Tracking mice behavior in naturalistic settings with the Kalman filter and smoother

1 Introduction

A central aim of current neuroscience is to understand the relation between neural circuits and behavior. Many important behaviors are only observed in naturalistic settings. Therefore, signal processing methods to accurately monitor behavior in these settings are essential.

Here we describe and evaluate the use of the Kalman Filter and Smoother to perform inference in the Discrete Time Wiener Process Acceleration model, a linear dynamical system model used for tracking. We use these inference methods to track the position of mice while freely exploring a large arena.

2 Methods

2.1 Mice foraging in naturalistic setting

Videos were recorded of mice foraging in a circular arena (diameter two meters) with two food patches (Figure 1).

2.2 Linear Dynamical System (LDS) model

You may want to fix the problem I marked in this section of your poster.

2.3 Inference

2.3.1 Inference problems

Here I would insert the prediction, filtering and smoothing equations.

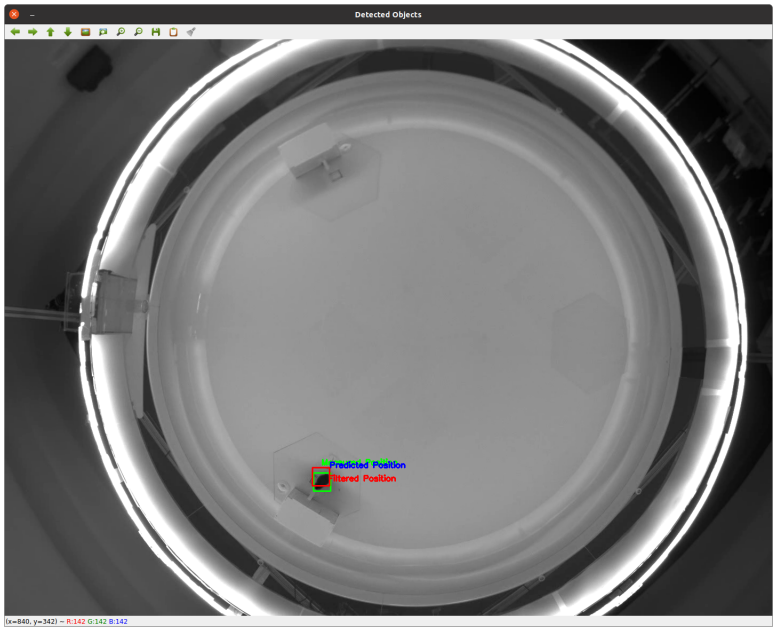

Figure 1: Video frame from a foraging mouse.

2.3.2 Kalman filter

I would put the Kalman filter equations.

2.3.3 Kalman smoother

I would put the Kalman smoother equations. You may want to comment that, as you will show in the results, the Kalman smoother yields more accurate state estimates than the Kalman filter. However, the Kalman filter can be used for online tracking, while the Kalman smoother cannot.

2.4 Discrete Wiener process acceleration (DWPA) model

$$\mathbf{x}_n = [x_n, \dot{x}_n, \ddot{x}_n, y_n, \dot{y}_n, \ddot{y}_n]$$
$$\mathbf{x}_n = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \mathbf{x}_{n-1} + \mathbf{w}_n, \quad \mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, Q), \quad Q = \begin{bmatrix} \gamma_x^2 \tilde{Q} & \mathbf{0} \\ \mathbf{0} & \gamma_y^2 \tilde{Q} \end{bmatrix}$$
$$\mathbf{y}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_n + \mathbf{v}_n, \quad \mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, R), \quad R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$
$$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{m}_0, V_0)$$

2.5 Computer vision functionality

We extracted the mouse position in each video frame using thresholding, dilation, contour finding functions in the OpenCV library [1].

2.6 Code availability

Python code implementing the Kalman filter, Kalman filter and DWPA model can be found at [2].

3 Results

We first verified the accuracy of the DWPA model to estimate the position, velocity and acceleration of a simulated mouse (Section 3.1). Then we used this model to track the motion of a real mouse (Section 3.2).

3.1 Simulated mouse

We simulated the DWPA model with parameters $\mathbf{m}_0 = [0, 0, 0, 0, 0, 0]^\intercal$, $V_0 = \text{diag}(0.001)$, $\gamma_x = \gamma_y = 1e4$ and $\sigma_x = \sigma_y = 100$ (Section 2.4), generating samples of true positions, velocities and accelerations (blue points in Figures 2a, b, c) and noisy measured positions (black points in Figure 2a). Note that the measured positions contain a substantial amount of noise (compare the blue and black points in Figure 2).

We then run the Kalman filter (Section 2.3.2, red traces in Figures 2a, b and c) and smoothing (Section 2.3.3, green traces in Figures 2a, b and c). Despite the large amount of noise in the simulations, the Kalman filter

and smoothing estimates were very accurate. Position estimates were more accurate than velocity ones, which in turn were more accurate than acceleration estimates. As expected, smoothed estimates were more accurate than filtered ones for positions, velocities and accelerations.

3.2 Real mouse

We first detected the horizontal and vertical positions of the mouse using the computer vision functions described in Section 2.5 (black points of Figure 3a). Note that when the mouse is behind the top feeder, or entering the nest, black dots are missing, indicating that the computer vision functions failed to detect the mouse position.

These missing positions are interpolated by the Kalman filter (red points in Figure 3a) and smoother (green points in Figure 3a). We observe that sometimes, the interpolations of the Kalman filter are incorrect (e.g., missing observations next to the top feeder). This problem is not present in Kalman smoother estimates (green points in Figure 3a).

Figures 3b,c show velocity and acceleration estimates. These estimates are reasonable. Velocities are at a minimum when the mouse is near the patches or nest, and are at a maximum when the mouse is traveling between patches. Accelerations are at a maximum when the mouse enters the arena and when it leaves the food patches.

4 Ongoing work

In this work we only discussed inference methods. That is, we assumed we knew the model parameters and tried to infer the model states. However, in most cases the model parameters are unknown and one needs to learn them from data. In the last month of my fellowship I will study and apply learning techniques for linear dynamical systems models. We plan to present our learning results at [3].

5 Summary

In this project I **learned** an important time series model, the linear dynamical systems model, as well as statistical methods to optimally infer the model states, the Kalman filter and smoother. I studied the mathematical basis of

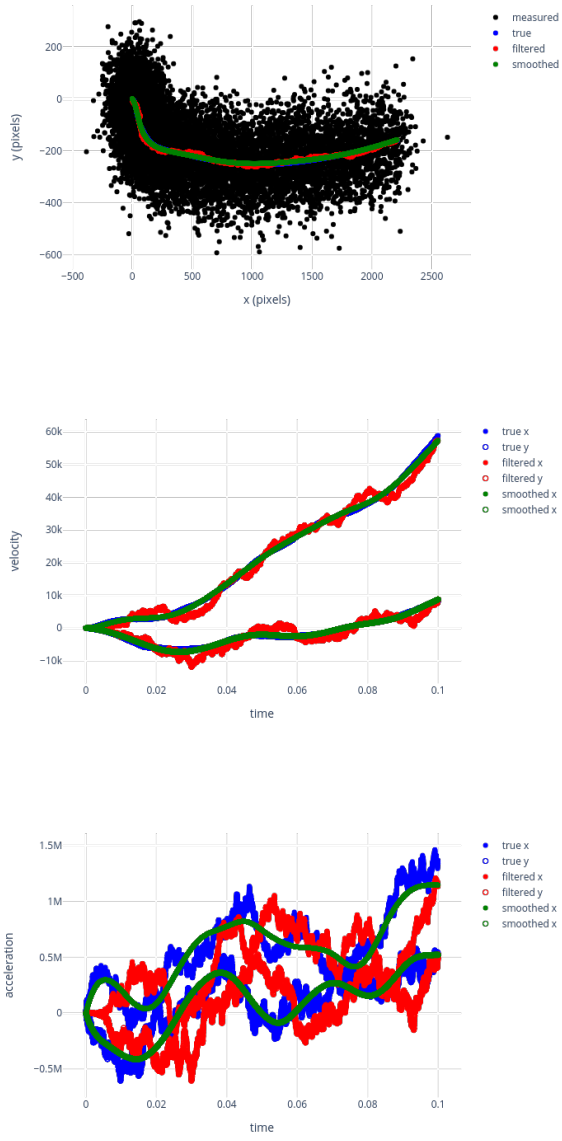

Figure 2: Simulated mouse: filtered positions, velocities and accelerations using true parameters.

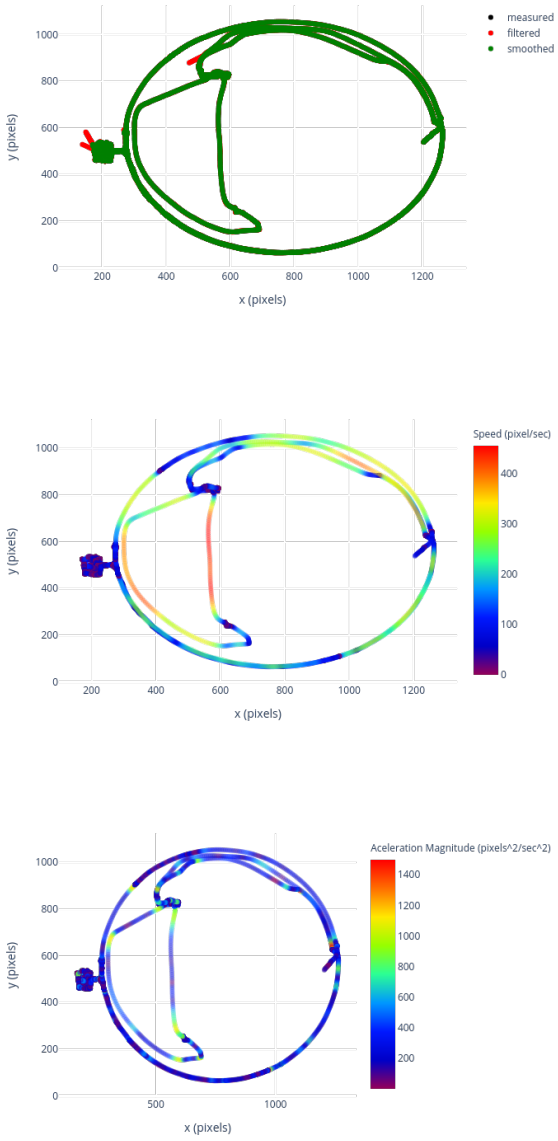

Figure 3: Real mouse: filtered positions, velocities and accelerations using true parameters.

the inference methods and implemented them in Python. I then used simulated data (for which I knew the true positions, velocities and accelerations) to **evaluate the accuracy** of the Kalman filter and smoother inferences. Finally, I **applied** the above methods to estimate the position, velocity and acceleration of **foraging mice**.

I feel fortunate to have learned a seminal model for time series data, and associated inference methods, and to have applied them to understand an important behavior.

References

[1] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[2] https://github.com/Ash530888/Kalman-Smoother

[3] A. Quershi, D. Campagner, M. Javadzadeh No, J. Rapela (submitted). Tracking freely moving mice using computer vision, statistical inference and statistical learning techniques. 44th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Glasgow, Scotland, UK. 2022.2022,