ACVM - Particle filter tracking

Sebastijan Trojer

I. Introduction

In this project we implemented 3 different motion models, namely random walk, near-constant velocity and near-constant acceleration model, for the Kalman filter, and a particle tracking filter, using those same models. We evaluated the Kalman filter on some artificial data to better understand where which model performs better, and evaluated the particle filter tracker on the VOT2014 dataset [1].

II. Experiments

A. Kalman filter

In the first part we implemented a motion models that we used with the Kalman filter, specifically we were tasked with finding the appropriate matrices, namely the system matrix A, the observation matrix C, noise covariance matrix Q and the observation noise covariance matrix R. We implemented random walk (RW), nearly-constant velocity (NCV) and nearly constant acceleration (NCA) models using Kalman filter. The matrices for each of the models are in the appendix.

We evaluated the models on curve from the slides, square and sine wave. The results are shown on Figure 2. Notice that when q is high and r is low, the filter is very responsive to the measurements, closely fitting the data, which in a practical application, could mean a noisy estimate. On the other hand, if q is low and r is high, the filter relies more on the motion model, rather than the measurements, so the estimations are smoother, however further away from the measurements. We can notice that a very high r and low q causes the estimations to rely on measurements far too little to be useful in a real-life scenario. The filter seems to perform the best when the ratio between q and r is more reasonable and the best value depends on our process. Figure 1 shows the same plot, but we added some random noise to the data, to outline how filters with different parameters perform on noisy measurements.

We can also outline some differences between the models from the Figure 2. For example we can notice that the random walk filter struggles with dynamic motion, such as sprials. It also very noticably fails at high r value, however in the second column, the performance of random walk is still comparable to that of the other two models. Similarly, the NCV model also fails at high r parameter values, however the fail is much less extreme. It is able to fit the smooth trajectories much better (spiral, sine), but it is still unable to follow a jagged path well (square). NCA seems to have the best performance overall, however it still cannot compensate for poor $\frac{q}{r}$ ratio. The NCA estimation could be further improved by fine-tuning the P matrix, to find the appropriate ratio of uncertainty.

B. Particle Filter

In the next part we implemented particle filter with NCV motion model and evaluated it on the VOT2014 dataset using the VOT toolkit. The results for different parameters are given in Table I. We used color histograms as the visual model. From the results, we can notice a large dependency of speed on the number of particles, a filter with 10 particles works almost 10× faster than one with 100 particles. Another interesting observation is that a low particle count with large spread performs much worse, than a somewhat larger particle count on a smaller spread (q), which indicates, that a lower q might be more suitable, as particles tend to spread apart during tracking, and at least few 10s of particles are required for consistent tracking, or the model will not be able to capture enough information. Given how the tracking speed slows down with the particle count, we found that about 100 particles offers the best balance between tracking performance and speed. The previous observation about a smaller spread being more suitable is confirmed in all of the experiments as tests with q values over 0.5 consistently underperform the ones with $q \leq .5$. Overall we got relatively good tracking performance on the dataset, especially when compared to the trackers we implemented in the past, such as the correlation based tracker and the mean shift tracker, although it comes at the cost of speed.

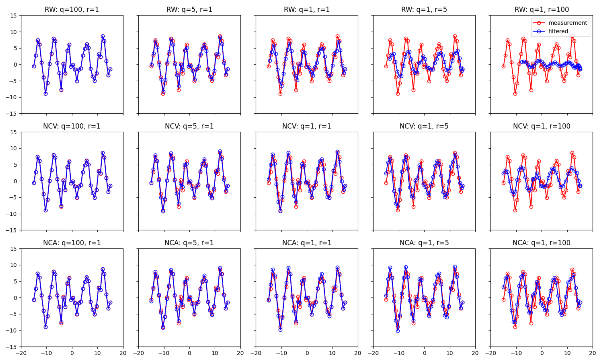

Figure 1. Kalman filters on a noisy sinusoid. Notice how larger r values tend to approximate a true sinusoid better, whereas high q follows noise too much.

N	q	Avg. Overlap	Failures	Avg. FPS
10	0.3	0.49	50	573.34
10	0.5	0.49	41	532.16
10	0.7	0.50	58	512.84
10	0.9	0.49	55	542.79
50	0.1	0.54	17	143.62
50	0.3	0.56	21	128.65
50	0.5	0.54	24	142.65
50	0.7	0.55	21	138.97
50	0.9	0.55	26	144.04
100	**0.1**	**0.54**	**19**	**74.90**
100	0.5	0.54	22	72.05
100	0.7	0.55	24	75.86
100	0.9	0.55	24	73.28
200	0.1	0.54	20	37.60
200	0.3	0.56	23	39.55
200	0.5	0.54	24	38.37
200	0.7	0.55	24	39.11
200	0.9	0.57	25	38.47

Table I
Evaluation results for Particle Filter with varying N and q.

During testing, we also visualized the tracker and the particles, to track performance visually. Two examples of the visualized particles are shown in Figure 3. In most cases, the tracker

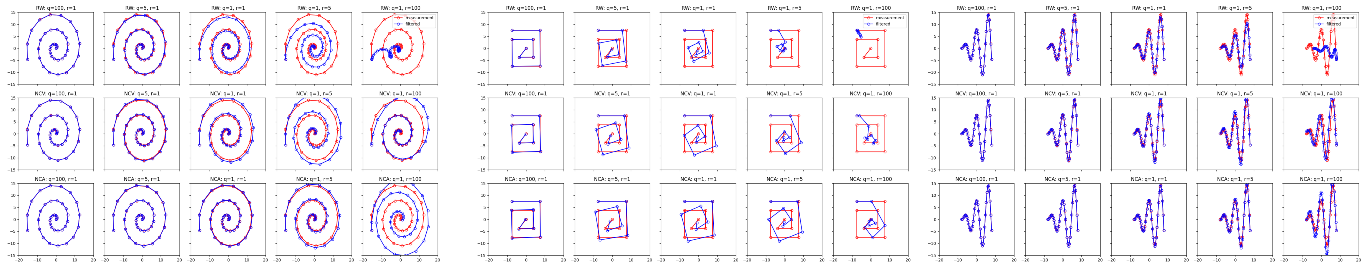

Figure 2. RW, NCV and NCA models on three different samples of data with different values of process noise q and measurement noise r.

managed to stay on target, like on the left example, however one of the biggest problems are that the tracker sometimes drifts, like in the second example. In this specific example, the problem was most likely in the similar color composition of the arm and the model relying on the measurements too much, rather than having a balance between the motion model and measurements.

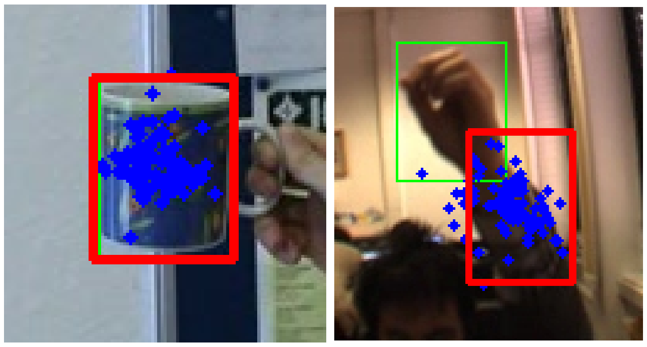

Figure 3. An example where the tracker was positioned well (left) and poorly (right) on the target, with visualized particles.

We also compared particle filter tracker using all three different motion models. The results are shown in Table II. Surprisingly,, even the RW particle filter significantly outperformed the NCA filter. We even tried evaluating it at several other values of q, however, it still performed the worst by far. The best result we were able to achieve was by increasing the particle count to 300, which slowed it down significantly, to about 23 FPS, however, that yielded an average overlap of 0.55 and 31 failures. In the original setting, however, both RW and NCV motion models beat it both in average overlap and number of failures. The reason for this might be that NCA makes an unrealistic assumption of smooth acceleration, whereas in the videos there are either relatively constant speed (no acceleration) or swift and abrupt movements (non-constant acceleration). The RW outperforms it since it is a simpler and therefore more robust model, which can, similarly as NCA, handle more such realistic motions, since they are based on fewer assumptions. NCA is also more complex, making it more sensitive to the parameter values, which would require additional experimentation.

Motion model	**Avg. Overlap**	**Failures**	**Avg. FPS**
RW	0.49	44	60.45
NCV	0.54	19	74.90
NCA	0.54	114	63.88

Table II
Evaluation results particle filter with random walk, near-constant velocity and near-constant acceleration motion models. We set the parameters to $N = 100$ and $q = 0.1$, since that's where NCV performed best in the first part.

C. *Exploring different color spaces*

We also explored how the tracker performs in different color spaces, namely RGB, HSV, Lab and YCrCb. To simplify the experiments, we used the best performing model from the last task, using the NCV model with q set to 0.1 and 100 particles. The results are given in Table III. The performance was relatively similar given any color space, however the HSV color space performed the best. The reason why HSV works better is because it separates the hue from the intensity, which can make the visual model more robust to lighting changes and shadows, and this specific dataset does have a few examples exhibiting such lighting changes. RGB has the same performance as the BGR that we used before, since it is a color space with simply permuted channels. Similarly to HSV, Lab also separates color from intensity, but is more susceptible to low quality images and noisy data, which might be the reason it didn't outperform the HSV model. YCrCb is also robust to lighting changes, however it did not outperform the other color spaces in our case. An approach that could leverage the YCrCb better would be to update the visual model faster, to capture the illumination changes better, since the RGB appears to encode the raw color values better.

Color space	**Avg. Overlap**	**Failures**	**Avg. FPS**
RGB	0.53	19	79.00
HSV	0.54	15	77.48
Lab	0.52	21	73.65
YCrCb	0.51	20	76.71

Table III
Tracker performance using different color spaces for the histograms.

III. Conclusion

We successfully implemented 3 different motion models and thoroughly compared them. We tested them using Kalman filter and evaluated how they perform in different scenarios, and then paired them with the particle tracker to implement a robust tracker. We tested different parameter sets to find the optimal one and, compared how each motion model performs on the data and discussed why.

Appendix

The appendix contains all the matrices used for the different motion models.

a) Random Walk Model:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q = q \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = r \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\text{State: } \begin{bmatrix} x & y \end{bmatrix}^\top$$

b) Near-Constant Velocity Model:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$Q = q \cdot \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix}, \quad R = r \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

$$\text{State: } \begin{bmatrix} x & y & 0 & 0 \end{bmatrix}^\top$$

c) Near-Constant Acceleration Model:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q = q \cdot \begin{bmatrix} \frac{1}{20} & 0 & \frac{1}{8} & 0 & \frac{1}{6} & 0 \\ 0 & \frac{1}{20} & 0 & \frac{1}{8} & 0 & \frac{1}{6} \\ \frac{1}{8} & 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{8} & 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{6} & 0 & \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{6} & 0 & \frac{1}{2} & 0 & 1 \end{bmatrix}$$

$$R = r \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{State: } \begin{bmatrix} x & y & 0 & 0 & 0 & 0 \end{bmatrix}^\top$$

References

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