COMS W4733: Computational Aspects of Robotics

Homework 3

Problem 1: 1-Dimensional Kalman Filter (16 points)

To gain deeper intuition for the Kalman filter equations, it is helpful to see what the updates look like for a 1-dimensional problem. Suppose a linear system has the following dynamics:

$$\begin{aligned} x_{k+1} &= f_k x_k + g_k u_k + v_k \\ y_k &= h_k x_k + w_k \end{aligned} \tag{1}$$

Everything is a scalar quantity, and the process and measurement noise are both distributed according to univariate Gaussian distributions: $v_k \sim \mathcal{N}(0, q_k)$, $w_k \sim \mathcal{N}(0, r_k)$.

(a) Write out the complete set of Kalman filter equations for the estimates of the state mean $\hat{x}$ and variance p. You should have equations for $\hat{x}_{k+1|k}$, $p_{k+1|k}$, $\hat{x}_{k+1|k+1}$, and $p_{k+1|k+1}$ in terms of $\hat{x}_{k|k}$, $p_{k|k}$, u_k, y_{k+1}, and the system parameters.

(b) Suppose that we receive a noise-free measurement at time T; i.e., $r_T = 0$. Show what happens to $p_{T|T}$.

(c) Show that $p_{k+1|k+1}$ is always no larger than $p_{k+1|k}$.

(d) An *asymptotically stable* system has $|f_k| < 1$ and $q_k = 0$ (no process noise). Show what happens to the covariance estimate p_k as $k \to \infty$.

Problem 2: Process and Input Noise (16 points)

The standard transition model for a linear dynamical system is as follows:

$$\mathbf{x}_{k+1} = F_k \mathbf{x}_k + G_k \mathbf{u}_k + \mathbf{v}_k \tag{2}$$

You may find it helpful to review the derivation of the covariance matrix $P_{k+1|k}$ for this problem. You may omit details or steps that are identical to those in the lecture slides if you are able to specify exactly which simplifications or properties you are using.

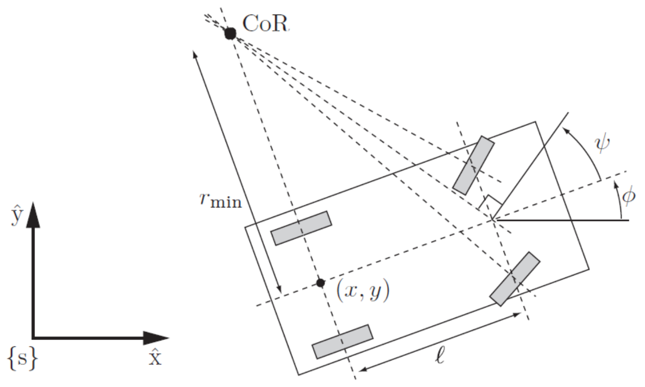
(a) Suppose that the process noise $\mathbf{v}_k$ is independently distributed according to a Gaussian distribution with nonzero mean, i.e. $\mathbf{v}_k \sim \mathcal{N}(\overline{\mathbf{v}}_k, Q_k)$. Find the new prediction equations for $\hat{\mathbf{x}}_{k+1|k}$ and $P_{k+1|k}$, given $\hat{\mathbf{x}}_{k|k}$ and $P_{k|k}$.

(b) Suppose that the process noise is distributed according to a zero-mean Gaussian distribution, but that the *inputs* are also independently distributed according to a Gaussian distribution: $\mathbf{u}_k \sim \mathcal{N}(\overline{\mathbf{u}}_k, U_k)$. Find the new prediction equations for $\hat{\mathbf{x}}_{k+1|k}$ and $P_{k+1|k}$, given $\hat{\mathbf{x}}_{k|k}$ and $P_{k|k}$.

Problem 3: Ackermann Car EKF (18 points)

Suppose we have a robotic vehicle that uses *Ackermann steering*, which also provides an approximate model for many front-wheel drive vehicles (see the figure below). The state vector is $\mathbf{x} = (x, y, \phi, \psi)$, where (x, y) track the car's inertial position, ϕ tracks the car's heading, and ψ tracks the the steering angle of the front wheels. The control inputs are the car forward velocity u_1 and steering angular velocity u_2. The (nonlinear) motion model is given by

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k + u_1(k)\cos\phi \\ y_k + u_1(k)\sin\phi \\ \phi_k + \frac{1}{l}u_1(k)\tan\psi \\ \psi_k + u_2(k) \end{bmatrix} + \mathbf{v}_k. \tag{3}$$



(a) Linearize the model by deriving the Jacobian dynamics matrix F_k.

(b) Compute $F_k F_k^T$. This is the update to an identity covariance matrix ($P = I$) due to the dynamics.

(c) At what headings ϕ are each of the covariance components of x and y minimized, respectively? Briefly explain your answers, referring to what you know about the car's motion model.

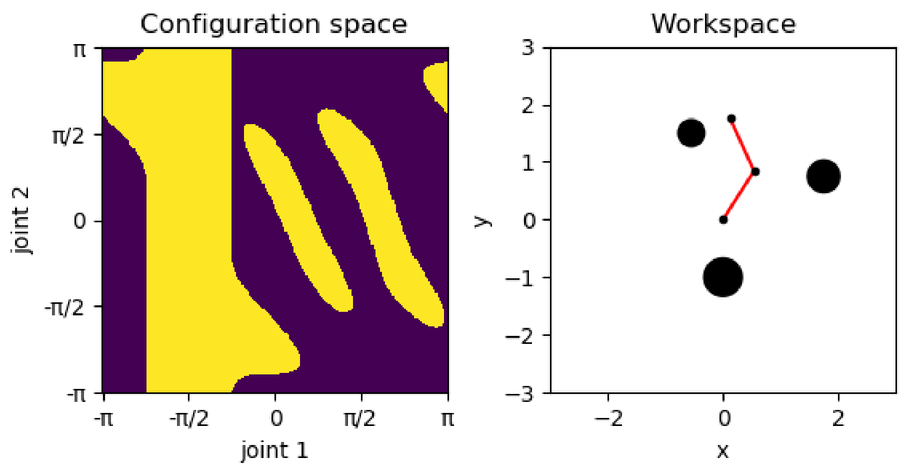
(d) Which covariance components become unbounded as $\psi \to \frac{\pi}{2}$? Again, briefly explain.

Problem 4: Non-Euclidean PRM

NOTE: We are providing significant scaffolding Python code for this portion of the assignment. We strongly recommend that you complete this problem using this code to avoid spending time on creating and debugging visualizations and support procedures.

Problem and Code Description

You will be implementing PRM construction for a RR robot arm. Remember that the configuration space of a RR arm is a torus, which is not Euclidean. In addition, it is often not trivial to construct representations of workspace obstacles in the configuration space. Below is an example of the two spaces side by side for an environment with three circular workspace obstacles.



We are providing the code file that generated the above figure, which you will also complete to generate probabilistic roadmaps on these C-spaces. Here is a brief rundown of the relevant portions of the code.

- At the top are several robot and PRM parameters. You are free to modify these as you develop your code. We will also ask you to show outputs for specific combinations of parameters.
- We define two classes for a RR arm and a search node. You don't need to worry about these.
- Several utility functions are provided. One helps to detect collisions for a given arm configuration. The other two implement a distance metric on toroidal space that can be used by a nearest neighbors computation.[1]
- You will mainly be working with the PRM functions `sample_points` and `construct_prm`, both called by `plan_route`. As indicated by their names, they will implement the two stages of the PRM algorithm, sampling points and graph construction.
- The rest of the code includes a Dijkstra search function and a number of visualization functions. You may modify these as needed, but there should be no need to do so. Note that `visualize_spaces` is commented out in the main function; this produced the above figure and is not used by the PRM. Remember that a benefit of sampling methods is that they do not need an explicit representation of the entire C-space.

4.1: Generating Samples (15 points)

Write the `sample_points` function to generate `N_SAMPLE` random, collision-free samples on the configuration space. Both coordinates should lie within the bounds $-\pi \leq q \leq \pi$. You should also append the start and goal coordinates to the lists of samples. We recommend that you sample the space uniformly to start with, and then implement more sophisticated strategies if necessary.

You should return your samples in two lists, one containing the coordinates of the first joint and the second containing the coordinates of the second joint.

[1]The way that this works is we make eight permutations of a single configuration, adding or subtracting 2π to one or both components. Then we return the one closest to a reference configuration, so that the Euclidean distance between them is equal to the toroidal distance.

4.2: PRM Construction (15 points)

Write the `construct_prm` function to take in your generated samples and construct a roadmap. Since this process typically requires nearest-neighbors computations, we recommend that you utilize sklearn.neighbors.NearestNeighbors. The number of neighbors to use is given by `N_KNN`, while the maximum length of an edge is given by `MAX_EDGE_LENGTH`. You should also use the `torus_dist` distance metric that we have provided.

You should include an incremental collision check when creating edges. However, you have to take into account the toroidal nature of the space here as well. Given two configurations q_1 and q_2, the shortest path between them is the Euclidean path between the pair (q_1, q_2'), returned by `toroid_to_euclidean_points(q1, q2)`.

Once you've successfully implemented both functions, the `main()` function will call `plan_route()`, which will call your functions to build the PRM and then search for a path. The static roadmap and arm environment will show for 5 seconds, followed by an animation of the path that is found. The latter will include red markers appearing on the roadmap indicating the nodes taken, while the arm moves to the corresponding configuration for each. If no path was found in the roadmap, the animation will not run, and a failure message will be printed to the console.

4.3: Outputs and Discussion (20 points)

Find combinations of PRM parameters that can successfully find paths for the two environments included at the top level (the parameters we provide are not necessarily optimal or effective). The three-obstacle environment starts at $(1, 0)$ and finishes at $(-3, -1)$; the four-obstacle environment starts at $(-3, 1)$ and finishes at $(-0.5, 0.5)$. Include the final still images of the PRM and workspace for both in your writeup, and also save the animations to submit as videos. The videos should be saved in a publicly accessible location like Google Drive or YouTube, and links should be provided in your writeup (make sure to test these yourself!). You'll also submit copies of these video files, described below. Finally, answer the following questions (a few sentences for each should suffice).

(a) Discuss some tradeoffs of having more versus fewer samples when generating the PRM vertices. Around what was the minimum number of samples that you found to be consistently successful on both environments?

(b) Discuss some tradeoffs of having more versus fewer neighbors for connecting PRM edges. What was the smallest number of neighbors that you found for which a connected graph could be constructed in both environments?

(c) Discuss some tradeoffs of having a larger versus smaller maximum edge length. Around what edge length did you find to successfully generate connected graphs in both environments?

Submission

You should have one document containing your solutions, responses, and figures for all written questions. At the end of the document, create an appendix with printouts of all code that you wrote or modified for problem 4 (you should not include the entire file). Archive your code file(s) together with the video files you generated for problem 4. Submit both your document and code/video archive separately on Gradescope.