Exercise 4: Advanced tracking

Žan Pušenjak, *ACVM 2024/25, FRI, UL*

I. Motion models and Kalman filter

First step to a functional tracker was the implementation of the Kalman filter. The algorithm depends on two parameters:

- r - denotes how much the filter trusts the dynamic model (models uncertainty).
- q - denotes how much the filter trusts the observation model (models uncertainty).

We implemented random Walk (RW), nearly-constant velocity (NCV), and nearly-constant acceleration (NCA) models and used them on different functions with various parameter combinations. The results can be observed in the appendix.

Observing the results we can clearly see, that a higher q parameter matches the actual data completely (since the observations can be accurate), where lowering the q and increasing the r parameter increases the dynamic model influence and the difference between the ground truth and our predictions becomes increasingly larger.

II. Particle tracker

We implemented a particle tracker with different dynamic models and compared their performance. Further we tried to find the best set of tracker parameters.

We used the sequences from the VOT challenges[1] more specifically the VOT14 set of challenges that include of twenty-five videos hand picked to test the reliability of trackers.

A. Implementation

We implemented the particle tracker using the hellinger distance between color histograms and the following formula

$$l(i) = e^{-\frac{(d_i^{HELL})^2}{2\sigma_h^2}}$$

to calculate the likelihood. The σ_h parameter was set to 0.1 since setting it any higher than that allowed particles far away to have a likelihood significant enough and the tracker would quickly drift if a similar object came close to the tracked one.

Based on the findings of previous assignments we used the following parameters:

- $\sigma = 1$ - Since we used the Epanechnikov kernel[2] we needed to determine its σ parameter.
- $N_{bin} = 16$ - the number of bins for each color channel

We updated the template based on the parameter α with the formula $q_{k+1} = \alpha q_k + (1 - \alpha)p_k$ where q is the template histogram and p is the current found object histogram.

Surprisingly a higher $\alpha = 0.05$ than one used with the means-shift tracker showed better results.

B. Different models and parameter tuning

Lastly we used grid search to try and find the best possible set of parameters for our dataset. Because of computational reasons we used approximately five possible values for each of the parameters. Note that $\alpha = 0.05$ was fixed according to previous findings.

[1] https://www.votchallenge.net/

[2] https://en.wikipedia.org/wiki/Epanechnikov_distribution

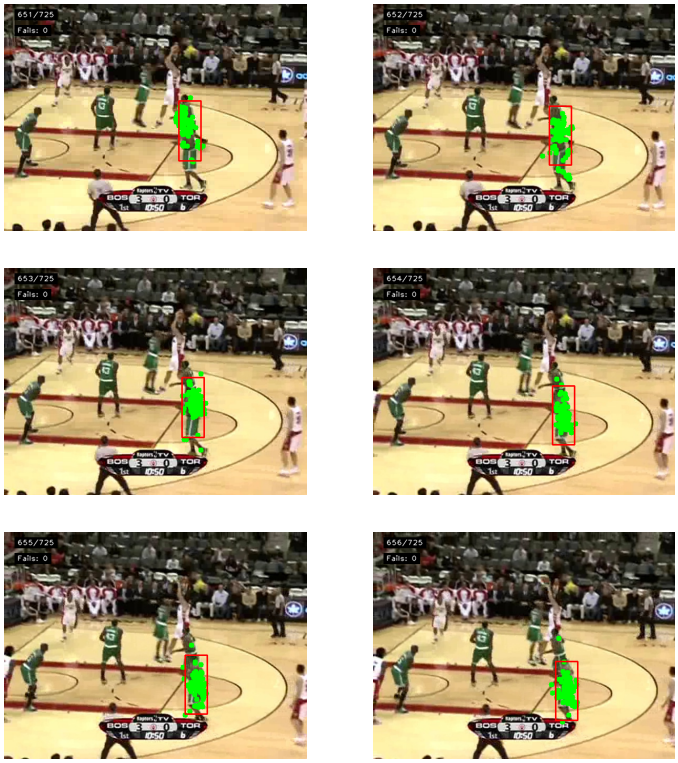

Figure 1: Example of NCA model drift.

Additionally for RW and NCV models the q was calculated based on the formula $\frac{min(w,h)}{6}$ where w, h correspond to the patch width and height, but the value was fixed at 0.1 for the NCA model, since any greater value allowed the particles to move everywhere across the screen. The problem with the q parameter is, that if the target is moving fast a higher q is needed, but a higher q value also means a less stable tracker, since the particles are much more uncertain.

Note that ecause the performance of the NCA model was not good, best score of 135 failures, we only plotted the RW and NCV model performance.

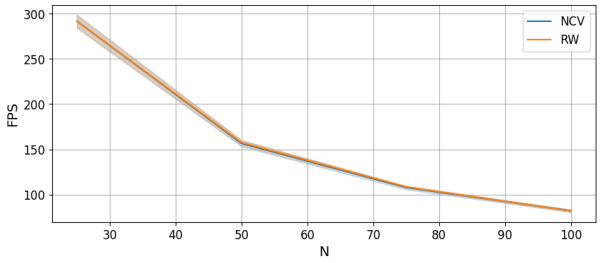

Figure 2: Average FPS on VOT14 challenge at different parameters

We can see that a higher number of particles yields a better performance overall, but is much more computationally expensive, so we have to make a decision about the resources used

and our performance.

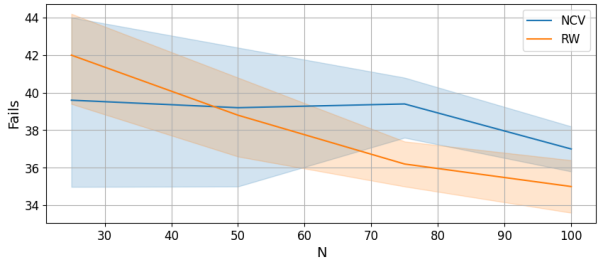

Figure 3: Total fails on VOT14 challenge at different parameters

Also, the decision of the model is very important. Since almost no entity moves with constant acceleration, the NCA model does not achieve good results. If we look at the frames, we can observe, that the particles move around radically. Looking at the `basketball` sequence1, when the player goes near another player the tracker is able to drift off to the other player because the particles are moving around so fiercely.

The best performance of **33 fails** and **81 FPS** was achieved with RW model with hundred particles. *Note that the results varied, sometimes the NCV model achieved better results*

Running the tracker with those parameters with the VOT evaluation toolkit[3] were able to achieve an average overlap of 0.5, 35 failures and an average FPS of 77.03.

[3]https://github.com/alanlukezic/pytracking-toolkit-lite

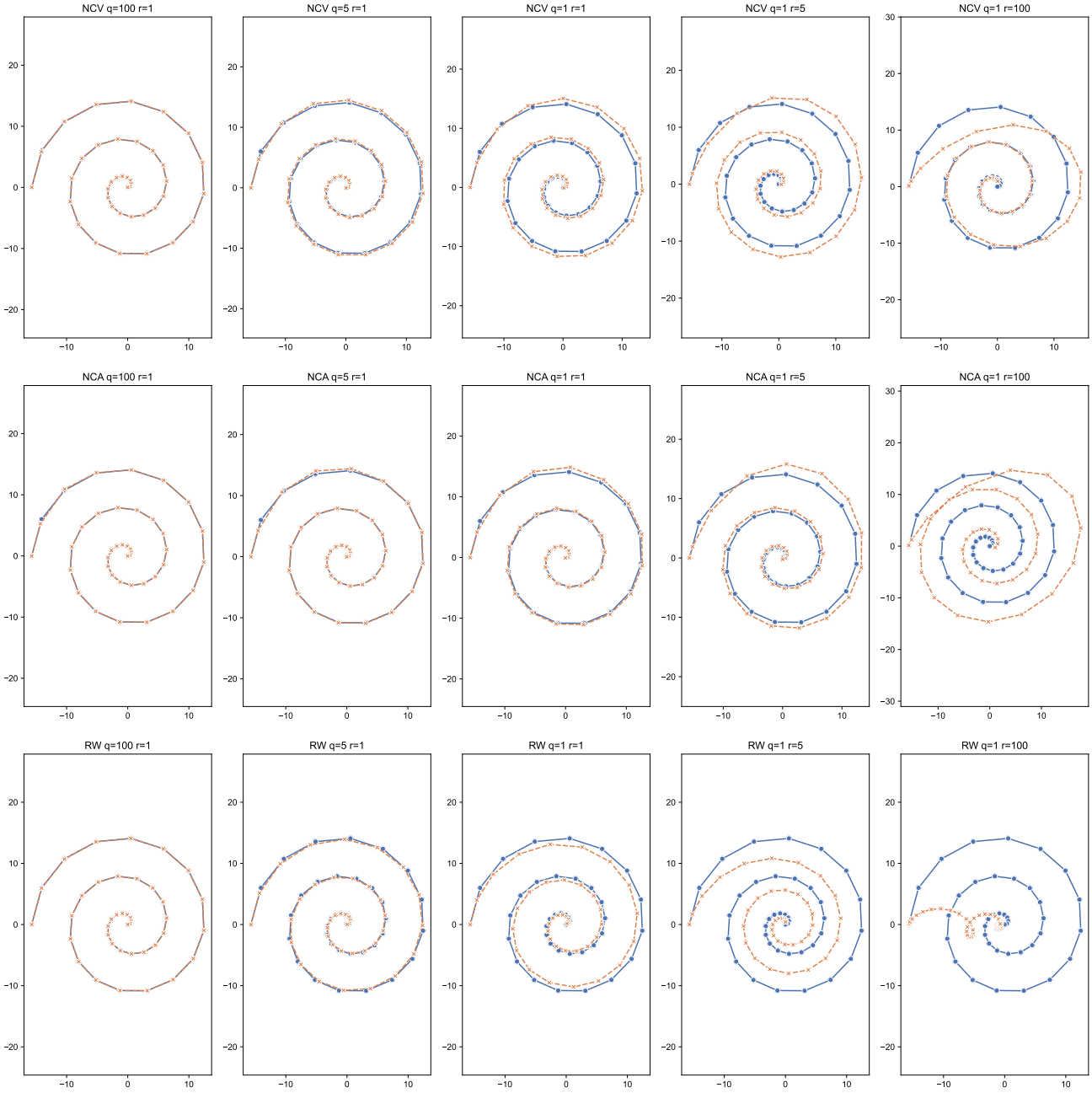

Figure 4: Kalman filter on a spiral pattern.

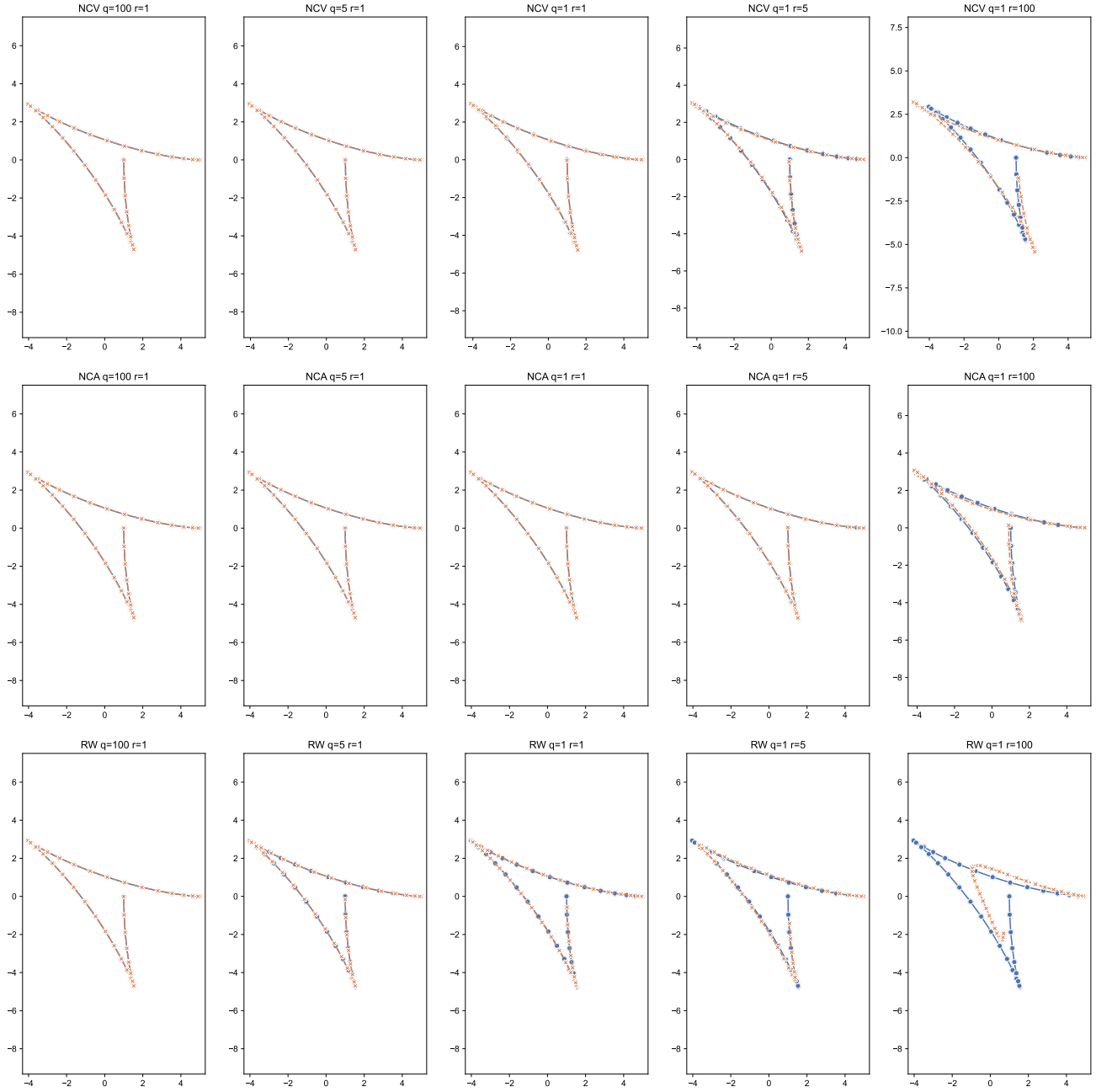

Figure 5: Kalman filter on a hypocycloid pattern.

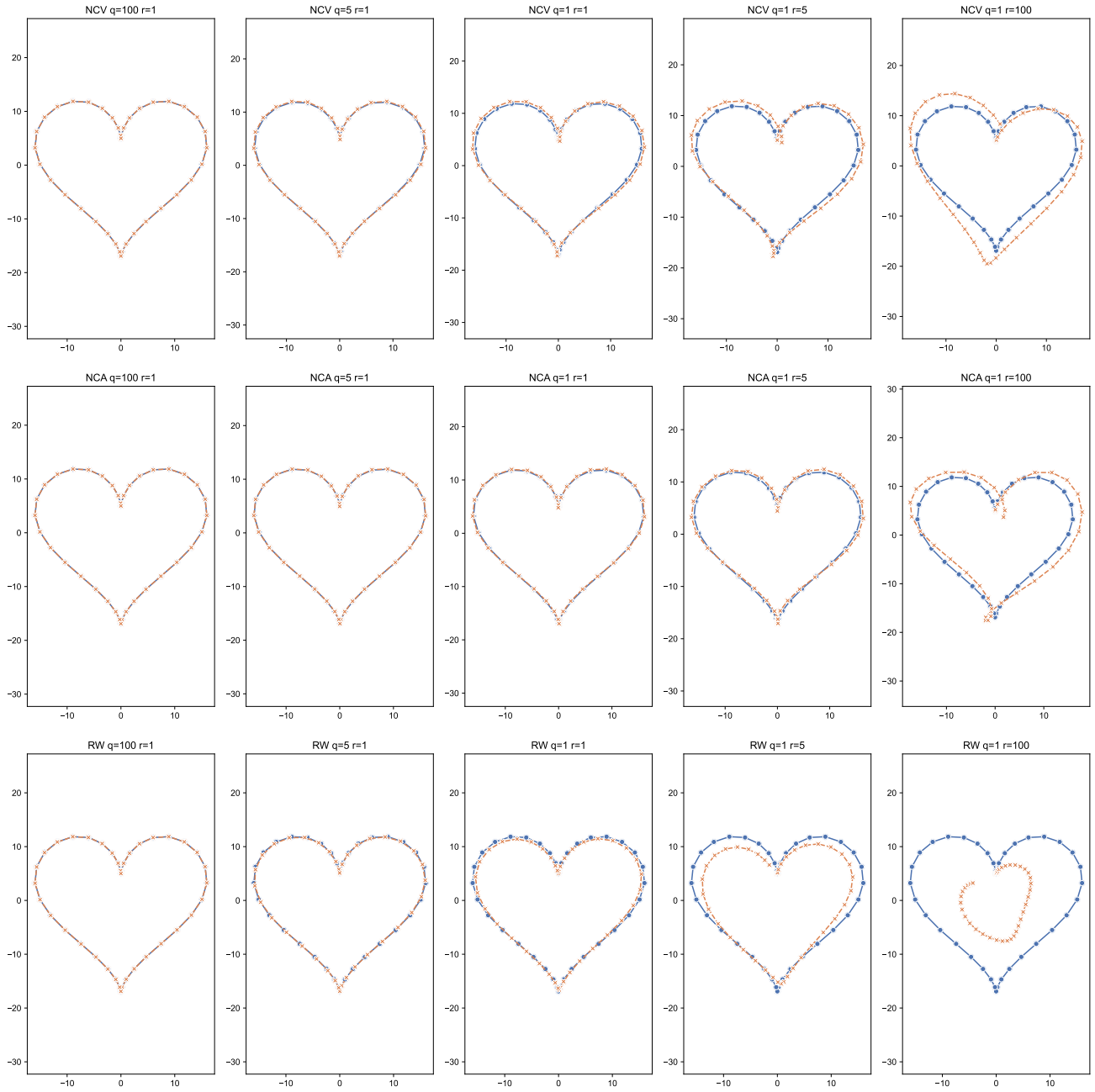

Figure 6: Kalman filter on a heart pattern.

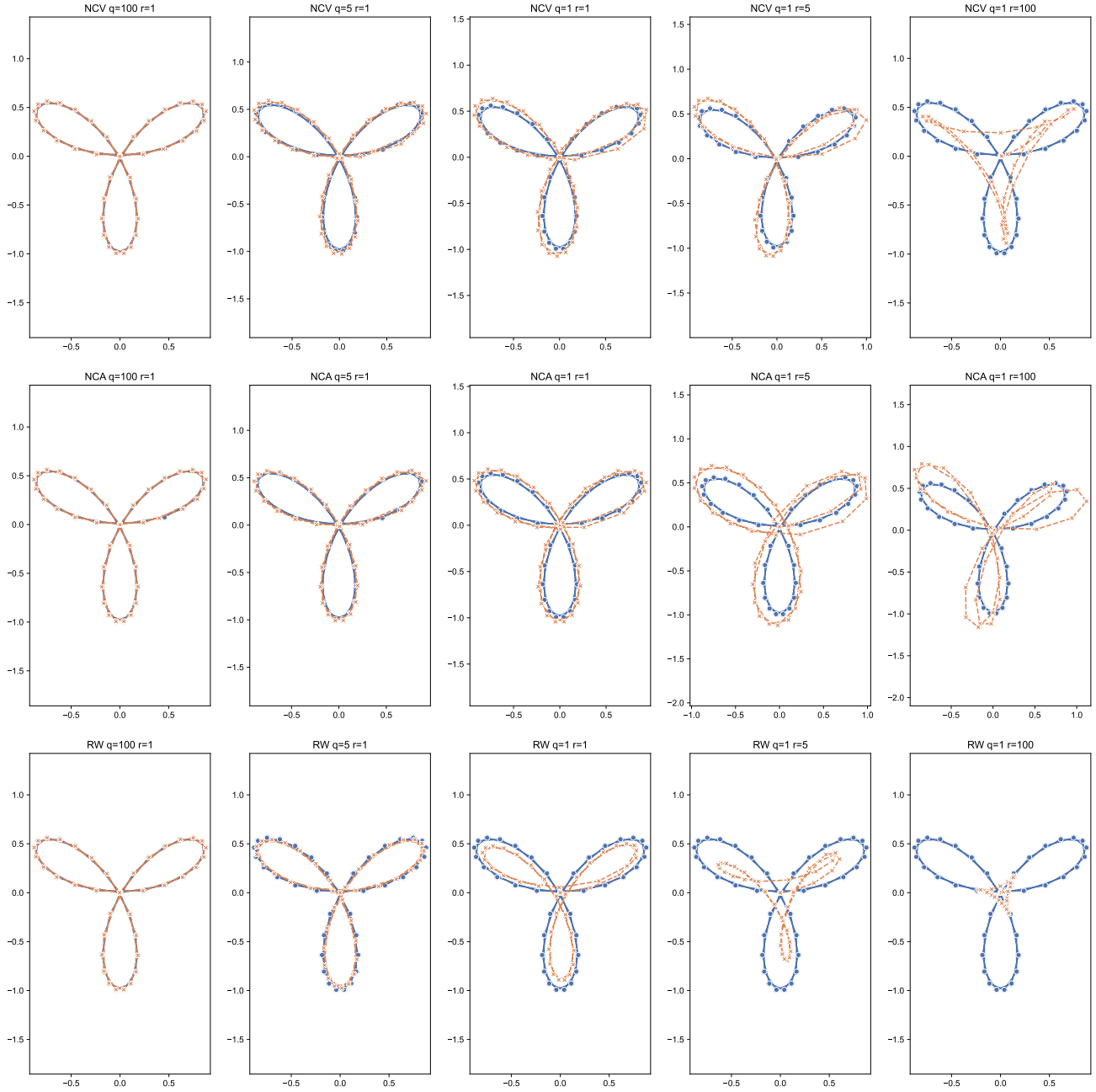

Figure 7: Kalman filter on a rose pattern.