Advanced tracking

Urša Zrimšek

Introduction

In this exercise we implemented more advanced tracking approach - combination of motion and visual model. We combined a tracker that uses particle filter sequential optimization, comparing histograms extracted at each particle, and Kalman filter motion prediction. For motion prediction we tried different models: random walk, nearly constant velocity and nearly constant acceleration model.

Kalman filter

For the first part of this exercise we analyzed three different motion models, Random Walk (RW), Nearly-Constant Velocity (NCV) and Nearly-Constant Acceleration (NCA) models, using Kalman filter. For testing we used artificially created trajectories instead of tracking, to be able to easier compare the Kalman filter parameters.

For each model we defined the state x and the transition matrix F, $\dot{x} = Fx$, so that it corresponded with the assumptions of the model. We then needed to derive the matrices that recursively determine the states and observations of each model. States are determined by the dynamic model:

$$x_k = \Phi x_{k-1} + w_k$$

and observations follow observation model:

$$y_k = Hx_k + v_k,$$

where $w_k \sim N(0, Q)$ and $v_k \sim N(0, R)$ are noise sequences.

When we solve the differential equation for discrete time steps δt we get the solution:

$$\Phi(\Delta t) = e^{\Delta t F}.$$

If we assume that there is some noise and our assumptions are not exact, $\dot{x} = Fx + Lw$, where L is a matrix that selects to which components of the state we want to add noise and w is noise sequence specified by its covariance q, we get the solution for Q:

$$Q = \int_0^{\Delta t} (\Phi(s)L)q(\Phi(s)L)^T ds.$$

The results for each model can be seen in the appendix of the report. H is a $2 \times n$ matrix, with n being the number of components in the state, with identity at the left side, otherwise zeros (also written in appendix) and $R = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$, with r being the variance of the components of observation model.

We applied these models to different trajectories, and tested different values of q and r. The results can be seen on figure 1. Parameters q and r represent uncertainties of the motion and observation models respectively, so it is not surprising to see that on the left, where motion model variance is bigger, the predictions are almost the same as the original data, and on the other hand, on the right they differ the most. But which parameters are better depends on what we think is the truth. That can be best explained with the last trajectory, where we can guess that the true data should be a circle in the middle of the inside and outside points of the star. If that is our assumption, then the best parameters for RW are $q = 1$ and $r \in [1, 5]$. For $r = 1$ we still have the shape of the star and for $r = 5$ we are already too close to the inner circle and for bigger r the the model output stays too close to the initial point. We can see that also on RW applied to other sequences. For NCV model, the best parameters in this case would be $q = 1$ and $r = 100$, so those that make our model be more dependent on the motion model. But other two trajectories suggest, that this could be too much. For NCA the results look similar, but even with this parameters, we can't reach a shape similar to a circle.

Therefore we can conclude that usually $r : q = 100 : 1$ gives too much importance to motion model, and $r : q = 1 : 100$ to the observation model, but to find the optimal parameters, we need to know what kind of trajectory are we expecting.

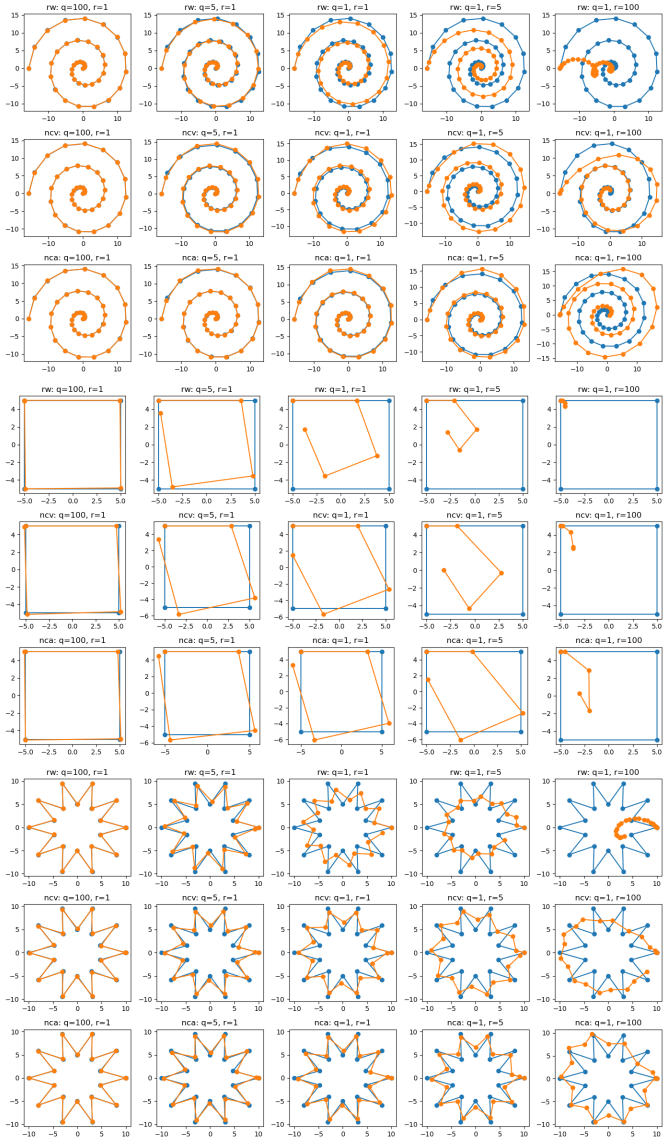

Figure 1. **Motion models applied to different trajectories.** For each trajectory the original is drawn with blue, and the output of the model with orange. The upper represents RW, middle NCV and lower NCA. Columns represent different values of parameters q and r.

Particle filter

Nearly-constant velocity model

Let's look at how NCV model performs on different sequences from *VOT2014* (sequence: number of fails, IoU, tracking speed):

- basketball: 0, 0.7, 50 FPS
- bicycle: 0, 0.49, 78 FPS
- bolt: 1, 0.61, 55 FPS
- car: 0, 0.43, 72 FPS
- david: 1, 0.49, 35 FPS
- diving: 0, 0.37, 44 FPS
- drunk: 0, 0.44, 33 FPS
- fernando: 3, 0.39, 13 FPS
- fish1: 1, 0.33, 83 FPS
- fish2: 2, 0.48, 50 FPS
- gymnastics: 0, 0.63, 49 FPS
- hand1: 0, 0.51, 68 FPS
- hand2: 4, 0.45, 65 FPS
- jogging: 2, 0.73, 62 FPS
- motocross: 4, 0.51, 28 FPS
- polarbear: 0, 0.47, 38 FPS
- skating: 2, 0.44, 42 FPS
- sphere: 0, 0.47, 38 FPS
- sunshade: 0, 0.56, 68 FPS
- surfing: 0, 0.78, 85 FPS
- torus: 0, 0.66, 65 FPS
- trellis: 1, 0.40, 49 FPS
- tunnel: 6, 0.39, 47 FPS
- woman: 3, 0.65, 60 FPS

For these evaluation we ran the tracker with 70 particles, $q = 0.5$, 16 bins in the histogram and template update speed of 0.02. With red we tagged the problematic fails and speed. We can see that once again there is problem with really fast sequences like hand2 and motocross, and because we are using color histograms as the visual model, our tracker fails when we have big changes in light - for example in tunnel and skating. Fernando, that has a big target size is very slow. We could solve this problem by taking less particles where we have a big target size, since we can see that on most other sequences the speed is not so problematic. Number of failures represents robustness of the filter, and IoU represents accuracy. But we didn't take accuracy into account when determining the best parameters, since it is usually better on the same sequence when we have more fails (new bounding box makes IoU bigger if the target changes size or shape). Altogether we have 30 fails, average IoU is 0.52 and average speed is 55 FPS.

Comparison of motion models

We compared all motion models with different values of parameter $\hat{q}$. This parameter is used when calculating the covariance matrix of the motion model. Parameter q used in the formulas in the appendix is calculated as $\hat{q}$ times half of the length of the shorter size of the target. It determines how much noise do we add to the new positions of the particle in each step. That means that bigger $\hat{q}$ makes the particles more spread out over the entire image. Consequently this gives more influence to the visual model, since a lot of the particles land on positions away of the target's way. If the target is really not there, they have small weights and are then sampled with less probability in the next step. But if our motion model is wrong, and the target is somewhere else, this uncertainty makes it possible for the particles to still land on it. Then those will have large weights and will be more important to the calculation of new position, and will also have bigger probability of being sampled. It also makes it possible to find our target again even if we completely lose it (in case we couldn't reinitialize it).

In table I are the comparisons. In each cell we have 2 numbers: number of failures/average IoU. We compared them with the same parameters as above.

Table I
Tracker performance with different $\hat{q}$.

$\hat{q}$	0.1	0.5	1	5	10
RW	80/0.41	44/0.46	34/0.49	**24/0.52**	32/0.53
NCV	42/0.48	**29/0.52**	32/0.52	42/0.54	51/0.54
NCA	128/0.51	128/0.5	**124/0.51**	207/0.5	283/0.49

We can see that NCA model performs the worst. The reason could be that nearly-constant acceleration assumption is not correct for these sequences. When observing separate sequences it doesn't perform better than NCV model on any of them. We also see that it jumps around a lot and wanders away from the target. With big enough $\hat{q}$ RW outperforms both other models. It is better on sequences fernando (1 fail), fish2 (1 fail), tunnel (6 fails) and woman (2 fails). But on fernando we can observe that it gets stuck on the background and the only reason it doesn't fail is the big size of the bounding box. Also on the tunnel, average IoU is very small (0.26), so we can conclude that this two models perform roughly the same. But there is no sequence where it would perform worse, so it should still be our model of choice, if we don't exactly know the assumptions we should make for the target movement. If we look at tracking this model also has the steadiest bounding box. The difference in the size of $\hat{q}$ can be explained with the random noise being the only movement that the particle have in the RW model. So it is better for it to be bigger.

Number of particles

The most computationally intensive part of our algorithm is the extraction of color histograms. In each step we extract them for each particle, which means that the number of particles greatly affects the speed of the model. For all three models the average speed on all sequences with 70 particles is around 55 FPS. If we lower the number of particles to 50, it rises to around 75 FPS. Number of failures with NCV model is then 37 and with RW we have 29 fails – with best $\hat{q}$ for each model. Here we see another advantage of RW model - even with lower number of particles, it still works as good as NCV with 70. Even with 30 particles we still have only 31 fails with fast tracker that can handle 122 FPS. NCV has twice as many fails with 30 particles.

If we take 100 particles, the speed falls under 40 FPS, and with 120 almost to 30 FPS, which is too slow for real time use. With NCV we reach 27 fails with 100 or 120 particles, which is not even that better. The biggest improvement is with NCA model, that can reach only 69 fails with 100 particles, but that is still much worse than other two models. RW model performs even worse with 100 particles then with 70 - it has 26 fails.

We didn't compare accuracy with different numbers of particles, since we can again see that it usually rises with number of fails and from that we can't conclude we have a better tracker.

Conclusion

This tracker proved better than all the previous ones we implemented. The biggest advantage could also be the flexibility between accuracy and speed. We can easily set the number of particles so that it adjusts to our needs. We could also set it automatically at the initialization, depending on the size of the target, so that the speed would be constant, no matter the size of our target (of course, if it is reasonable). Since we only tried one, we could probably also combine it with different visual models, to reach even better results.

Appendix

Random walk

For random walk model we set:

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

and then calculate:

$$\boldsymbol{\phi} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q} = q \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Nearly-constant velocity model

Setting state and transition matrices to:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

we derive:

$$\boldsymbol{\phi} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{Q} = q \begin{bmatrix} \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix}$$

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

Nearly-constant acceleration model

For the last model the set up is:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} \mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

which leads us to:

$$\boldsymbol{\phi} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{3\Delta t^2}{2} & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{3\Delta t^2}{2} \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Q} = q \begin{bmatrix} \frac{9\Delta t^5}{20} & 0 & \frac{3\Delta t^4}{8} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{9\Delta t^5}{20} & 0 & \frac{3\Delta t^4}{8} & 0 & \frac{\Delta t^3}{2} \\ \frac{3\Delta t^4}{8} & 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{3\Delta t^4}{8} & 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{2} & 0 & \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix}$$

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