_0, \pi(b_0)\big) \int p\big(z_{2:L} | b_0, \pi(b_{0:L-1}), z_1\big) \left[\sum_{l=1}^{L-1} r\big(b_l, \pi(b_l)\big) + r(b_L)\right] dz_{2:L} dz_1 =$$

$$= r\big(b_0, \pi(b_0)\big) + \int p\big(z_1 | b_0, \pi(b_0)\big)\, V_1^\pi(b_1) dz_1$$

Differences from the objective function J

In the objective function formulation J, we assume a set of known actions. The value of V^π uses a policy $\pi(b)$ to create actions. As such, actions are generated online and π can be optimized online.

Hands-on tasks 1 – GPS

Given:

1. Mobile robot navigating in a 2D environment.
2. Motion and observation models
3. $p(x_{k+1}|x_k, a_k), p(z_k|x_k) = x_{k+1} = f(x_k, a_k) + w\ , z_k = h(x_k) + v$
   Where:

$$w \sim N(0, \Sigma_w)\ , v \sim N(0, \Sigma_v)$$
$$f(x_k, a_k) = F \cdot x_k + a_k \ with\ F = I_{2x2}\ and\ a_k \epsilon \mathbb{R}^2$$
$$z_k \in \mathbb{R}^2\ and\ h(x_k) = x_k$$

4. $b(x_k) \doteq p(x_k) = N(\mu_k, \Sigma_k)$
5. A given prior $b(x_0) = b_0$

a. Implement a function *PropagateUpdateBelief*

We decided to go with a filtering approach and implement a Kalman filter (ProbabilisticRobotics chapter 3.2.1) for the following reasons:

1. $p(x_{t+1}|u_t, x_t)$ is a linear function and corrupted by Gaussian noise.
2. $p(z_t|\ x_t)$ is a linear function and corrupted by Gaussian noise.
3. $bel(x_0) \doteq p(x_0) = N(\mu_0, \Sigma_0)$

We provide a pseudo code of the algorithm below:

1: **Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**
2: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
3: $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$
7: return μ_t, Σ_t

Table 3.1 The Kalman fi lter algorithm for linear Gaussian state transitions and measurements.

Figure 1 Kalman Filter Implementation

b. Implement *SampleMotionModel*

Given a robot state x and action a we generate the next state $x' \sim p(x'|x, a)$ as such:

$$x' = F \cdot x + a + w$$

Where:

$$w = N(0, \Sigma_w)$$

$$F = I_2$$

To sample from the distribution, we used Julia's *MersenneTwister* generator

c. Implement a function *GenerateObservation*

We generate an observation $z \sim p(z|x)$ such:

$$z = H \cdot x + v$$

Where:

$$v = N(0, \Sigma_v)$$

$$H = I_2$$

Same as with the motion model, we sample the gaussian distribution using Julia's *MersenneTwister* generator.

d. Simulation of our 'GPS' model

(i + ii) generating actions and observations

In this simulation, we generated actions and observations at each time step as follows:

Table 1 GPS simulation

```
for _ in 1:T-1

        x_gt = SampleMotionModel(𝒫, ak, x_gt)

        z = GenerateObservation(𝒫, x_gt)

        #generate beliefs
        x_deadreckoning = PropagateBelief(x_deadreckoning, 𝒫, ak)
        x_kalman = PropagateUpdateBelief(x_kalman, 𝒫, ak, z)
```

The parameters used:

$$F = I_2; \Sigma_w = 0.1^2 I_2$$

$$H = I_2; \Sigma_v = 0.01^2 I_2$$

$$b_0 \sim N(\begin{bmatrix}0\\0\end{bmatrix}, I_2)$$

$$a_k = \begin{bmatrix}1\\0\end{bmatrix} \; \forall \, k \in [0,9]$$

$$x_0|_{ground\ truth} = \begin{bmatrix}0.5\\0.2\end{bmatrix}$$

(iii+iv) We achieved the following results

We present below the trajectory made by the robot (ground truth) against both beliefs.
As expected, the dead reckoning beliefs expand with time, and the Kalman filter generated beliefs are superior to them. In fact, they are so accurate that their covariance is too small to display.

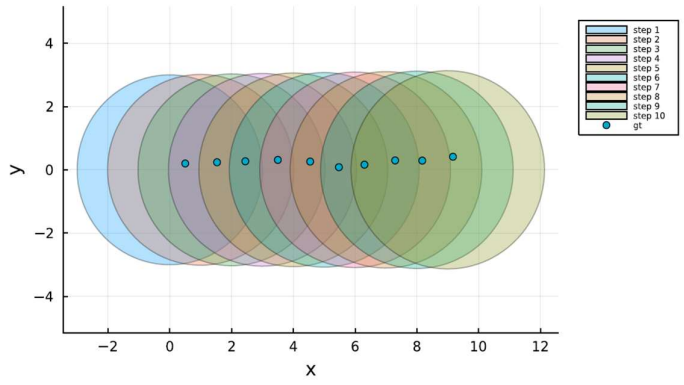

Figure 2 Ground Truth trajectory and dead reckoning propagated beliefs

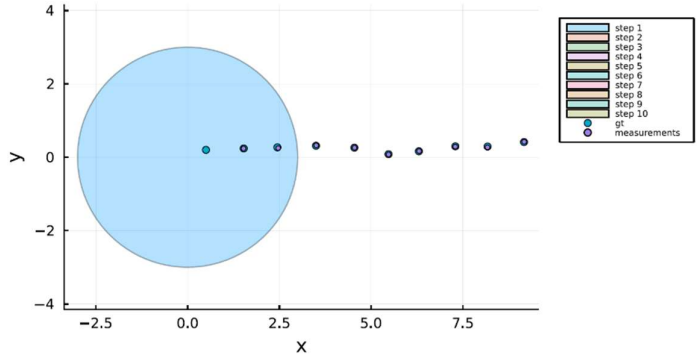

Figure 3 Ground Truth trajectory and Kalman filter beliefs

Hands-on tasks 2 – Beacons

Our map now consists of 9 beacons x^b where $b = \{1{:}\,9\}$ with **known locations**, spread in a rectangular form where the bottom left corner of that rectangle is placed in $[0,0]$.

a.+b. Implement the observation model

The observations model with parameter set $\{H, d, r_{min}, \Sigma_v\}$ for a given beacon located at x^b:

$$z_k^{rel}(x_k, x^b) = \begin{cases} \{H(x_k - x^b) + v(x_k, r), b\} & \|x_k - x^b\|_2 < d \\ \emptyset & otherwise \end{cases}$$

Where:

$$H = I_2$$

$$v \sim N(0, \Sigma_v)$$

$$\Sigma_v = (\alpha \cdot \max(\|x_k - x_b\|_2, r_{min}))^2$$

Notice that a successful measurement is a tuple of both the relative location and the index of the beacon. In practice, we loop over all the beacons to check which, if any, provides a successful measurement.

In this simulation, we assume only one beacon is in range of the robot at any given moment. As such, the robot may obtain at most 1 observation at each time step.
Given that the beacon's locations are known, this allows us to write a simpler function which returns the global location

$$z_k^{global}(x_k, x^b) = \begin{cases} x_k + v(x_k, r) & \|x_k - x^b\|_2 < d \\ \emptyset & otherwise \end{cases}$$

While this function is different from what we were asked to provide in section (b), it does serve the same exact purpose as:

$$z_k^{global}(x_k, x^b) = z_k^{rel}(x_k, x^b)[1] + x^b$$

c. simulation - only state estimation

(i) Repeat the simulation from Hands-on tasks 1 – GPS

In this section we only simulated one trajectory $a_k = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \forall\, k \in [0,99]$.

The simulation's parameters are as follows:

$$F = I_2; \Sigma_w = 0.1^2 I_2$$

$$H = I_2; \Sigma_v = (0.01 \max(\|x_k - x_b\|_2, r_{min}))^{\,2} I_2$$

$$d = 1; r_{min} = 0.1$$

$$b_0 \sim N(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, I_2)$$

$$x_0|_{ground\ truth} = \begin{bmatrix} -0.5 \\ -0.2 \end{bmatrix}$$

As expected, the dead-reckoning belief's covariance expands with time, while the Kalman based belief's shrinks quickly when a beacon is within range.
We present the figures below:

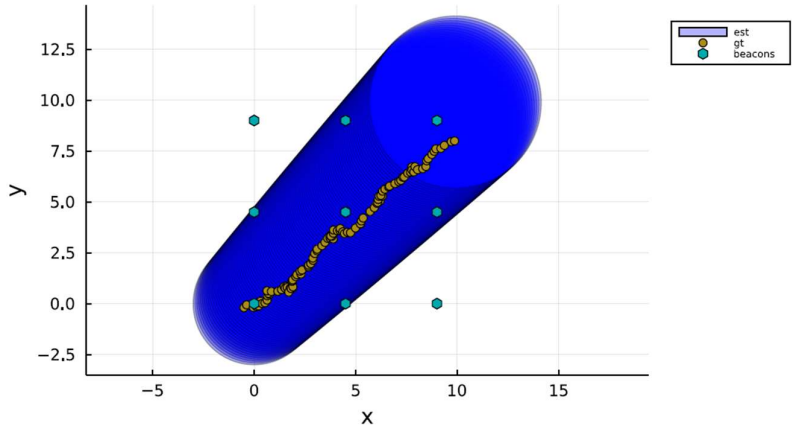

Figure 4 Beacons Scenario: Ground Truth trajectory and dead reckoning beliefs

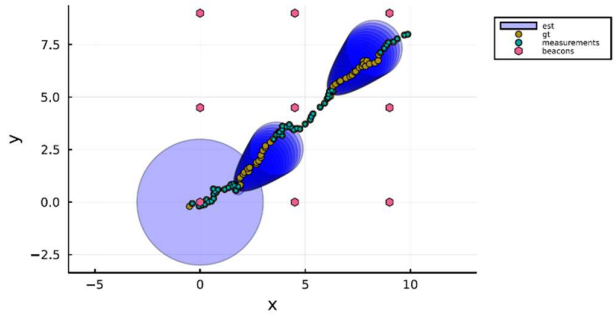

Figure 5 Beacons Scenario: Ground Truth trajectory and Kalman beliefs for dynamic measurement covariance

(ii) Repeat the simulation from the previous section with fixed Σ_v

We repeat the simulation from the previous section with a static measurement's noise covariance:

$$\Sigma_v = 0.01^2 I_2$$

To the naked eye, the resulting belief look the same.

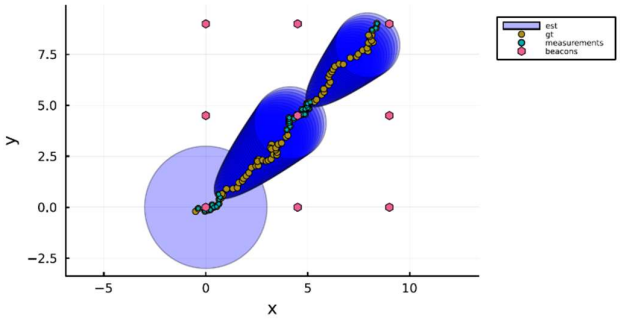

Figure 6 Beacons Scenario: Ground Truth trajectory and Kalman beliefs for dynamic measurement covariance

(iii+iv) Present and compare the errors between the ground truth and the estimations for each of the methods (dynamic vs. static covariance).

We drew the error norms $\left\| x_k|_{ground\ trut} - b_k.\mu \right\|_2$, and a measure of the estimation confidence, $tr(b_k.\Sigma)$, for each of the methods, showing they are very similar to the point where estimation with dynamic covariance provides no real benefit.

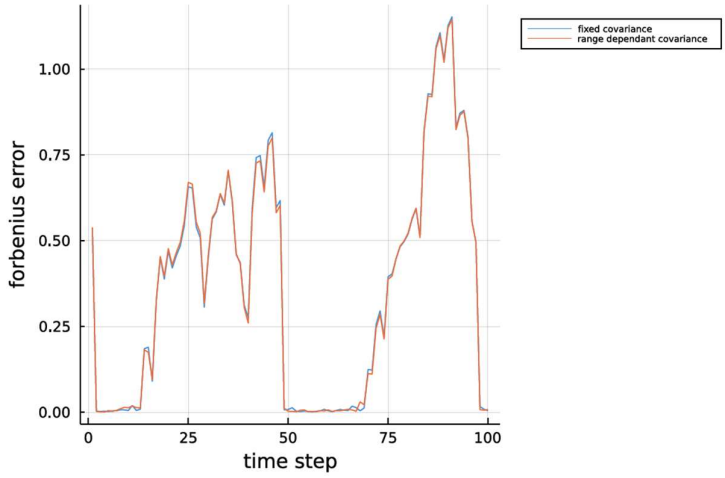

Figure 7 Beacon Scenario: Frobenius norm of errors against time

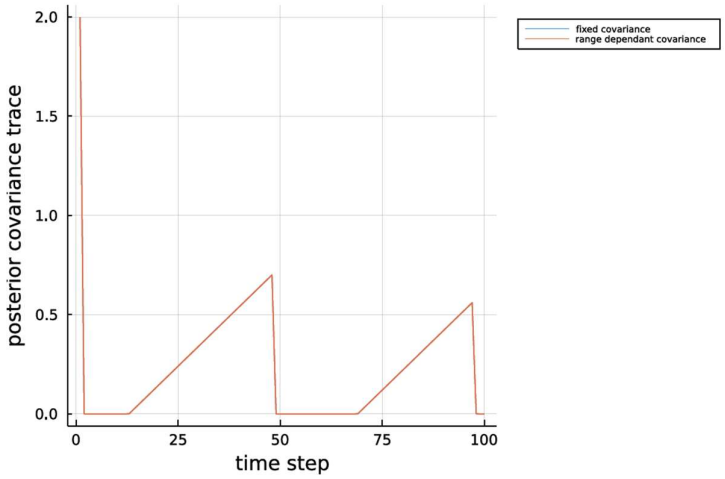

Figure 8 Beacon Scenario: trace of belief's covariances against time

.d Active estimation

In this section we test 10 different trajectories each containing a repeating action for 100 time steps.

$$a_j = [0.1, 0.1j]$$

$$A_j = \{a_{j,0} \dots a_{j,99}\}$$

For each trajectory we estimated the objective function $J(b_0, A_j, cost, cost_{terminal})$

Where the cost functions are:

$$cost(b, a) = cost_{terminal}(b, a) = \det(b.\Sigma)$$

The objective function was estimated in two ways for each we present a recursive formulation provided, for convenience at the first-time step

1. Maximum likelihood:
   $J(b_0, A_j, cost, cost_{terminal}) = cost(b_0, a_0) + J(b_1, A_j[2{:}end], cost, cost_{terminal})$
   Where $b_1 = kalamnFilter(b_0, a = a_0, z = b_0.\mu, modelParameters)$

This method allows us to draw the trajectory's estimated evolution in time as there is only one hypothesis belief estimated for each previous belief.

2. Sigma points:

$$J(b_0, A_j, cost, cost_{terminal}) = cost(b_0, a_0) + \sum_i w_i \cdot J(b_i, A_j[1{:}end], cost, cost_{terminal})$$

Where:

$$b_i = kalamnFilter(b_0, a = a_0, z = \sigma_i, modelParameters)$$
$$\sum w_i = 1$$

The sigma points and weights $\{w, \sigma\}_i$ are calculated in the same manner as they are for Unscented Kalman filter [Kalman and Bayesian Filters in Python, Roger R Labbe Jr, 2020] in a function we created called $generateSigmaPoints(\mu, \Sigma, \beta, \alpha, n)$.
We chose $\beta = 2, alpha = 1$ and $n = 3$.

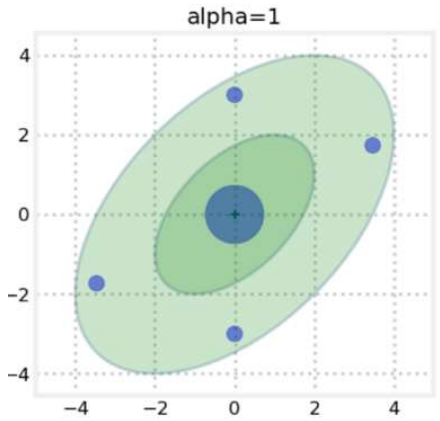

Figure 9 Sigma Points example, Roger R Labbe Jr, 2020

On both cases, the 5th trajectory, $a_5 = [0.1, 0.5]$, was deemed best, as the 'beacon density' along its path is the highest.

not exactly what is supposed to happen -3

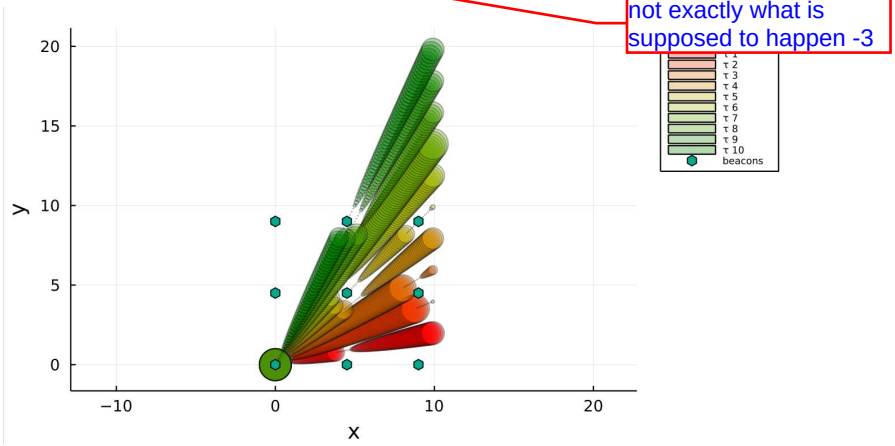

Figure 10 Trajectories and belief estimations for ML method

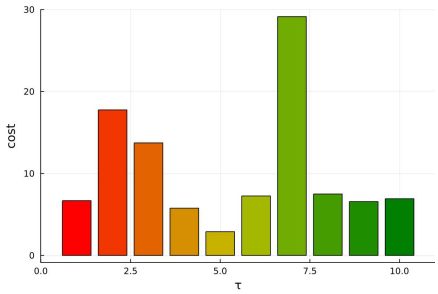

Figure 11 J per trajectory for ML method

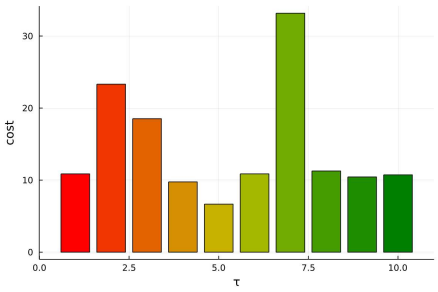

Figure 12 J per trajectory for sigma points method

The objective functions values are higher in the sigma points methods, as we allow for measurement error to affect the update section of the Kalman filter (innovation is not zero as it is in ML). In the sigma points method, we average over worst estimations.