Inferring Mouse Behavior in Naturalistic Settings With the Kalman Filter and Smoother

Aishah Qureshi 1,2 and Dario Campagner 3,4 and Mitra Javadzadeh 3,6 and Joaquín Rapela 4,6
1 Queen Mary University of London
2 Simons Collaboration on the Global Brain Undergraduate Research Fellow
3 Sainsbury Wellcome Centre for Neural Circuits and Behaviour, University College London
4 Gatsby Computational Neuroscience Unit, University College London
6 Simons Collaboration on the Global Brain Undergraduate Research Mentor

Acknowledgements

Miss Aishah Qureshi was supported by the generous Simons Collaboration on the Global Brain Undergraduate Research Fellowship, where Joaquin Rapela and Mitra Javadzadeh acted as her mentors.

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 infer 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 (DWPA) 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. Mice Foraging in Naturalistic Setting

Videos were recorded of mice foraging in a well-lit circular arena (diameter two meters) with two food patches using high-quality cameras.

Output of detect.py [2]

3. Computer Vision Functionality

We used the frame differencing technique to detect moving objects in video frames. This involved calculating the absolute difference between every eight frames and the background model. To get the background model, we took the median of fifty randomly selected frames. We then extracted the mouse position in the resulting frame by using thresholding, dilation, contour finding functions in the OpenCV library [1].

Python code implementing the computer vision functionality, the DWPA model, as well as inference and learning in linear dynamical systems can be found at [2].

4. The Linear Dynamical Systems Model

$x_n = Ax_{n-1} + w_n \qquad w_n \sim N(w_n|0, Q) \qquad x_n \in \mathbf{R}^M$

$y_n = Cx_n + v_n \qquad v_n \sim N(v_n|0, R) \qquad y_n \in \mathbf{R}^N \qquad n = 1 \ldots T$

$x_0 \sim N(x_0|m_0, V_0)$

Inference

Predicting $P(x_n|y_1, \ldots, y_{n-1}) = N(x_n|x_{n|n-1}, P_{n|n-1})$

Filtering $P(x_n|y_1, \ldots, y_n) = N(x_n|x_{n|n}, P_{n|n})$

Smoothing $P(x_n|y_1, \ldots, y_N) = N(x_n|x_{n|N}, P_{n|N})$

5. Kalman Filter

$$x_{1|0}, P_{1|0} \rightarrow x_{1|1}, P_{1|1} \rightarrow x_{2|1}, P_{2|1} \rightarrow x_{2|2}, P_{2|2} \rightarrow \ldots \rightarrow x_{T|T}, P_{T|T}$$

$$x_{1|0} = m_0, P_{1|0} = V_0$$

Predict

$x_{n|n-1} = A\, x_{n-1|n-1}$ predicted mean

$P_{n|n-1} = A\, P_{n-1|n-1} A^T + Q$ predicted covariance

Update

y_n (observation at time n) has to be supplied

$y_{n|n-1} = C\, x_{n|n-1}$ predicted observation

$\tilde{y}_n = y_n - y_{n|n-1}$ innovation

$S_n = C\, P_{n|n-1} C^T + R$ innovation conditional covariance

$K_n = P_{n|n-1} C^T S_n^{-1}$ Kalman gain

$x_{n|n} = x_{n|n-1} + K_n \tilde{y}_n$ filtered mean

$P_{n|n} = (I_M - K_n C)\, P_{n|n-1}$ filtered covariance

Update Predict

Initial prediction

6. Kalman Smoother

$$x_{T|T}, P_{T|T} \rightarrow x_{T-1|T}, P_{T-1|T} \rightarrow x_{T-2|T}, P_{T-2|T} \rightarrow \ldots \rightarrow x_{1|T}, P_{1|T}$$

$x_{T|T}$ and $P_{T|T}$ are initialised from the last step of the Kalman Filter.

$x_{n|T} = x_{n|n} + C_n (x_{n+1|T} - x_{n+1|n})$ smoothed mean

$P_{n|T} = P_{n|n} + C_n (P_{n+1|T} - P_{n+1|n})\, C_n^T$ smoothed covariance

$C_n = P_{n|n} A^T\, P_{n+1|n}^{-1}$

7. Discrete Time Wiener Process Acceleration (DWPA) Model [3]

Motion equations

$\xi(t+1) = \xi(t) + \xi'(t)\Delta t + \xi''(t)\Delta t^2/2 + r(t)\Delta t^2/2$
$\xi'(t+1) = \xi'(t) + \xi''(t)\Delta t + r(t)\Delta t$
$\xi''(t+1) = \xi''(t) + r(t)$

Where **Δt** is the **timestep**, and r(t) is random noise that unpredictably changes acceleration. Without r(t), acceleration would be constant.

$$\mathbf{x}_n = [\xi_x, \xi'_x, \xi''_x, \xi_y, \xi'_y, \xi''_y]$$

$$\mathbf{x}_n = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \mathbf{x}_{n-1} + \mathbf{w}_n, \quad \mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, Q), \quad Q = \begin{bmatrix} \gamma_x^2 \tilde{Q} & \mathbf{0} \\ \mathbf{0} & \gamma_y^2 \tilde{Q} \end{bmatrix}$$

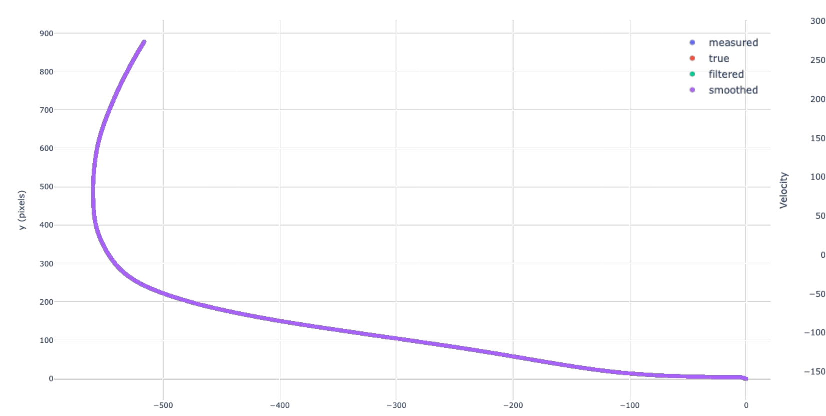
$$\mathbf{y}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_n + \mathbf{v}_n, \quad \mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, R), \quad R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{m}_0, V_0)$$

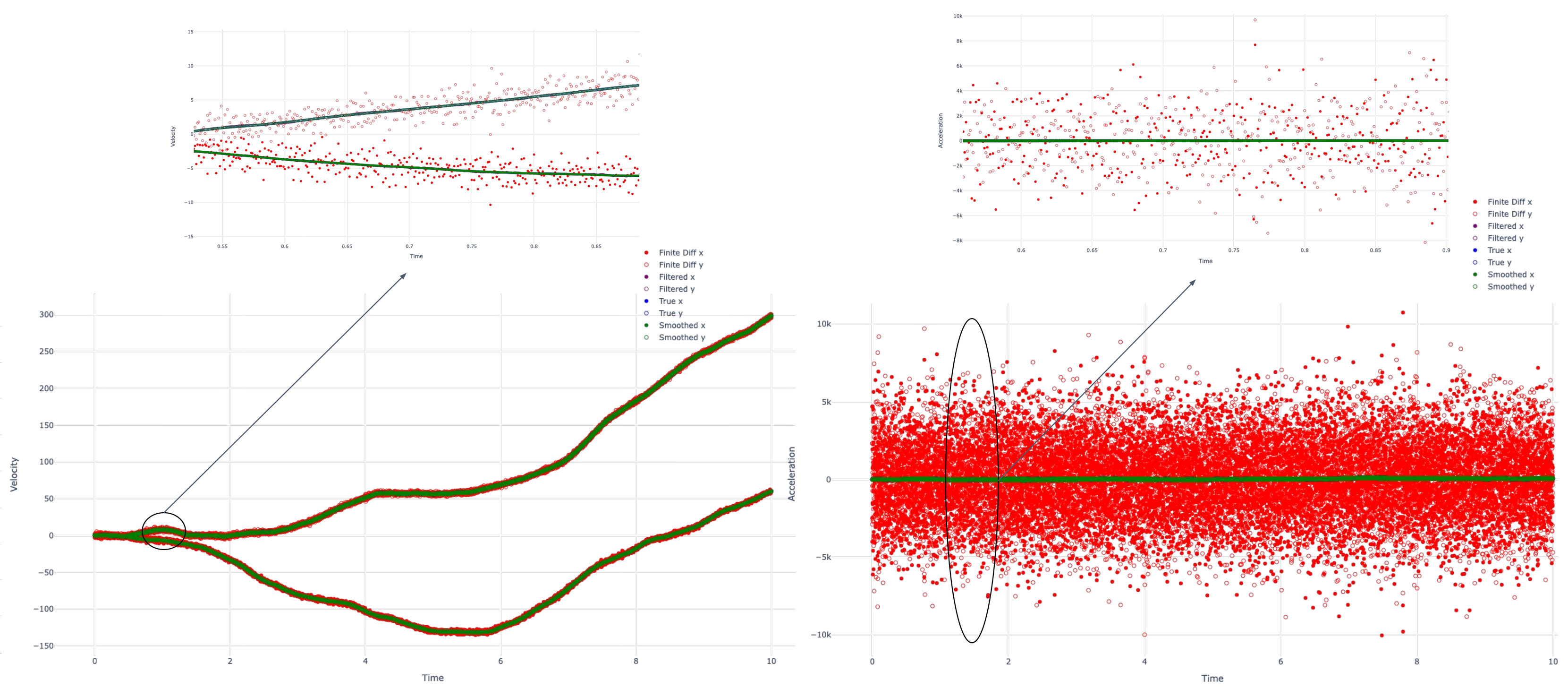
Note that red symbols indicate the model's free parameters.

8. Simulated Mouse

Figures generated by simulation.py [2].
Parameters: $\Delta t = e^{-3}$, $m_0=[0, 0, 0, 0, 0, 0]$, $V_0=\text{diag}([0.001, 0.001, 0.001, 0.001, 0.001, 0.001])$, $\sigma_x = \sigma_y = e^{-3}$, $\gamma_x = \gamma_y = 1.0$



(a) Positions: there is so little noise that measured and true are indistinguishable

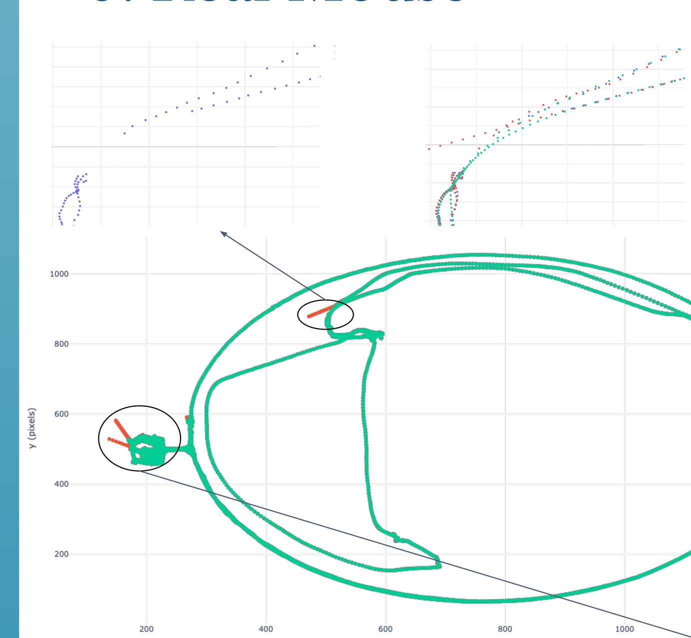


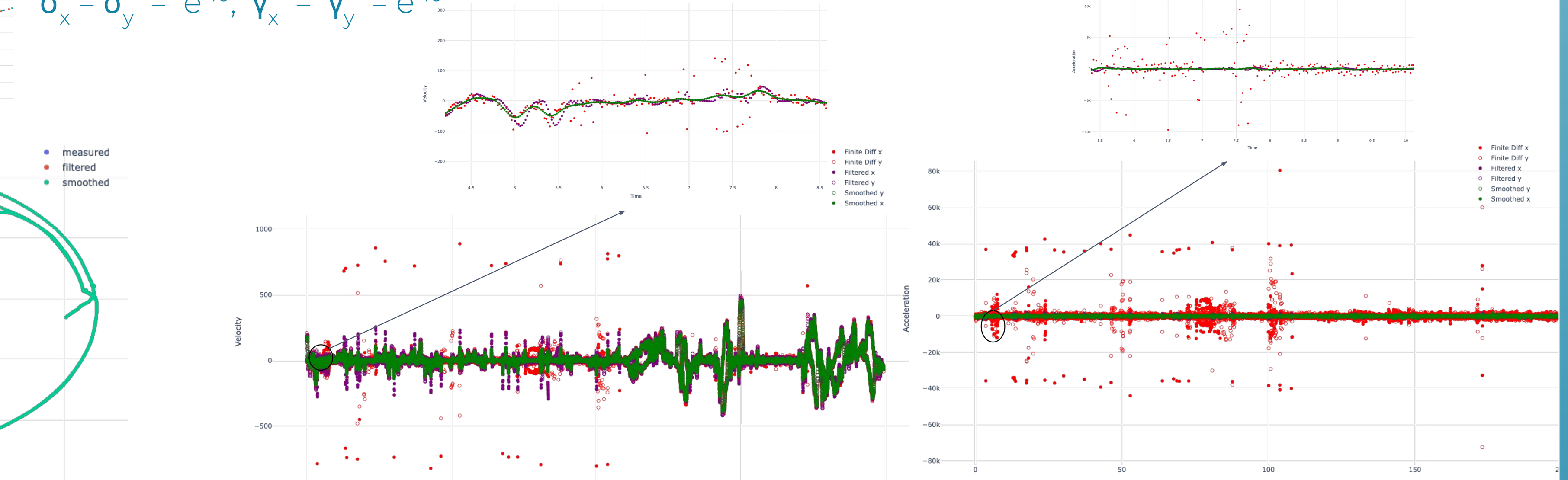
(b) Velocities: All velocity estimates are accurate, but those from the Kalman filter and smoother are more accurate than estimated by the finite differences method.

(c) Accelerations: acceleration estimates from the Kalman filter and smoother are accurate, but not those from the finite difference method. Finite difference estimates of acceleration are substantially worse than those of velocity.

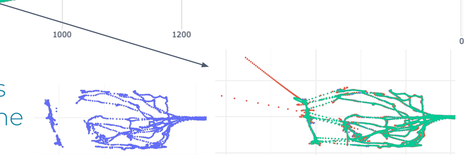
9. Real Mouse

Figures are generated by detect.py [2].
Parameters: $m_0=[273, 0, 0, 542, 0, 0]$, $V_0=\text{diag}([0.001, 0.001, 0.001, 0.001, 0.001, 0.001])$
$\sigma_x = \sigma_y = e^{-10}$, $\gamma_x = \gamma_y = e^{-10}$





(a) Positions: estimates of missing positions by the Kalman smoother, but not those by the Kalman filter, are reasonable.



(b) Velocities: Overall, Kalman filter and smoother estimates of velocity agree with those of the finite difference method, but appear to be less noisy.

(c) Accelerations: Consistently with Figure 9c, acceleration differences between estimates by the finite difference methods and those by the Kalman filter and smoother are orders of magnitude larger than velocity differences.

10. Conclusions

We used computer vision techniques to acquire good estimates of positions from a high-quality video of a mouse foraging in an arena. We provide a repository with Python code for inference and learning in linear dynamical systems. We used this code to infer positions, velocities and accelerations of mice, even at times where mice were not visible (Figure 9a). For simulated data, velocities and accelerations inferred by the Kalman Filter and Kalman Smoother were more accurate than those extracted using the conventional finite difference method (Figures 8a-8c). For position measurements from a foraging mouse, the Kalman smoother, but not the Kalman filter, provided reasonable position estimates at times with missing measurements. Consistently with the simulated data, velocity and acceleration estimates of the foraging mouse by the Kalman filter and smoother appeared less noisy than those estimates by the finite differences method (Figures 9a-9c).

References

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

[2] https://github.com/Ash530888/LDS_KF_KS.git

[3] Bar-Shalom, Y., Kirubarajan, T. and Li, X., 2007. Estimation with applications to tracking and navigation. New York, NY: Wiley.