Exercise 4: Advanced tracking

Nikolay Vasilev

I. Introduction

The **recursive Bayes** filters have become a popular approach for estimating the state of dynamic systems, especially in the field of computer vision. The primary idea behind these filters is to use probability theory to estimate the state of the tracked object based on the available measurements.

In the first part of this paper, we present a detailed description of the implementation of three motion models: *Random Walk (RW)*, *Nearly-Constant Velocity (NCV)*, and *Nearly-Constant Acceleration (NCA)*, using the **Kalman filter** [1]. We evaluate each model with different values of parameters q and r and on three different curves.

Furthermore, we propose a **Particle filter** [2] tracker that uses the *NCV* motion model and a color histogram as a visual model. The proposed tracker is evaluated on the *Visual Object Tracking (VOT)* sequence dataset, specifically the *VOT14* [3], using different parameters, motion models, number of particles and colorspaces for the generation of the histogram. This will help us observe different sequences, find the failure cases and improve them by decreasing the failures and increasing the accuracy and FPS.

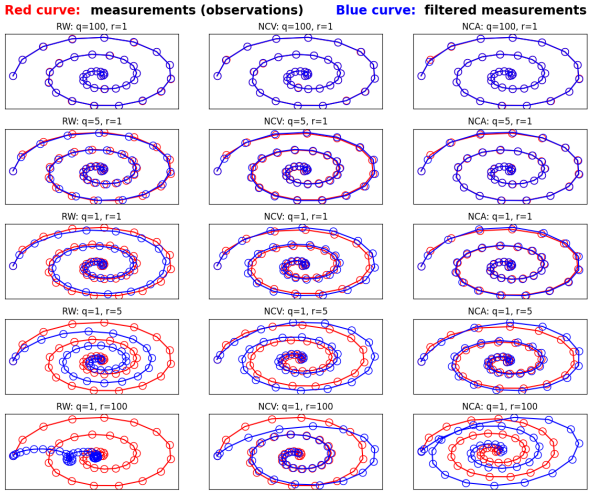

Figure 1. Kalman filter method on a spiral curve using RW, NCV and NCA motion models and different values of parameters q and r.

II. Experiments

The recursive Bayes filters work by recursively updating the posterior probability density function (PDF) of the object's state, given the prior PDF and new measurements. The prior PDF represents the estimated state of the object at the previous time step, while the new measurements provide information about the current state.

In the following section, we will implement the two main types of recursive Bayes filters used in target tracking – the Kalman filter and the Particle filter.

A. Motion models and Kalman filter

The Kalman filter assumes that the state of the object follows a linear Gaussian model, and it updates the posterior PDF using a linear combination of the prior and the likelihood function of the measurements.

The experimentation is done by applying the Kalman filter with three motion models - RW, NCV and NCA. We used the matrices (x, F, , L, Q and H), which are visible in the Appendix (page 3), and different parameters of q and r to apply all three motion models on three different curves - spiral (Figure 1), epicycloid (Figure 2) and hypocycloid (Figure 3). From the evaluation of all curves, we can clearly see that the increasing of the parameter r has a negative impact on the performance of all motion models. Meanwhile, by decreasing the parameter q, both measurements overlap and almost completely match, which means that the performance improves. Following this logic, we can see that the best optical parameters here, which are also the same for all motion models, are $q = 100$ and $r = 1$.

B. Tracking with Particle filter

The Particle filter uses a set of particles to represent the posterior PDF and updates them based on the likelihood of the measurements. In the second part of our assignment, we have

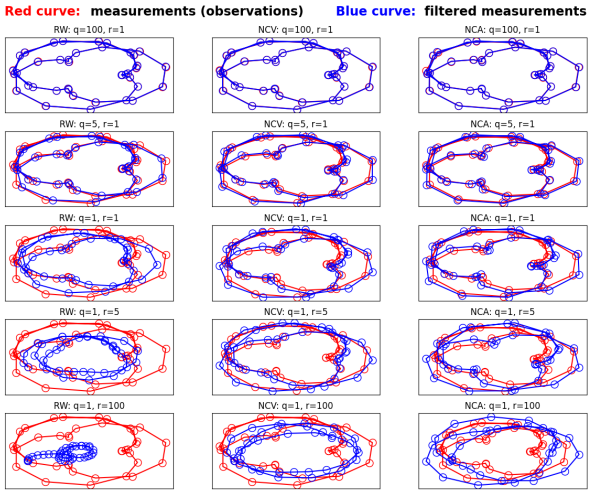

Figure 2. Kalman filter method on a epicycloid curve using RW, NCV and NCA motion models and different values of parameters q and r.

implemented the Particle filter tracker using color histogram as a visual model. The performance was evaluated on the VOT14 sequence dataset with total numbers of errors, average overlap (intersection of prediction and ground truth boxes) and the speed in frames per second (FPS), using different q values, motion models, number of particles and color spaces for the color histogram.

The first thing, that we can easily figure out, is that the parameter q greatly affects the tracker's performance. In our tracker, we have implemented q so it is proportion to the target's size. We use a parameter q_factor, which help us

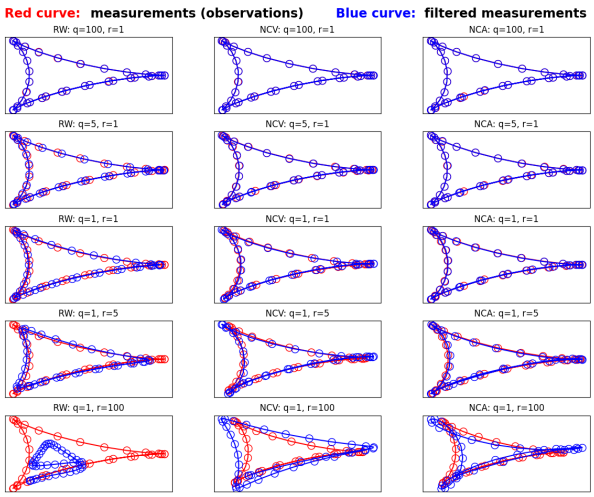

Figure 3. Kalman filter method on a hypocycloid curve using RW, NCV and NCA motion models and different values of parameters q and r.

calculate q the following way:

$$q = max(0, int((H_{img} * W_{img})/(H_{reg} * W_{reg}) * q\_factor))$$

, where H_{img} and W_{img} are the height and width of the current image (frame), and H_{reg} and W_{reg} are the height and width of the searching region. In Table I, we can see how the performance of our model changes with different q values. As expected, from the results we can see that too small q factor (too small q value) makes the performance worse. This happens, because in this case the tracker can't handle well large movements of the target. On the other hand, too big q factor is also a problem, since too large value of the parameter q means that the particles are going over the entire image. From the experiment the best value of the parameter q proves to be, when the q factor is 0.1.

Table I
IMPACT OF THE Q FACTOR ON THE PERFORMANCE AND PROCESSING TIME OF THE PARTICLE FILTER

Parameter	Overlap	Failures	FPS
$q\_factor = 0.05$	0.40	97	12.04
$q\_factor = 0.1$	0.41	88	11.87
$q\_factor = 1$	0.40	136	11.74
$q\_factor = 5$	0.40	156	12.03
$q\_factor = 50$	0.41	248	12.00
$q\_factor = 100$	0.42	334	11.41

As we can see in Table II, the number of particles is another attribute, which affects greatly the performance. It is clear that with increasing the particles, the performance gets better, which is why the best fit for this parameter is 500.

The last important parameter, whose impact on the performance we have to check, is the color space. From Table III, we figure out that the best color space is HSV. This is expected, since it separates the color information from the brightness information, which makes it perfect for tracking. On other hand the YCRGB color space gives the worst performance, since it separates the image into luminance and chrominance components. These components may not capture the color information as accurately as the other color spaces, which is why it does not perform so well for tracking.

Table II
IMPACT OF THE NUMBER OF PARTICLES ON THE PERFORMANCE AND PROCESSING TIME OF THE PARTICLE FILTER

Parameter	Overlap	Failures	FPS
$num\_particles = 10$	0.39	199	273.02
$num\_particles = 50$	0.40	118	94.42
$num\_particles = 100$	0.39	99	55.43
$num\_particles = 150$	0.41	99	36.98
$num\_particles = 200$	0.41	112	28.52
$num\_particles = 500$	0.41	88	11.87

Table III
IMPACT OF THE COLOR SPACE ON THE PERFORMANCE AND PROCESSING TIME OF THE PARTICLE FILTER

Parameter	Overlap	Failures	FPS
$color\_space = HSV$	0.41	88	11.87
$color\_space = LAB$	0.40	111	11.91
$color\_space = RGB$	0.39	94	12.05
$color\_space = YCRCB$	0.39	126	12.05

To get the best results we used the parameters that we discovered from the experiments to fit the best our tracker. We also found the values of other important parameters, such as $bins = 12$ (number of bins of the color histogram), $alpha = 0.05$ (used to update the visual model), $distance_sigma = 0.1$ (used to calculate the probability from the Hellinger distance), $kernel_sigma = 0.5$ (used to calculate the Epanechnikov kernel). In Table IV, we can see the results our tracker achieved

Table IV
IMPACT OF THE MOTION MODEL ON THE PERFORMANCE AND PROCESSING TIME OF THE PARTICLE FILTER USING THE BEST FITTED PARAMETERS

Parameter	Overlap	Failures	FPS
$motion\_model = NCV$	0.41	88	11.87
$motion\_model = RW$	0.39	99	12.10
$motion\_model = NCA$	0.43	134	11.28

on different motion models using these parameters. The NCV motion model seems to be working the best, since it has pretty reasonable accuracy and the least failures. Anyway, even though the NCA model has the most failures, it also has the biggest accuracy. This give us the freedom to choose, which motion model better fits our tracker depending on the use case.

III. CONCLUSION

We can conclude that these findings highlight the potential of the Particle filter, especially when it comes to handling non-linear motions. Depending on the use case, we can choose a corresponding motion model, since that can significantly affect the trade-off between robustness and accuracy. We believe that the Particle filter may be further improved by implementing adaptive scaling, occlusion handling, other visual models. Even just by focusing on the sequences, which generate the most errors and/or lowest accuracy, we can find techniques and parameter, which better fit the tracker and increase its processing time and performance for to best fit our tracking use case.

References

[1] P. Senna, I. N. Drummond, and G. S. Bastos, "Real-time ensemble-based tracker with kalman filter," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2017, pp. 338–344.

[2] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An adaptive color-based particle filter," *Image Vision Comput.*, vol. 21, no. 1, p. 99–110, jan 2003. [Online]. Available: https://doi.org/10.1016/S0262-8856(02)00129-4

[3] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, and G. Fernández, "The visual object tracking vot2014 challenge results," in *Computer Vision - ECCV 2014 Workshops*. Cham: Springer International Publishing, 2015, pp. 191–217.

Appendix

1. Matrices for the **Random Walk (RW)** model:

$$\text{state} = \begin{bmatrix} x & y \end{bmatrix}$$

$$\mathrm{F} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \Phi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathrm{L} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathrm{Q} = \begin{bmatrix} \Delta T * q & 0 \\ 0 & \Delta T * q \end{bmatrix}$$

$$\mathrm{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathrm{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$

2. Matrices for the **Nearly-Constant Velocity (NCV)** model:

$$\text{state} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}$$

$$\mathrm{F} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \Phi = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathrm{Q} = \begin{bmatrix} \frac{\Delta T^3 * q}{3} & 0 & \frac{\Delta T^2 * q}{2} & 0 \\ 0 & \frac{\Delta T^3 * q}{3} & 0 & \frac{\Delta T^2 * q}{2} \\ \frac{\Delta T^2 * q}{2} & 0 & \Delta T * q & 0 \\ 0 & \frac{\Delta T^2 * q}{2} & 0 & \Delta T * q \end{bmatrix}$$

$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \mathrm{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$

3. Matrices for the **Nearly-Constant Acceleration (NCA)** model:

$$\text{state} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & a_x & a_y \end{bmatrix}$$

$$\mathrm{F} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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

$$\mathrm{L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

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

$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathrm{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$