

CS598PS – Machine Learning for Signal Processing

Missing Data and Dynamical Systems

12 October 2017

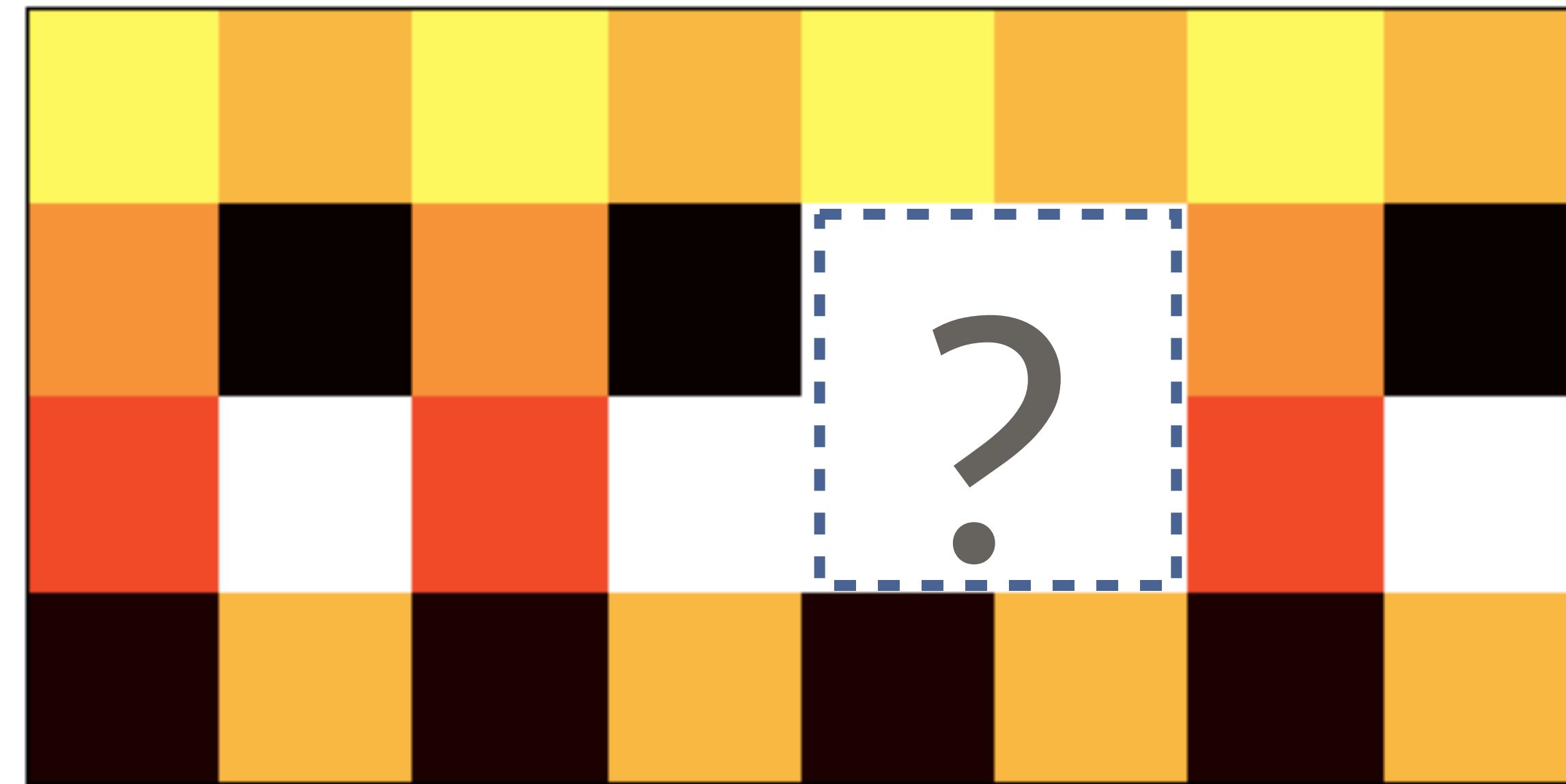
Today's lecture

- Dealing with missing data
- Tracking and linear dynamical systems

Missing data

- You don't always have all your data!
 - e.g. cell phone drops, gaps in pictures, ...
- How do we deal with that?

A simple case

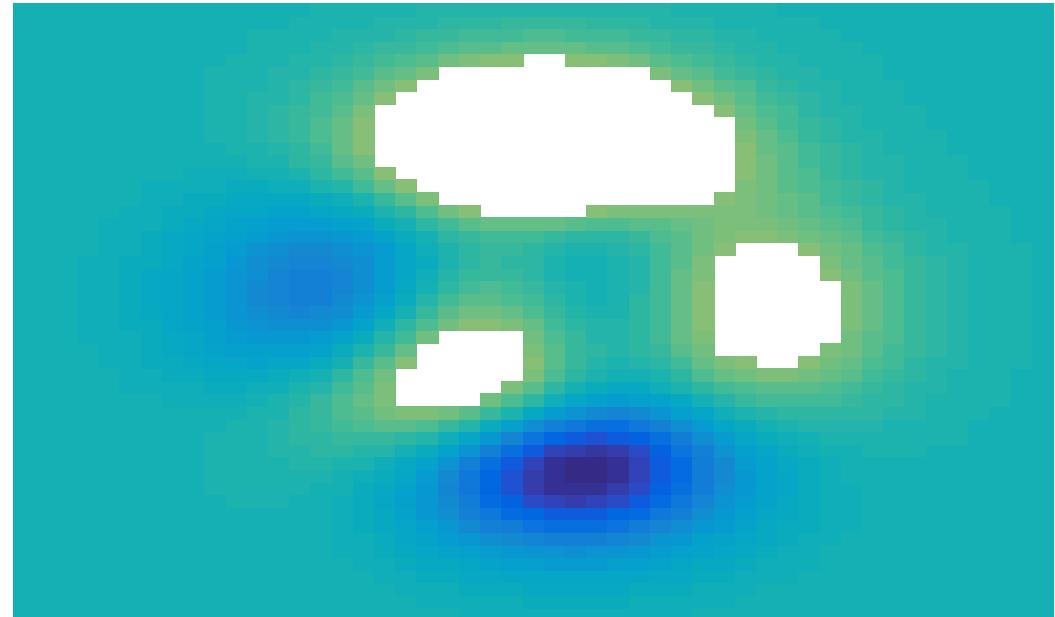
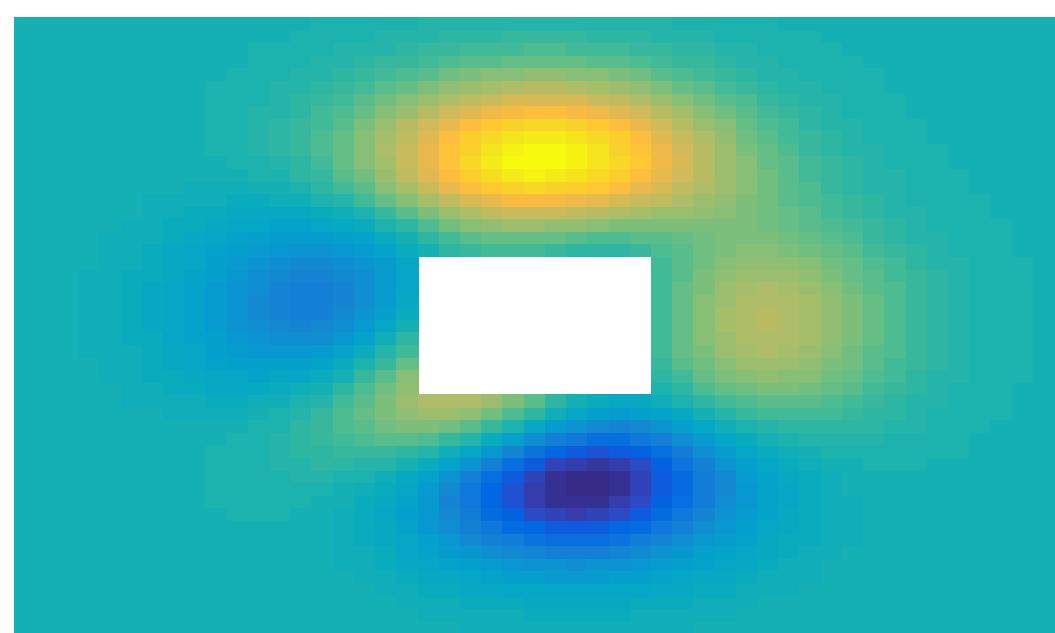
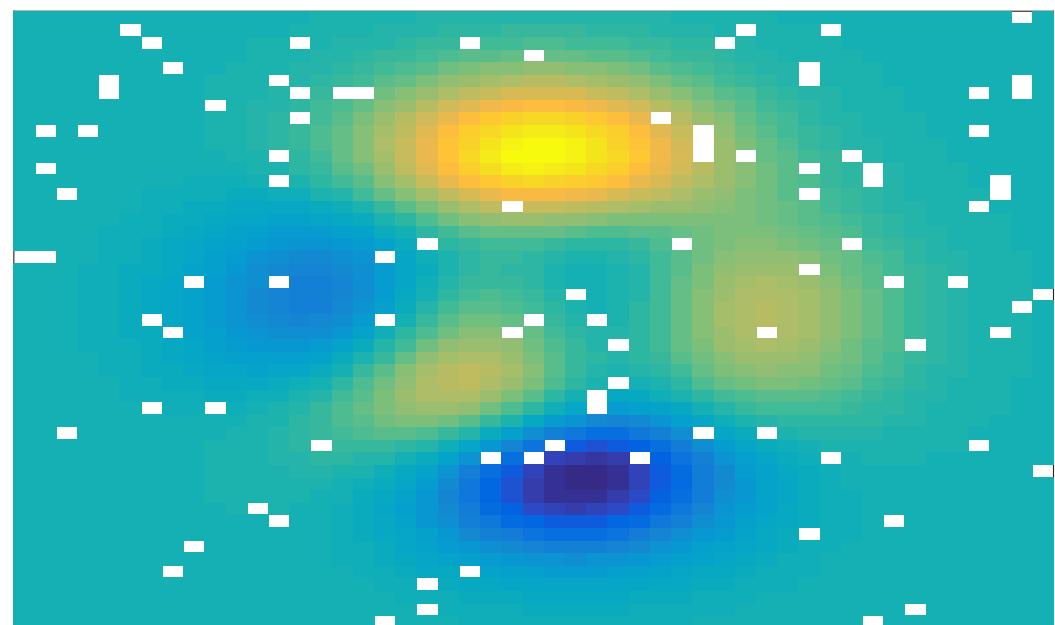


A real-world case!



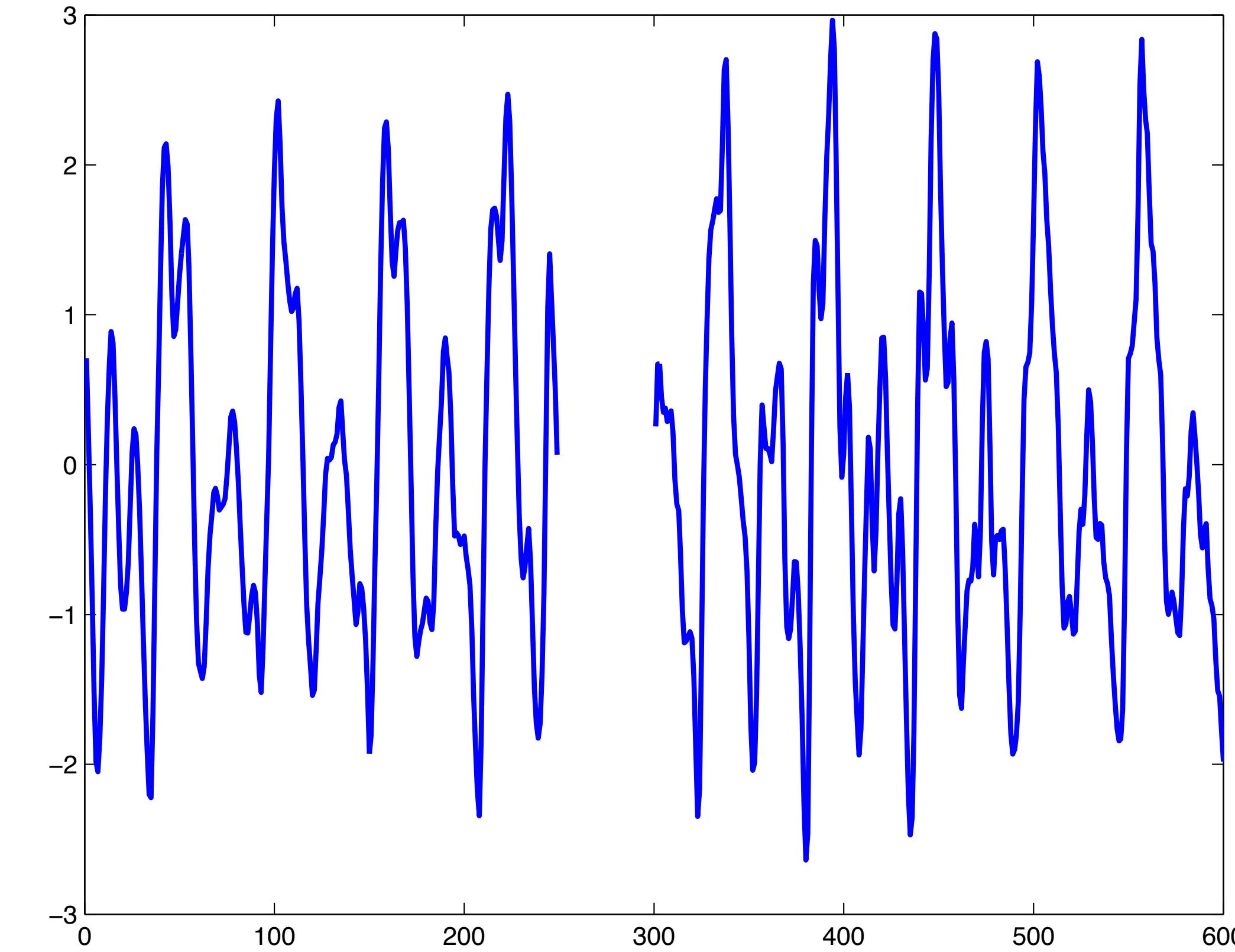
General classification of cases

- Missing completely at random (MCAR)
 - “Missingness” is really random
- Missing at random (MAR)
 - “Missingness” depends on some variable
 - But not on the missing data values
- Not missing at random (NMAR)
 - None of the above



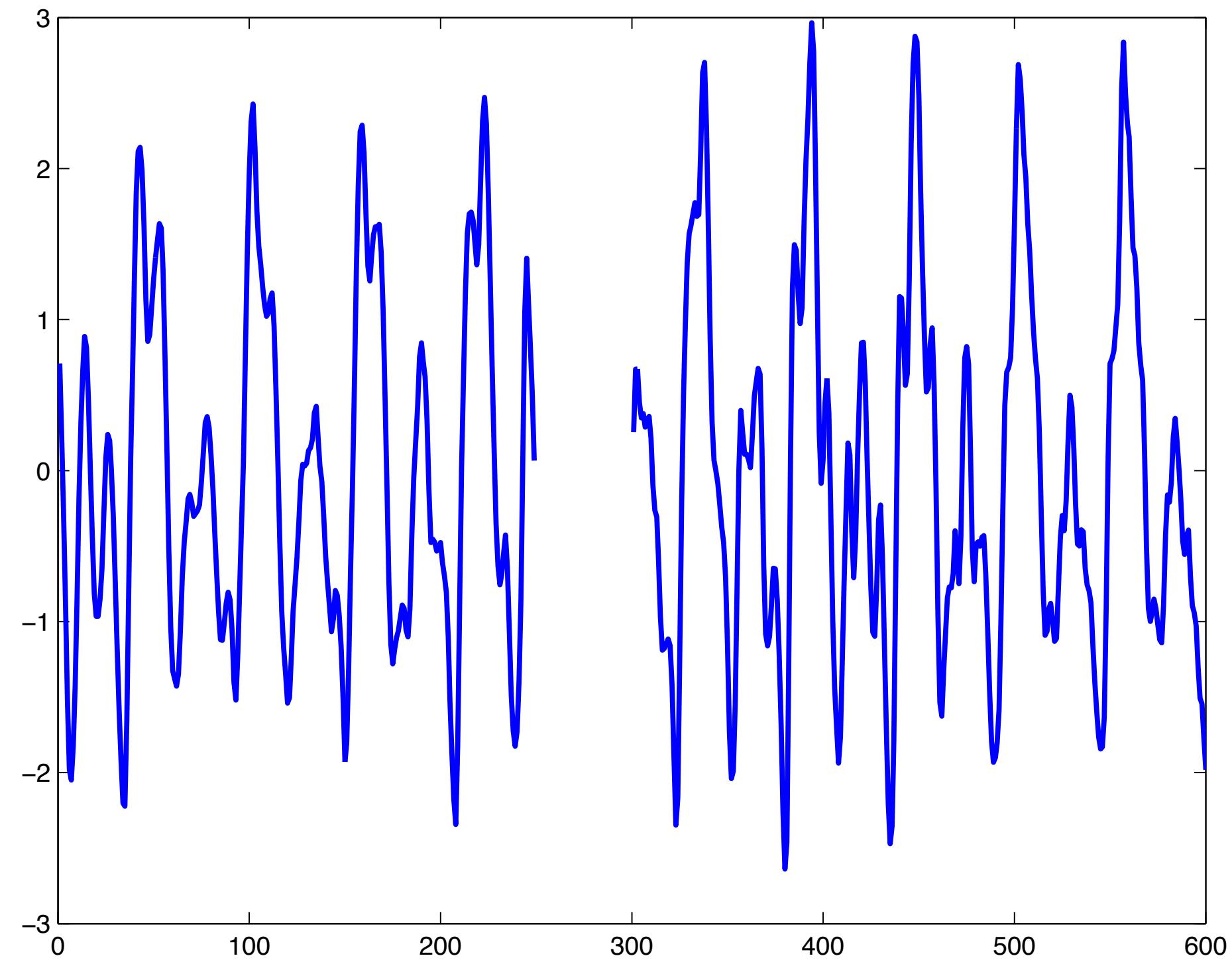
Starting with 1D

- Assume a small gap in 1D data
 - Can we find the missing values?



Key observations

- Temporal structure
 - The signal is somewhat periodic
- We could predict the future from the past
- How do we formalize this?



A predictive model

- Predict current sample from the past
 - Using a weighted average of preceding samples

$$x_t = \sum a_i x_{t-i} + e_t$$

- Autoregressive (AR) model
 - We can learn the coefficients a using various methods

Rewriting the model

- Linear algebra notation

$$\mathbf{e} = \mathbf{A} \cdot \mathbf{x}$$

$$\mathbf{A} = \begin{bmatrix} -a_p & \dots & -a_1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & -a_p & \dots & -a_1 & 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \dots & 0 & 0 & -a_p & \dots & -a_1 & 1 & 0 & 0 \\ 0 & \dots & 0 & 0 & -a_p & \dots & -a_1 & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 & -a_p & \dots & -a_1 & 1 \end{bmatrix}$$

Filling gaps

- Model the input sequence as:

$$\mathbf{e} = \mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot (\mathbf{U} \cdot \mathbf{x}_u + \mathbf{K} \cdot \mathbf{x}_k) = \mathbf{A}_u \cdot \mathbf{x}_u + \mathbf{A}_k \cdot \mathbf{x}_k$$

- \mathbf{x}_u and \mathbf{x}_k are the unknown and known samples
- \mathbf{U} and \mathbf{K} are repositioning matrices, e.g.:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}_{\text{Unknown}} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_4 \end{bmatrix}_{\text{Known}}$$

Isolating the unknowns

- Final form isolates unknowns as a variable
 - These are values that should conform to the model
 - Assuming they aren't missing at random!
- We thus want to find \mathbf{x}_u so that we minimize e
 - i.e. fill in the blanks with least unpredictable samples

$$\mathbf{A}_u \cdot \mathbf{x}_u + \mathbf{A}_k \cdot \mathbf{x}_k = \mathbf{e}$$

Solving for the gaps

- Set the derivative to zero

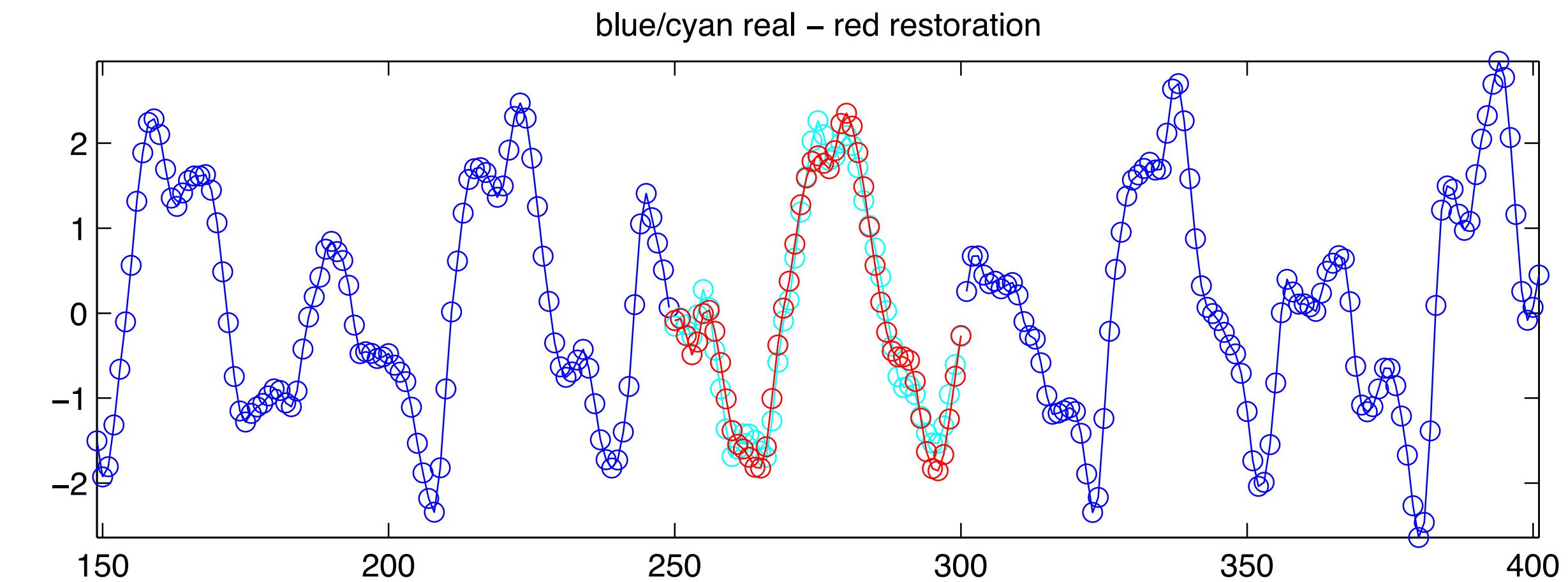
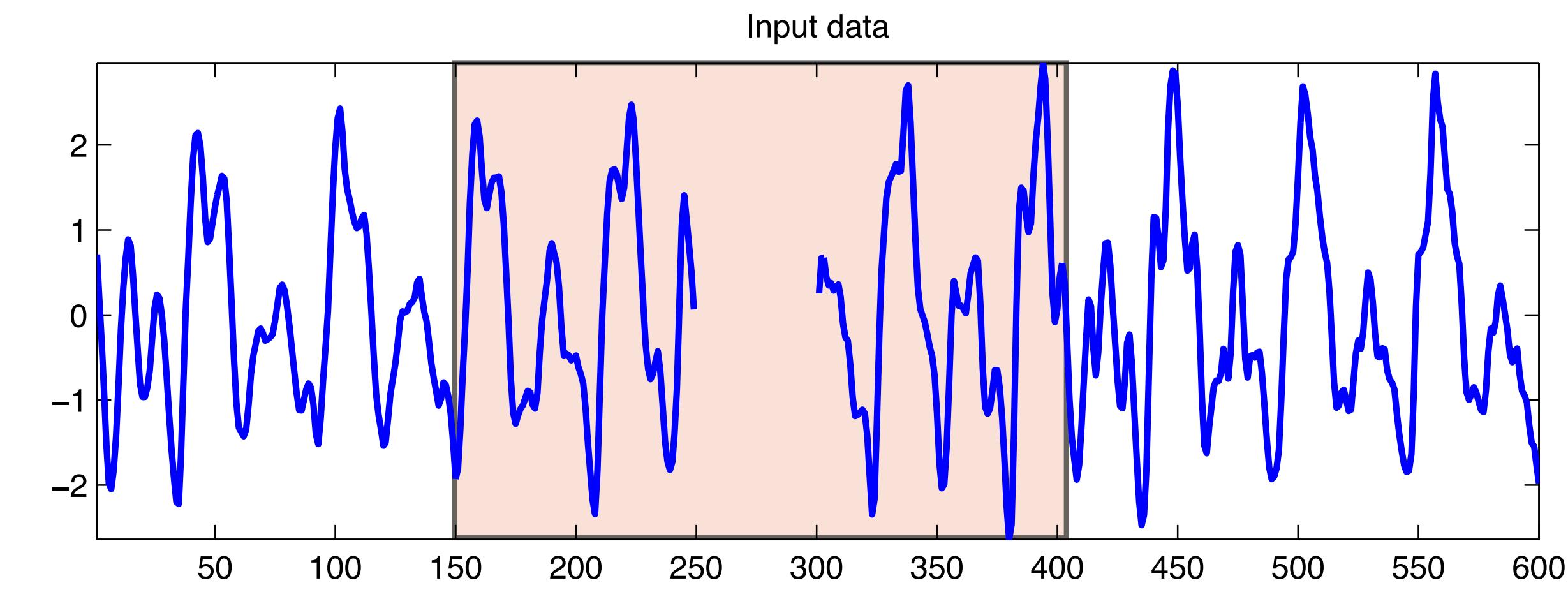
$$\frac{\partial \mathbf{e}^\top \cdot \mathbf{e}}{\partial \mathbf{x}_u} = 2\mathbf{e}^\top \frac{\partial \mathbf{e}}{\partial \mathbf{x}_u} = 2\left(\mathbf{A} \cdot \mathbf{x}_u + \mathbf{A}_k \cdot \mathbf{x}_k\right)^\top \cdot \mathbf{A}_u = 0$$

- Solve for unknown samples

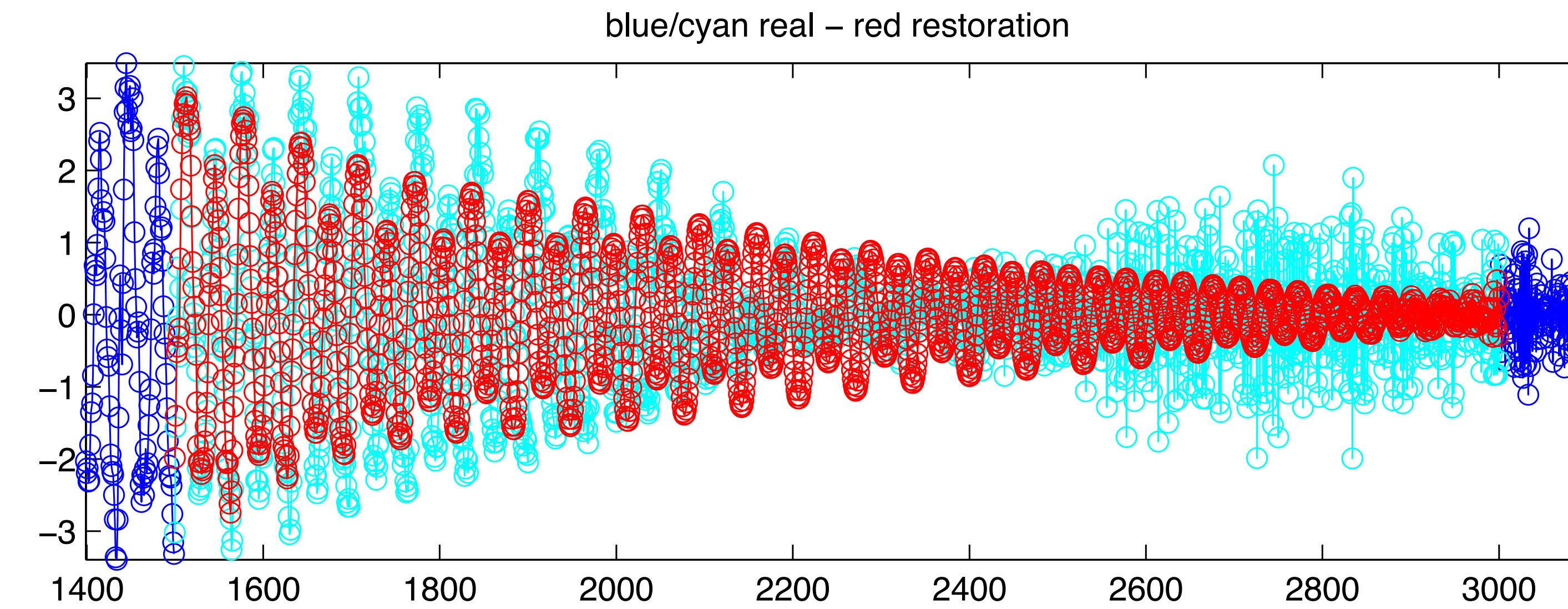
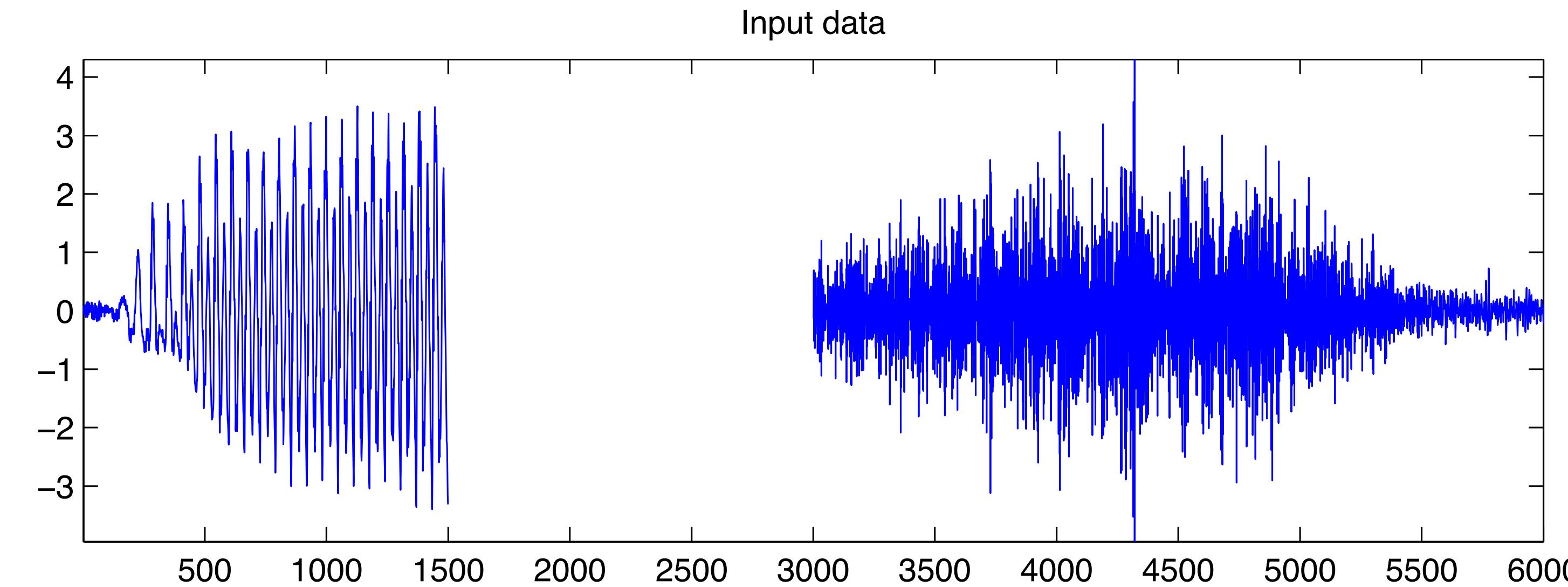
$$\hat{\mathbf{x}}_u = -\left(\mathbf{A}_u^\top \cdot \mathbf{A}_u\right)^{-1} \cdot \mathbf{A}_u^\top \cdot \mathbf{A}_k \cdot \mathbf{x}_k = -\mathbf{A}_u^+ \cdot \mathbf{A}_k \cdot \mathbf{x}_k$$

Results

- Works pretty well!



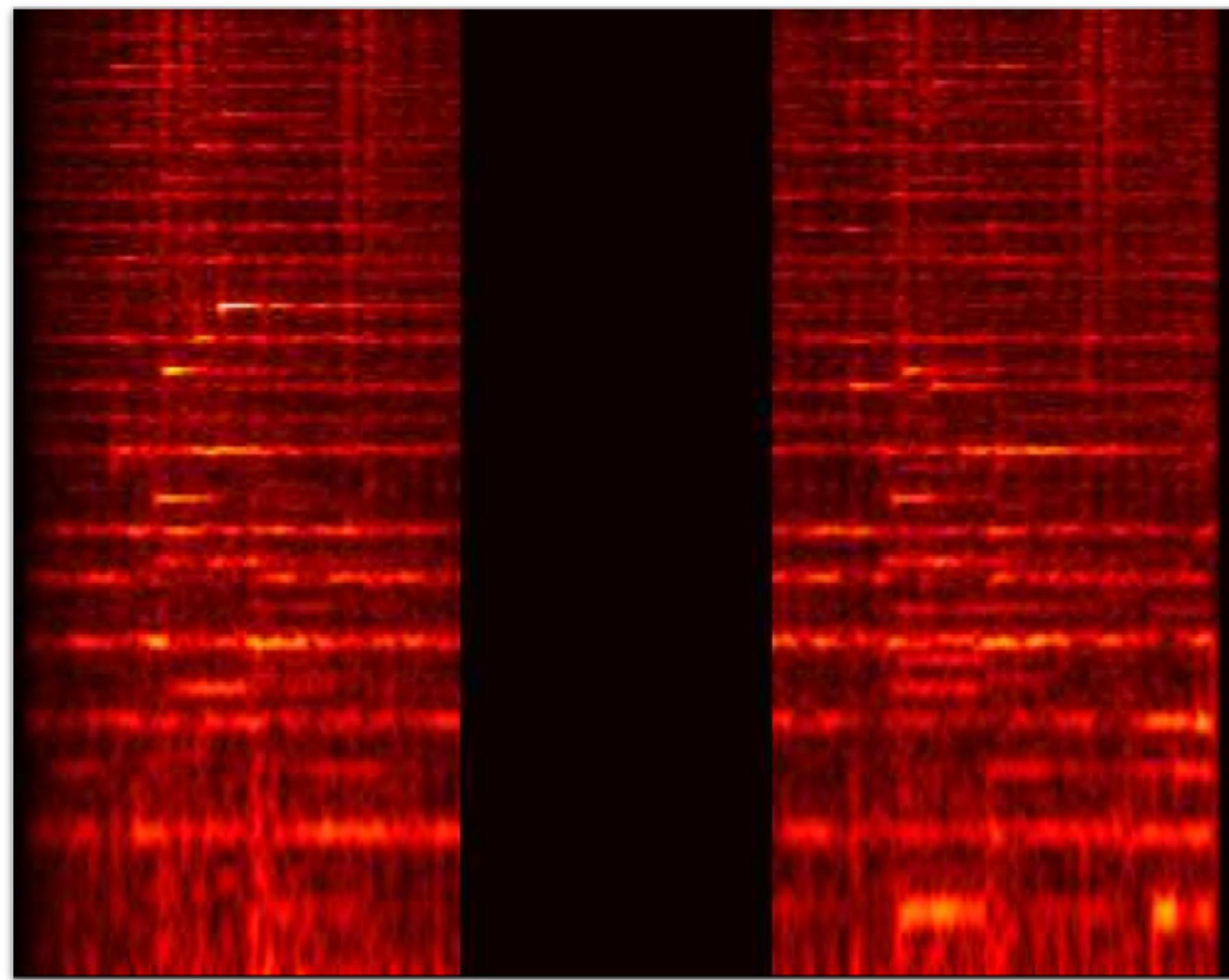
But not for large windows!



Looking for a new idea

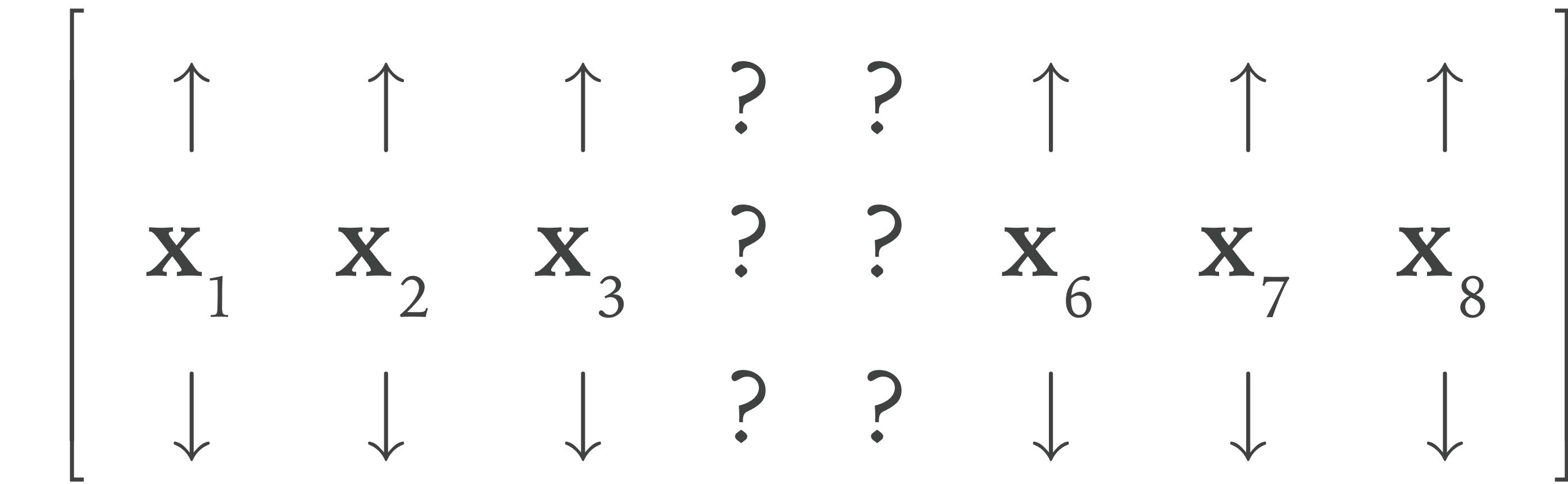


- We should move away from the sample level, and look at a coarser scale
 - e.g. a time-frequency view



A simpler idea

- Find sections most similar to the gap edges
 - Replace missing sections with their neighbors

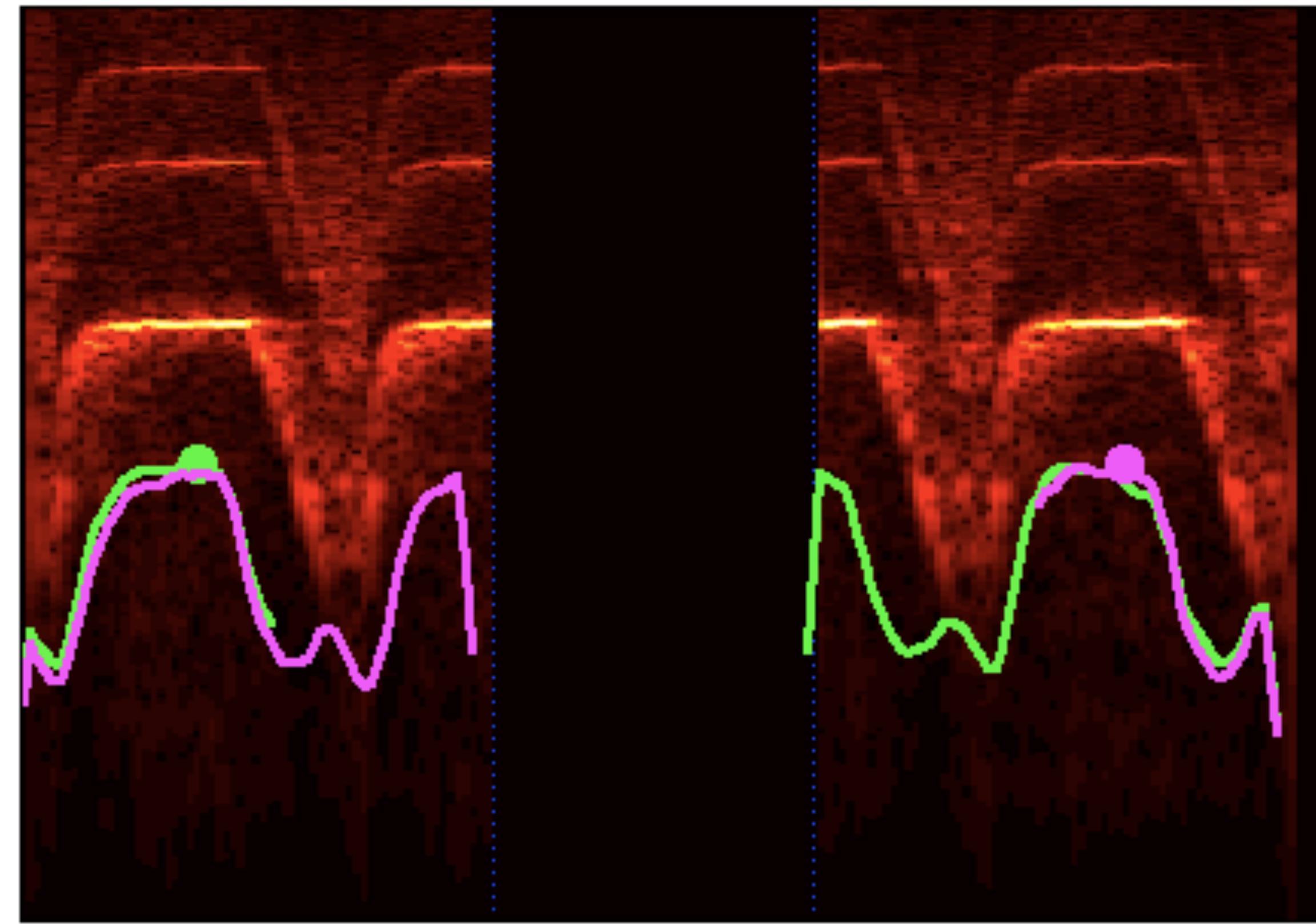


- How do we find the close match?

In action

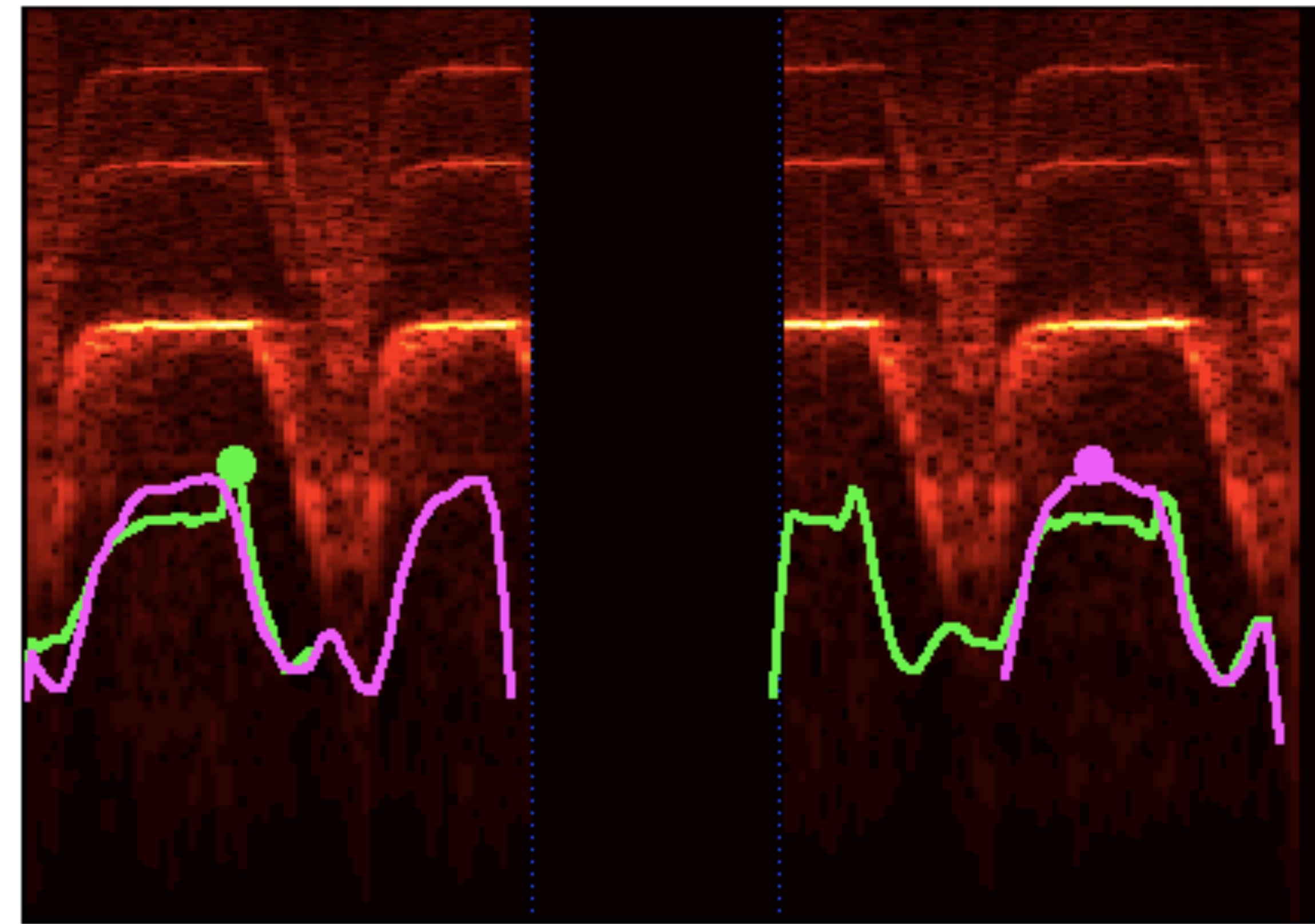


- First iteration



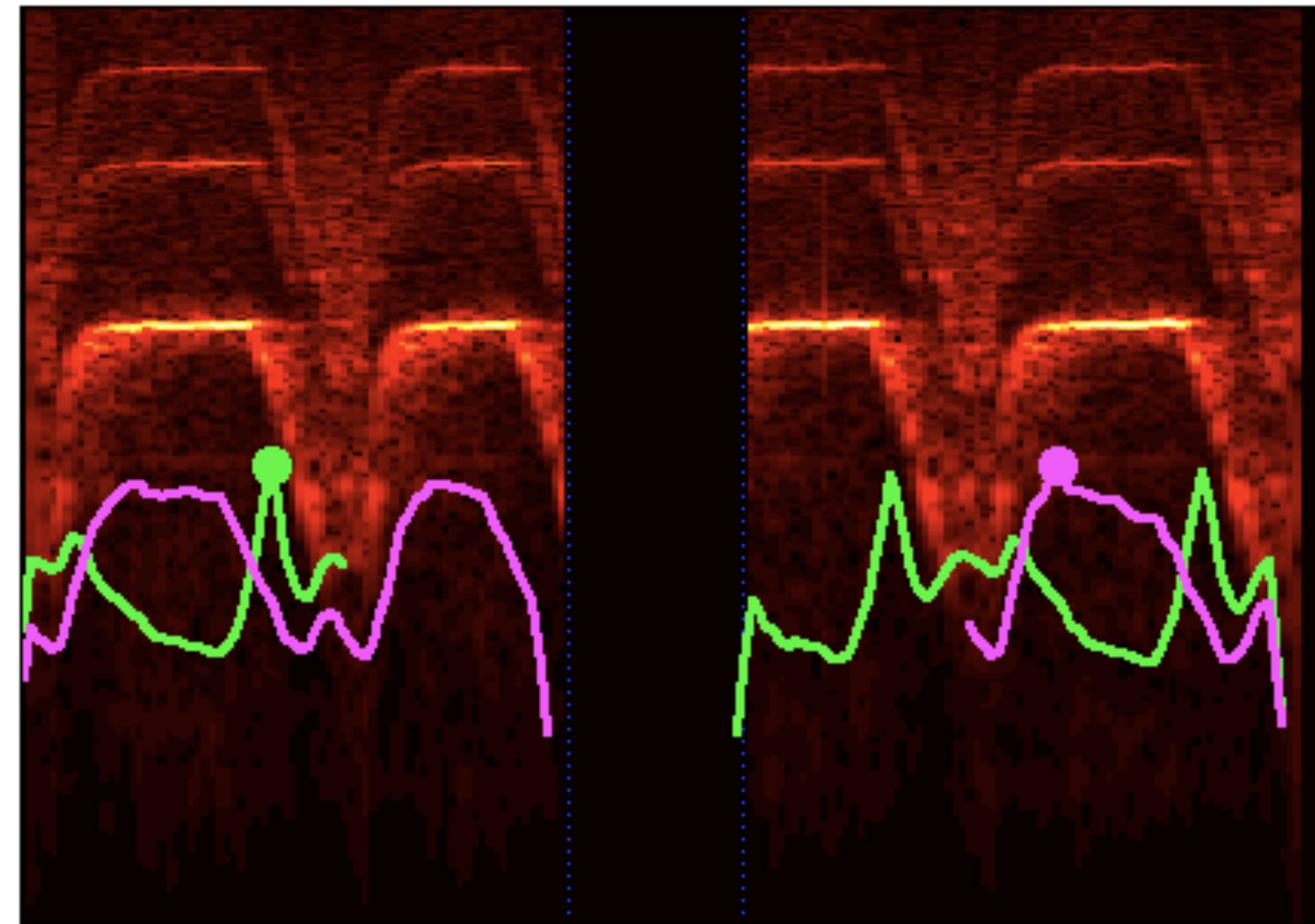
In action

- Second iteration



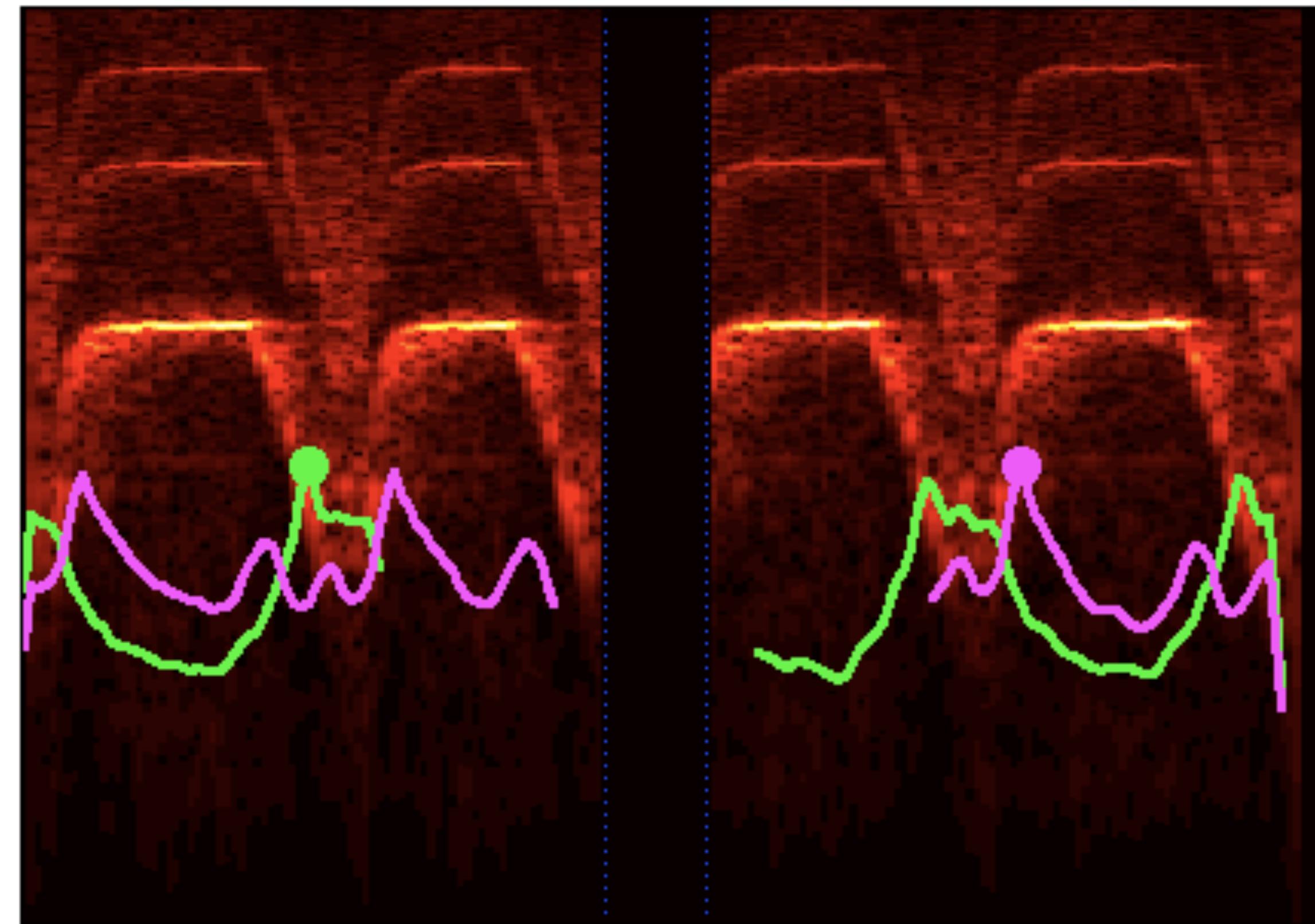
In action

- Third iteration



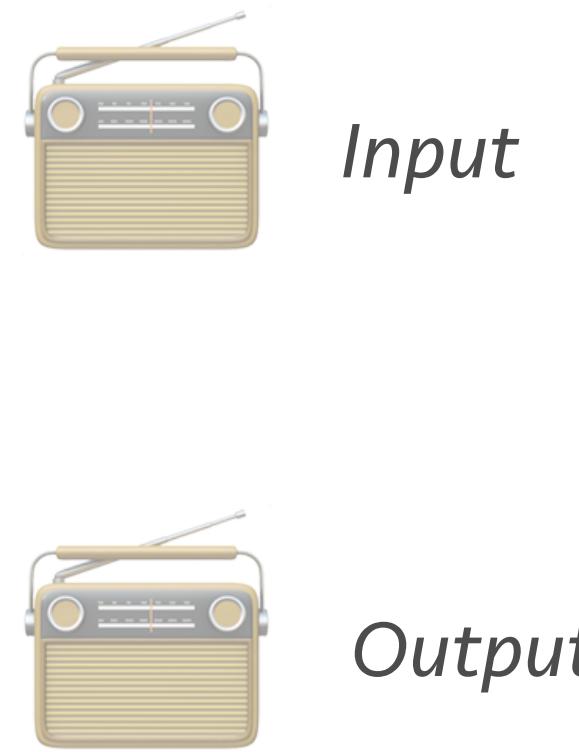
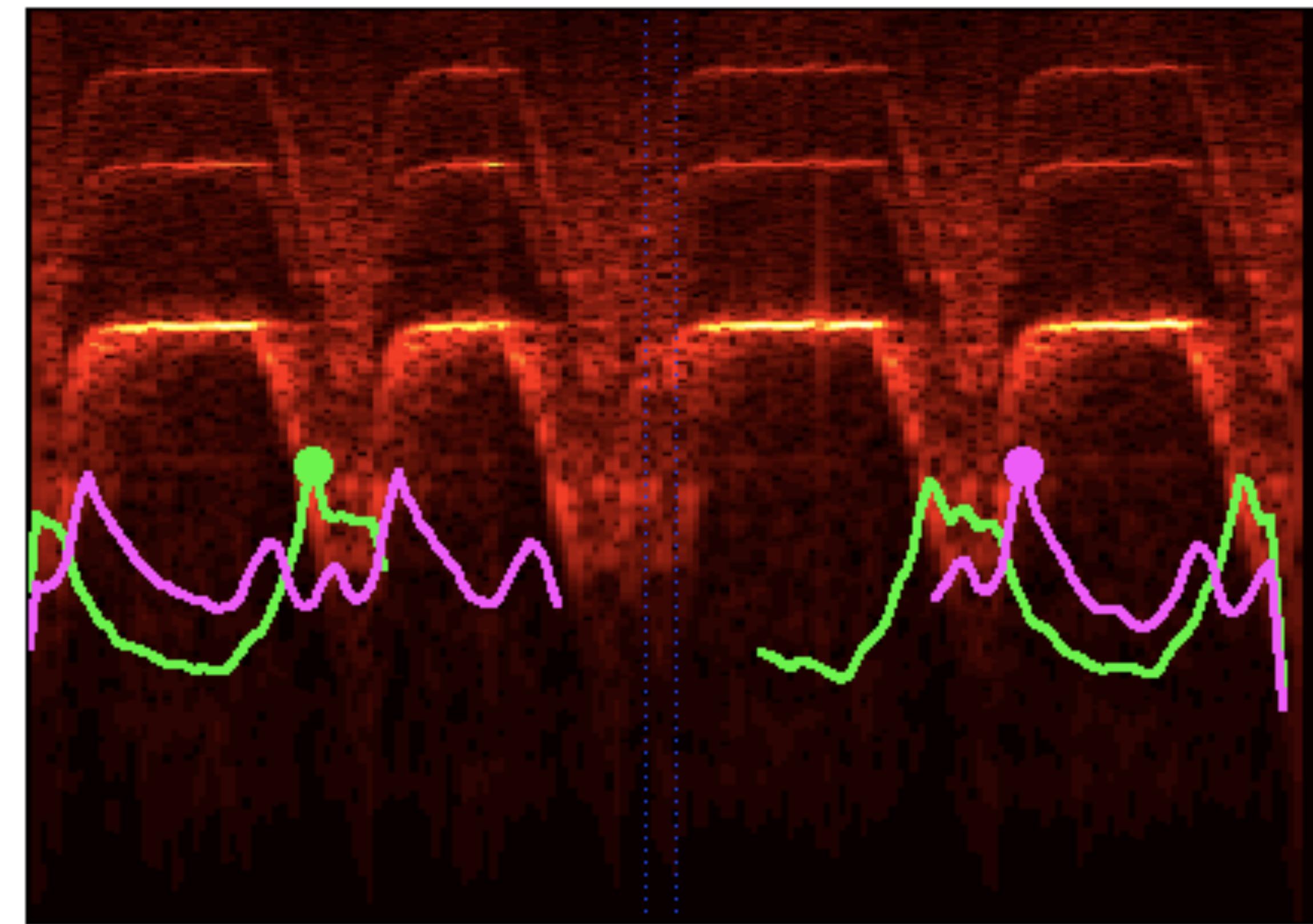
In action

- Fourth iteration

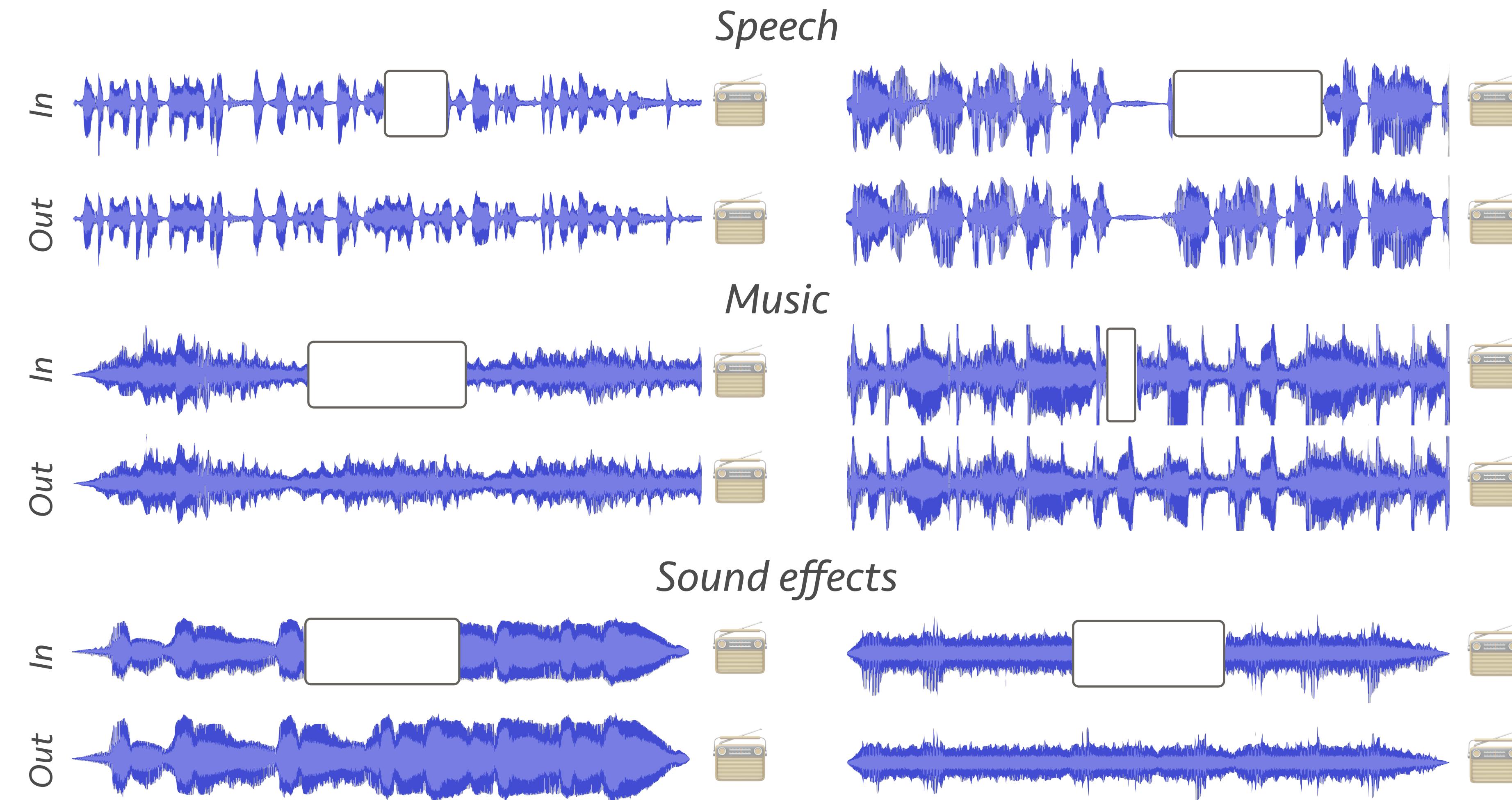


In action

- Final iteration

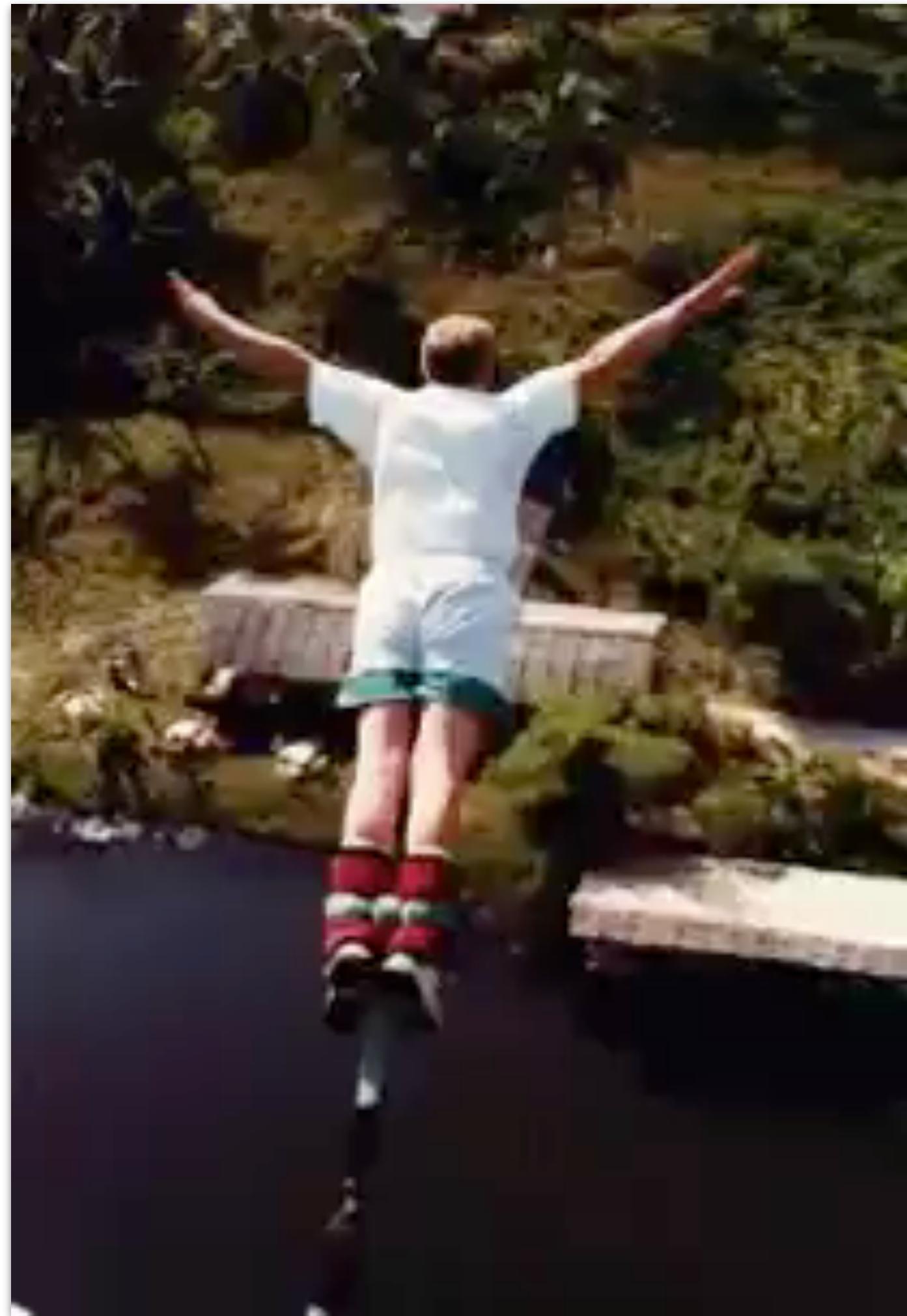


Some examples



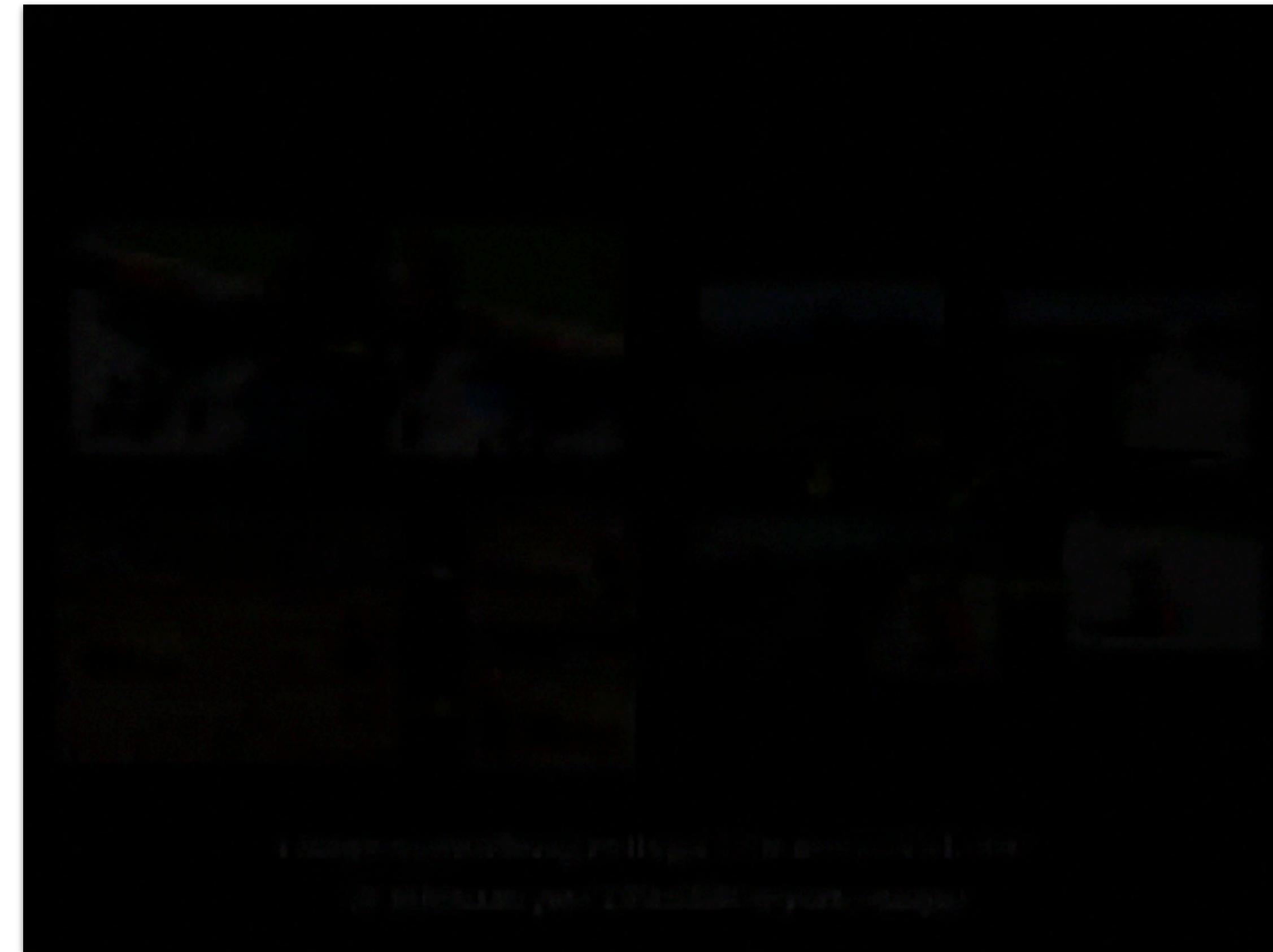
What about images?

- Basic idea is the same
 - Find a neighbor and blend it
- A bit more complicated
 - We need to blend carefully, we can't just add
 - Poisson blending
- Also more computationally intensive (2d search!)



A real-world example

- Inpainting, recomposing, warping the truth!



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing
Connelly Barnes, Eli Shechtman, Adam Finkelstein, Dan B Goldman

The statistical viewpoint

- Nearest-neighbor search was local
 - We ignore global data structure
- Missing data should conform to a statistical model of the input
 - i.e. they shouldn't be outliers

SVD-based imputation

- Simple global approach

- Replace missing data with some values

$$x_u = E\{x\} \quad \text{or} \quad x_u \sim \mathcal{N}(\mu, \sigma^2) \quad \text{or} \dots$$

- Perform an SVD approximation of the data

$$\mathbf{X} \approx \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^\top$$

- Replace missing data with SVD approximation

$$\mathbf{x}_u = \left(\mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^\top \right)_u$$

- Repeat!

Why?

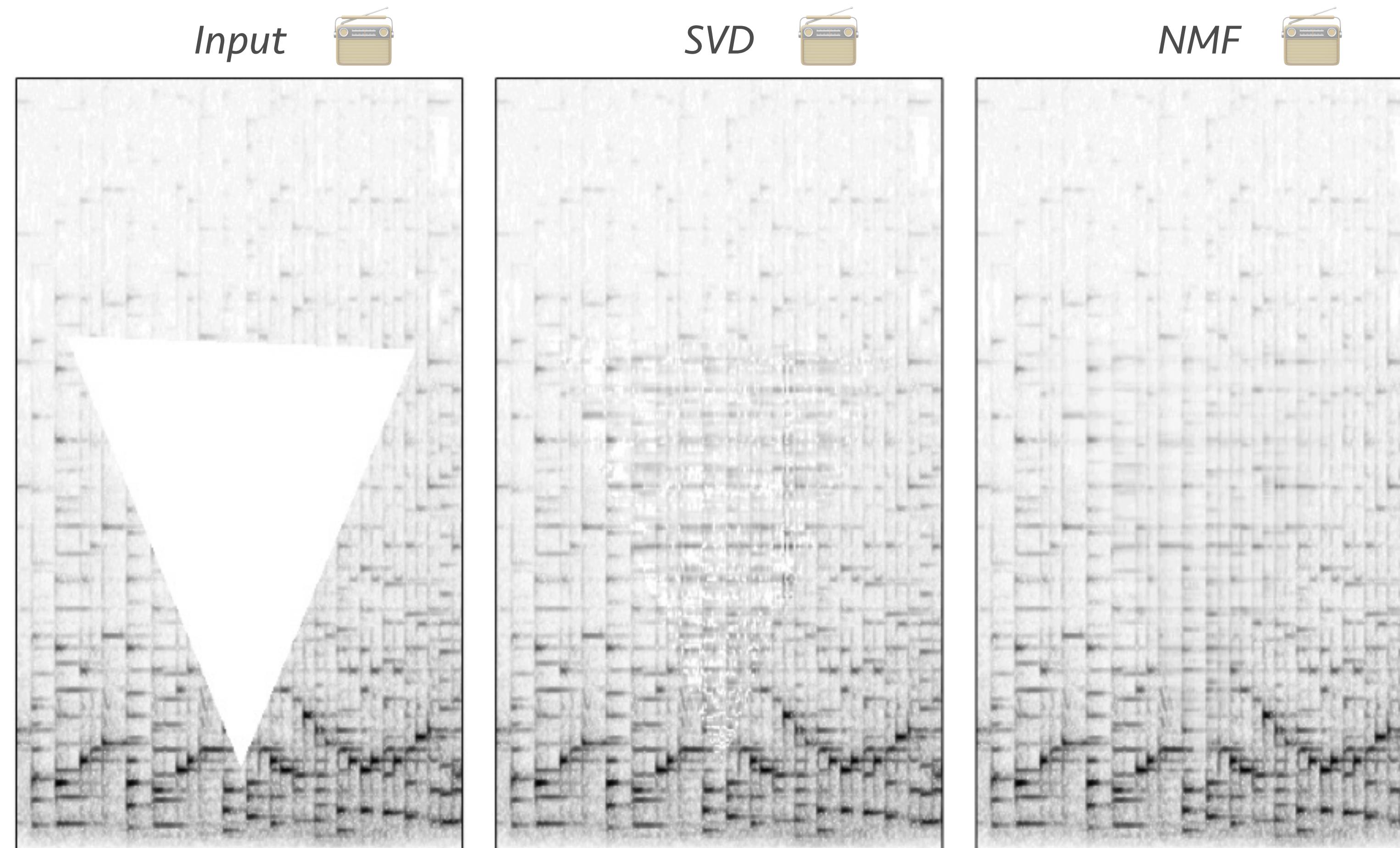
- Known data will dominate statistics
 - Assuming they are enough!
- Each approximation will try to conform to the global statistics of the input
 - i.e. made up values will not be random
- With each new iteration, that conformance to global statistics will increase

Variations

- We don't have to use an SVD
 - e.g. our data might be non-negative
 - So use NMF, or probabilistic models, or whatever makes sense
- Any global statistical model would work
 - Just make sure that it fits the data well

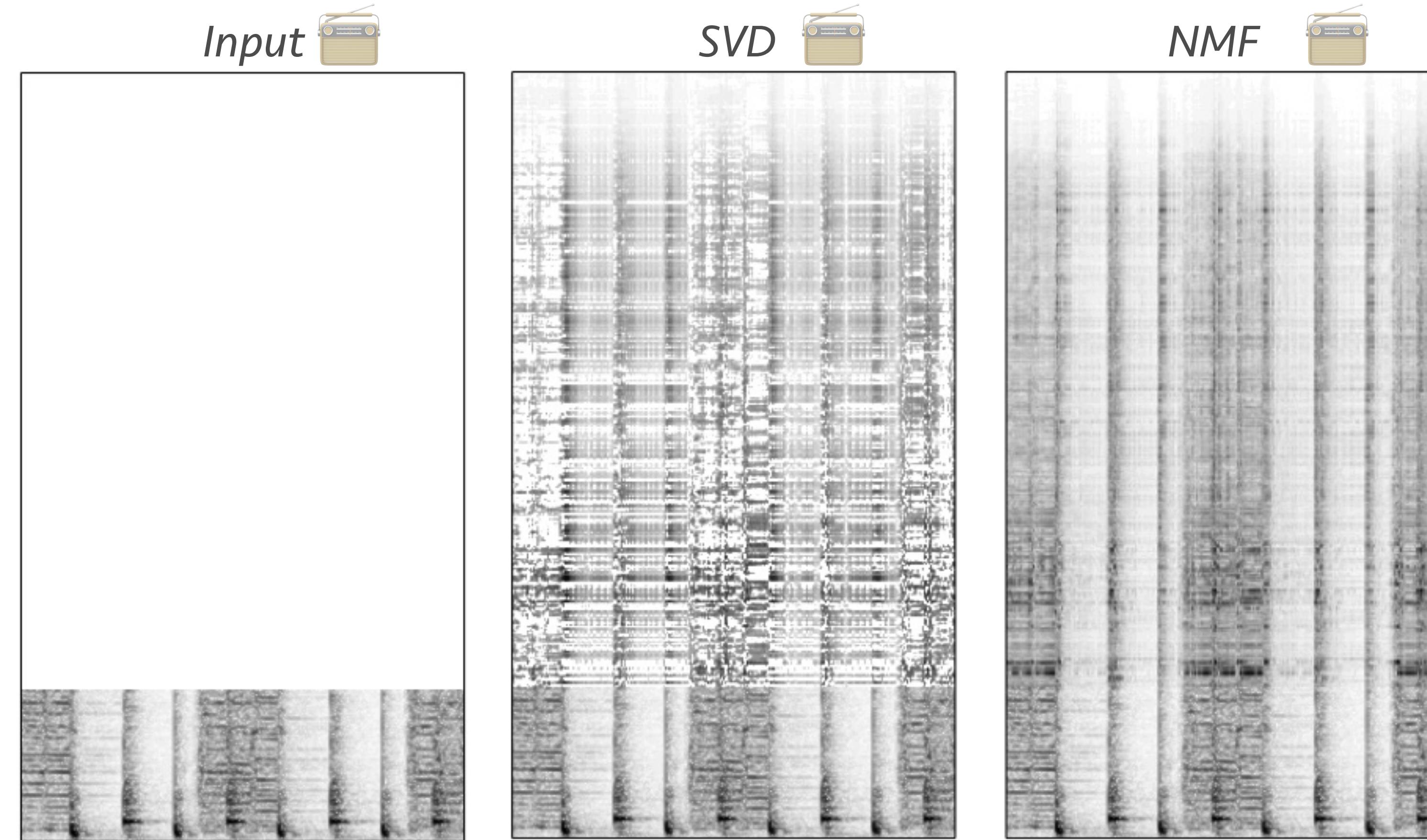
Example

- Learn from input and fill-in missing values



Learning from outside

- Bandwidth expansion
 - Learn model from other recordings



Tracking

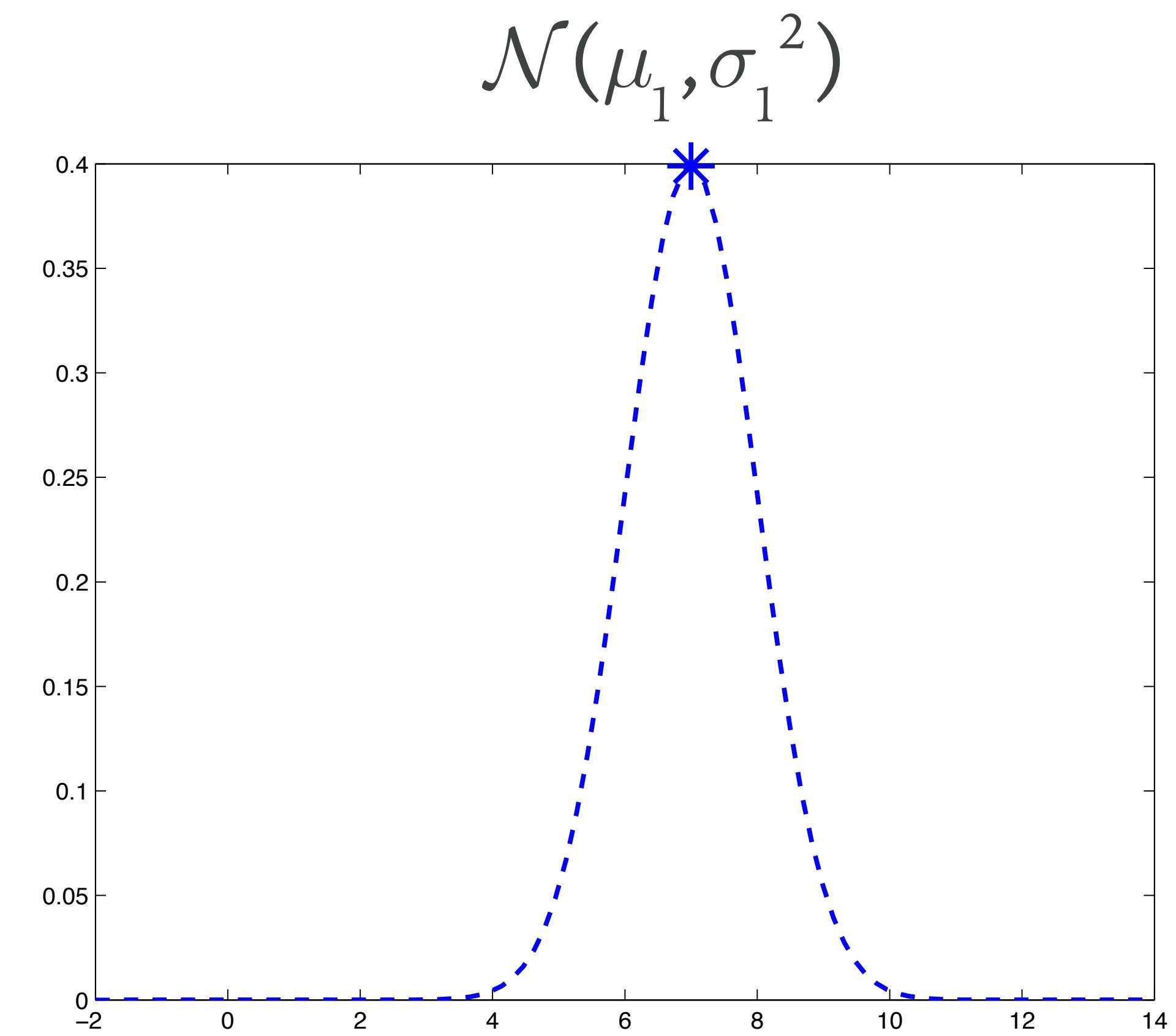
- Tracking things that evolve in time
 - missiles, faces, finances, etc ...
- A time-series model again

A simple problem

- I'm looking at a star which is about "there"
 - So does my drunken friend
- How do we consolidate our observations?

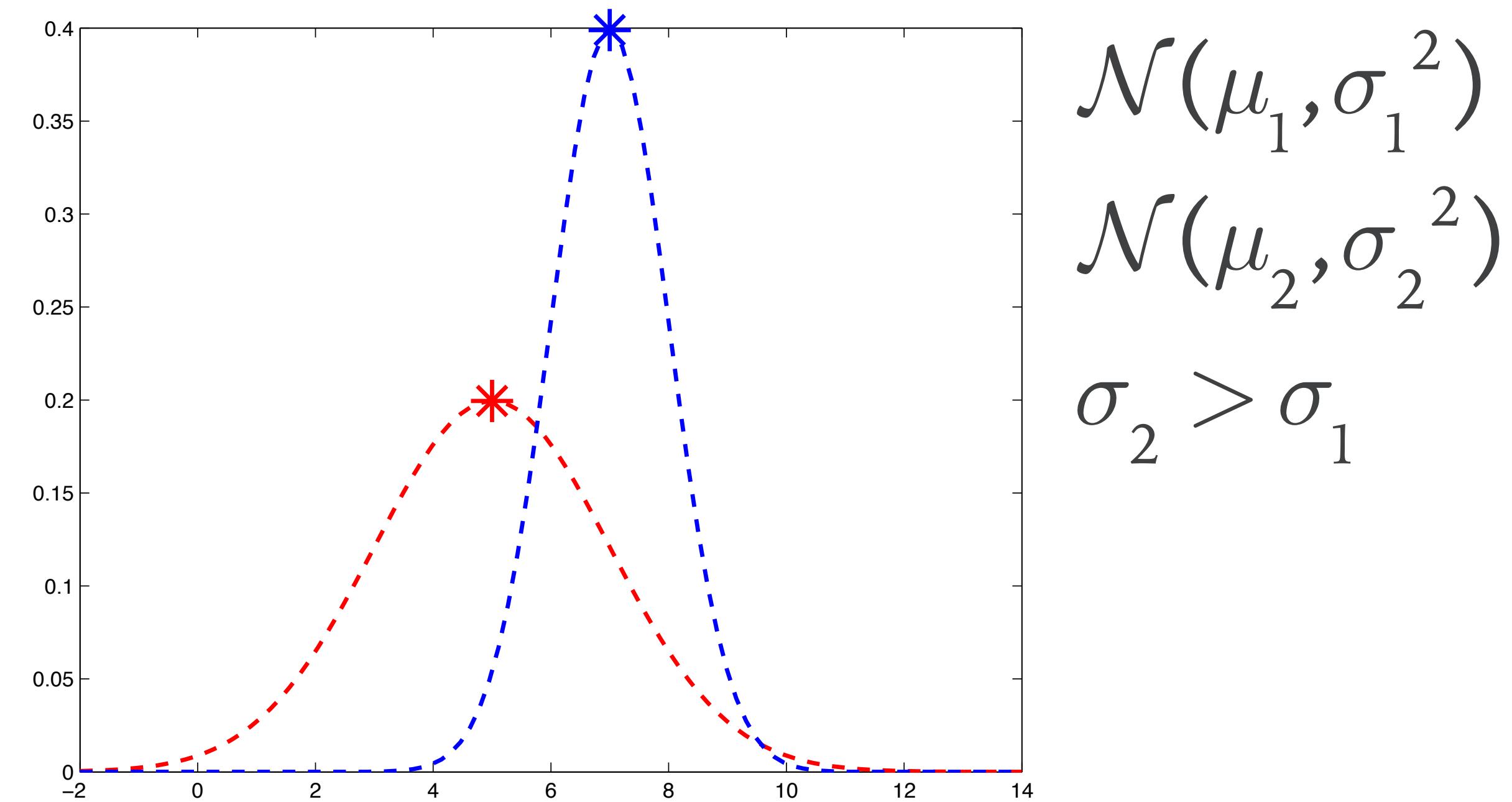
An uncertain estimate

- My observation is a Gaussian that helps me account for some uncertainty
 - Mean is what I think is right
 - Variance is an indication of how sure I am



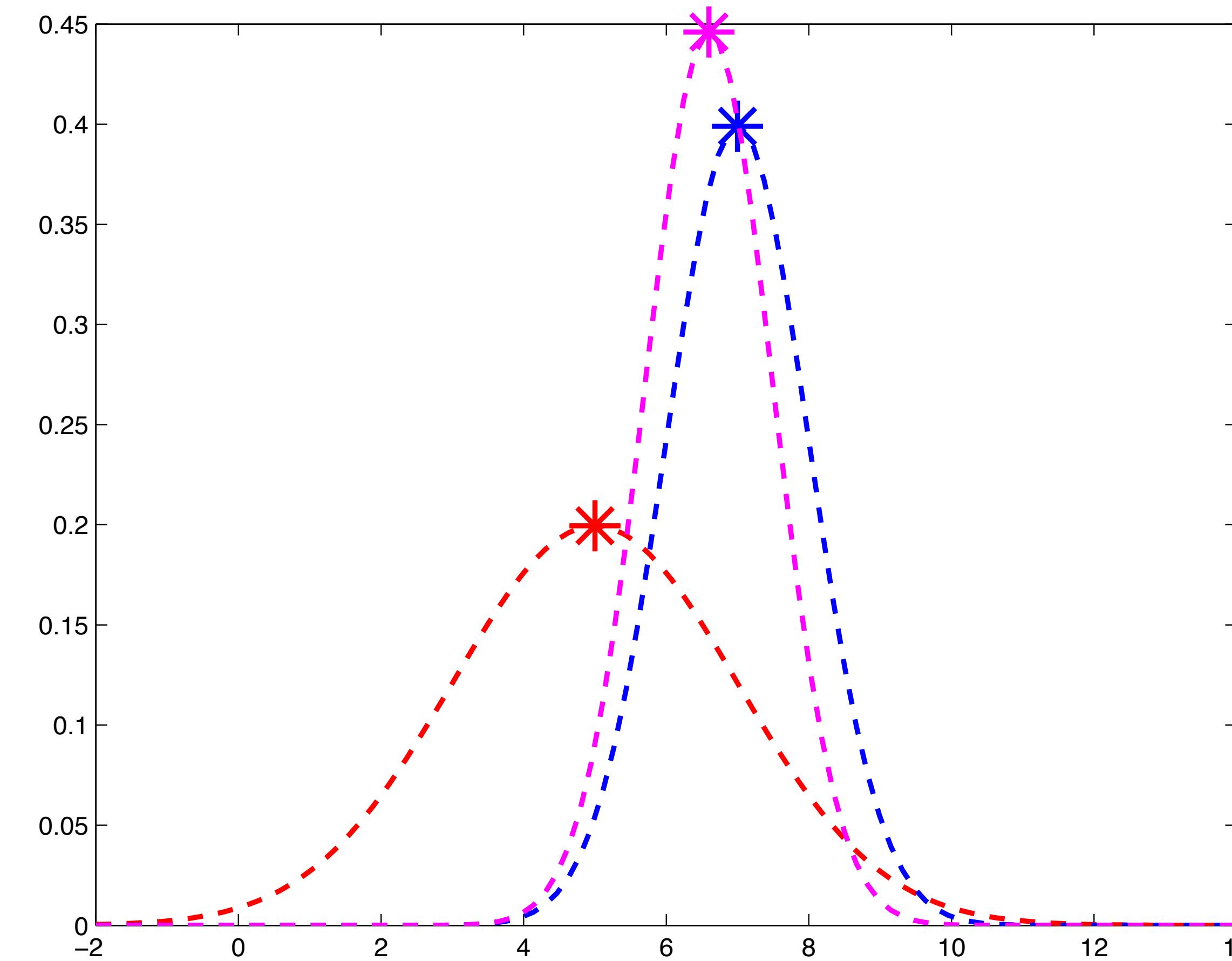
More uncertainty

- My drunken friend's estimate is unreliable
 - Therefore his variance is higher



Consensus estimate

- The consensus estimate is proportional to a Gaussian



$$\mathcal{N}(m, s) \propto \mathcal{N}(\mu_1, \sigma_1) \mathcal{N}(\mu_2, \sigma_2)$$

$$K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$$

$$m = \mu_1 + K(\mu_2 - \mu_1)$$

$$s = (1 - K)\sigma_1^2$$

Serializing this process

- Assume a noisy time series

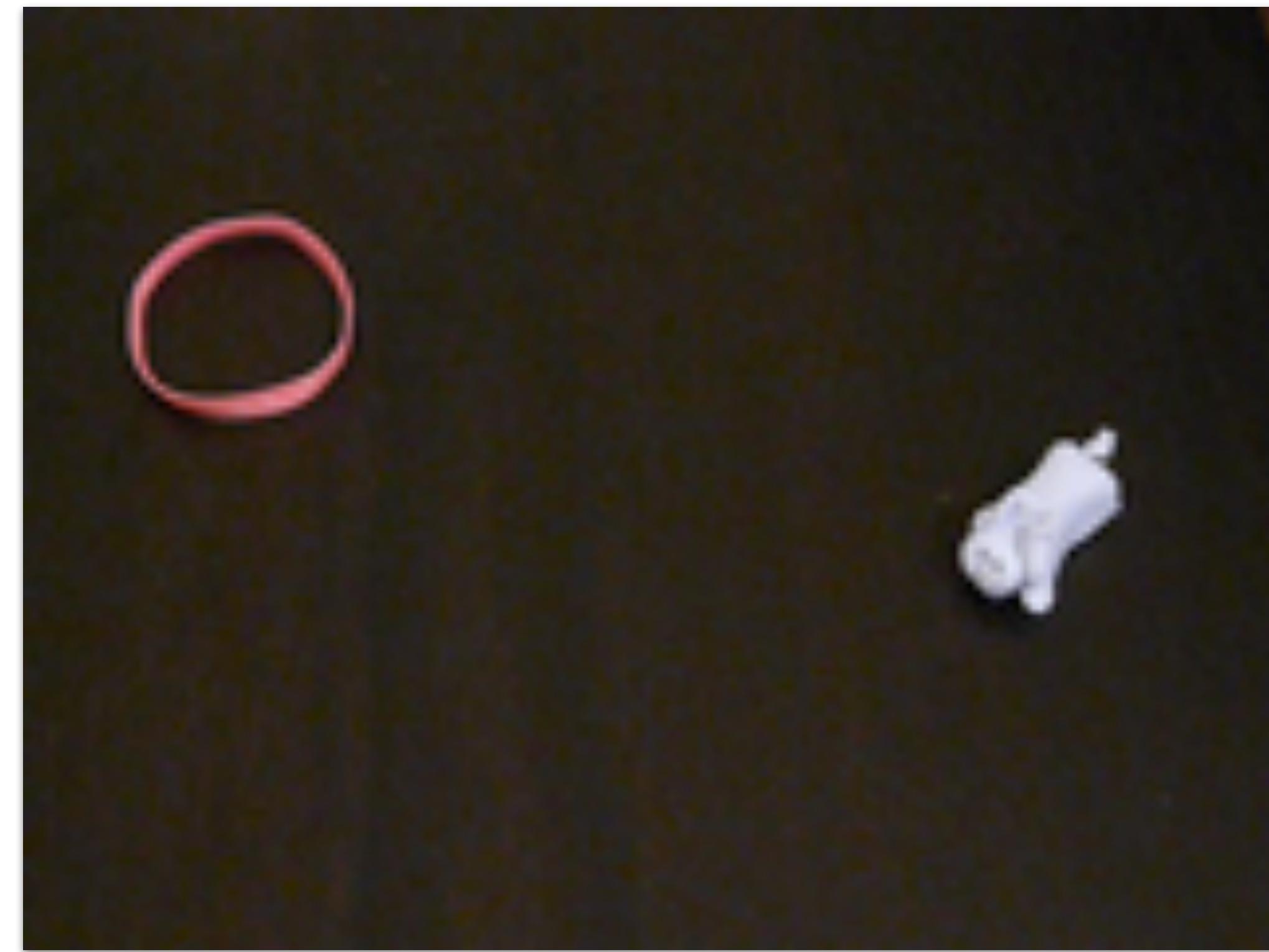
$$z_t = z_{t-1} + e_t$$

$$e_t \sim \mathcal{N}$$

- Can we use that consensus idea for removing noise in successive values?
 - What is the model here?

Example case

- Track the position of a moving ball



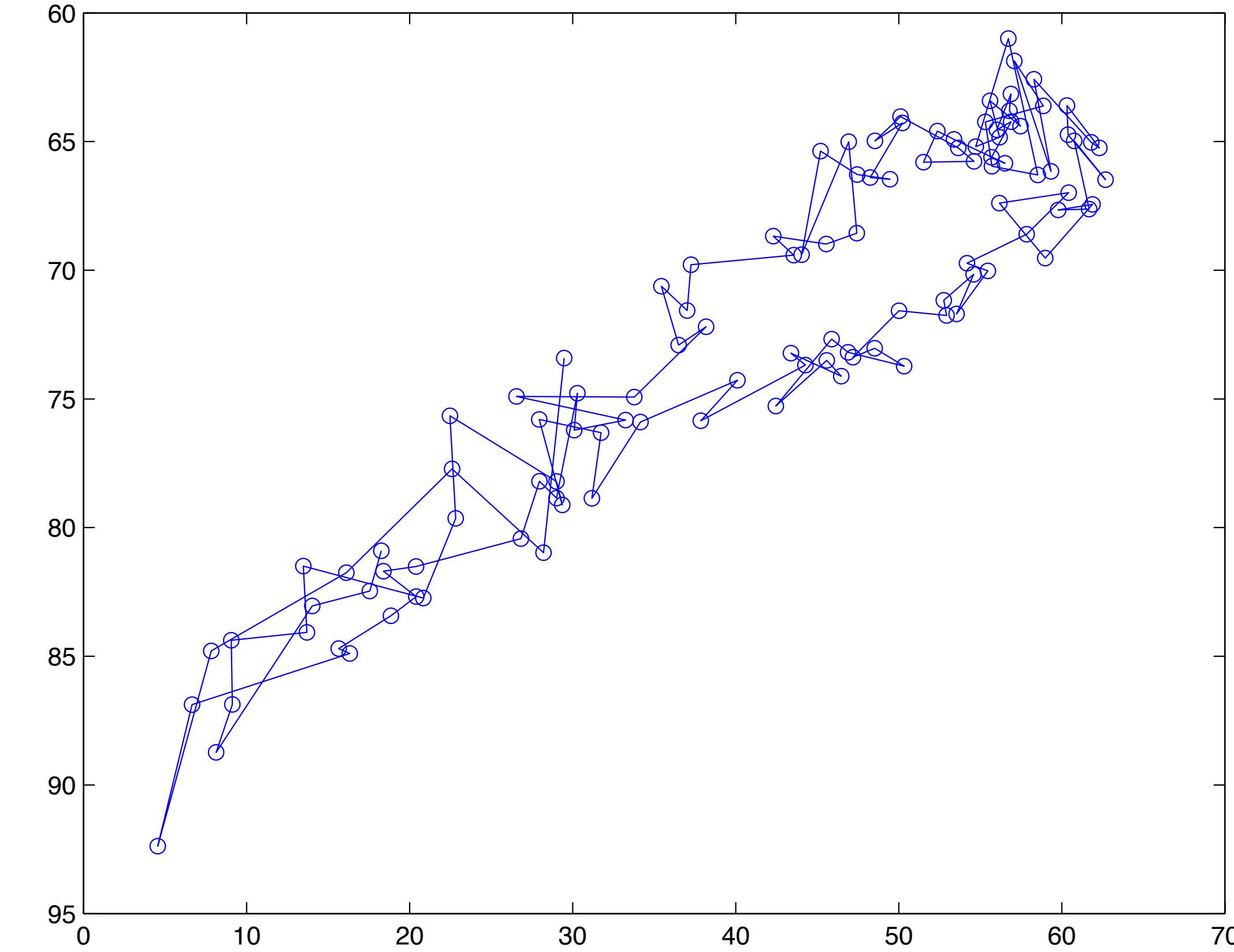
Some simple processing

- Background removal
 - Subtract constant part, find center of remaining pixels



The problem

- The position estimate is very noisy
 - Lets see how we can clean it up



Making a model

- Defining a transition process
 - Looks familiar?

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \mathbf{e}_t \rightarrow \mathbf{z}_t = \mathbf{A} \cdot \mathbf{z}_{t-1} + \mathbf{e}_{t-1}$$

$$\mathbf{A} = \mathbf{I}$$

$$\mathbf{e}_t \sim \mathcal{N}$$

- In short, the next input will be a random value away from the current input

Starting with it

- Initial estimate will be the first input: $\hat{z}_1 = z_1$
- With a given certainty: $P_1 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$
- And two noise models: $Q = \begin{bmatrix} q_x & 0 \\ 0 & q_y \end{bmatrix}$
*Process noise
(how much
the data changes)*
- $R = \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix}$
*Measurement noise
(how reliable my
measurements are)*

Setting up sequential estimates

- Predict future value

$$\bar{z}_t = \hat{z}_{t-1} \quad \longleftarrow \quad \text{We don't expect change other than noise}$$

$$\bar{P} = P_{t-1} + Q \quad \longleftarrow \quad \text{And we accumulate the uncertainty}$$

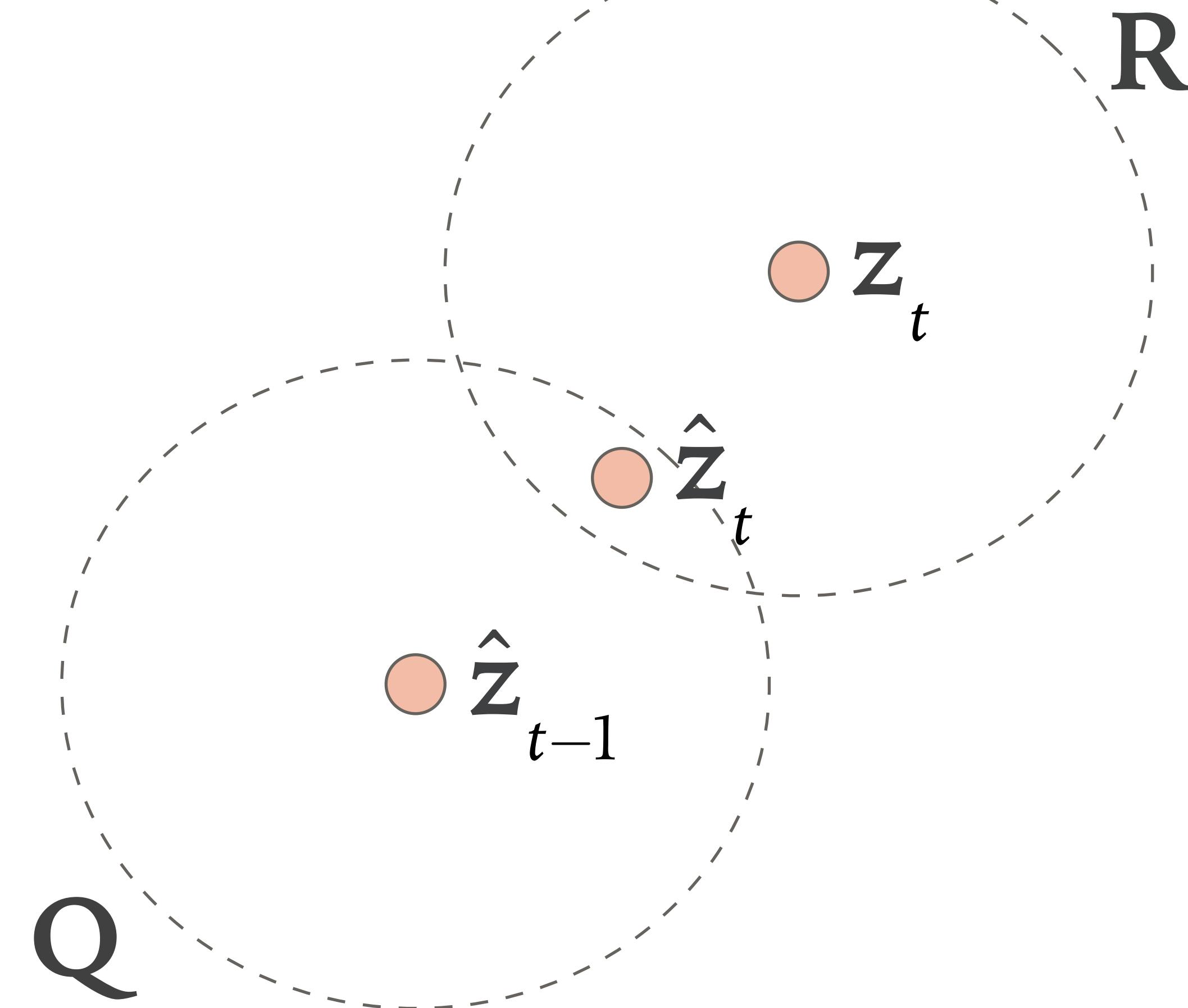
- Correct that estimate given new input

$$K = P^- \cdot (P^- + R)^{-1} \quad \longleftarrow \quad \text{Get uncertainty of a consensus estimate}$$

$$\hat{z}_t = \bar{z}_t + K \cdot (z_t - \bar{z}_t) \quad \longleftarrow \quad \text{And the estimate itself}$$

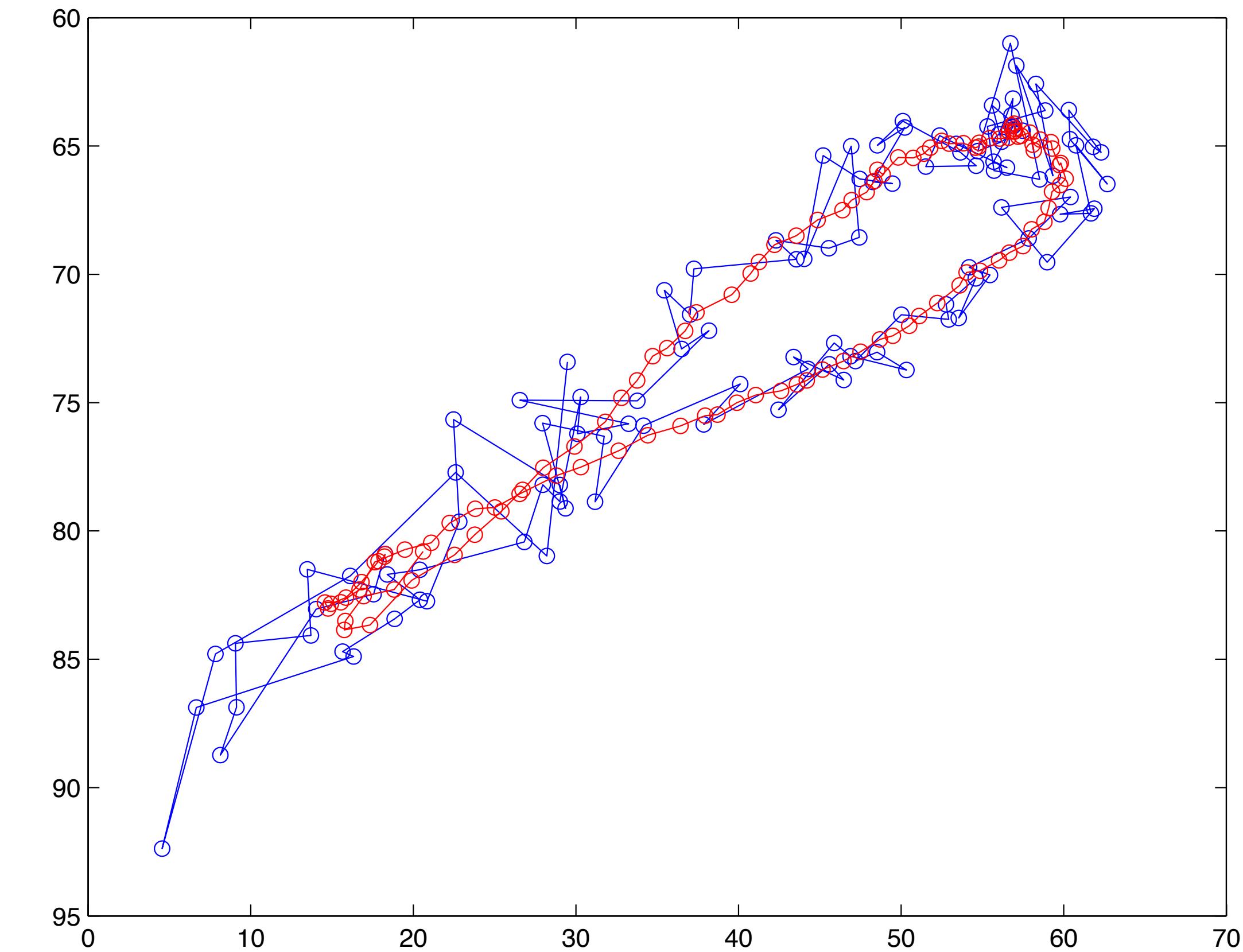
$$P_t = (I - K) \cdot P^- \quad \longleftarrow \quad \text{Get new uncertainty}$$

Single step



Example output

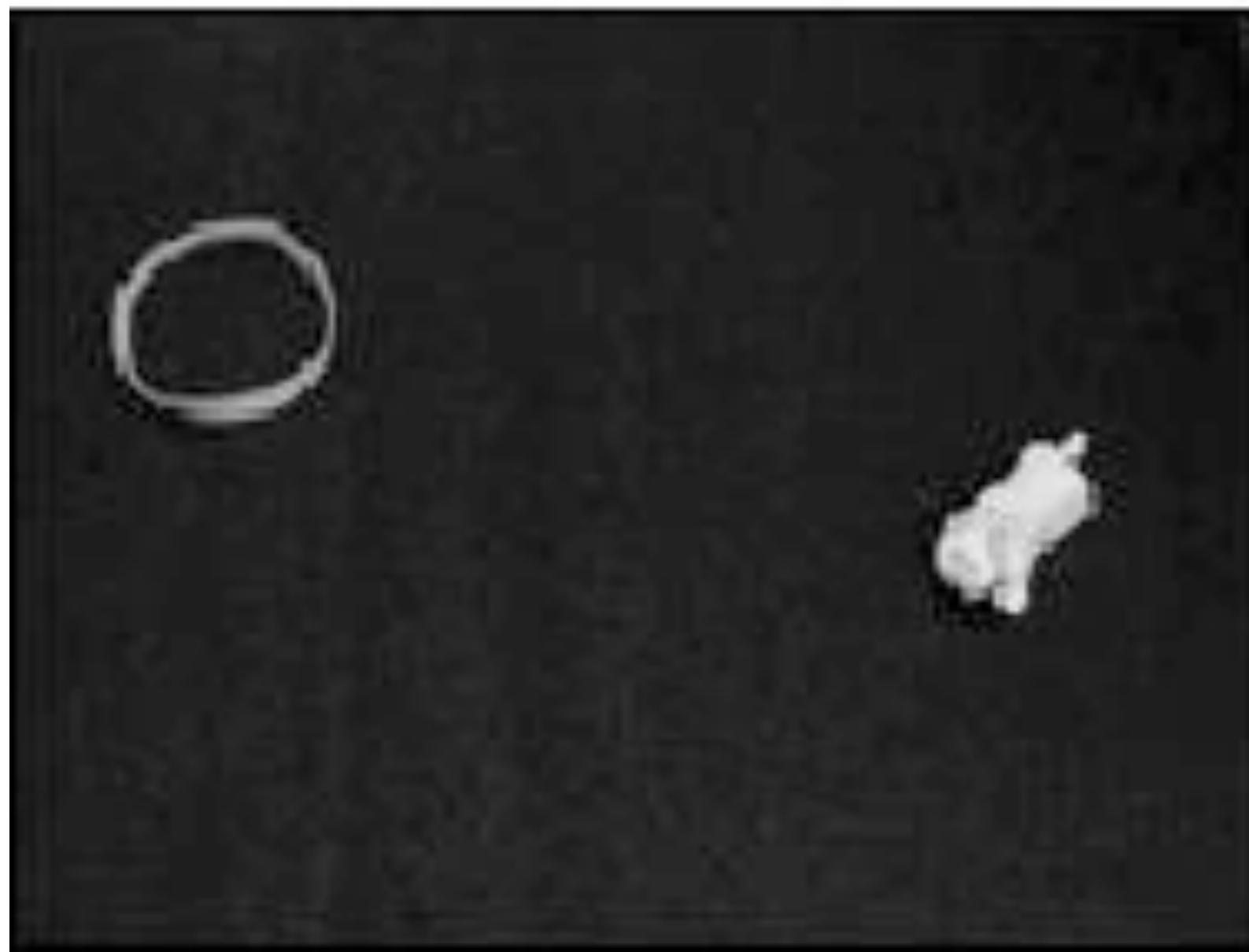
- Estimate is closer to ground truth



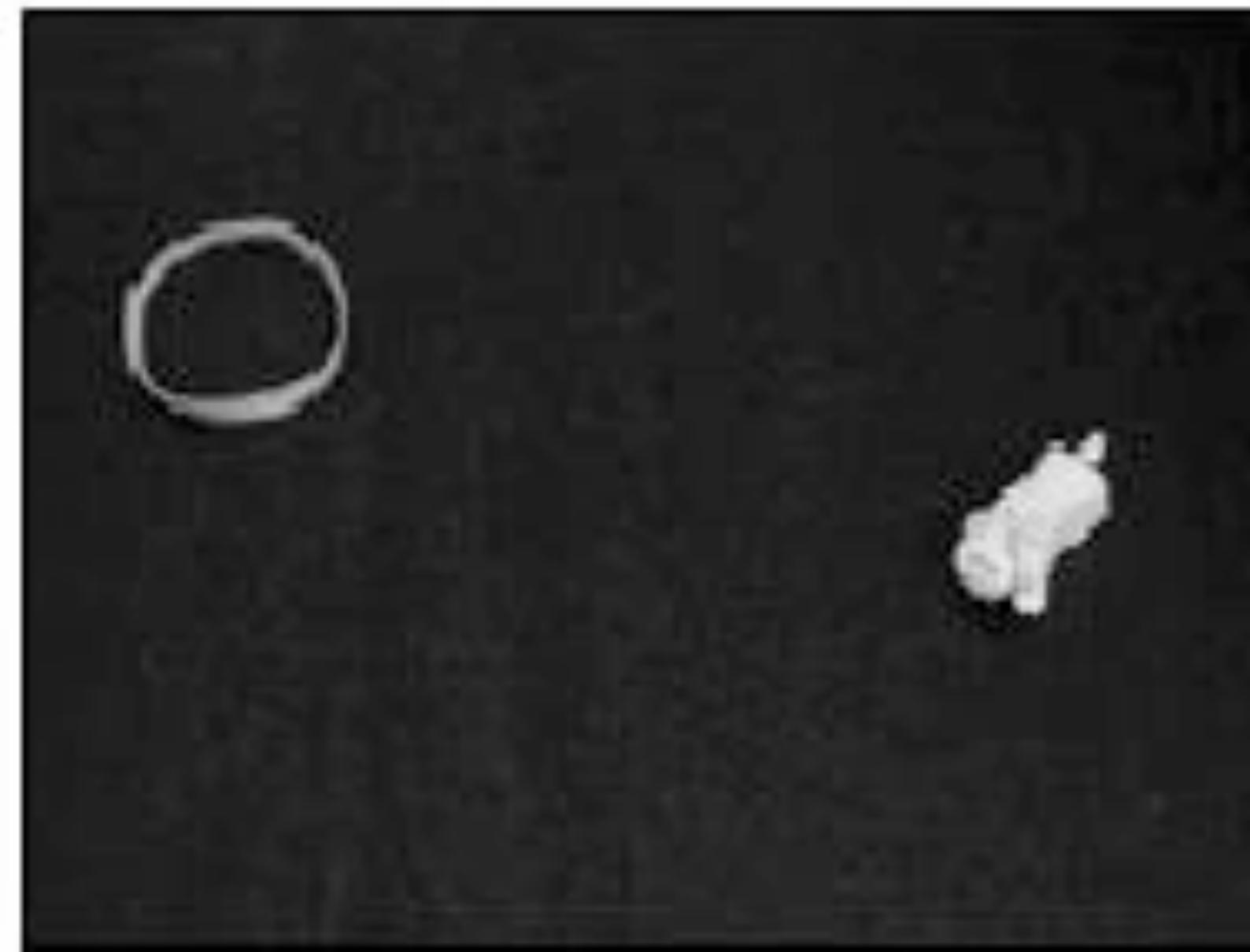
Example output

- Tracking is more smooth
 - But it lags a bit (why?)

Noisy input



Filtered output



Complicating the process

- Adding velocity information
 - Internal state representation

$$\mathbf{z}_t = \begin{bmatrix} x_t & y_t & \frac{dx}{dt} & \frac{dy}{dt} \end{bmatrix}^\top$$

- Transition forces constant velocity

$$\mathbf{z}_t = \mathbf{A} \cdot \mathbf{z}_{t-1} + \mathbf{e}_t = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix}$$

Measurements

- What we measure is position only:

$$\mathbf{w}_t = \mathbf{H} \cdot \mathbf{z}_t + \mathbf{v}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \mathbf{z}_t + \mathbf{v}_t$$

- Internal state is not same as measurement
 - And it has its own noise

The Kalman filter

- Predict future value

$$\mathbf{z}_t^- = \mathbf{A} \cdot \hat{\mathbf{z}}_{t-1} \quad \longleftarrow \quad \text{Change state according to model}$$

$$\mathbf{P}^- = \mathbf{A} \cdot \mathbf{P}_{t-1} \cdot \mathbf{A}^\top + \mathbf{Q} \quad \longleftarrow \quad \text{Accumulate the uncertainty}$$

- Correct that estimate given new input

$$\mathbf{K} = \mathbf{P}^- \cdot \mathbf{H}^\top \cdot \left(\mathbf{H} \cdot \mathbf{P}^- \cdot \mathbf{H}^\top + \mathbf{R} \right)^{-1} \quad \longleftarrow \quad \text{Get uncertainty of a consensus estimate}$$

$$\hat{\mathbf{z}}_t = \mathbf{z}_t^- + \mathbf{K} \cdot \left(\mathbf{w}_t - \mathbf{H} \cdot \mathbf{z}_t^- \right) \quad \longleftarrow \quad \text{And the estimate itself}$$

$$\mathbf{P}_t = \left(\mathbf{I} - \mathbf{K} \cdot \mathbf{H} \right) \cdot \mathbf{P}^- \quad \longleftarrow \quad \text{Get new uncertainty}$$

More elaborate extensions

- Add acceleration, etc ...
 - More complex internal state
 - More accurate models given various inputs
- Use this to track moving processes
 - Missile over an ocean
 - Finger over a touchscreen
 - Cars on the highway
 - ...

And yet more extensions

- Non-linear processes

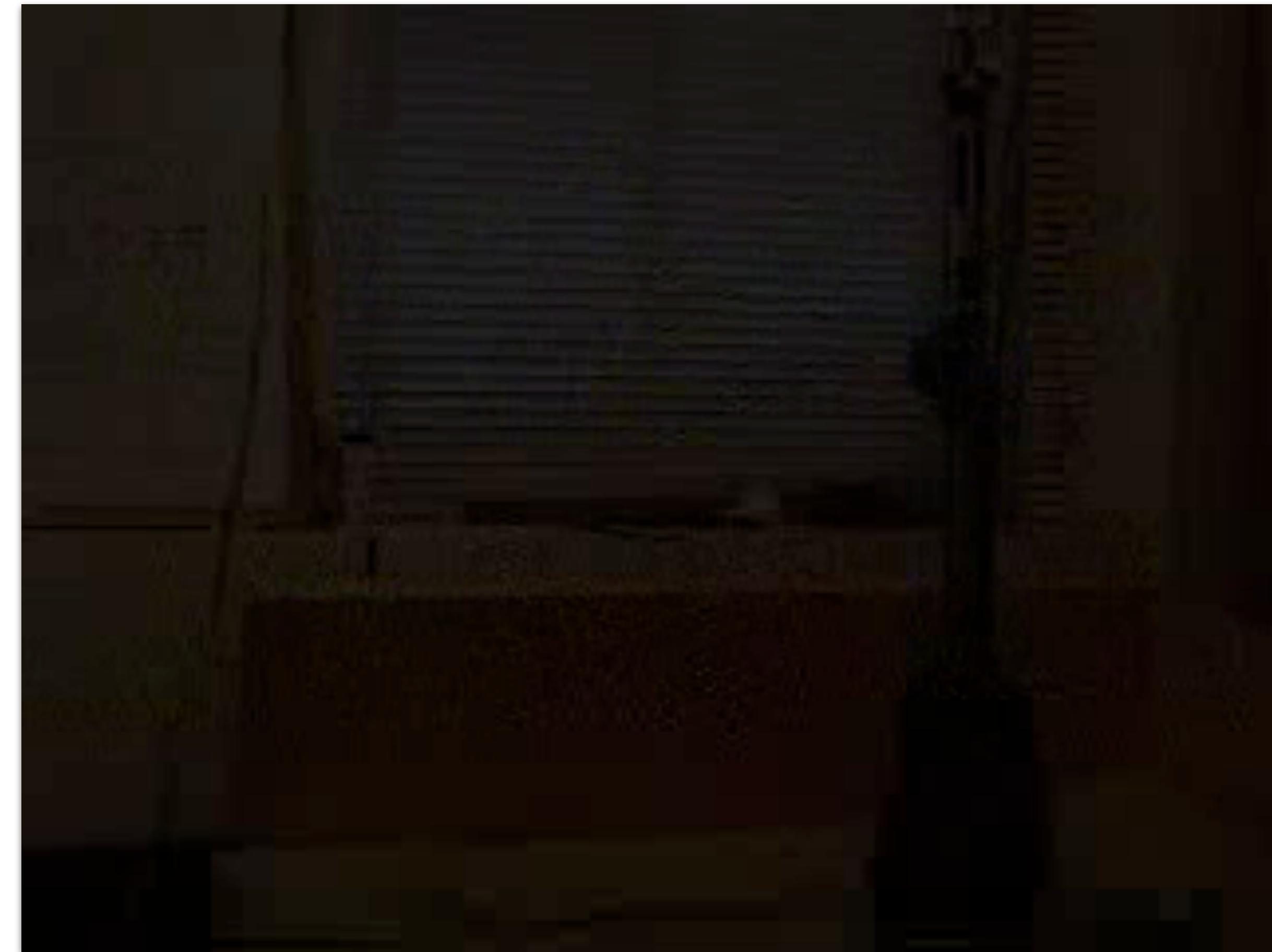
$$\mathbf{z}_t = \mathbf{f}(\mathbf{z}_{t-1}) + \mathbf{e}_t, \quad \mathbf{w}_t = \mathbf{h}(\mathbf{w}_{t-1}) + \mathbf{v}_t$$

- Extended Kalman filter
- Unscented Kalman filter (for very nonlinear)

- Particle filters

- Sampling based method
- Bypasses Gaussian assumptions

An example



In the real world

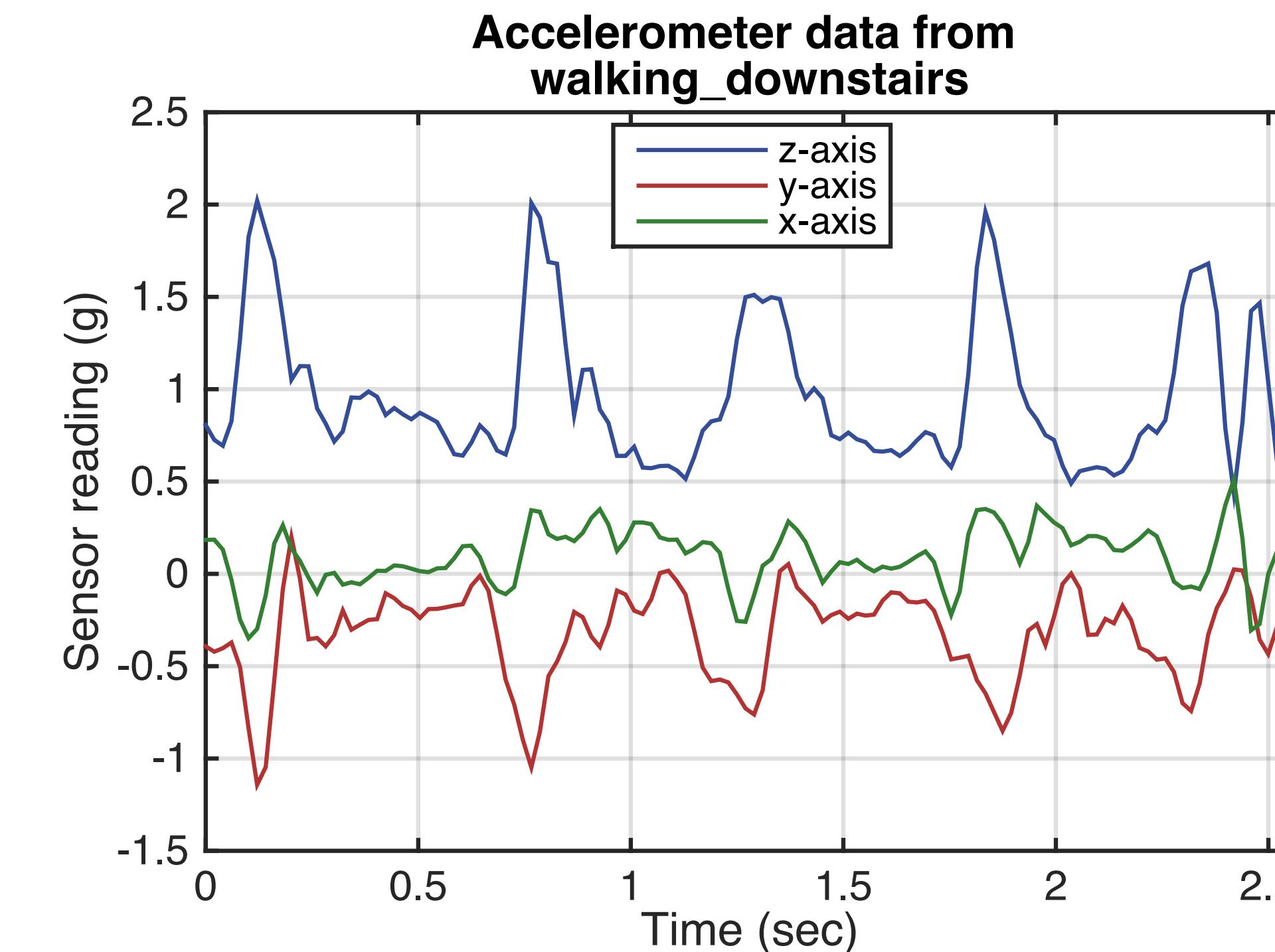
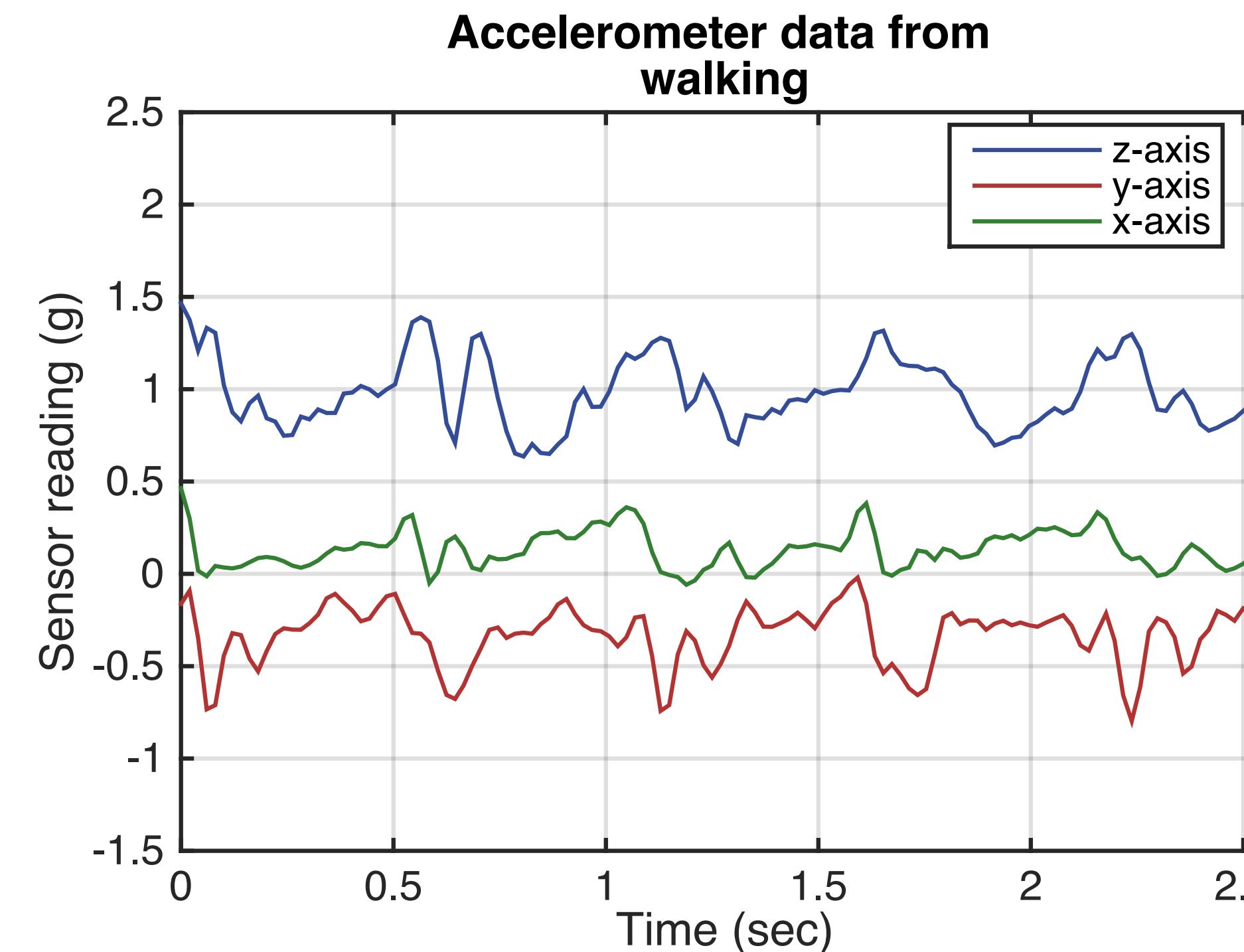


Using dynamical systems to classify

- We can also classify sequences using dynamical models
 - So far we only did prediction, tracking and smoothing
- Dynamical models can be fit on training sequences
 - And evaluated on new sequences that we want to classify
- Goodness of fit (for now) can be used for classification

An example

- 3D Accelerometer data from a cell phone
 - Classes are user activity (sitting, walking, going upstairs, etc).
 - Each activity has a unique temporal pattern



The VAR model (Vector AutoRegression)

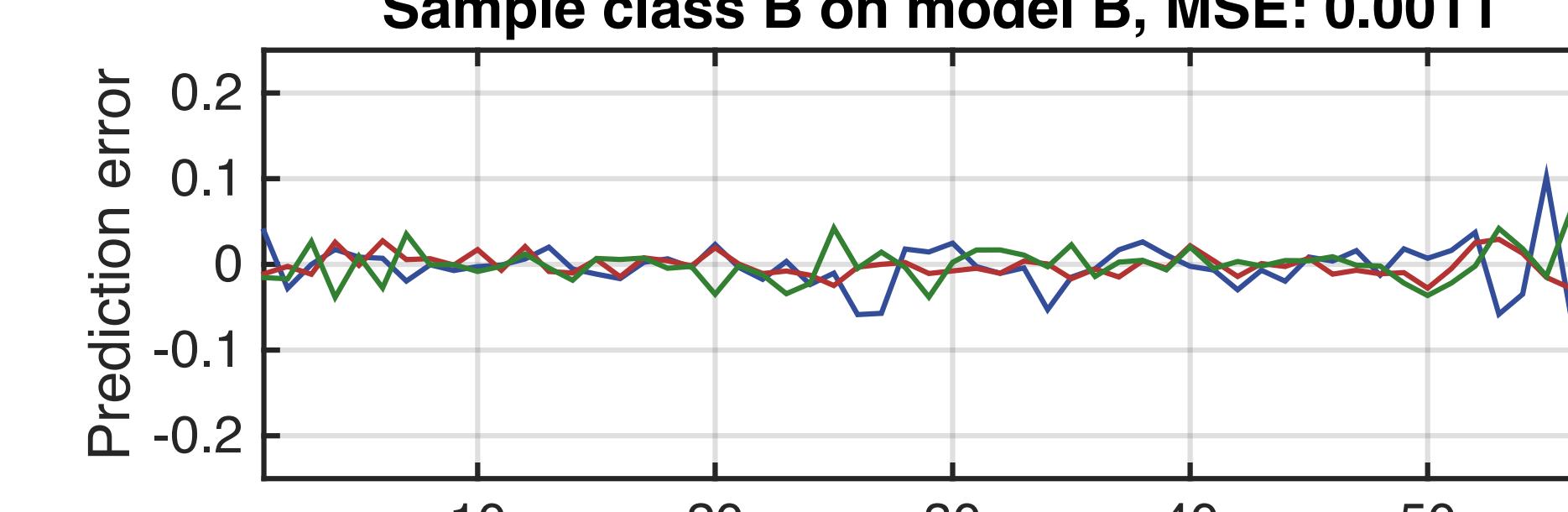
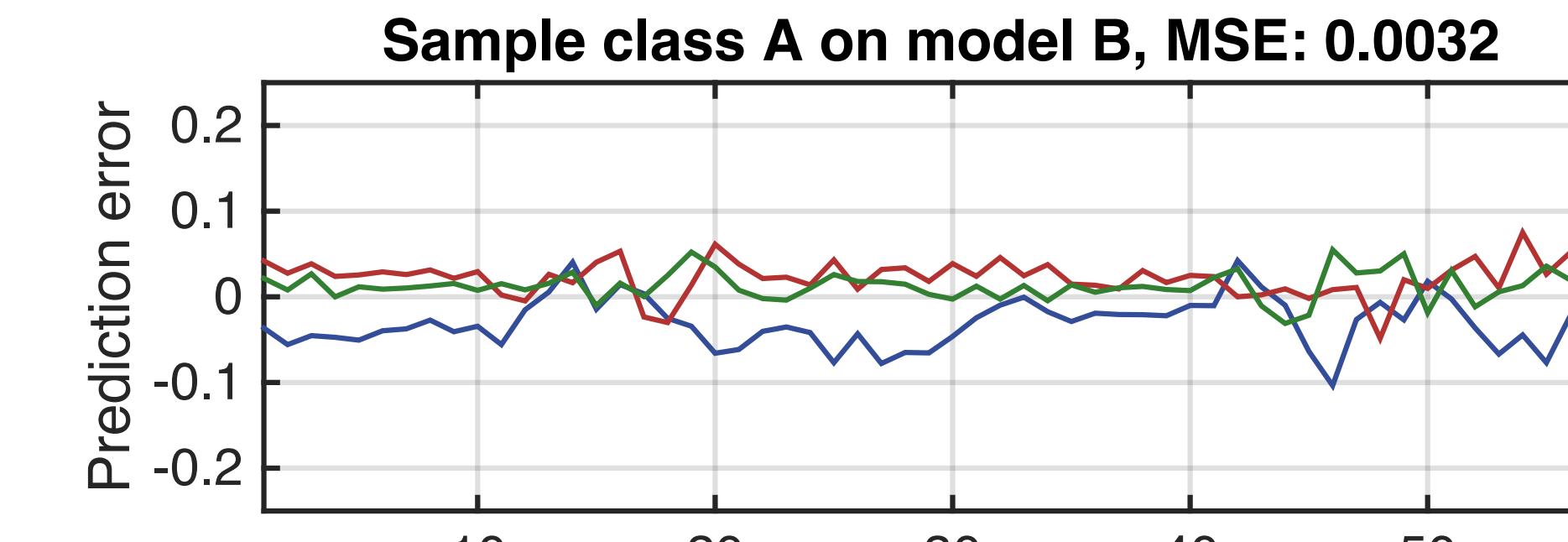
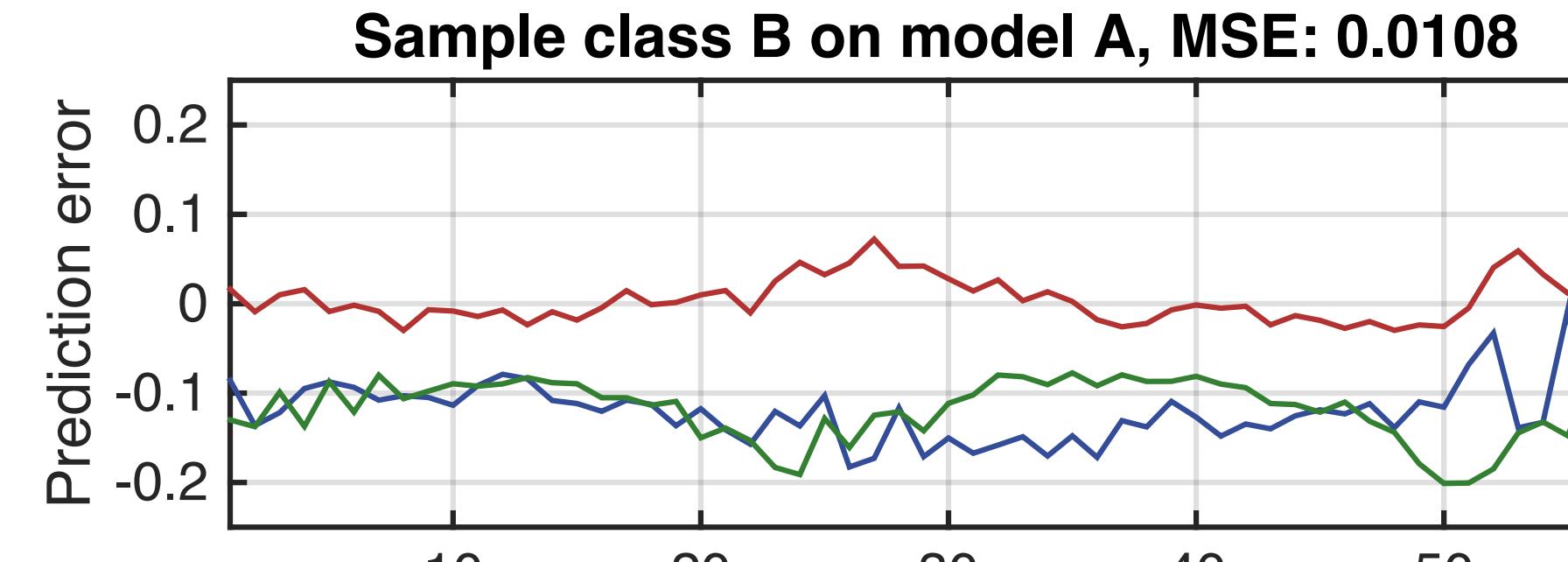
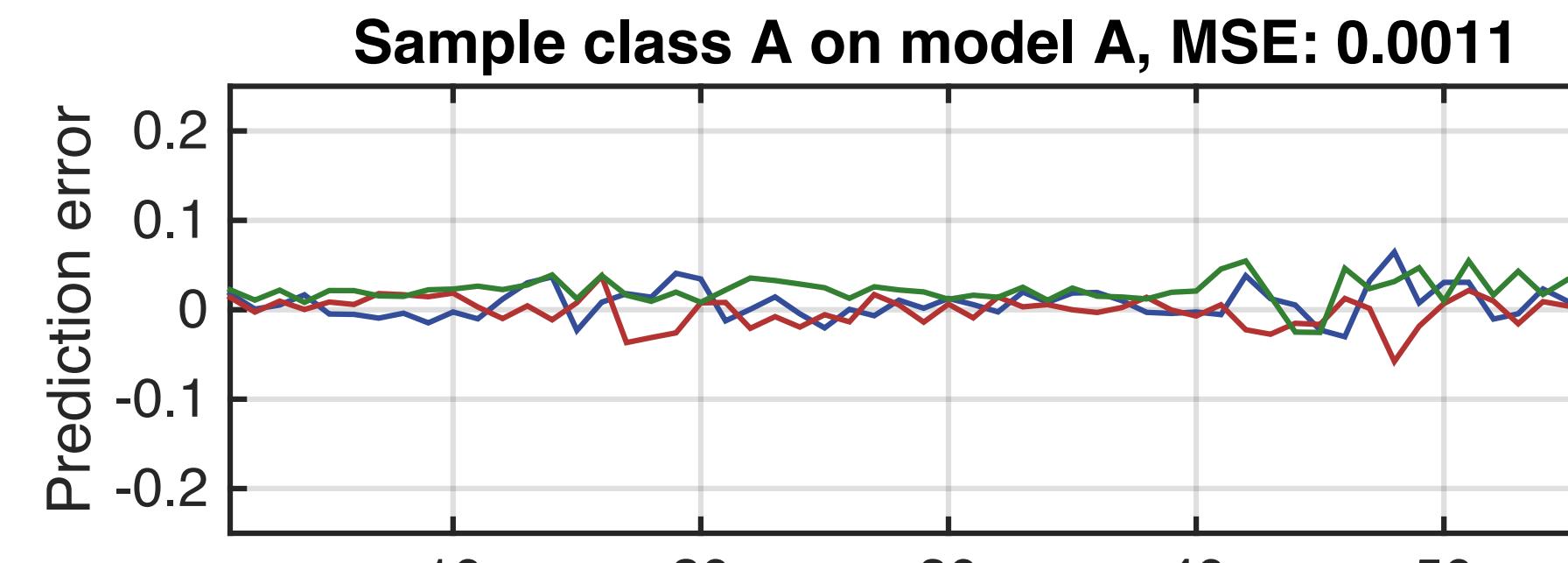
- We can make a linear model to predict future samples using the past

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} + \mathbf{e}_t$$

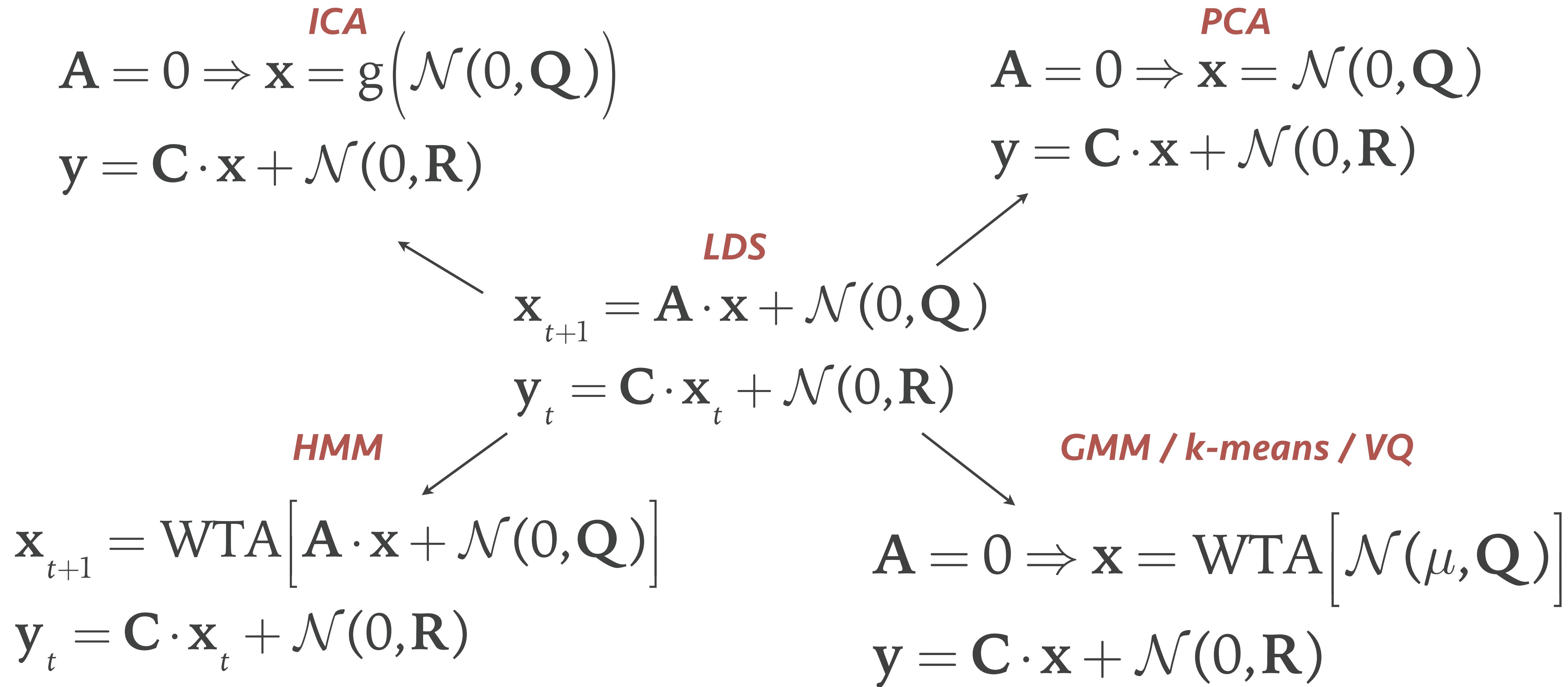
- Matrix \mathbf{A} should be unique to each class
 - Captures the temporal structure of each class
 - Can easily estimate \mathbf{A} with e.g. least squares

Predicting new inputs

- Get each model's prediction error on new samples
 - Model with least error belongs to the same class as input
 - Can also reformulate probabilistically to get likelihoods instead

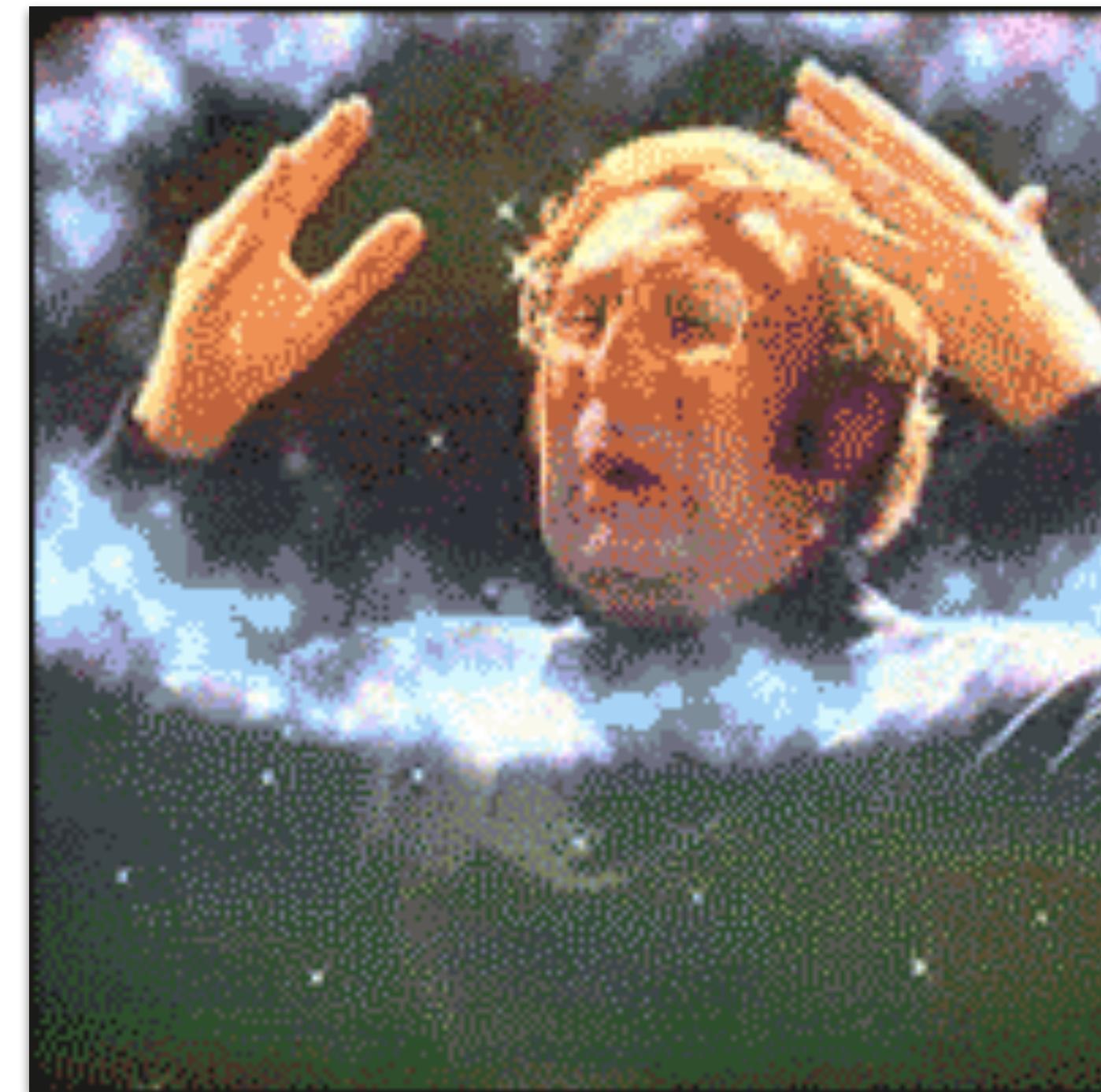


One big happy family



So it's all really the same model!

- Like I said in the beginning of this class, DSP and ML are pretty much the same equation over and over



Recap

- Missing data techniques
 - AR models → using temporal dynamics
 - NN, SVD, NMF → using context information
- Tracking and prediction
 - Kalman filter *et al.*
- The broader Linear Dynamical System family

Next week

- Source separation
 - Array methods using machine learning
 - Monophonic signal separation methods

Reading

- Missing data in audio
 - <http://www-sigproc.eng.cam.ac.uk/~sjg/springer/index.html>
 - <http://www.cs.illinois.edu/~paris/pubs/smaragdis-jspst10.pdf>
- Inpainting (and more) in images
 - <http://www.cs.princeton.edu/gfx/pubs/Barnes 2009 PAR/index.php>
- The Kalman filter
 - http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
- LDS
 - <http://cs.nyu.edu/~roweis/papers/lds.pdf>