Università
Ca' Foscari
Venezia

Computer Vision

Flow and tracking

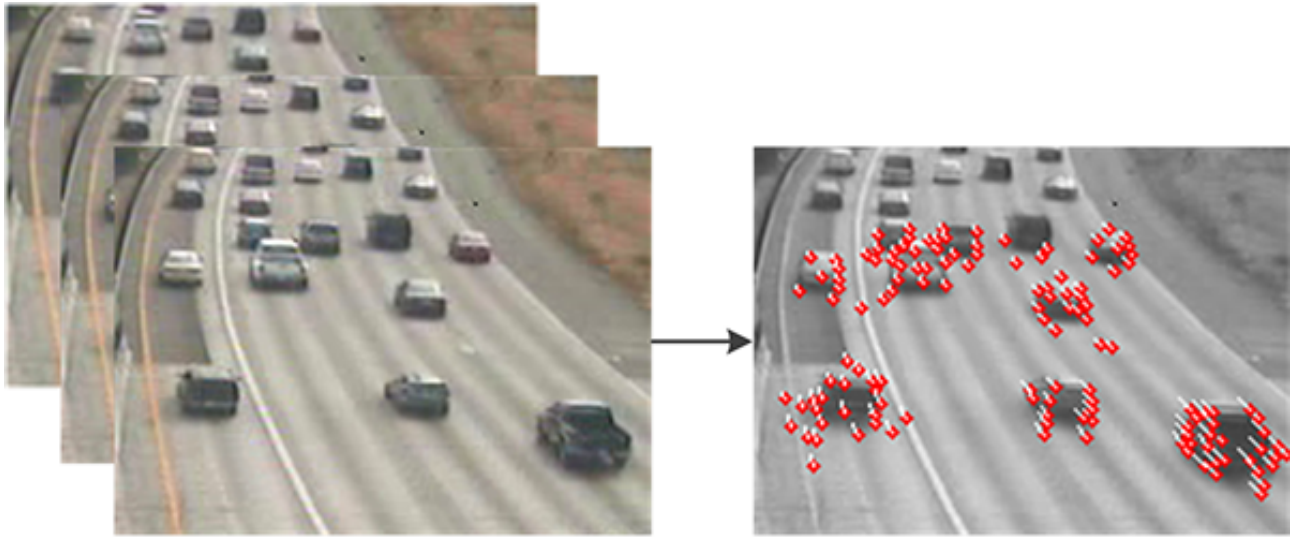
Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco
DAIS, Ca' Foscari University of Venice
Academic year 2018/2019

Università
Ca' Foscari
Venezia

Our goal

Track the position and movement of an object across a video sequence

- Multiple video frames
- Usually small delay between each frame
- In many cases the computational complexity matters! (aim for real-time solutions)



Motion field

What we would like to estimate is the 2D vector field of velocities of the image points induced by the relative motion between the viewing camera and the observed objects

This motion is the projection of the 3D velocity field onto the image plane

... But the motion field cannot be really observed with a single camera...

Optical Flow

What we observe in an image sequence is the temporal variation of the intensity level of each pixel.

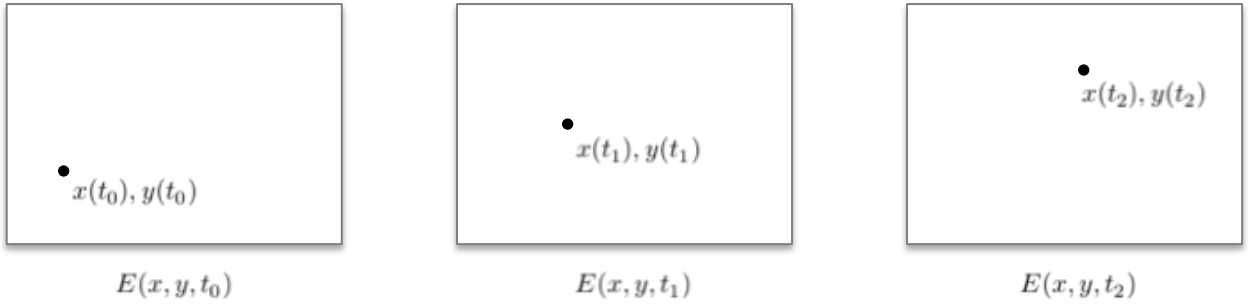
We call Optical Flow the observed 2D displacements of brightness patterns in the image.

Optical Flow is the only thing we can aim to estimate from the image... So what is the relation wrt the MF?

Optical Flow

Let's model our frame sequence as a function $E(x, y, t)$ that we assume to be continuous and differentiable (as many times as needed) in both the spatial and the temporal domain.

Consider a scene point moving through the image sequence:

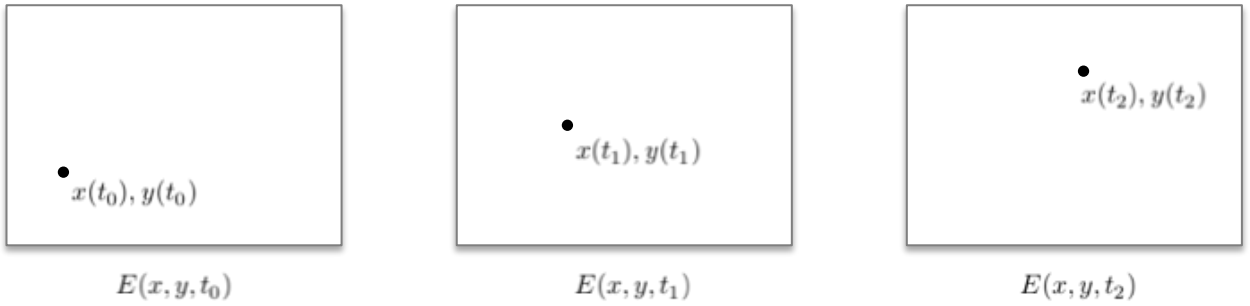


Assumption: The apparent brightness (or color) will remain constant during the movement

Università Ca' Foscari Venezia

Brightness constancy equation

Assumption: The apparent brightness (or color) will remain constant during the movement



$$E(x(t), y(t), t) = \text{constant}$$

Taking derivative wrt. t

$$\frac{dE(x(t), y(t), t)}{dt} = 0$$

$$\frac{\partial E}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial E}{\partial t} = 0$$

Using the chain rule

Università Ca' Foscari Venezia

Brightness constancy equation

$$\frac{\partial E}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial E}{\partial t} = 0$$

The spatial gradient of each image (can be computed via convolution)

Università Ca' Foscari Venezia

Brightness constancy equation

$$\frac{\partial E}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial E}{\partial t} = 0$$

The optical flow $\vec{v} = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}\right) = (u, v)$ we want to estimate

Brightness constancy equation

$$\frac{\partial E}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial t} + \boxed{\frac{\partial E}{\partial t}} = 0$$

The image derivative across frames (essentially the difference between consecutive frames)

Brightness constancy equation

In vector form:

$$(\nabla E)^T \vec{v} + E_t = 0$$

How many unknowns and equations per pixel?

$$E_x u + E_y v + E_t = 0$$

Brightness constancy equation

In vector form:

$$(\nabla E)^T \vec{v} + E_t = 0$$

How many unknowns and equations per pixel?

$$E_x u + E_y v + E_t = 0$$

3 known coefficients

Brightness constancy equation

In vector form:

$$(\nabla E)^T \vec{v} + E_t = 0$$

How many unknowns and equations per pixel?

$$E_x u + E_y v + E_t = 0$$

2 unknowns, one equation... of a line!

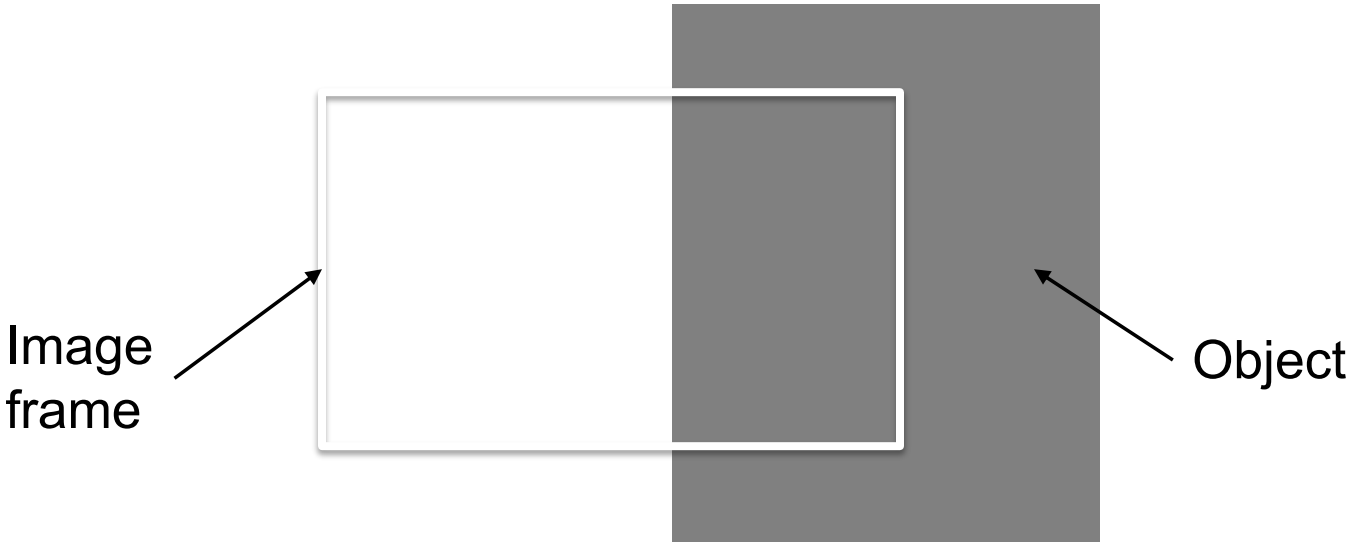
Università Ca' Foscari Venezia

The aperture problem

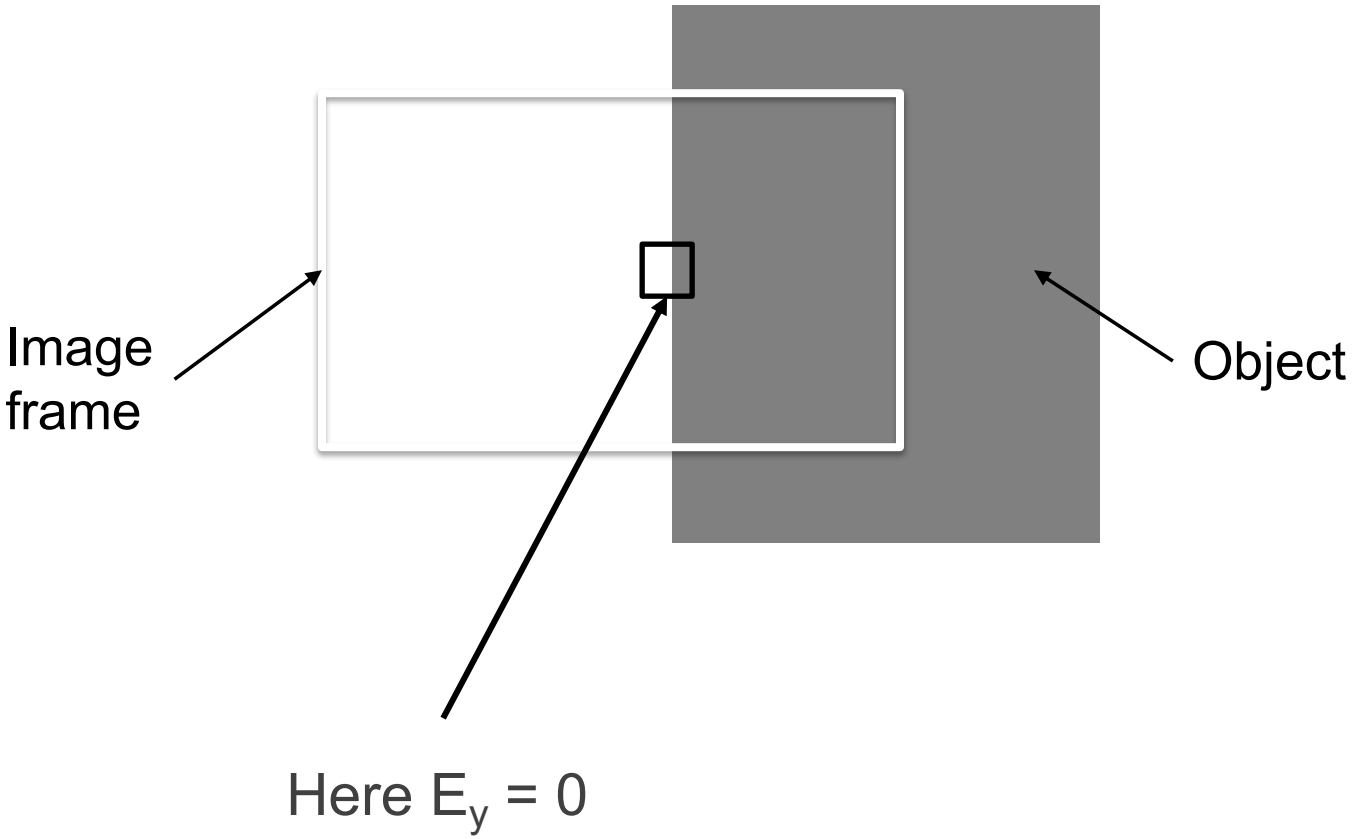
The component of the optical flow orthogonal to the spatial image gradient is not constrained by the image brightness constancy equation

In other terms, **only the flow in the gradient direction (normal flow) can be determined**

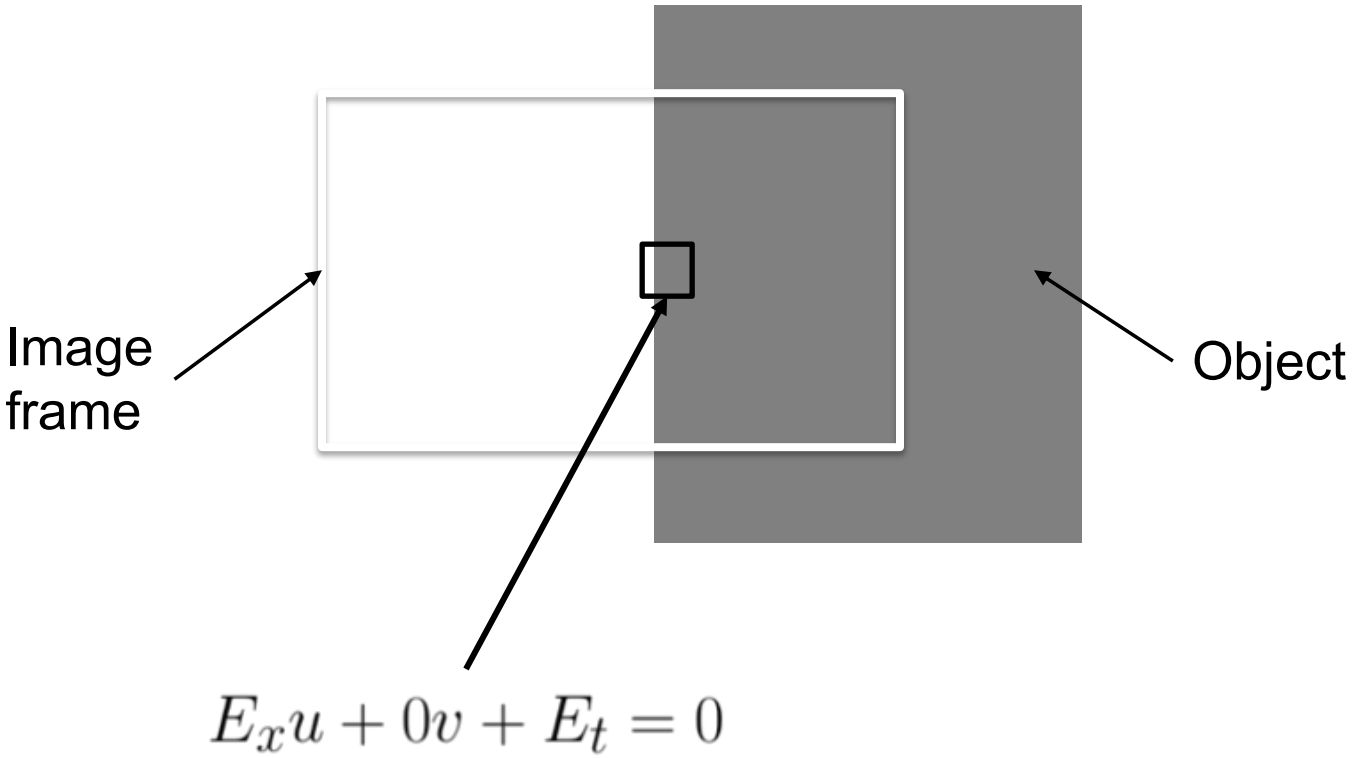
Ex:



The aperture problem



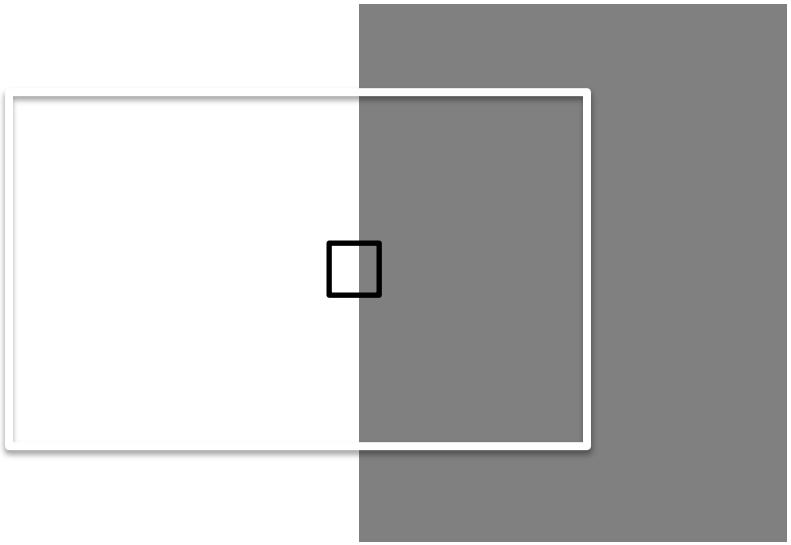
The aperture problem



$$E_x u + 0v + E_t = 0$$

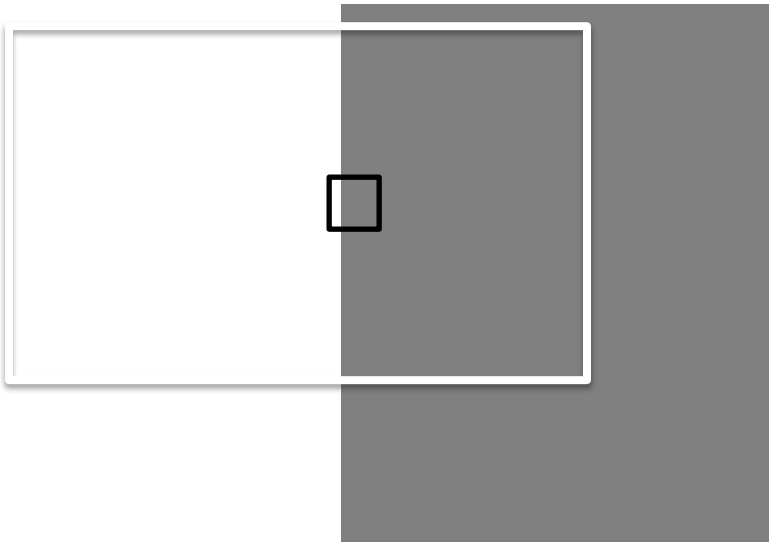
This implies that any additional vertical movement of the object produces a valid solution!

The aperture problem



Università
Ca' Foscari
Venezia

The aperture problem



The aperture problem

Optical Flow vs. Motion Field

The optical flow is the approximation of the motion field under the assumptions of:

- Lambertian surfaces
- Point-wise light source at infinity
- No photometric distortion

The error is:

- Small at points with high spatial gradient
- Exactly zero if the brightness gradient is parallel to the object motion

Computing the OF

Differential techniques

- Based on the spatial and temporal variations of the image brightness at all pixels
- Used to compute **dense** flow
- Ex: Lukas-Kanade flow

Matching techniques

- OF is estimated by feature matching only on a sparse subset of image points
- Used to compute **sparse** flow
- Ex: KLT tracker

Differential techniques

They are based on solving systems of partial differential equations, usually requiring the computation of second and high-order derivatives of the image brightness.

The most simple differential technique for flow estimation was proposed by B.D. Lukas and T. Kanade and is based on a least-squares fitting of the flow parameters:

- Is not iterative, therefore is fast and less biased by possible discontinuities of the motion field
- Involve only first-order derivatives: less sensitive to noise

Lukas-Kanade flow

$$E_x u + E_y v + E_t = 0$$

One equation per pixel is not enough... So we make an additional assumption:

The optical flow is well approximated by a **constant vector** within **any small patch** of the image plane

Suppose that we use a patch Q with size NxN (ex. 5x5). If we suppose that (u,v) is constant for every pixel in Q we obtain a system of 25 equation in 2 unknowns

Lukas-Kanade flow

$$\text{argmin}_{u,v} \sum_{\mathbf{p_i} \in Q} (E_x(\mathbf{p_i})u + E_y(\mathbf{p_i})v + E_t(\mathbf{p_i}))^2$$

Or, in matrix form: $\mathbf{argmin}_\beta \|\mathbf{y} - \mathbf{X}\beta\|^2$

where

$$\mathbf{X} = \begin{pmatrix} E_x(\mathbf{p_1}) & E_y(\mathbf{p_1}) \\ E_x(\mathbf{p_2}) & E_y(\mathbf{p_2}) \\ \vdots & \vdots \\ E_x(\mathbf{p_{N^2}}) & E_y(\mathbf{p_{N^2}}) \end{pmatrix} \quad \mathbf{y} = - \begin{pmatrix} E_t(\mathbf{p_1}) \\ E_t(\mathbf{p_2}) \\ \vdots \\ E_t(\mathbf{p_{N^2}}) \end{pmatrix}$$

$$\beta = (u, v)^T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Lukas-Kanade flow

$$\beta = (u, v)^T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Solving the least-squares problem depends by inverting the matrix $\mathbf{A} = \mathbf{X}^T\mathbf{X}$:

- $\mathbf{A}$ should be invertible (ie. $\lambda_1, \lambda_2 \neq 0$)
- $\mathbf{A}$ should be well conditioned:

  large λ_1 and $\frac{\lambda_1}{\lambda_2}$ not too large (ie. both large)

Lukas-Kanade flow

$$\mathbf{X} = \begin{pmatrix} E_x(\mathbf{p_1}) & E_y(\mathbf{p_1}) \\ E_x(\mathbf{p_2}) & E_y(\mathbf{p_2}) \\ \vdots & \vdots \\ E_x(\mathbf{p_{N^2}}) & E_y(\mathbf{p_{N^2}}) \end{pmatrix}$$

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} = \begin{pmatrix} \sum E_x E_x & \sum E_x E_y \\ \sum E_y E_x & \sum E_y E_y \end{pmatrix}$$

$\mathbf{A}$ is exactly the structure tensor used in the Harris corner detector!

- Corners are places in which both λ_1, λ_2 are big, and this also is the case in which the LK flow works best
- Aperture problem disappear at corners!

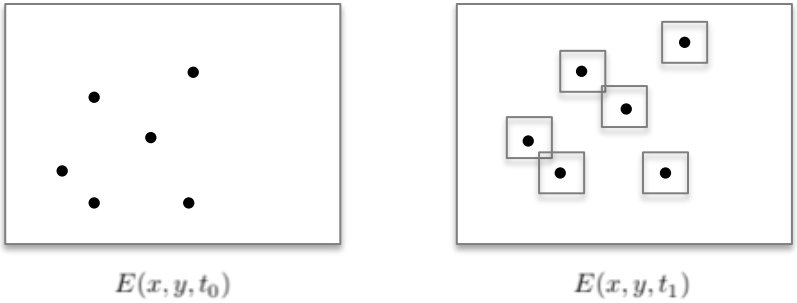
Corners are a good place to compute optical flow

Feature-based techniques

Feature-based techniques estimate the optical flow at feature points only to obtain a sparse field

General idea:

- Find corners in the first image
- Search for corresponding intensity patches in the second image



KLT Algorithm

Algorithm born from the combined work of:

Lukas-Kanade (goal: how should we track patches from frame to frame?) and **Tomasi-Kanade** (goal: how should we select good features?)

Algorithm:

1. Extract corners
2. For each corner, compute the displacement using Lukas-Kanade method
3. Store the displacement of each corner and update the corner position
4. Add more corners
5. Repeat step 2-4
6. Return long trajectories for each corner point

Correlation-based techniques

Some techniques, instead of using LK to find the displacement, simply compute normalized correlation between patches centered around features of each image.

Assumption: In small period of times (or small motions), corners tend to remain corners

Ex:

D. Nister, O. Naroditsky, J. Bergen, "Visual Odometry", CVPR 2004

Visual Odometry technique

- Extract corners in first and second image
- Extract image patches around each corner
- Find matching pairs (a,b) where:
  - a is a corner patch from the first image
  - b is a corner patch from the second image
  - b is the best match for a (according to normalized correlation)
  - **a is the best match for b**

The last condition was proven to be a good heuristic to get an approximate result of the linear assignment problem

Linear assignment problem

We have N agents and N tasks. Any agent can be assigned to perform any task, incurring some *cost* that may vary depending on the agent-task assignment.

Goal: assign exactly one agent to each task and exactly one task to each agent in such a way that the *total cost* of the assignment is minimized

An optimal solution exists but is computationally too intensive for tracking.

Linear assignment problem

Approximate solution:

- Each agent rank the tasks in order of cost
- Each task rank the agents in order of cost

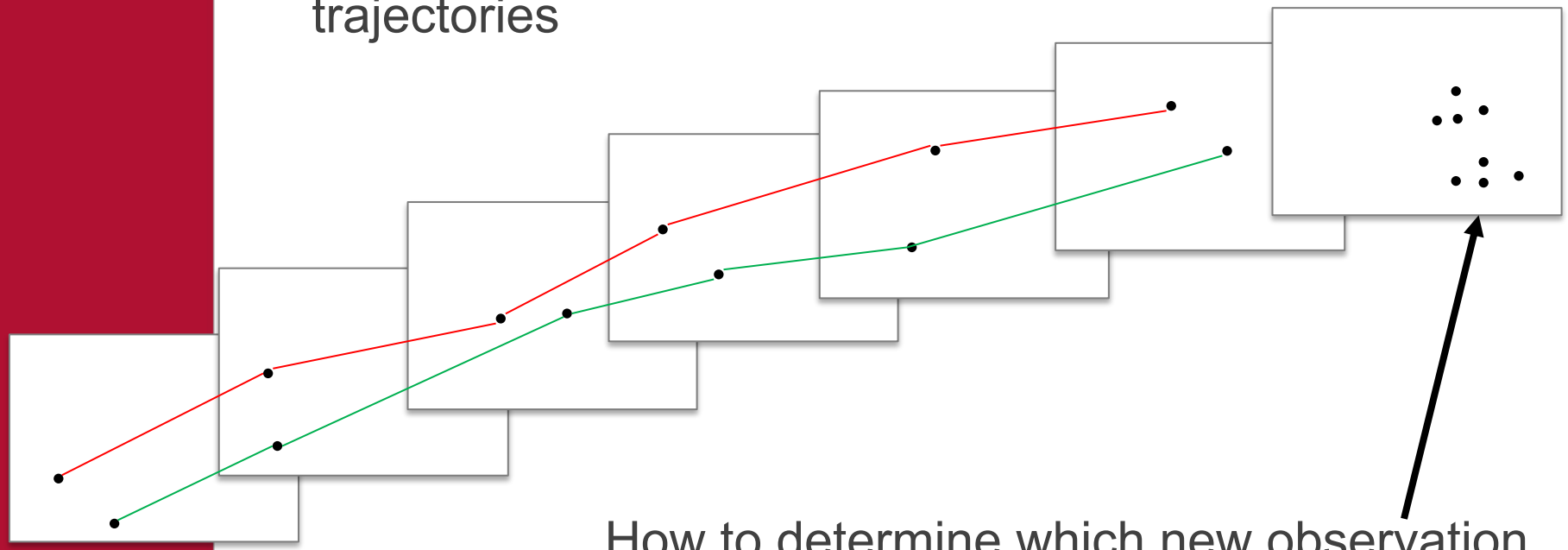
Agents and tasks can pair up **only if** each is the best match for the other, and viceversa

NOTE: this will eliminate many potential good matches, but overall we obtain a close to optimal solution.

Data association

So far we have seen the matching/tracking problem between two frames.

In a frame sequence we also have the problem to mantain a continuity of identity and generate trajectories

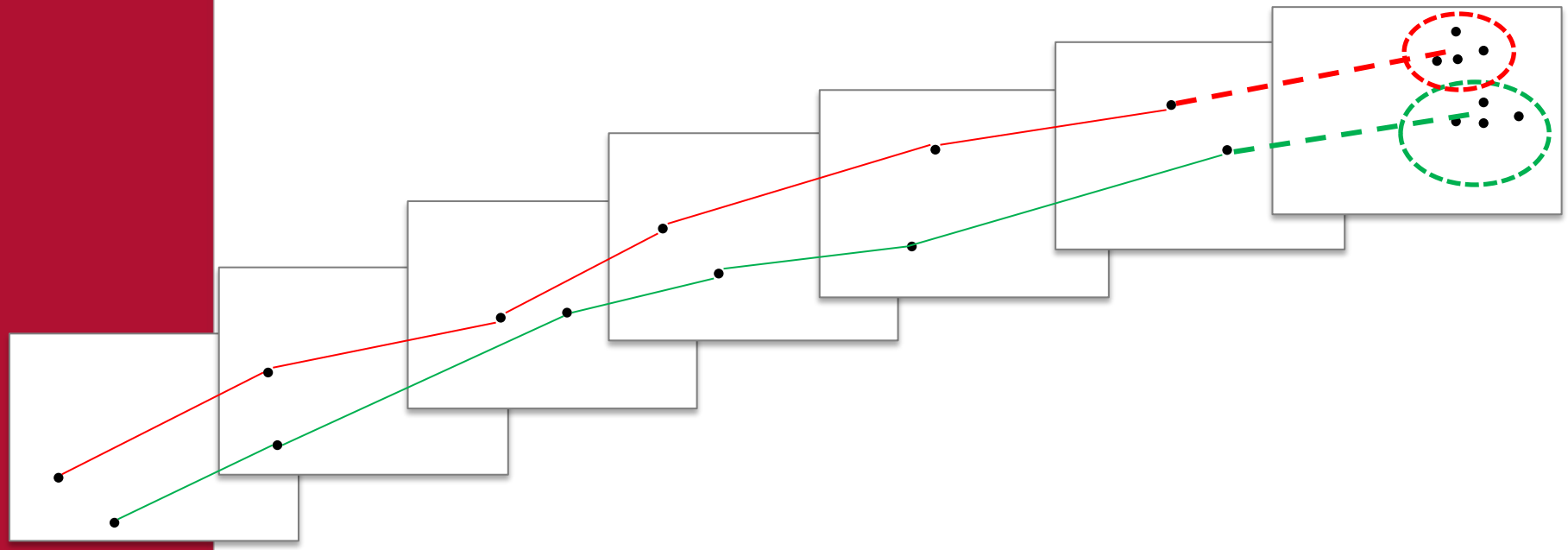


How to determine which new observation should be added to each track?

Data association

Main idea:

- Predict the next tposition and take the observation closest to that prediction
- ...or at least restrict the search to a gating region



Tracking as inference problem

The interesting properties of a feature (like position, velocity, etc) are modelled as a discrete time random variable. (for example, let $\mathbf{X}_i$ model the position of a feature at time i.)

The "true" position at each frame x_1, x_2, x_3 … is **not** directly observable, but it can be inferred by a sequence of noisy measurements y_1, y_2, y_3 … also modelled as a random variable $\mathbf{Y}_i$.

With this setting, the tracking is divided in 3 steps…

Tracking as inference problem

Prediction: Suppose that we have seen $y_0, y_1, \ldots y_{i-1}$. What can we expect for the internal state of the system at time i? In other words, we would like to obtain $P(\mathbf{X}_i|Y_0=y_0, \ldots \mathbf{Y}_{i-1} = y_{i-1})$

Data association: What are the measurements obtained in the i-th frame that can be used to obtain a better estimate of the current state $\mathbf{X}_i$?

Correction (update): When we identified the set of relevant measurements y_i for this frame, how to update $\mathbf{X}_i$? In other words, we would like to obtain $P(\mathbf{X}_i|Y_0=y_0, \ldots \mathbf{Y}_{i-1} = y_{i-1}, \mathbf{Y}_i = y_i)$

Tracking as inference problem

We will also pose the following useful simplifications:

1. **Only the last frame matters:**

   $P(\mathbf{X}_i \mid \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{i-1}) = P(\mathbf{X}_i \mid \mathbf{X}_{i-1})$. The transition distribution $P(\mathbf{X}_i \mid \mathbf{X}_{i-1})$ defines the **system model** and its uncertainties as a Markov sequence

2. **Measurements depends only on the current state:**

   This dependence is modeled by specifying the distribution of the measurement given the state $P(\mathbf{Y}_i \mid \mathbf{X}_i)$. This define the measurement model

With the above simplifications, the problem is tractable as a **Bayes inference problem**. Different system and measurement models result in problems that can be more or less computationally expensive to solve

Linear Kalman Filter

The simplest scenario is when:

- System and measurement models are **linear**
- Noise is assumed to be zero-mean Gaussian
- **Pdfs are all Gaussian**

System model:

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \quad \mathbf{v}_{k-1} \sim N(0, \mathbf{Q}_{k-1})$$

Measurement model:

$$\mathbf{y}_k = H_k\mathbf{x}_k + \mu_k \quad \mu_k \sim N(0, \mathbf{R}_k)$$

System model

State vector at time k

Vector that account for noise in the system model (ie. The model is an approximation of the real one)

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \qquad \mathbf{v}_{k-1} \sim N(0, \mathbf{Q}_{k-1})$$

State transition matrix (the model is linear!). Note that the matrix can change among the steps

Noise PDF is fully defined by its covariance matrix

Measurement model

Vector that accounts for noise in the measurements

$$\mathbf{y}_k = H_k \mathbf{x}_k + \mu_k \qquad \mu_k \sim N(0, \mathbf{R}_k)$$

Matrix mapping the current state $\mathbf{x}_k$ to the measurements

Measurement noise PDF is fully defined by its covariance matrix

Linear Kalman Filter

We assume to have an initial state, Normally distributed with mean $\mathbf{x}_0$ and covariance $\mathbf{P}_0$

Prediction step:

$$\hat{\mathbf{x}}_k = F_{k-1}\mathbf{x}_{k-1}$$
$$\hat{\mathbf{P}}_k = F_{k-1}\mathbf{P}_{k-1}F_{k-1}^T + \mathbf{Q}_{k-1}$$

Update step:

$$\tilde{\mathbf{y}_k} = \mathbf{y}_k - H_k\hat{\mathbf{x}}_k$$
$$\mathbf{S}_k = H_k\mathbf{P}_kH_k^T + \mathbf{R}_k$$
$$\mathbf{K}_k = \mathbf{P}_kH_k^T\mathbf{S}_k^{-1}$$
$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\tilde{\mathbf{y}}_k$$
$$\mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T$$

Linear Kalman Filter

We assume to have an initial state, Normally distributed with mean $\mathbf{x}_0$ and covariance $\mathbf{P}_0$

Prediction step:

$$\hat{\mathbf{x}}_k = F_{k-1}\mathbf{x}_{k-1}$$ ← Predicted state

$$\hat{\mathbf{P}}_k = F_{k-1}\mathbf{P}_{k-1}F_{k-1}^T + \mathbf{Q}_{k-1}$$ ← Predicted state covariance

Update step:

$$\tilde{\mathbf{y}_k} = \mathbf{y}_k - H_k\hat{\mathbf{x}}_k$$ ← Measurement residual

$$\mathbf{S}_k = H_k\mathbf{P}_kH_k^T + \mathbf{R}_k$$ ← Measurement residual covariance

$$\mathbf{K}_k = \mathbf{P}_kH_k^T\mathbf{S}_k^{-1}$$ ← Kalman gain

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\tilde{\mathbf{y}}_k$$ ← Updated state estimate

$$\mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T$$ ← Updated state covariance

Example

Suppose that we track a single feature.

Hidden state vector:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$$

Example

Suppose that we track a single feature.

Hidden state vector:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$$

State transition matrix:

$$F = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example

Suppose that we track a single feature.

Hidden state vector:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$$

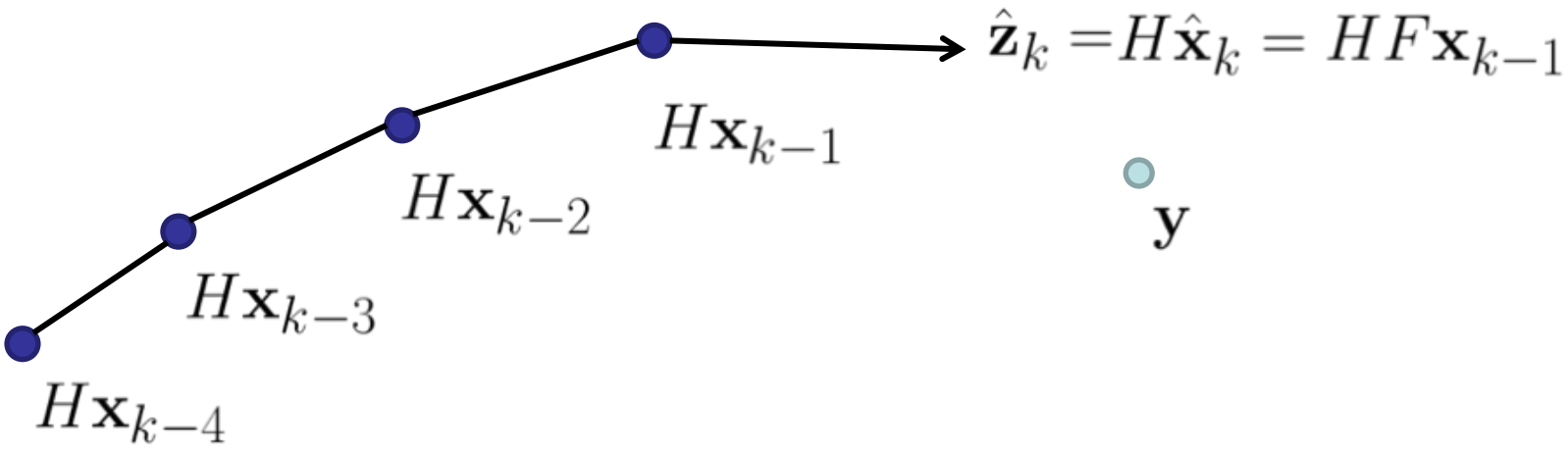
State transition matrix:

$$F = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Measurement matrix:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Tracking

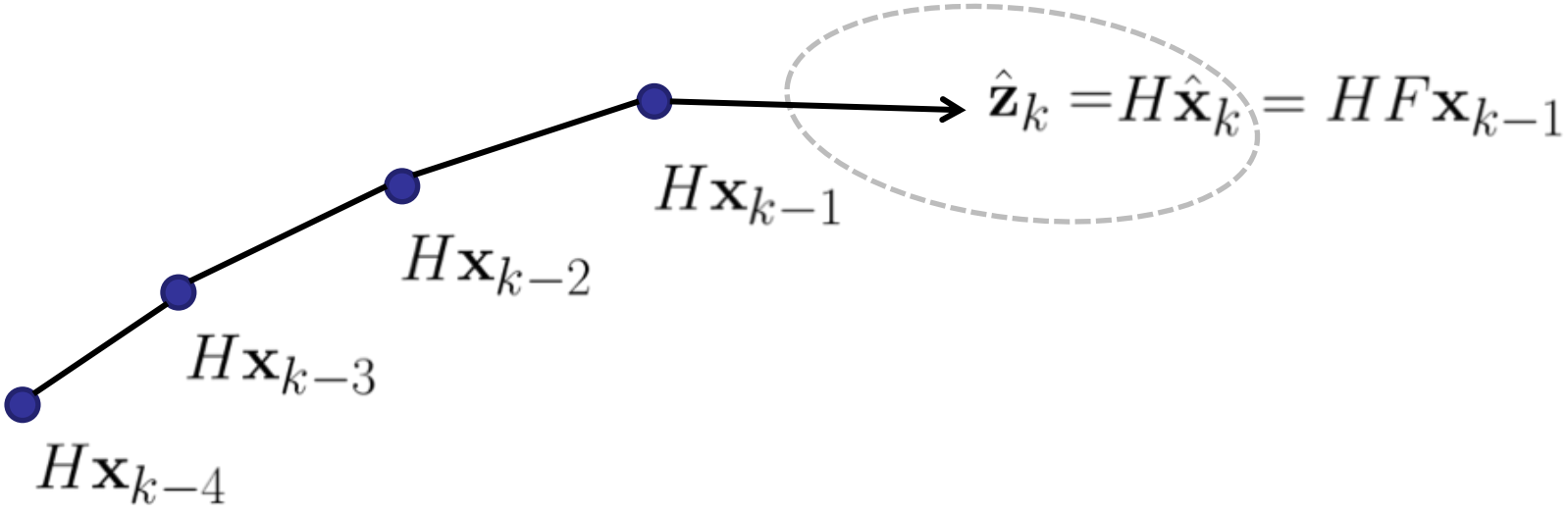


Usually we check the Mahalanobis distance

$$d = \sqrt{(x-\mu)^T \mathbf{C}^{-1}(x-\mu)}$$

between a new measurement **y** and the predicted state mapped in measurement space $\hat{\mathbf{z}}_k$

Tracking

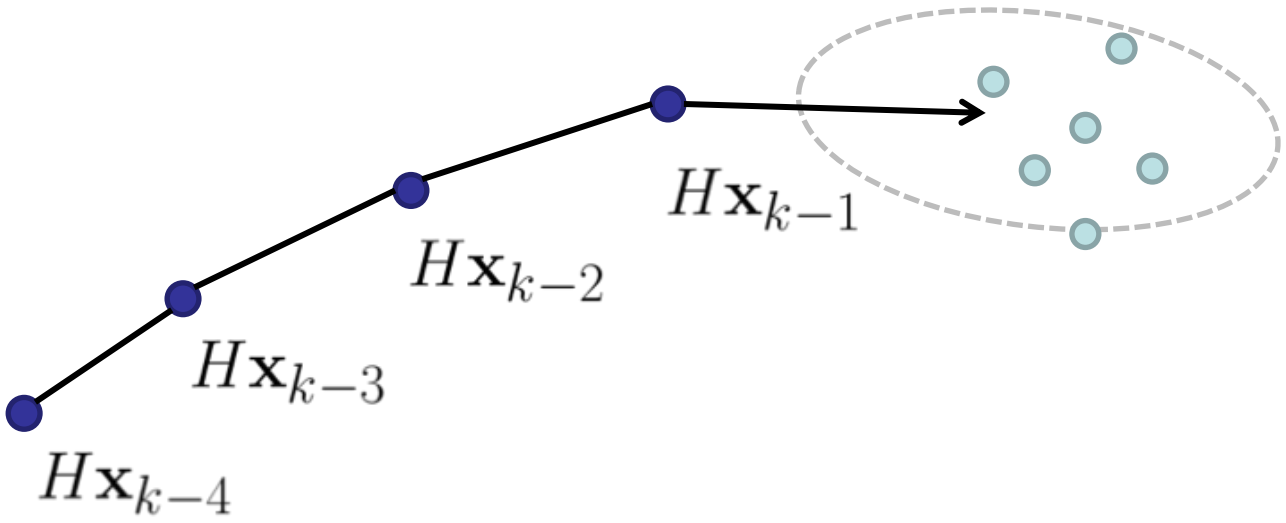


We set a threshold G, and accept a measurement if

$$\tilde{\mathbf{y}}^T S^{-1} \tilde{\mathbf{y}} < G$$

Such formula defines an ellipsoidal gating region depending on the measurement residual covariance S

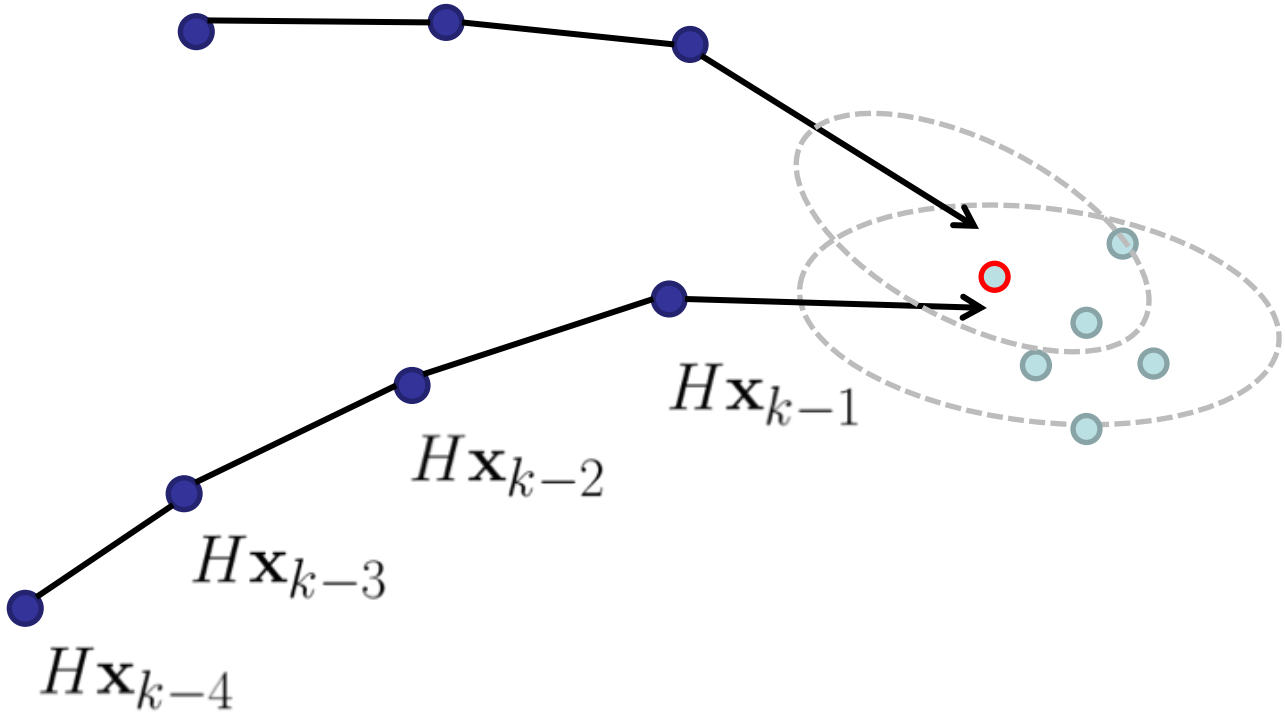
Global Nearest Neighbor



We could have multiple possible observations (features) to be incorporated into track.

GNN: Take the feature maximizing a "score" against the track. The score can be based on Mahalanobis distance between the feature and the predicted location (like $e^{-\text{dist}}$) or similarity of appearance.

Global Nearest Neighbor



Problem: If we do GNN for each track we can have a contention for the same observation...

Linear assignment problem, again...

M tracks, N features to assign

We build a MxN matrix of matching scores:

$$X = \begin{matrix} 0.95 & 0.76 & 0.62 & 0.41 & 0.06 \\ 0.23 & 0.46 & 0.79 & 0.94 & 0.35 \\ 0.61 & 0.02 & 0.92 & 0.92 & 0.81 \\ 0.49 & 0.82 & 0.74 & 0.41 & 0.01 \\ 0.89 & 0.44 & 0.18 & 0.89 & 0.14 \end{matrix}$$

Goal: Chose one number from each row and one number for each column such that the sum of the chosen numbers is maximized.

How many possible combinations?

Linear assignment problem

Possible solutions:

- Simplex method (but it has an exponential worst-case complexity)
- See the problem as maximal matching in a weighted bipartite graph (solved via max-flow algorithms like Ford-Fulkerson)
- Use approximate solutions like:

J. Kosowsky and A. Yuille. *The invisible hand algorithm: Solving the assignment problem with statistical physics*, *Neural Networks*, 7:477-490, 1994