Univerza v Ljubljani

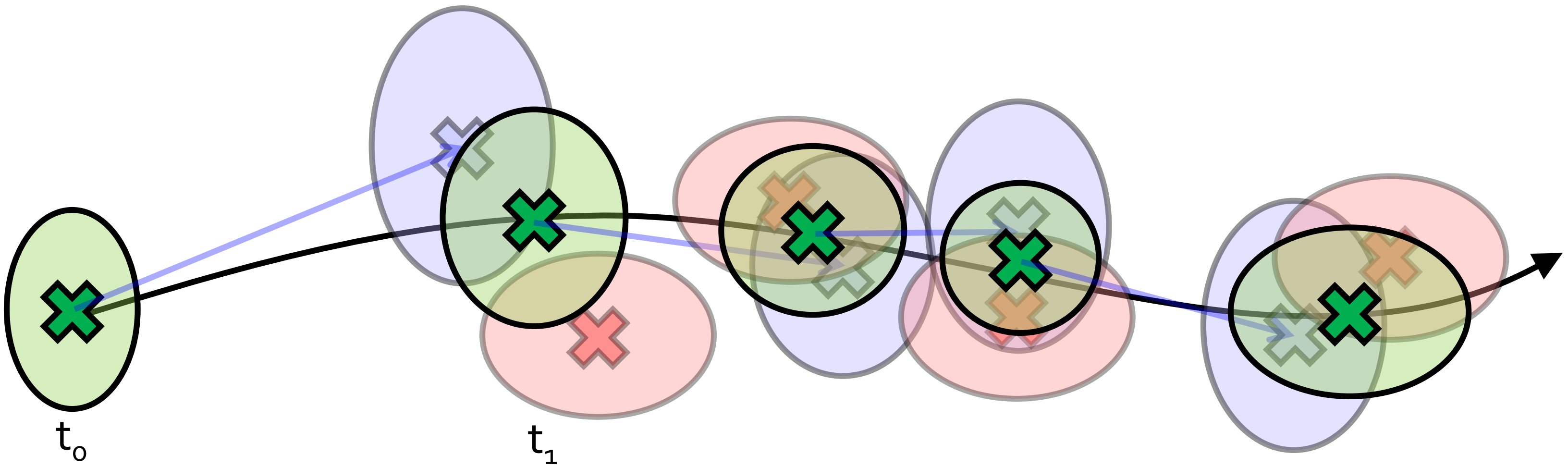
Motion Models and Particle Filter tracker

Advanced Computer Vision Methods
Exercise 4

Visual Cognitive Systems Laboratory,
Faculty of Computer and Information Science,
University of Ljubljana

Motion models: idea

- We have some measurements
  - Typically: noisy
- We want the measurements to be more *consistent*
- Also think about this as smoothing



- Green: our current state (i.e., position) at time t (combined mesurement and prediction)
- Red: measurement (i.e., position from tracker, radar, …)
- Blue: prediction (i.e., from motion model)

Motion model: Kalman filter

- Implement Kalman filter for motion prediction
- Use the theory from lectures
- Most of the Kalman filter is already implemented (see exercise code material: `kalman_update`)
- All you need to do is define input/output elements
- All these elements depend on how you define your state

Motion model: Kalman filter

[x_new, V_new] = `kalman_step(A, C, Q, R, y, x, V)`

- A: System matrix (on lect. slides: Fi or Φ)
- C: Observation matrix (on lect. slides: H)
- Q: System covariance (on lect. slides: Q)
- R: Observation covariance (on lect. slides: R, …)
- y: Current observation (measurement, on lect. slides: y_k)
- x: Prior mean (previous state, on lect. slides: x_k)
- V: Prior covariance (previous covariance, on lect. slides: P_k)

Motion model: Kalman filter

- Define your state x (position, ?velocity?, ?acceleration?)
- Define transition matrix F: ($\dot{x} = Fx$)
- From F obtain Fi – also called Φ
  - RBF1: slide 37
  - If we process frame-by-frame: $\Delta T = 1$
- Define L and derive Q (integrate using Fi and L)
  - RBF1: slide 41
- Define observation matrix H (how y is obtained from x)
  - RBF2: slide 14
- Define observation covariance R
- Initialize x and y, initialize prior covariance P

Motion model: Kalman filter

- System covariance Q depends on parameter q
  - RBF1: slide 41

$$Q = Q(q, \Delta T) = q\, Q(\Delta T)$$

- Observation covariance R

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

Start with q=1, r=1
and find the best values

- Prior covariance P

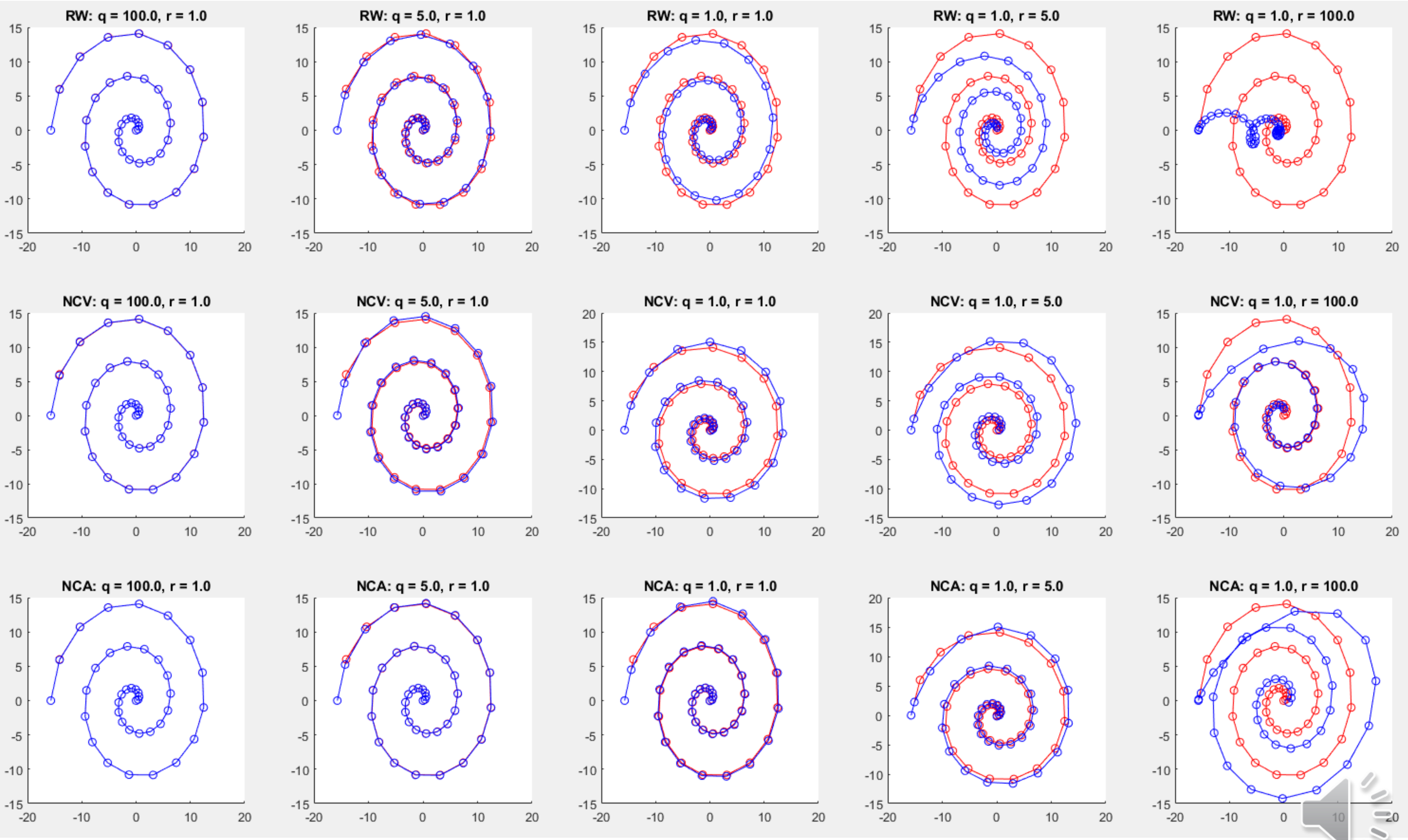
$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \Rightarrow P = \begin{bmatrix} \alpha w & 0 & 0 & 0 \\ 0 & \alpha h & 0 & 0 \\ 0 & 0 & \beta w & 0 \\ 0 & 0 & 0 & \beta h \end{bmatrix}$$

w, h: target width and height

Covariance for speed should be
larger than the position
at the initialization step

Kalman filter: example

Red curve: measurements (observations); Blue curve: filtered measurements

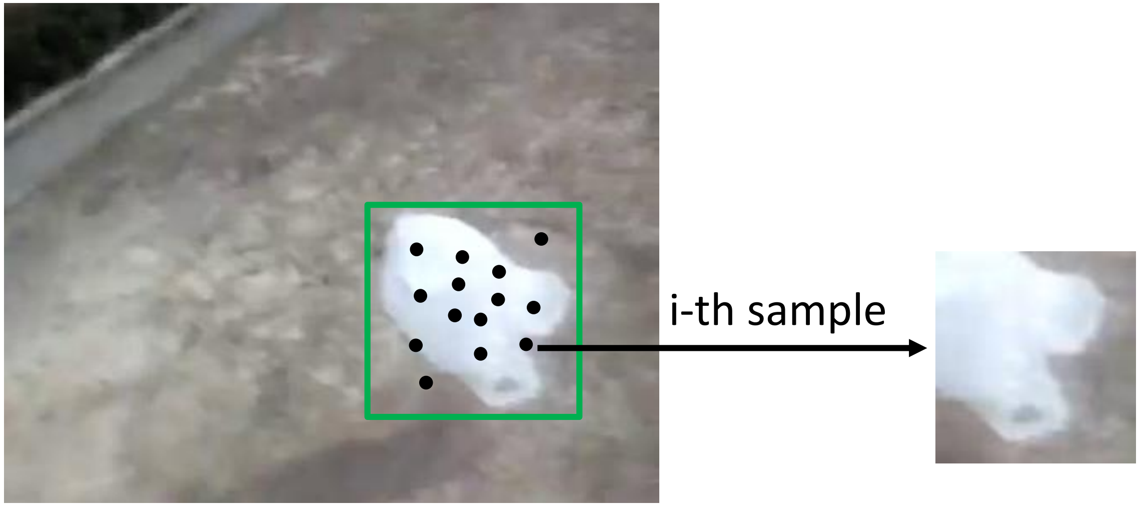


Kalman filter: questions

- Test different motion models
  - RW, NCV, NCA
  - In the report include the following elements for each motion model: state x, F, Φ, L, Q and H
    (You can include these elements on the third page of the report as an Appendix)
- What are the optimal parameters for different motion models?
- How the parameters effect performance?

Particle filter: idea

- Represent the target with N samples
  - Each sample has its own weight w_i

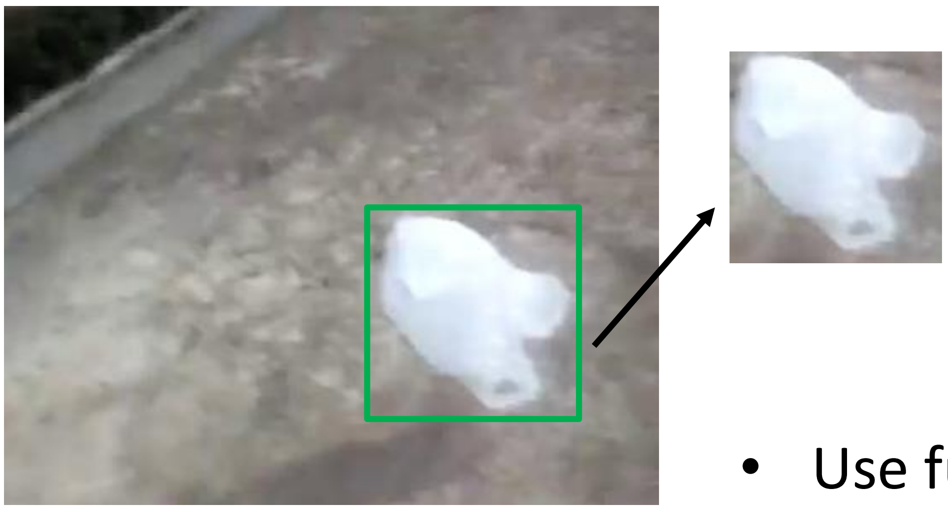


Possible visual representations of samples:
- Color histogram
- Template (NCC)
- Correlation filter

- Track the target by refining the sample set:
  - Keep the samples with high weight
  - Discard samples with low weight

Particle filter: tracker initialization

- First frame: obtain target visual representation
  - Let‘s choose color histogram for visual target representation



Example: extract color histogram $h^{(TAR)}$

- Use function for histogram extraction from Exercise 2 (mean-shift)
- You can use epanechnikov kernel for histogram extraction.

Particle filter: tracker initialization

- First frame: define (choose) motion model
  - Similar idea as Task 1 of this Exercise (but do not use `kalman_update` function)
- Define matrices:
  - Particle state: X = $[\mathrm{x}, \mathrm{y}, \dot{x}, \dot{y}]$ if NCV is chosen
  - System matrix: Φ
  - System covariance Q (experiment with different q, should be dependent on target size)

Particle filter: tracker initialization

- First frame: initialize N samples (particles)

Particles initialized on target center

Add noise sampled from: $\mathcal{N}(0, Q)$

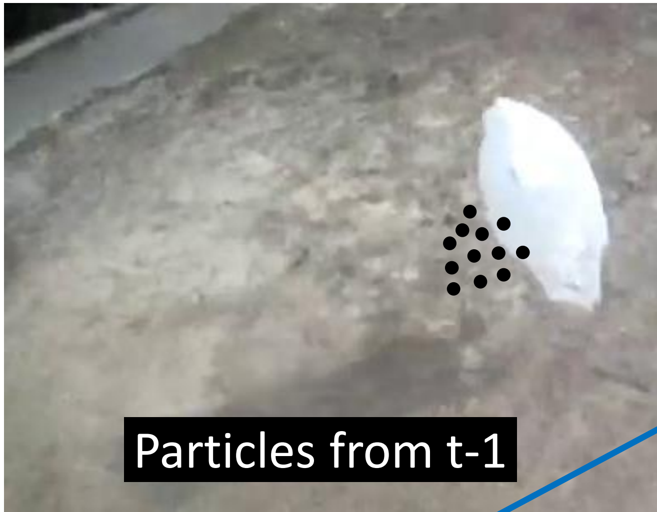
Multivariate normal distribution

See function `sample_gauss` from the material on Učilnica

- All weights are initialized equally i.e., $w_i = 1$

Particle filter: tracking iteration

- New frame (t):



Resampled particles from t-1

- Resample particles according to their weights (some of them will be choosen multiple times, some of them will not be choosen at all)
- Move each particle:
  deterministic shift + noise
  $\Phi\, X_{t-1}$ Sampled from: $\mathcal{N}(0, Q)$
- Re-calculate weights on new positions
- Calculate new position of the target (weighted mean of the position of the particles)
- Update target visual model (histogram $h^{(TAR)}$)

$$h^{(TAR)} = (1-\alpha)h^{(TAR)} + \alpha h_{NEW}^{(TAR)}$$

Compare current position with the target visual model
If target is represented with the color histogram:
Hellinger distance (see next slide)

Particle filter: tracking iteration

- How to re-calculate particle weight?

Let's see just particle i:
How similar is it to the target?

Extract patch from this position

Extract color histogram h_i from this patch

$\boxed{\text{Compare histogram}}$ h_i to the target model (histogram $h^{(TAR)}$)

Hellinger distance:

$$d_i^{(HEL)} = d(h_i, h^{(TAR)})$$

Convert distance to probability:

$$w_i = p_i = e^{-\frac{1}{2}\frac{(d_i^{(HEL)})^2}{\sigma^2}}$$

After localization update visual model (histogram - similar as in mean-shift exercise)

Particle filter: common mistakes

- Particles are going over entire image
  - Parameter q is too large
  - Weights of the particles are not correct (check distance measure, conversion to probability, sigma)
- Tracker doesn’t handle well large movements of the target
  - Parameter q is too small
- Parameter q: typically proportional to target size
- Number of particles N: start with 100,
  find optimal Price/Performance
- Advice for implementation: constantly **visualize** particles
  - Their positions and weights!

Particle filter: tracking example

Particle filter: open questions

- Which motion model did you choose and why?
- What visual model did you choose?
  - Color histogram (which colorspace?, number of bins?)
  - Template (NCC, SSE)
  - Correlation filter (probably slow?)
- How did you set the parameters, their impact to the tracking performance?
- What is the tracking speed?
  - How does it change with the number of particles?