Advanced tracking

Blaž Erzar

I. Introduction

In this exercise, we implemented two recursive Bayes filtering tracking methods, the *Kalman* and *particle filters*. For the Kalman filter, we use three motion models, random walk, nearly constant velocity and nearly constant acceleration. We run it on three curves and use the same motion models to implement the particle filter tracker, which is evaluated on the VOT14 dataset.

II. Experiments

A. Kalman filter

First, we run the Kalman filter on the spiral curve, see Figure 1. We can see that for high values of q and low values of r, all motion models perform similarly. Because the motion model uncertainty is high enough, they explore enough space and select the best state due to low measurement uncertainty. For higher values of r, the measurement is more uncertain and the random walk model performs the worst. NCV and NCA models perform better because they keep moving in similar directions.

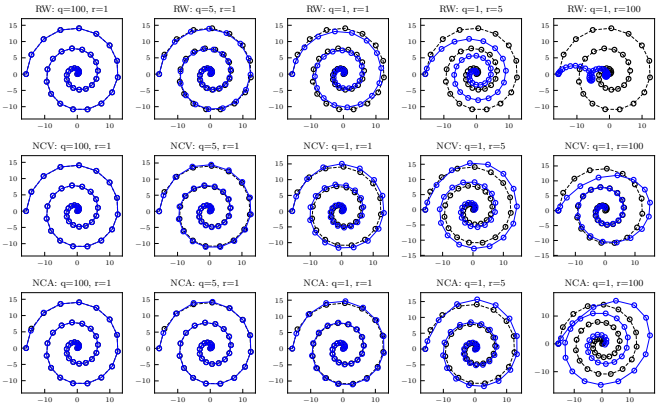

Figure 1: Kalman filter response (blue) on the spiral curve (black) for multiple motion models and parameters.

On the Hilbert curve in Figure 3, we can see similar behaviour. Again, models work best with low measurement uncertainty and random walk breaks down for high r. Here, we can more clearly see that NCV works best for high r.

In Figure 2 we can observe that for uncertain measurements, the random walk model not only lags behind the curve in the vertical direction, but the final horizontal position is also far off.

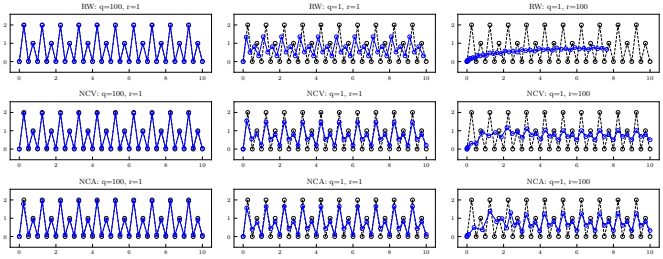

Figure 2: Kalman filter response (blue) on the sawtooth curve (black) for multiple motion models and parameters.

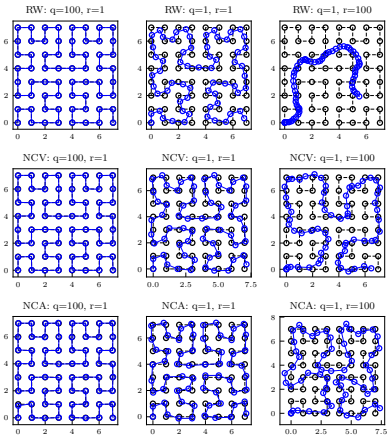

Figure 3: Kalman filter response (blue) on the Hilbert curve (black) for multiple motion models and parameters.

Matrices computed for the motion models are shown in the appendix.

B. Particle filter

The implemented tracker achieves **44 failures** and an average **overlap of 0.50** on the whole VOT14 dataset. The average speed is 159 frames per second. On many sequences, it achieves no or a single failure, e.g., *ball*, *hand2*, *torus*, but it fails 4 times on *fish1* and *fish2*, with up to 5 fails on *motocross*, *tunnel* and *woman* sequences.

We use 100 particles and $q = 0.6$, which is multiplied by the smallest of the bounding box sizes. The selection of this parameter is explained in the next section. We update the visual model in each step with the weight $\alpha = 0.05$. For it, we use a histogram with 16 bins per colour channel. We use the Hanning window for weights in the histogram computation. For the histogram distance, we use the Hellinger distance and convert it to probabilities using the Gaussian with parameter $\sigma = 0.05$, which was tuned by hand to produce a variate distribution of weights.

In each iteration, we remove particles which go out of bounds of the image. In case all particles are invalid, we return the previous position and resample the particles, but this case only happens when the tracking fails.

C. Motion models

To compare the motion models, we run all of them on multiple parameters. The comparison between the random walk and NCV is shown in Figure 4. For optimal selection of q, the NCV model performs better. NCA model, on the other hand, performs much worse. By hand-tuning we obtain **176 failures** and **overlap of 0.38**.

D. Number of particles

We also look at the effect of the number of particles on performance. We can see in Figure 5 that the performance is

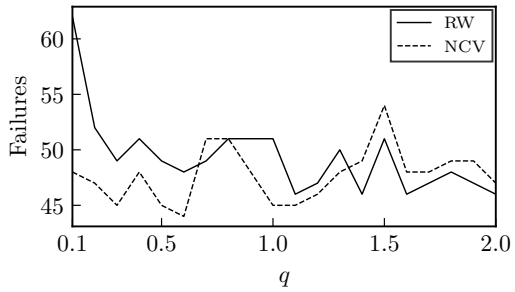

Figure 4: Tracking performance for RW and NCV motion models with different parameter q. In all cases, overlap is between 0.47 and 0.50.

best with 100 particles. With a higher number of particles, the performance does not improve due to diminishing returns and the fact that with 100 particles we already produce a particle close to the target which gets a high weight. The overlap starts at around 0.47 and increases to 0.49 at 50 particles. For higher values, it varies around 0.49 and 0.50.

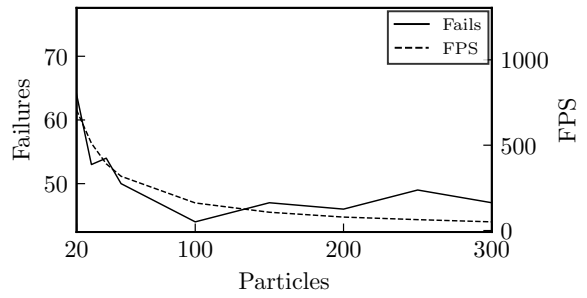

Figure 5: Tracking performance with different number of particles, NCV motion model and $q = 0.6$.

We can observe that with more particles, the framerate drops because of more loop iterations. The performance drops by half when going from 50 to 100 particles, but the differences get lower for higher values because we can model the performance with the function $\frac{1}{x}$.

III. Conclusion

As we can see, the Kalman filter can work with different motion models and does some sort of smoothing. In our case, the NCV model works best, which is also the case for the particle filter. With optimal parameters, we were able to achieve 44 failures on the VOT14 dataset, while showing that performance depends both on the parameter q and the number of particles. Increasing the number of particles can improve the performance, but we have to take into account that with it, the tracker framerate drops.

Appendix

For the **random walk** model, we define the space as $\mathbf{x} = [x, y]$. We are solving the differential equation $\dot{\mathbf{x}} = \mathbf{L}\mathbf{w}$, which means that $\mathbf{F} = \mathbf{0}$. Because the velocity is noisy, $\mathbf{L} = \mathbf{I}$. From these two matrices, we compute

$$\mathbf{\Phi} = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}, \qquad \mathbf{Q} = q \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Because the measurement takes location from the current state, $\mathbf{H} = \mathbf{I}$ and the uncertainty is defined as

$$\mathbf{R} = r \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

which is the same for all models.

For the **nearly constant velocity** model, we are solving the equation $\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{L}\mathbf{w}$ using the state $\mathbf{x} = [x, y, \dot{x}, \dot{y}]$. From this and the fact that the acceleration is noisy, we have

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

We again do the integration and obtain

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{Q} = q \begin{bmatrix} \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix}.$$

The measurement matrix is only different because the state is bigger:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

For the **nearly constant acceleration** model, the differential equation is the same, with matrices

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

and the state is $\mathbf{x} = [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]$. With integration, we get

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{Q} = q \begin{bmatrix} \frac{\Delta t^5}{20} & 0 & \frac{\Delta t^4}{8} & 0 & \frac{\Delta t^3}{6} & 0 \\ 0 & \frac{\Delta t^5}{20} & 0 & \frac{\Delta t^4}{8} & 0 & \frac{\Delta t^3}{6} \\ \frac{\Delta t^4}{8} & 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^4}{8} & 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{6} & 0 & \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^3}{6} & 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix}$$

The measurement matrix is again the same, just bigger:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Because we are processing sequences frame by frame, $\Delta t = 1$.