Saving Castaway Bob

Thelonious Cooper
EECS Dept. MIT
(Dated: May 14, 2024)

SETTING THE SCENE

Bob is enjoying his time on a new cruise with his family. Among the cruise attractions is a massive waterslide. While sliding down the slide Bob is met by catastrophe. Due to a part failure the structure partially collapses as Bob slides down, sending him careening into the sea. Unfortunately Bob is now lost at sea. He was fortunate enough to be wearing a life jacket since he can't swim so he is floating somewhere in the Ocean.

Our mission is to save Bob.

Thankfully Bob's phone has an SOS mode which broadcasts his GPS location every minute. Unfortunately this GPS signal is very weak and is only accurate to a 50m radius. We need a narrower range to be able to send out a rescue team.

METHODS

State-Space Model

Here is what we have thus far mathematically. We can observe the variable Y, which is the output of the GPS

$\vec{Y}[n] \sim \mathcal{N}(\vec{x}[n], \sigma_p \mathbb{I}_2)$

We also know that bob will have little effect on the bulk ocean dynamics, and will flow as a neutrally boyant particle embedded in the velocity field subject to random forces from fish and other unmodelable sources. For the purposes of this rescue mission we will neglect deep-water effects and treat the flowfield as 2d To make the math work out we'll express this challenge as a linear-state-space model. Which just means we will have a locally linear update rule governing a vector timeseries.

Bob floats along the u field as follows To start out we define

$$\vec{x} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \tag{1}$$

$$\vec{x}[n+1] = A\vec{x}[n] + \begin{bmatrix} 0 \\ 0 \\ F_x \\ F_y \end{bmatrix} \tag{2}$$

With Gaussian random forces

$$F_x, F_y \sim \mathcal{N}(0, \sigma_F \mathbb{I}_{\not\varepsilon}) \tag{3}$$

A is the matrix that integrates our velocity state into our position state and with a weak damping $\alpha < 1$ on velocity for numeric stability

$$\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix} \tag{4}$$

We can then only observe position, which we express as the following equation in terms of our state variables and observation matrix H.

$$\vec{y}[n] = H\vec{x}[n] + \vec{n_y} \tag{5}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6}$$

Where $n_y \sim \mathcal{N}(0, \sigma_y \mathbb{I}_2)$ is our "observation noise", and F_b is our "process noise"

Numerical Integration of Navier Stokes

Before generating our pictures its worth considering Bob's locale: the Ocean. Because Bob is lucky enough to have a life jacket of neutral boyancy, we will model Bob as an ideal lagrangian particle. Which is to say that the contribution of outside velocity fields is greater than any force on him. Another important factor is that we are considering a high-reynolds number flow because the lengthscale over which bob is moving is large while hes moving comparitively slowly. We implement this with WaterLily.jl a Julia library for fluid simulation. WaterLily.jl solves the unsteady incompressible 2D or 3D Navier-Stokes equations on a Cartesian grid. The pressure Poisson equation is solved with a geometric multigrid method. In our case we can use the shallow-water approximation and assume the flow regime is roughly 2d. A common approach employed in WaterLily.jl is to first solve the momentum equations, and have some numerical error, then correct by solving the poisson pressure equation.

Starting from the incrompressibility condition

$$\nabla \cdot \vec{u} = 0 \tag{7}$$

and the momentum equation

$$\partial_t u + u \cdot \nabla u = -\frac{1}{\rho}\nabla p + \eta \nabla^2 u \tag{8}$$

We can decouple the pressure and velocity by taking the divergence of the momentum equation to get

$$\nabla^2 p = -\rho \nabla \cdot (\partial_t u + u \cdot \nabla u) + \eta \nabla^2 u \tag{9}$$

To solve the equation WaterLily.jl uses the multigrid method. The multigrid method is an efficient algorithm for solving large-scale discretized partial differential equations (PDEs), particularly useful due to its rapid convergence rates. It operates by employing a hierarchy of grid resolutions to smooth out errors at different scales, typically involving a sequence of restriction and prolongation operations between coarser and finer grids. This approach is highly effective for a variety of PDEs, and recent advancements have integrated machine learning techniques to optimize the construction of multigrid solvers for specific problems.

The Kalman Filter

I promise we'll get to the fluids part in a bit but before we need to figure out how to estimate Bob's position from our noisy GPS signal.

We seek the distribution $X|Y \sim N(?, ?)$

Because we assumed our system had randomness that was purely gaussian, we know our target distribution will also be Gaussian, and we can do Linear Algebra to figure it out exactly.

I'll spare you the complete derivation, but I'll give a sketch.

We know that

$$\mathbb{E}[X[n+1]] = A\mathbb{E}[X[n]] \tag{10}$$

and

$$cov(X[n+1]) = Acov(X[n])A^T + \sigma_F \mathbb{I}_4 \tag{11}$$

In the space of linear-gaussian models, an optimal estimator is always linear. So we will assume our estimate will look like our naive estimate based on our previous state, plus some gain times an error term for to correct our estimate via our observation. This gain is called the Kalman Gain

$$\hat{x}[n] = \mathbb{E}[x[n]] + \mathcal{K}(y[n] - \mathbb{E}[y[n]|x[n]]) \tag{12}$$

We can solve for $\mathcal{K}$ by assuming it provides an estimate with the minimum possible variance. Which is to say $P := Cov(X[n])$ has small singular values. Because the covariance matrix is by definition positive semi-definite, we can bound the singular values by the eigenvalues. The solution can then be found via matrix calculus by setting the derivative of the $Tr(\hat{P})$ to zero.

Assuming the linear gain matrix $\mathcal{K}$ exists, the optimal estimate of the estimate covariance would be

$$\hat{P} = (\mathbb{I} - \mathcal{K}H)P(\mathbb{I} - \mathcal{K}H)^T + \mathcal{K}K_v\mathcal{K}^T \tag{13}$$

Thus we seek

$$\mathcal{K} = \arg\min_{\mathcal{K}} Tr(\hat{P}) \tag{14}$$

Using matrix calculus identities we can expand the previous statement

$$Tr(\hat{P}) = Tr(P) + Tr(\mathcal{K}HPH^T\mathcal{K}^T) \tag{15}$$

$$-2Tr(\mathcal{K}HP) + Tr(\mathcal{K}K_v\mathcal{K}^T) \tag{16}$$

We then take the derivative with respect to $\mathcal{K}$ and set it to 0. Note that, because all matrices involved are positive semi-definite, the solution will also be positive semi-definite, implying the critical point is a minimum without having to check the hessian.

$$\frac{\mathrm{d}}{\mathrm{d}\mathcal{K}} Tr(\hat{P})) = 0 = 2\mathcal{K}(HPH^T) - 2P^TH^T + 2\mathcal{K}K_v \tag{17}$$

Finally, via algebra we obtain the solution

$$\mathcal{K} = PH^T(HPH^T + K_v)^{-1} \tag{18}$$

where H is the observation matrix, K_v is the observation noise covariance and P is the predicted state error covariance

RESULTS

We begin with (figure 11) where we use the simplest possible estimator: an average of the last 3 gps pings.

We then move on to the rank 2 Kalman filter (figure 2) which observes the gps signal and estimates velocity and position covariances.

Lastly, we incorporate our knowledge of the velocity field to develop a rank 4 Kalman filter (figure 3) which has significantly better tracking accuracy and smaller estimated covariance.

CONCLUSION

We saved castaway Bob!

Should Bob somehow get lost at sea again there are some more complicated techniques we could experiment with for tracking and estimation. Among techniques not

FIG. 1. Fixed-Lag Mean Estimator of order 3

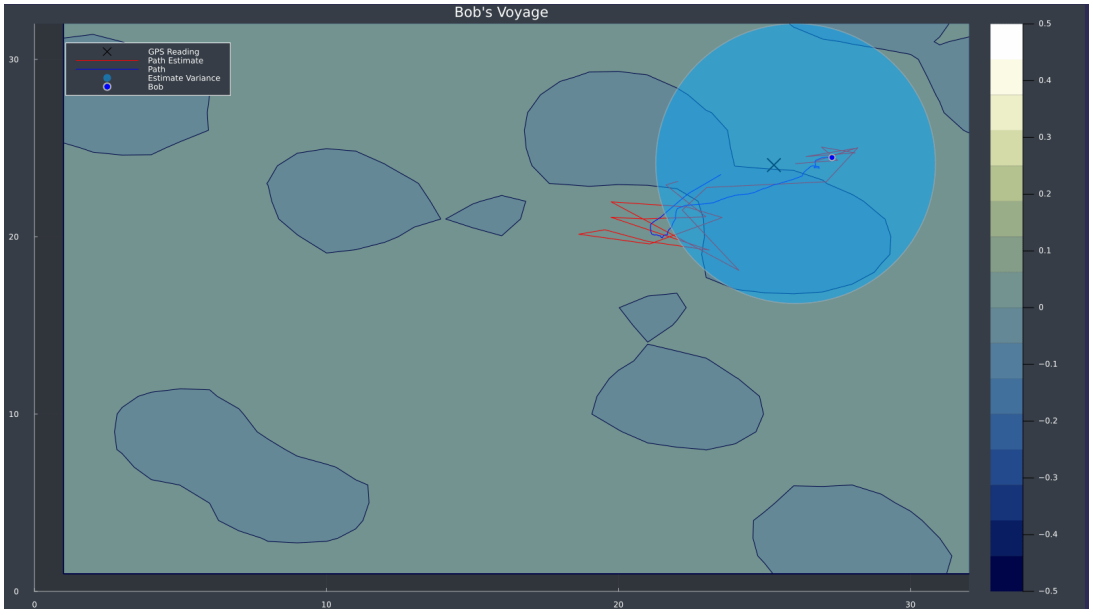

FIG. 2. Kalman Filter Estimator with rank 2

covered but of interest are extensions of the Kalman filter known as the unscented Kalman filter ([1]) and the Ensemble Kalman filter ([2]). Both of these methods have means of correcting for nonlinearities in the system dynamics by taking higher-order moments. Both of which are more computationally expensive.

[1] E. Wan and R. Van Der Merwe, The unscented kalman filter for nonlinear estimation, in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)* (2000) pp. 153–158.

[2] J. J. Gómez-Hernández, Ensemble kalman filtering, in *Encyclopedia of Mathematical Geosciences*, edited by B. S. Daya Sagar, Q. Cheng, J. McKinley, and F. Agterberg (Springer International Publishing, Cham, 2020) pp. 1–5.

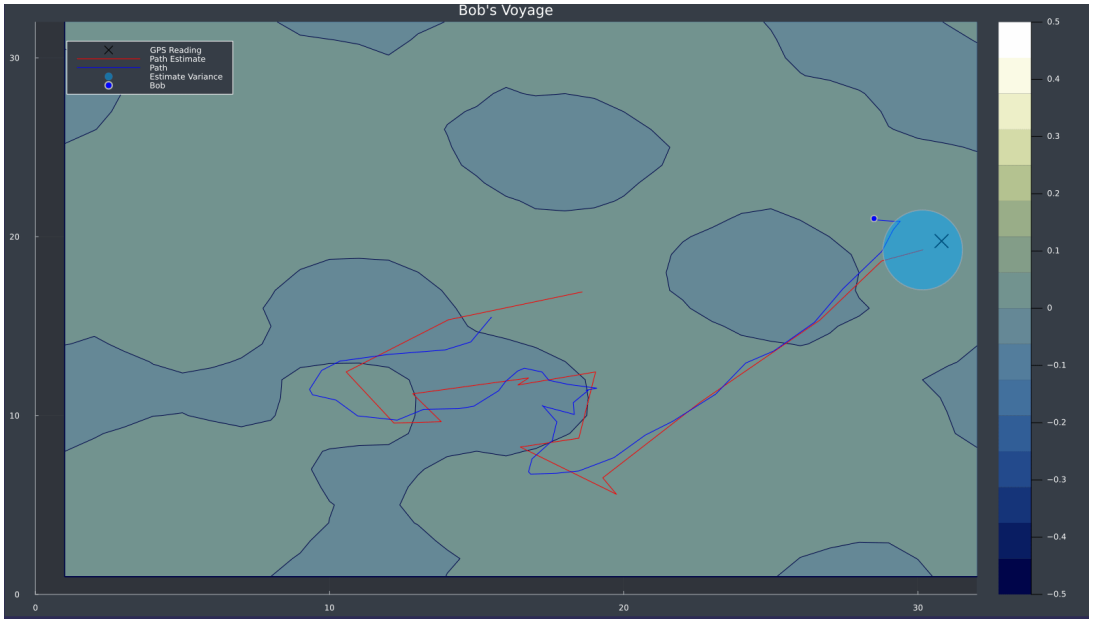

FIG. 3. Kalman Filter Estimator with rank 4