Worksheet: Linear Dynamical Systems

Joaquin Rapela and Aniruddh Galgali

March 8, 2024

1 Sampling from a linear dynamical system

Sample (i.e., simulate) observations from a linear dynamical system for tracking. Follow the equations on slide 16 of the Linear Dynamical Systems lecture. That is, first sample an initial state $\mathbf{x}_1$, then sample the remaining states $\mathbf{x}_2, \ldots, \mathbf{x}_N$, and finally simple the observations $\mathbf{y}_1, \ldots, \mathbf{y}_N$.

You may use the script doSimulateTrajectoryDWPA.py in the class repository, but you will need to complete a few missing parts in the module simulation.py. You should obtain a plot similar to that in Figure 1.

2 Tracking a foraging mouse

Here we are going to track a foraging mouse using linear dynamical systems[1]. For this you may want to use the script doTrackMouse.py. This script reads a video file and using the opencv computer vision library it estimates the filtering distribution of the center of mass of the foraging mouse. It plots a red point in the video frame at the position of the opencv estimate.

This opencv estimate is noisy and sometimes is not available due to occlusions. Here we use the Kalman filter to estimate the true center of mass of the mouse from the noisy observations. At each time point the Kalman filter provides the mean and covariance of the posterior distribution of the center of mass. The script plots a greeen point in the video frame at the location of this mean and a 95% confidence ellipse summarizing the variability of the posterior distribution. Figure 2 plots one example frame of the video generated by the script.

References

Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[1] The video for this example was generously provided by the Sainsbury Wellcome Centre Foraging Behaviour Working Group (2023). Aeon: An open-source platform to study the neural basis of ethological behaviours over naturalistic timescales, `https://doi.org/10.5281/zenodo.8413142`.

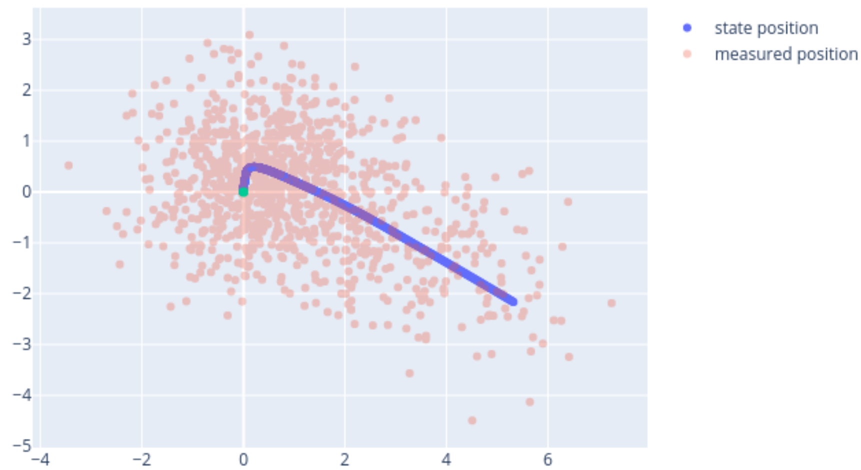

Figure 1: Simulated observations and state position components using the Discrete Wiener Process Acceleration model [Bar-Shalom et al., 2004, Section 6.3.3].

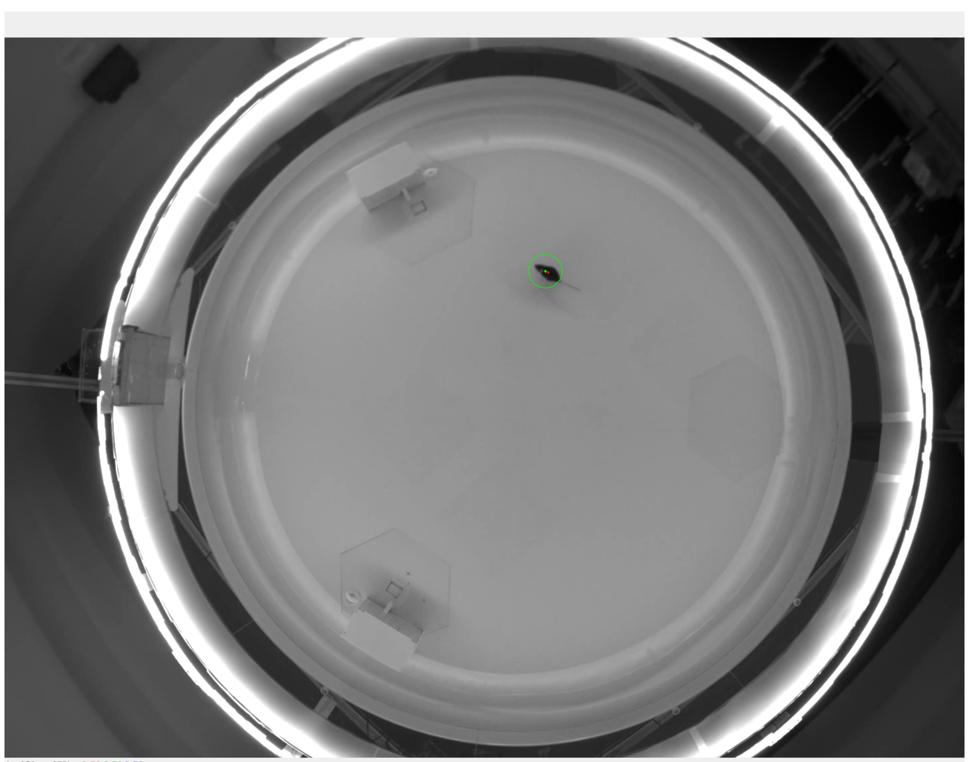

Figure 2: Example frame of the video generated by the doTrackMouse.py script. The opencv estimate of the mouse position is plotted in red, the Kalman filter estimateed mean of the position is plotted in green, and the 95% confidence ellipse of the estimated position is drawn in green.