



COMPUTER VISION

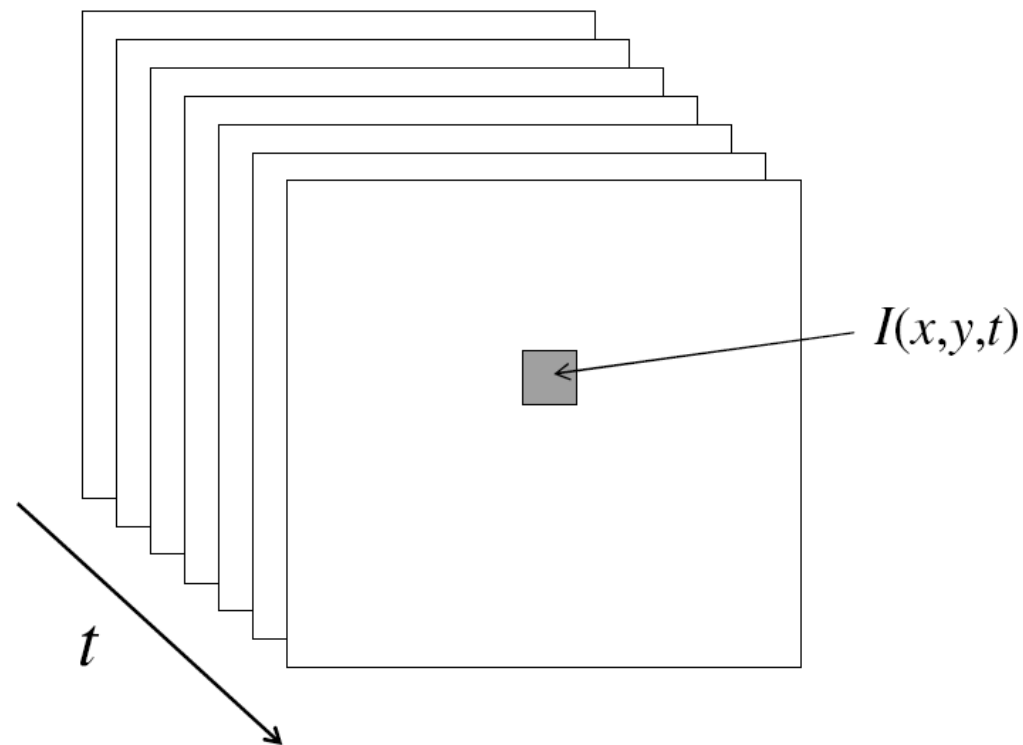
Motion and Tracking

Overview

- Motion
 - Analysis
 - Motion field
 - Estimation – optical flow
- Background Subtraction
 - Goal
 - Modelling the background
 - Mixture models for the background
- Tracking
 - Detection vs. tracking
 - Kalman filter for tracking
 - Issues

Video

- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)



Motion analysis

- A lot of information can be extracted from time varying sequences of images:
 - camouflaged objects are only easily seen when they move.
 - the relative sizes and position of objects are more easily determined when the objects move.
 - even simple image differencing provides an edge detector for the silhouettes of texture-free objects moving over any static background.

Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept



Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept



Motion analysis

- The analysis of visual motion can be divided into two stages:
 - the measurement of the motion, and
 - the use of motion data to segment the scene into distinct objects and to extract three dimensional information about the shape and motion of the objects.
- There are two types of motion to consider:
 - movement in the scene with a static camera,
 - and movement of the camera, or ego motion.
- Since motion is relative, these types of motion should be the same. However, this is not always the case, since if the scene moves relative to the illumination, shadow and specularities effects need to be dealt with.

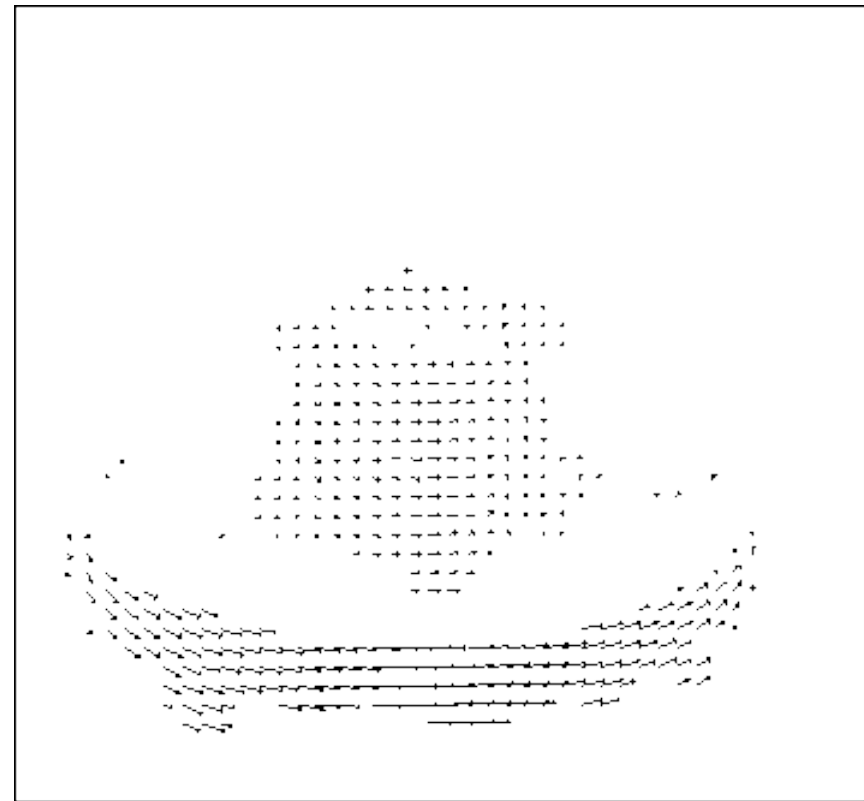
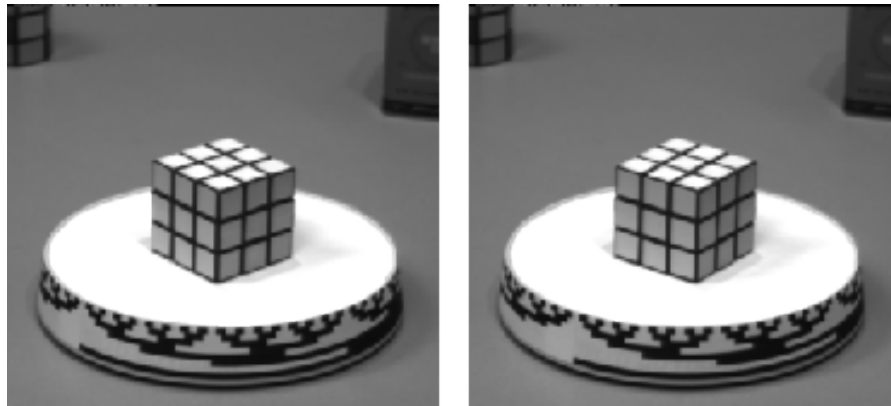


Uses of motion

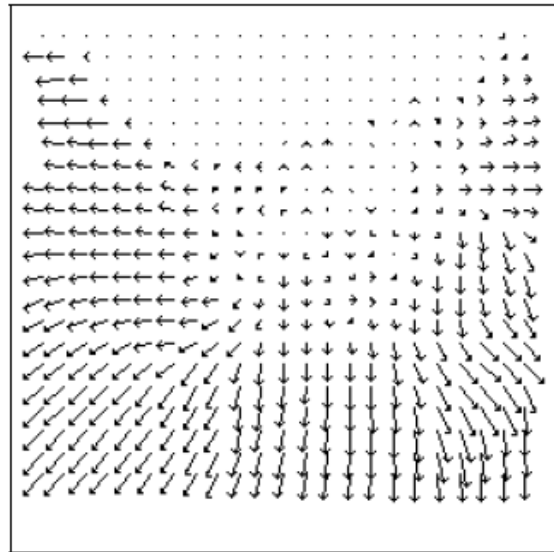
- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning dynamical models
- Recognizing events and activities
- Improving video quality (motion stabilization)

Motion field

- The motion field is the projection of the 3D scene motion into the image



Motion field + camera motion



Length of flow vectors inversely proportional to depth Z of 3d point

Figure 1.2: Two images taken from a helicopter flying through a canyon and the computed optical flow field.

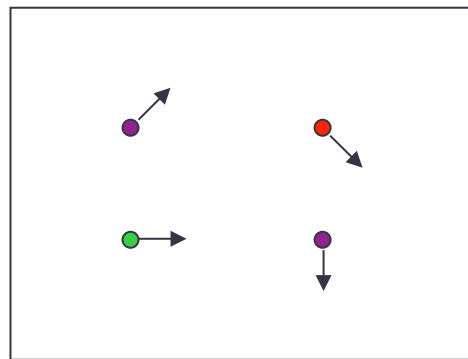
Figure from Michael Black, Ph.D. Thesis

points closer to the camera move more quickly across the image plane

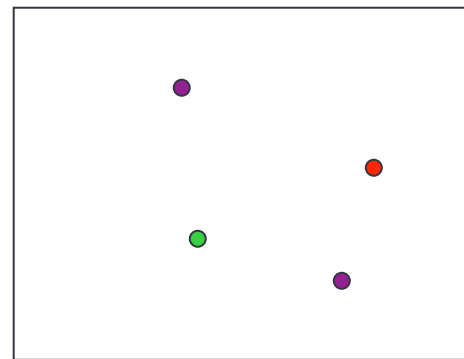
Optical flow

- **Definition:** optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion

Problem definition: optical flow



$H(x, y)$



$I(x, y)$

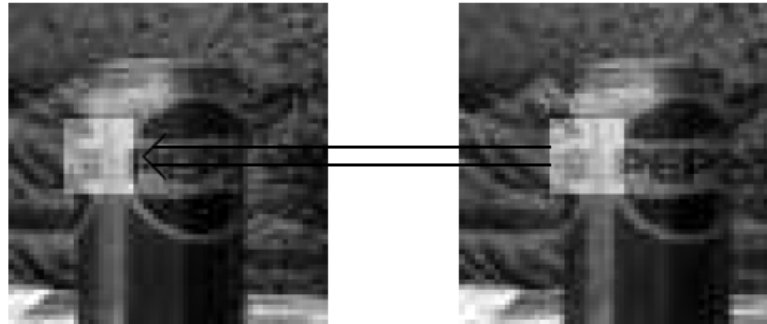
- How to estimate pixel motion from image H to image I?
 - Solve pixel correspondence problem
 - given a pixel in H, look for **nearby** pixels of the **same color** in I

Key assumptions

- **color constancy**: a point in H looks the same in I
 - For grayscale images, this is brightness constancy
- **small motion**: points do not move very far

This is called the optical flow problem

Brightness constancy



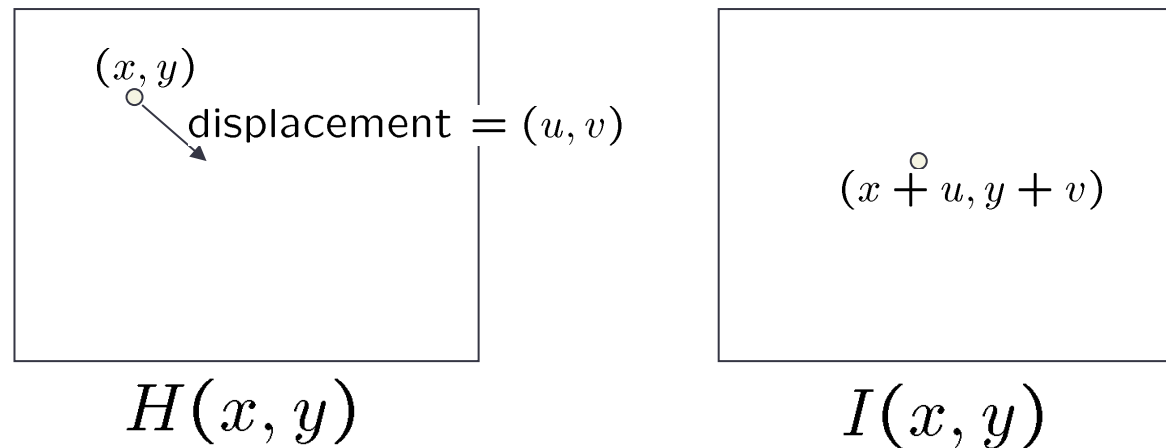
This assumption of brightness conservation implies that :

$$\frac{dI}{dt} = 0 \quad \Rightarrow \quad I_x u + I_y v + I_t = 0$$

where the partial derivatives of I are denoted by subscripts, and u and v are the x and y components of the optical flow vector.

This last equation is called the optical flow constraint equation since it expresses a constraint on the components u and v of the optical flow (we will look into this in more detail next).

Optical flow constraints (grayscale images)



- Let's look at these constraints more closely
 - **brightness constancy:** Q: what's the equation?

$$H(x, y) = I(x + u, y + v)$$

- **small motion:**

$$\begin{aligned} I(x + u, y + v) &= I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms} \\ &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \end{aligned}$$

Optical flow equation

- Combining these two equations

$$0 = I(x + u, y + v) - H(x, y)$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$\approx I(x, y) + I_x u + I_y v - H(x, y)$$

$$\approx (I(x, y) - H(x, y)) + I_x u + I_y v$$

$$\approx I_t + I_x u + I_y v$$

$$\approx I_t + \nabla I \cdot [u \ v]$$

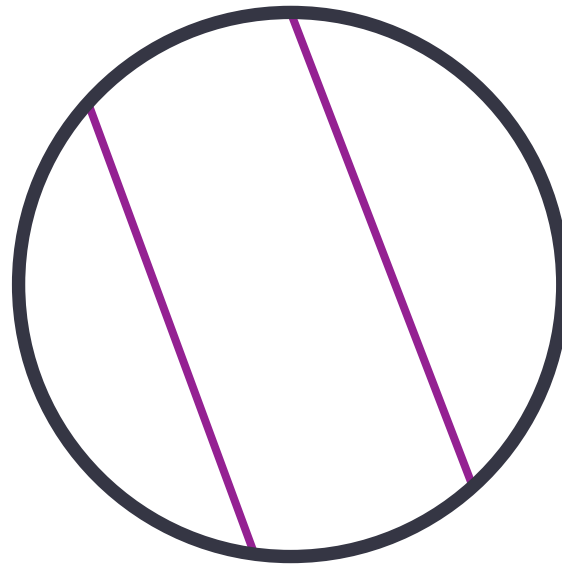
Optical flow equation

$$0 = I_t + \nabla I \cdot [u \ v]$$

- Q: how many unknowns and equations per pixel?

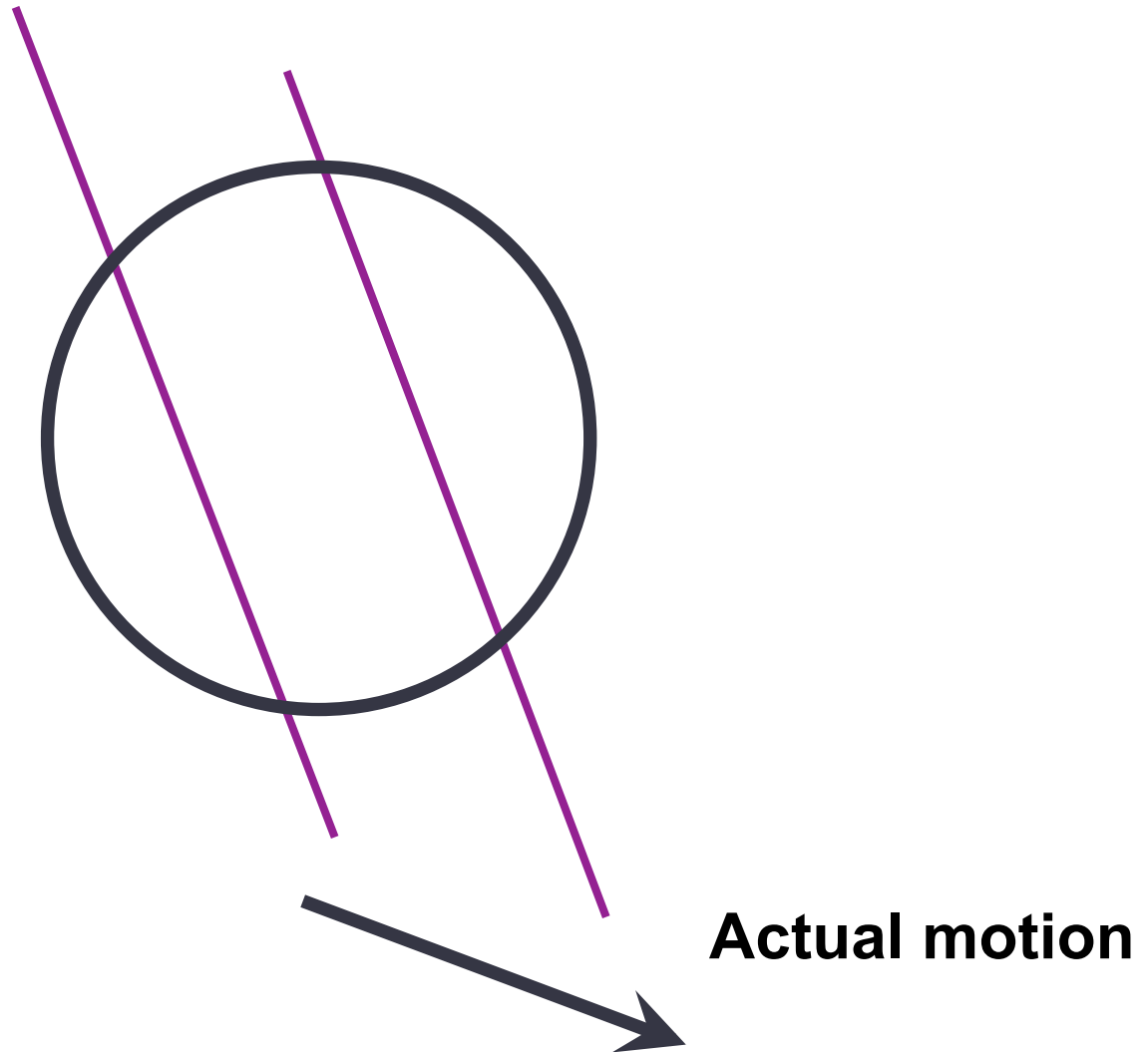
Intuitively, what does this ambiguity mean?

The aperture problem



Perceived motion

The aperture problem



The barber pole illusion



http://en.wikipedia.org/wiki/Barberpole_illusion

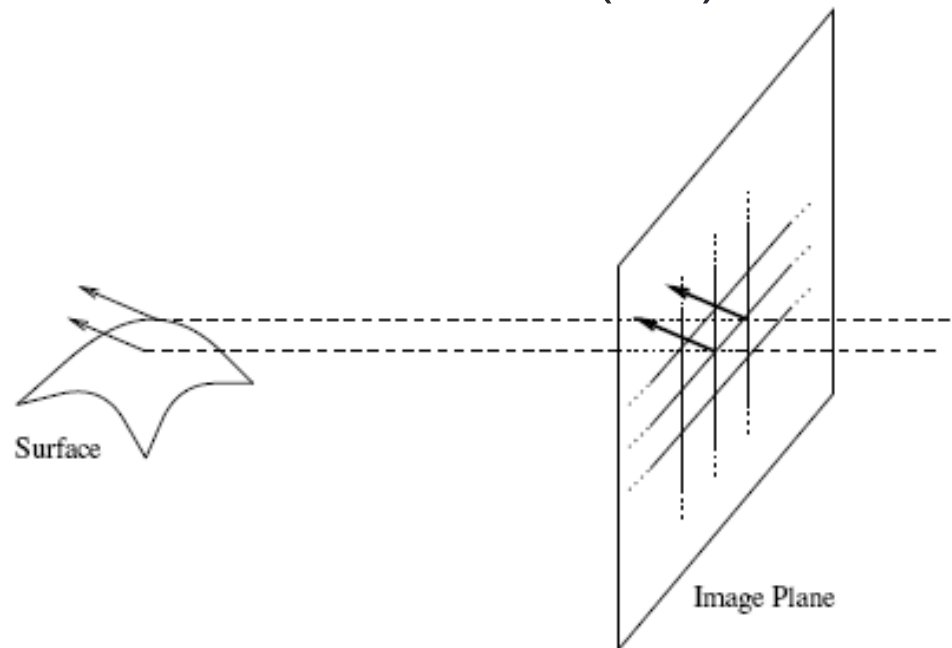
The barber pole illusion



http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm

Solving the aperture problem (grayscale image)

- How to get more equations for a pixel?
- **Spatial coherence** constraint: pretend the pixel's neighbors have the same (u,v)



Solving the aperture problem (grayscale image)

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)
 - If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p}_i) + \nabla I(\mathbf{p}_i) \cdot [u \ v]$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\begin{matrix} A & d & = & b \\ 25 \times 2 & 2 \times 1 & & 25 \times 1 \end{matrix}$$

Solving the aperture problem

Prob: we have more equations than unknowns

$$\begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 \quad 25 \times 1 \end{matrix} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- **minimum least squares solution given by solution (in d) of:**

$$\begin{matrix} (A^T A) & d = A^T b \\ 2 \times 2 & 2 \times 1 \quad 2 \times 1 \end{matrix}$$

$$\begin{matrix} \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} & \begin{bmatrix} u \\ v \end{bmatrix} = - & \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \\ A^T A & & A^T b \end{matrix}$$

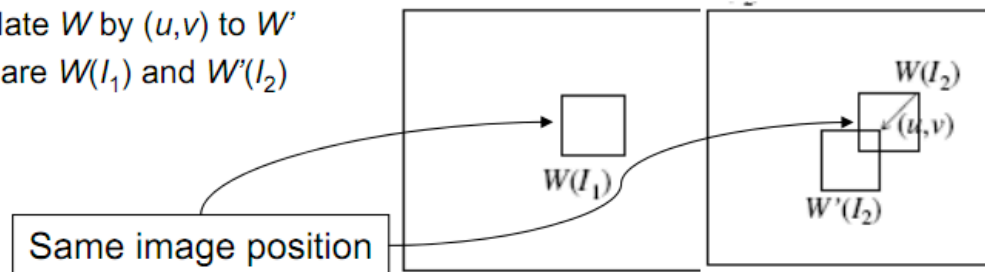
- The summations are over all pixels in the K x K window
- This technique was first proposed by Lucas & Kanade (1981)

Solving the aperture problem

- Algorithm

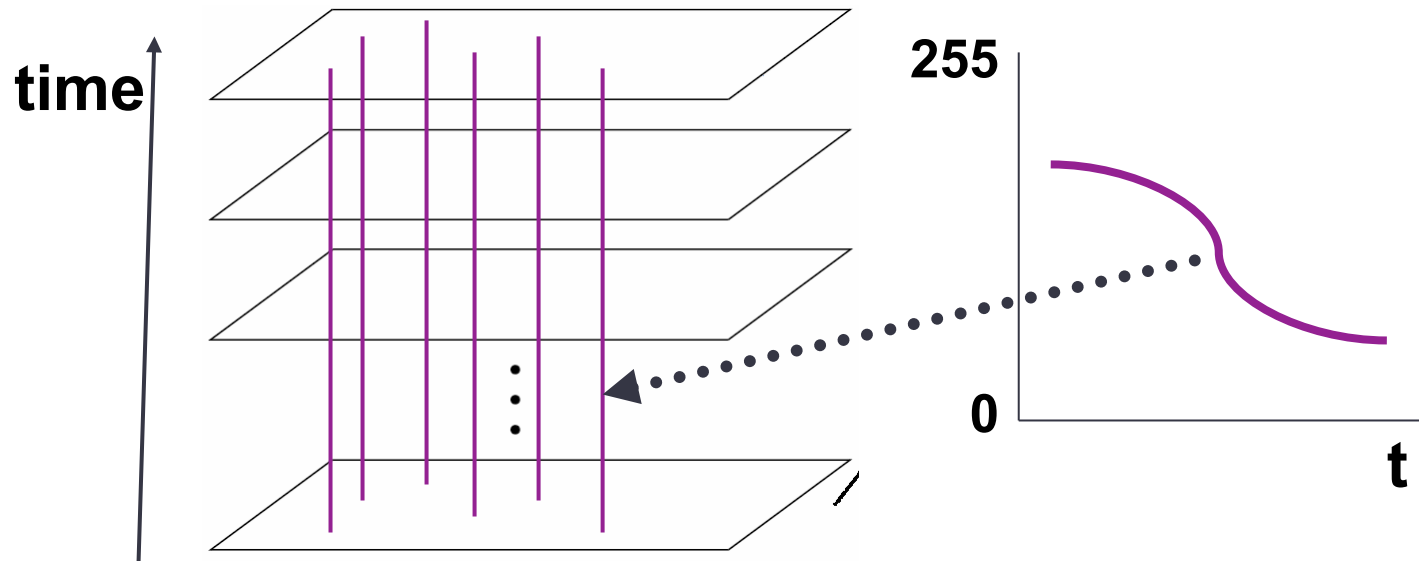
- Given Images I_1 and I_2
- Determine Feature at (x_0, y_0) in I_1
- Window W at (x_0, y_0)
- **Compute displacement in image (u, v)**
- I_t is the pixel difference between $W(I_1)$ and $W(I_2)$
- Translate W by (u, v) to W'
- Compare $W(I_1)$ and $W'(I_2)$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$



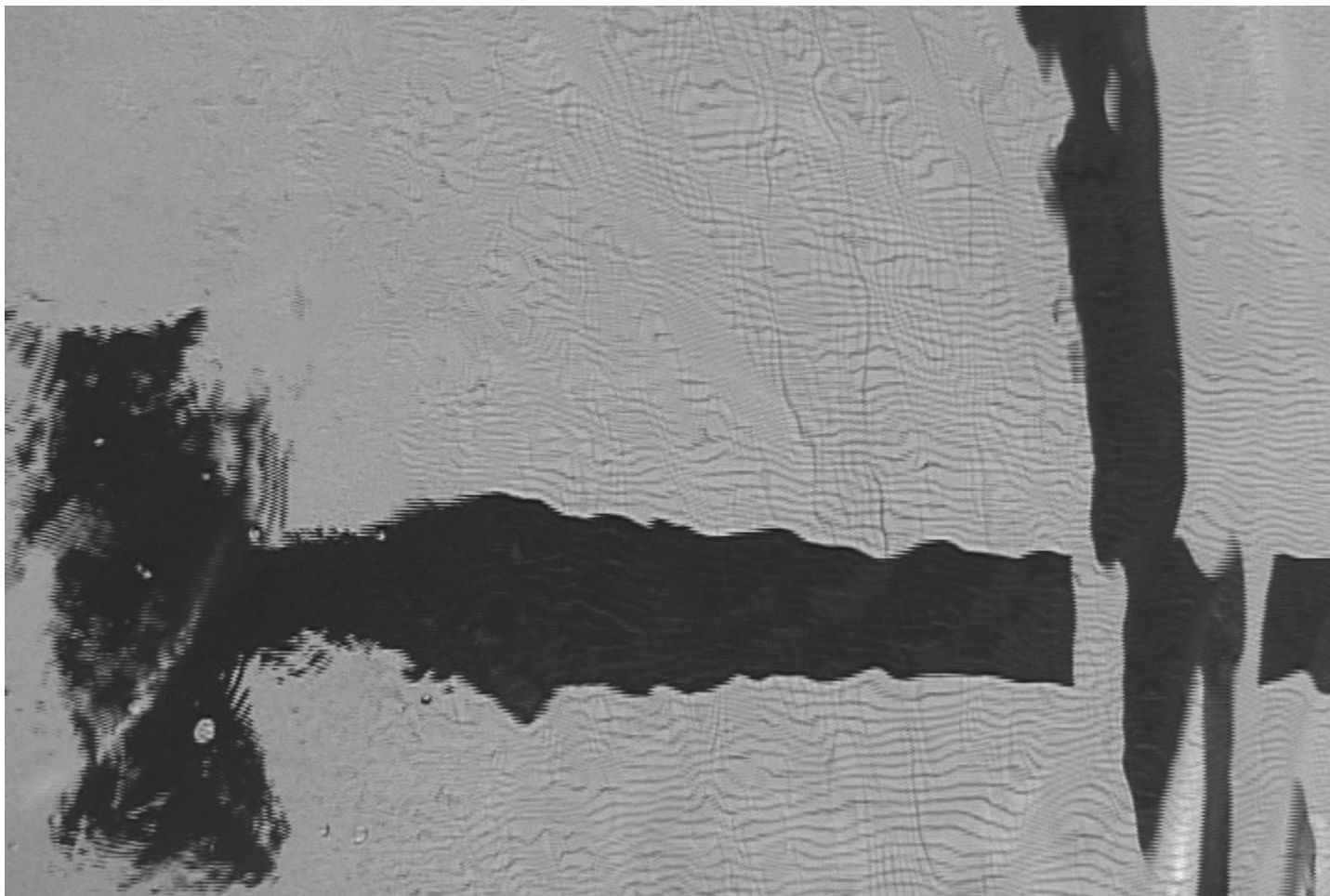
- If $W(I_1)$ and $W'(I_2)$ are too different, that means that the motion (u, v) is not quite correct and another iteration of the constant flow is applied, replacing W by W' .
- The iterations continue until (u, v) became very small or the windows match

Video as an “Image Stack”

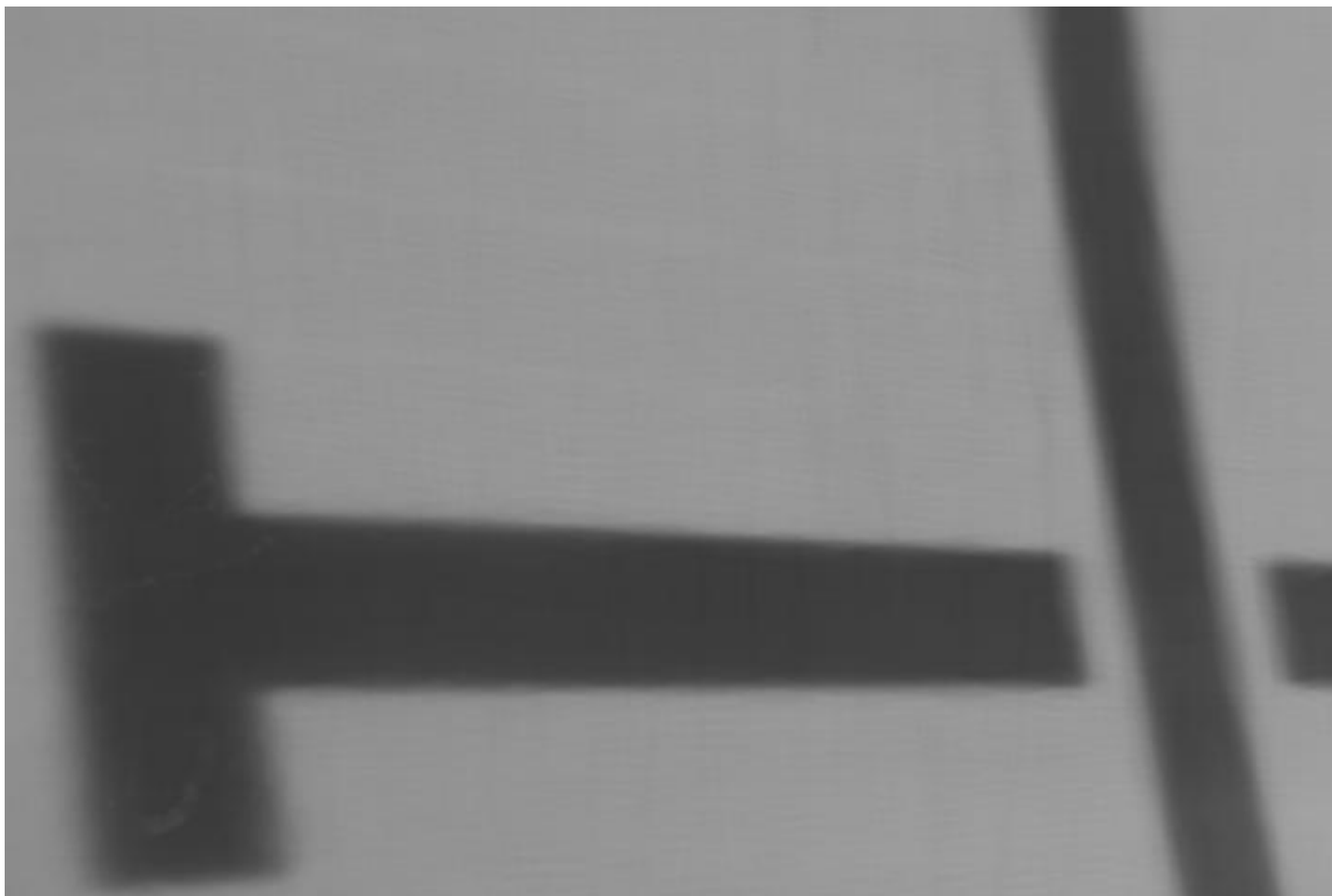


- We can look at video data as a spatio-temporal volume
 - If camera is stationary, each line through time corresponds to a single ray in space

Input Video



Average Image



Background subtraction

Background subtraction is a commonly used class of techniques for segmenting out objects of interest in a scene for applications such as:

- Surveillance
- Robot vision
- Object tracking
- Traffic applications
- Human motion capture
- Augmented reality

Background subtraction

- It involves comparing an **observed image** with an **estimate** of the image if it contained no objects of interest.
- The areas of the image plane where there is a significant difference between the observed and estimated images indicate the location of the **objects of interest**.
- The name **background subtraction** comes from the simple technique of subtracting the observed image from the estimated image and thresholding the result to generate the objects of interest.

Background subtraction

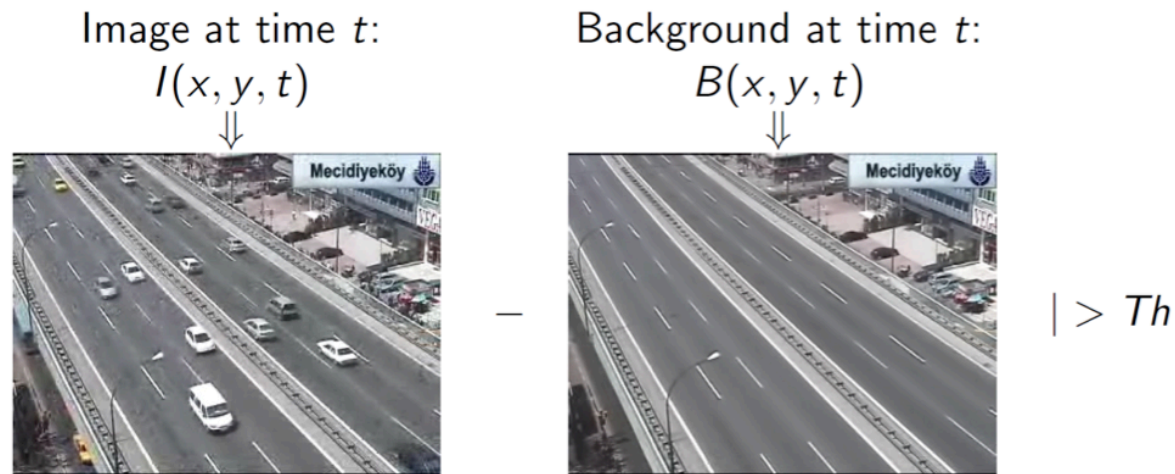
- Three important issues
 - **foreground detection** – how the object areas are distinguished from the background;
 - **background maintenance** – how the background is maintained over time;
 - **post-processing** – how the segmented object areas are postprocessed to reject false positives,



Background subtraction

- **Generic Algorithm**

- Create an image of the stationary background by averaging a long sequence
- Difference a frame from the known background frame



- Motion detection algorithms such as these only work if the camera is stationary and objects are moving against a fixed background

Background subtraction

- With **frame differencing**, background is estimated to be the previous frame. Background subtraction equation becomes:

$$B(x, y, t) = I(x, y, t - 1)$$
$$\Downarrow$$
$$|I(x, y, t) - I(x, y, t - 1)| > Th$$

- Depending on the object structure, speed, frame rate and global threshold may or may not be useful (usually not).

Background subtraction

- Another approach is to model the background using a **running average**. A pixel is marked as foreground if

$$|I_t - B_t| > \tau$$

where τ is a “predefined” threshold. The thresholding is followed by morphological closing with a 3x3 kernel and the discarding of small regions

- The background update is

$$B_{t+1} = \alpha I_t + (1 - \alpha) B_t$$








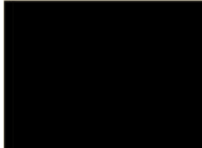













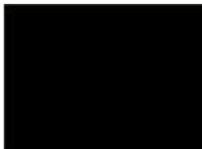






where α is kept small to prevent artificial “tails” forming behind moving objects

Background subtraction

- Two background corrections are applied:
 - If a pixel is marked as foreground for more than m of the last M frames, then the background is updated as $B_{t+1} = I_t$.
 - This correction is designed to compensate for sudden illumination changes and the appearance of static new objects.
 - If a pixel changes state from foreground to background frequently, it is masked out from inclusion in the foreground.
 - This is designed to compensate for fluctuating illumination, such as swinging branches.

J. Heikkila and O. Silven: A real-time system for monitoring of cyclists and pedestrians in: Second IEEE Workshop on Visual Surveillance, 1999

Background subtraction

Test Image							
	Chair moved	Light gradually brightened	Light just switched on	Tree Waving	Foreground covers monitor pattern	No clean background training	Interior motion undetectable
Ideal Foreground							
Adjacent Frame Difference							
Mean & Threshold							

Pros and cons

Advantages:

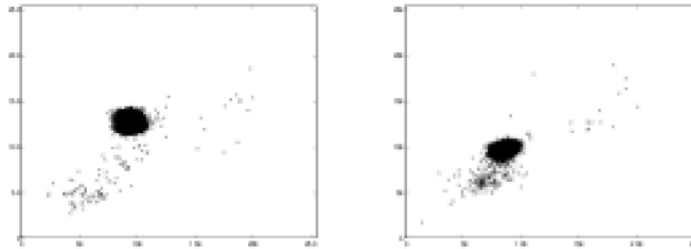
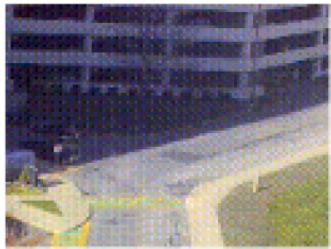
- Extremely easy to implement and use.
- Fast.
- Corresponding background models need not be constant, they change over time.

Disadvantages:

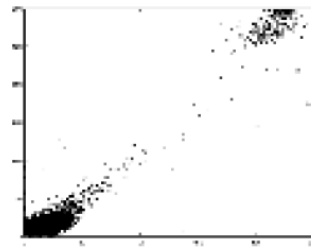
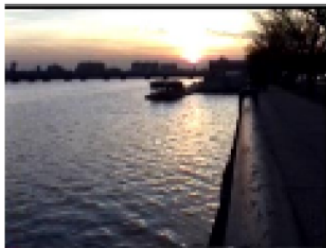
- Accuracy of frame differencing depends on object speed and frame rate
- Setting a global threshold Th

When will this basic approach fail?

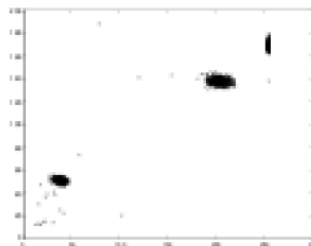
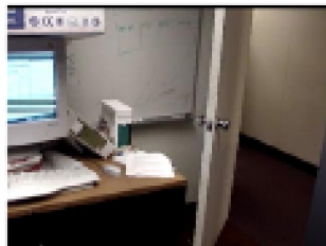
Background mixture models



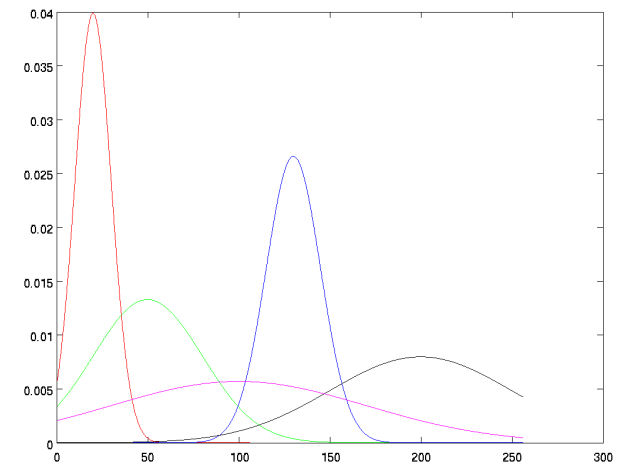
(a)



(b)



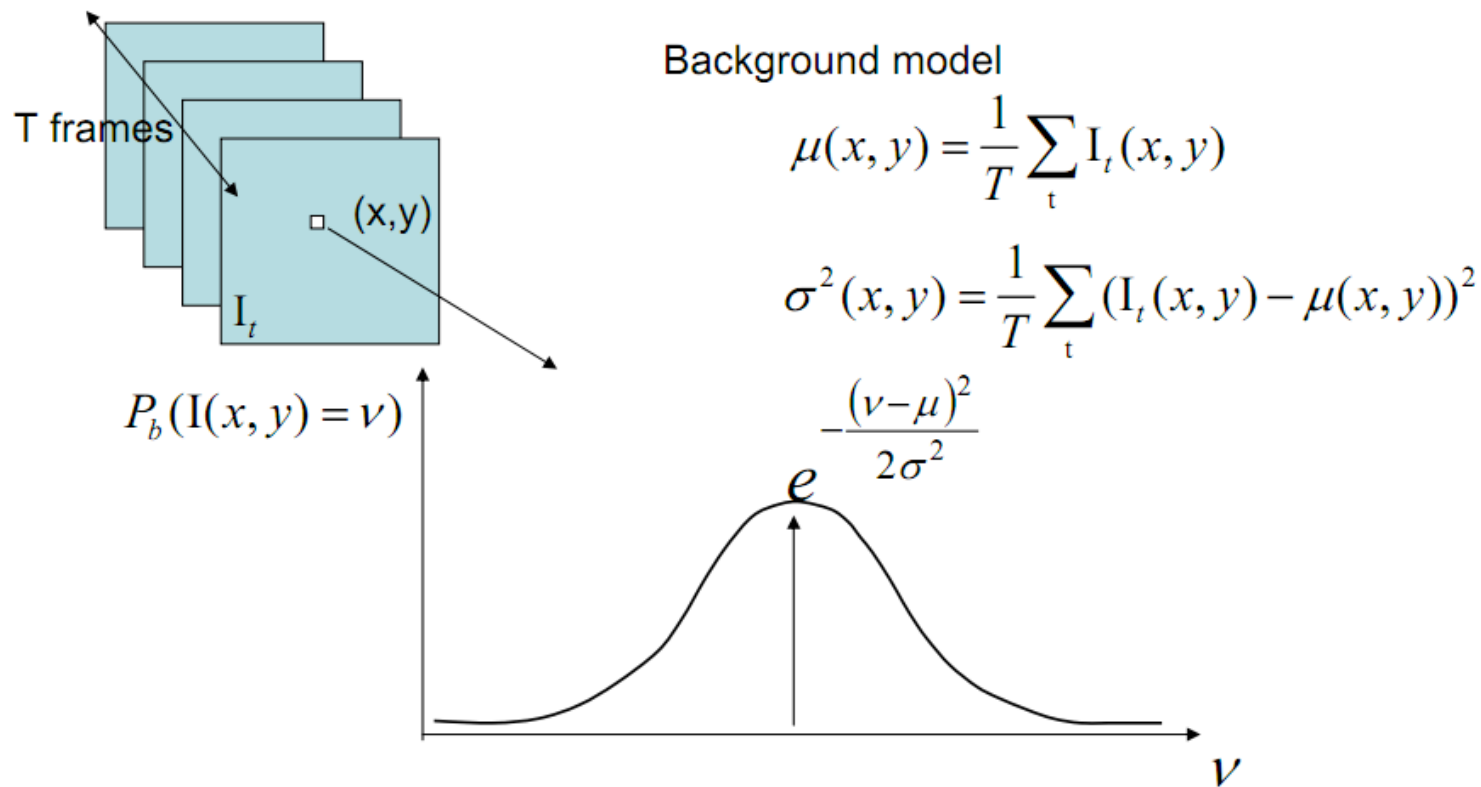
(c)



Idea: model each background pixel with a *mixture* of Gaussians; update its parameters over time.

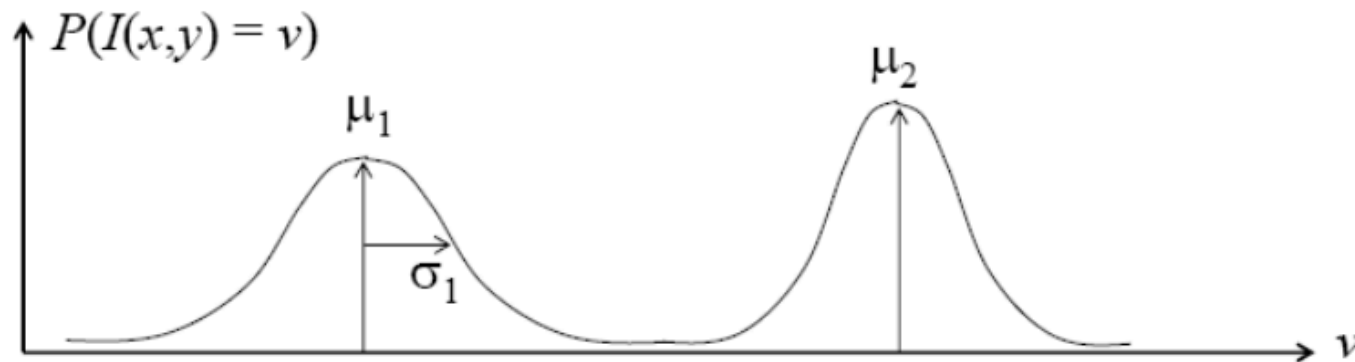
Background mixture models

- Adaptive Mixture of Gaussians:



Background mixture models

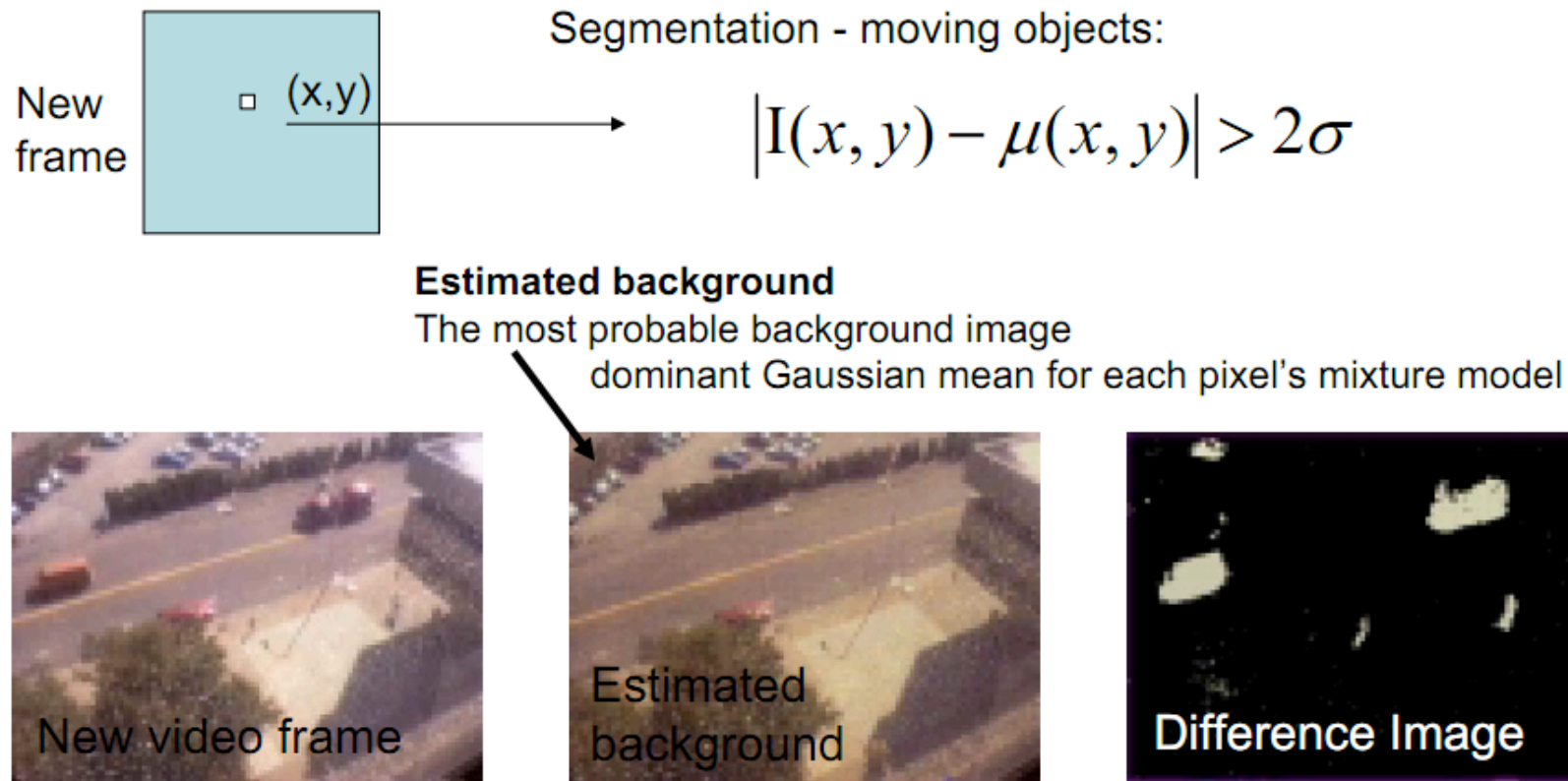
- Adaptive Mixture of Gaussians:



$$P_b(I(x,y) = v) = \sum_i w_i e^{-\frac{(v-\mu_i)^2}{2\sigma_i^2}}$$

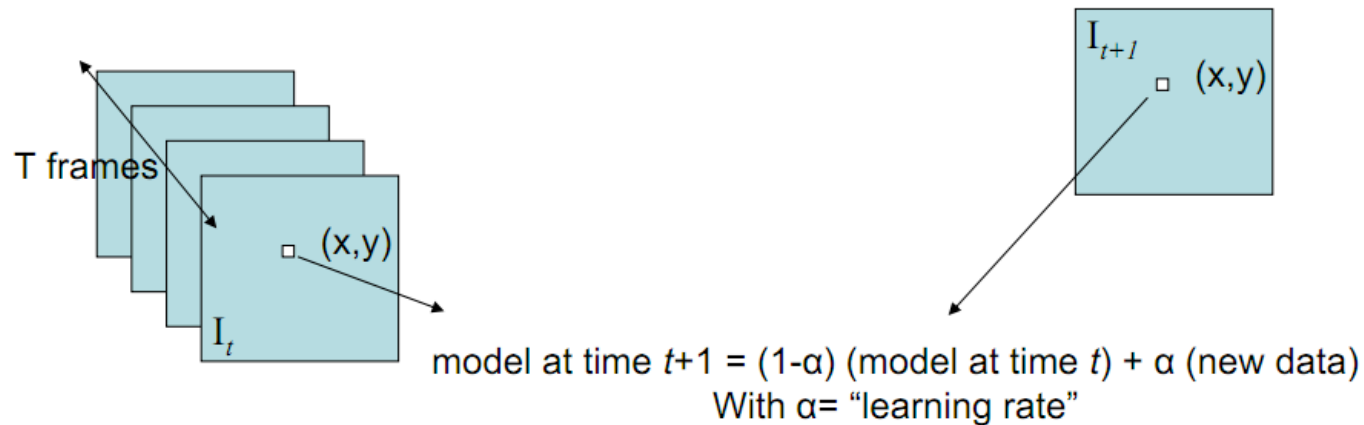
Background mixture models

- Adaptive Mixture of Gaussians:



Background mixture models

- Adaptive Mixture of Gaussians:



Update
Gaussian model

$$\rho = \alpha P_b(v_{t+1})$$

$$\mu_{t+1} = (1 - \rho)\mu_t + \rho v_{t+1}$$

$$\sigma_{t+1}^2 = (1 - \rho)\sigma_t^2 + \rho(v_{t+1} - \mu_{t+1})^2$$

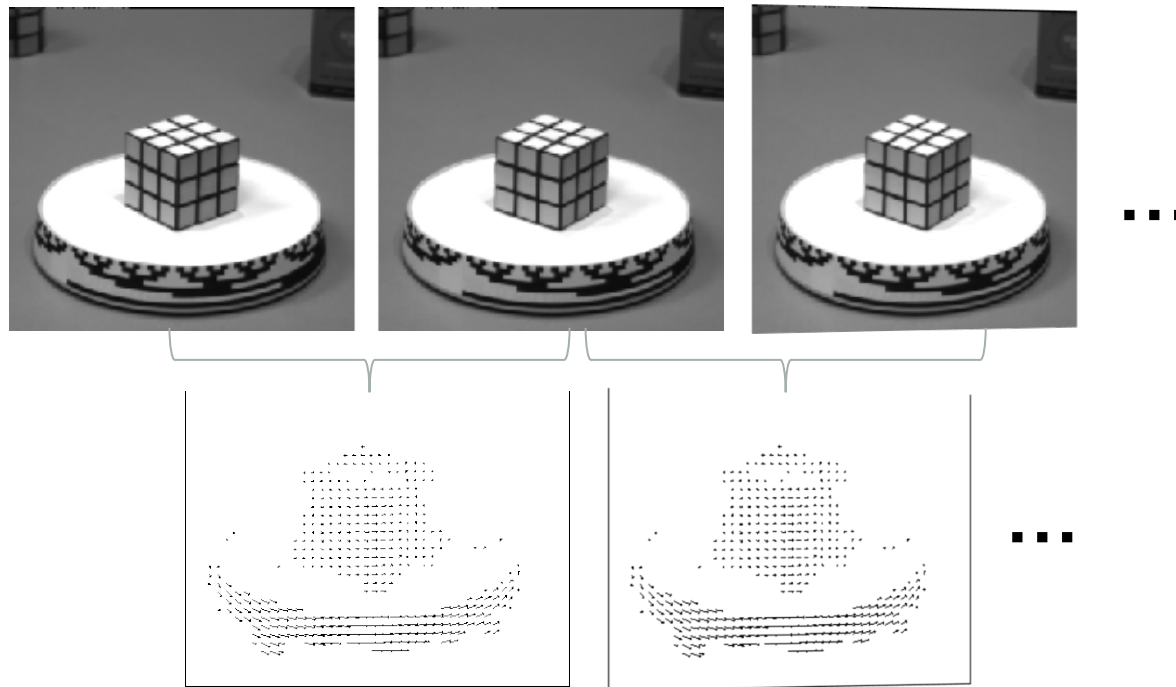
Tracking

- Object tracking is a crucial research issue in computer vision, especially for the applications where the environment is in continuous changing:
 - Robot Vision
 - mobile robot navigation,
 - applications that must deal with unstable grasps
 - Surveillance
 - Traffic applications
 - Human motion capture



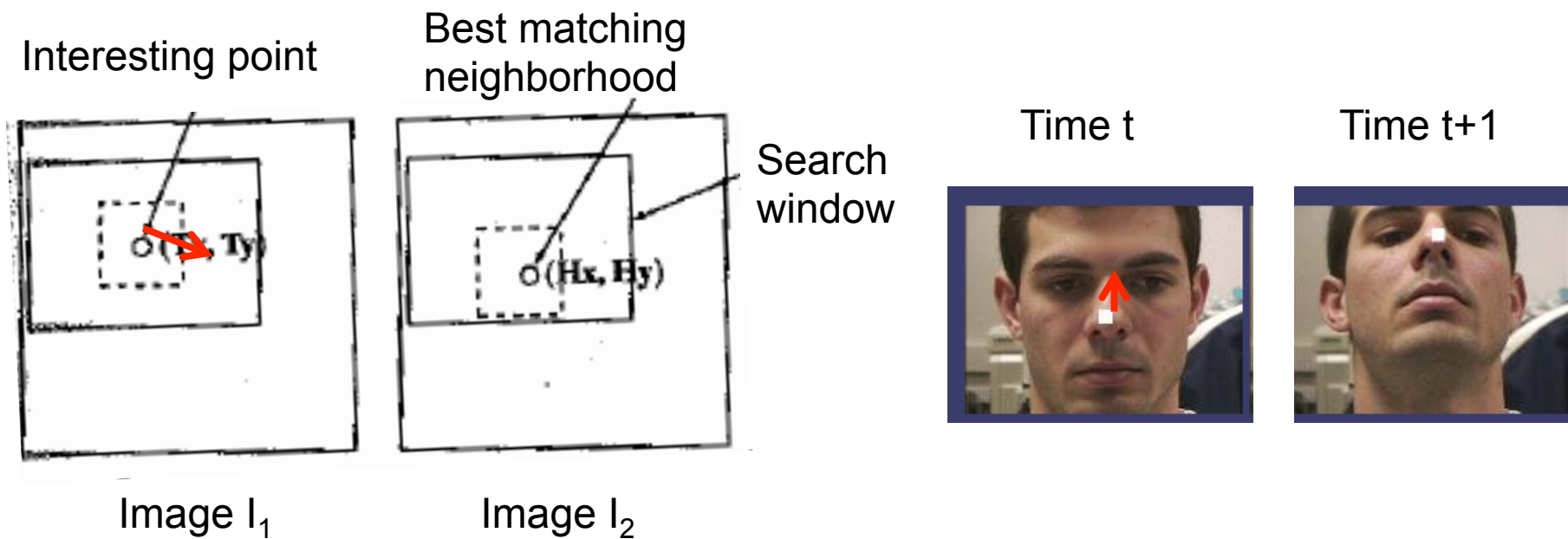
Optical flow for tracking?

If we have more than just a pair of frames, we could compute flow from one to the next:



But flow only reliable for small motions, and we may have occlusions, textureless regions that yield bad estimates anyway...

Feature-based matching for motion

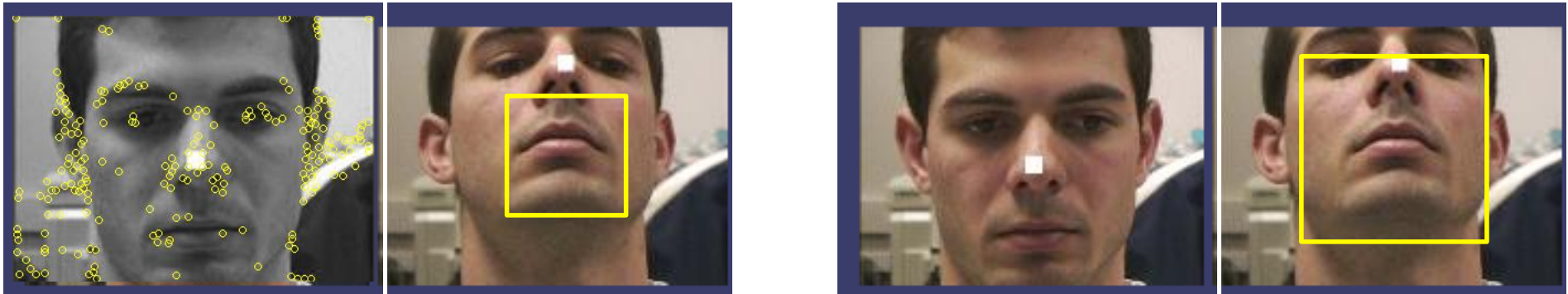


Search window is centered at the point where we last saw the feature, in image I_1 .

Best match = position where we have the highest normalized cross-correlation value.

Feature-based matching for motion

- For a discrete matching search, what are the tradeoffs of the chosen **search window** size?



- Which points to track?
 - Select interest points – e.g. corners
- Where should the search window be placed?
 - Near match at previous frame
 - **More generally, taking into account the expected *dynamics* of the object**

Detection vs. tracking



t=1



t=2

...



t=20



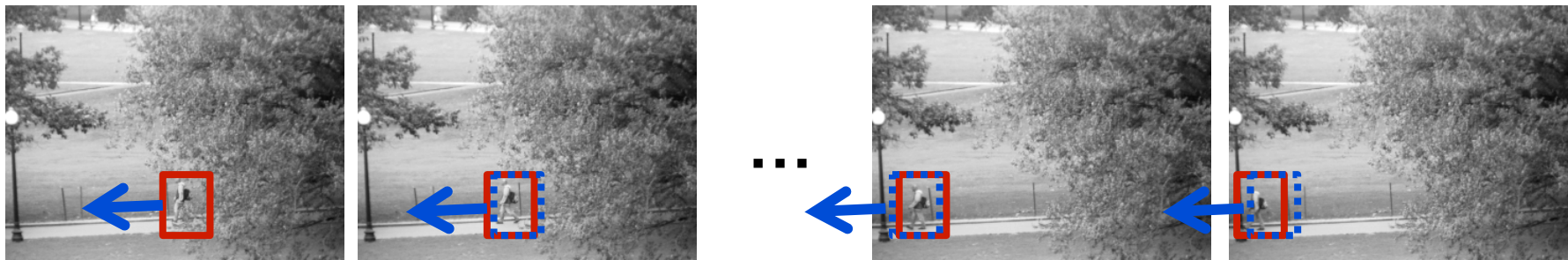
t=21

Detection vs. tracking



Detection: We detect the object independently in each frame and can record its position over time, e.g., based on blob's centroid or detection window coordinates

Detection vs. tracking



Tracking with *dynamics*: We use image measurements to estimate the position of the object, but also incorporate position predicted by dynamics, i.e., our expectation of object's motion pattern.

Tracking with dynamics

- Use model of expected motion to *predict* where objects will occur in next frame, even before seeing the image.
- **Intent:**
 - Do less work looking for the object, restrict the search.
 - Get improved estimates since measurement noise is tempered by smoothness, dynamics priors.
- **Assumption:** continuous motion patterns:
 - Camera is not moving instantly to new viewpoint
 - Objects do not disappear and reappear in different places in the scene
 - Gradual change in pose between camera and scene

Tracking as inference

- The *hidden state* consists of the true parameters we care about, denoted X .
- The *measurement* is our noisy observation that results from the underlying state, denoted Y .
- At each time step, state changes (from X_{t-1} to X_t) and we get a new observation Y_t .

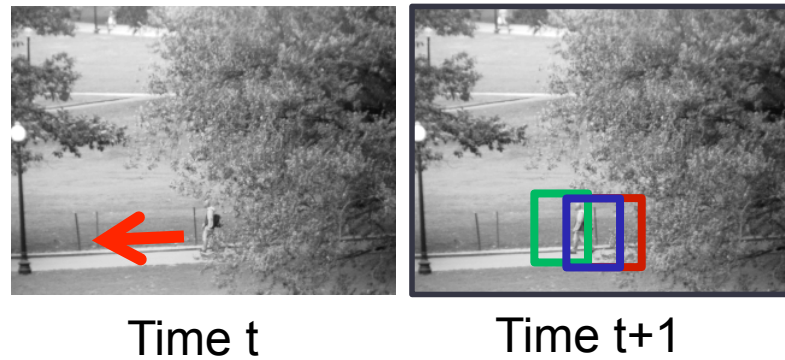
Hidden state : parameters of interest

Measurement : what we get to directly observe

Tracking as inference

- The *hidden state* consists of the true parameters we care about, denoted X .
- The *measurement* is our noisy observation that results from the underlying state, denoted Y .
- At each time step, state changes (from X_{t-1} to X_t) and we get a new observation Y_t .
- **Our goal: recover most likely state X_t given**
 - All observations seen so far.
 - Knowledge about dynamics of state transitions.

Tracking as inference: intuition



Belief

Measurement

Corrected prediction

Independence assumptions

- Only immediate past state influences current state

$$P(X_t | X_0, \dots, X_{t-1}) = P(X_t | X_{t-1})$$

dynamics model

- Measurement at time t depends on current state

$$P(Y_t | X_0, Y_0, \dots, X_{t-1}, Y_{t-1}, X_t) = P(Y_t | X_t)$$

observation model

Tracking as inference

- Prediction:

- Given the measurements we have seen **up to** this point, what state should we predict?

$$P(X_t | y_0, \dots, y_{t-1})$$

- Correction:

- Now given the **current** measurement, what state should we predict?

$$P(X_t | y_0, \dots, y_t)$$

Questions

- How to represent the known dynamics that govern the changes in the states?
- How to represent relationship between state and measurements, plus our uncertainty in the measurements?
- How to compute each cycle of updates?

Representation: We'll consider the class of *linear* dynamic models, with associated Gaussian pdfs.

Updates: via the Kalman filter.

Linear dynamic model

- Describe the *a priori* knowledge about
 - System dynamics model: represents evolution of state over time.

$$\mathbf{x}_t \sim N(\mathbf{D}\mathbf{x}_{t-1}; \Sigma_d)$$

$n \times 1$ $n \times n$ $n \times 1$

- Measurement model: at every time step we get a noisy measurement of the state.

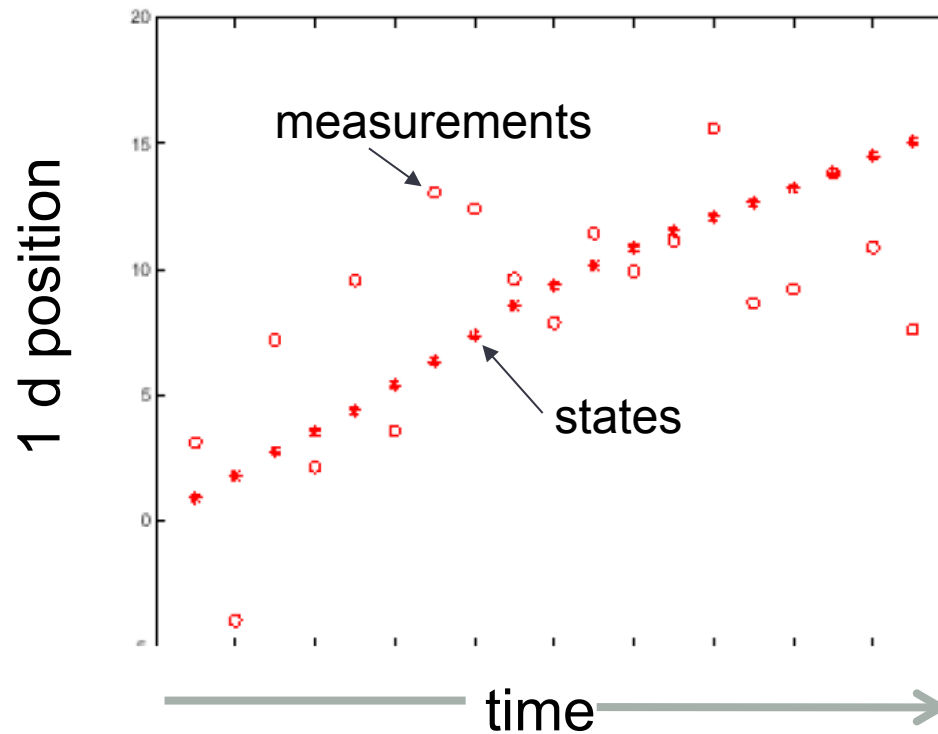
$$\mathbf{y}_t \sim N(\mathbf{M}\mathbf{x}_t; \Sigma_m)$$

$m \times 1$ $m \times n$ $n \times 1$

Example: Constant velocity (1D points)



1 d position



Example: Constant velocity (1D points)

$$\mathbf{x}_t \sim N(\mathbf{D}\mathbf{x}_{t-1}; \Sigma_d)$$

$$\mathbf{y}_t \sim N(\mathbf{M}\mathbf{x}_t; \Sigma_m)$$

- State vector: position p and velocity v

$$\mathbf{x}_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \quad p_t =$$

$$x_t = D_t x_{t-1} + noise =$$

- Measurement is position only

$$y_t = Mx_t + noise =$$

Questions

- How to represent the known dynamics that govern the changes in the states?
- How to represent relationship between state and measurements, plus our uncertainty in the measurements?
- How to compute each cycle of updates?

Representation: We'll consider the class of *linear* dynamic models, with associated Gaussian pdfs.

Updates: via the Kalman filter.

Kalman filter

- Method for tracking linear dynamical models in Gaussian noise
- The predicted/corrected state distributions are Gaussian
 - Only need to maintain the mean and covariance
 - The calculations are easy

Kalman filter

Know corrected state from previous time step, and all measurements up to the current one → Predict distribution over next state.

Know prediction of state, and next measurement → Update distribution over current state.

Receive measurement

Time update
("Predict")

Measurement update
("Correct")

$$P(X_t | y_0, \dots, y_{t-1})$$

$$P(X_t | y_0, \dots, y_t)$$

Mean and std. dev. of predicted state:

$$\mu_t^-, \sigma_t^-$$

Time advances: $t++$

Mean and std. dev. of corrected state:

$$\mu_t^+, \sigma_t^+$$

1D Kalman filter: Prediction

- Have linear dynamic model defining predicted state evolution, with noise

$$X_t \sim N(dx_{t-1}, \sigma_d^2)$$

- Want to estimate predicted distribution for next state

$$P(X_t | y_0, \dots, y_{t-1}) = N(\mu_t^-, (\sigma_t^-)^2)$$

- Update the mean:

$$\mu_t^- = d\mu_{t-1}^+$$

- Update the variance:

$$(\sigma_t^-)^2 = \sigma_d^2 + (d\sigma_{t-1}^+)^2$$

1D Kalman filter: Correction

- Have linear model defining the mapping of state to measurements: $Y_t \sim N(mx_t, \sigma_m^2)$

- Want to estimate corrected distribution given latest meas.:

$$P(X_t | y_0, \dots, y_t) = N(\mu_t^+, (\sigma_t^+)^2)$$

- Update the mean:

$$\mu_t^+ = \frac{\mu_t^- \sigma_m^2 + m y_t (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2}$$

- Update the variance:

$$(\sigma_t^+)^2 = \frac{\sigma_m^2 (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2}$$

Prediction vs. correction

$$\mu_t^+ = \frac{\mu_t^- \sigma_m^2 + m y_t (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2} \quad (\sigma_t^+)^2 = \frac{\sigma_m^2 (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2}$$

- What if there is no prediction uncertainty $(\sigma_t^- = 0)$?

$$\mu_t^+ = \mu_t^- \quad (\sigma_t^+)^2 = 0$$

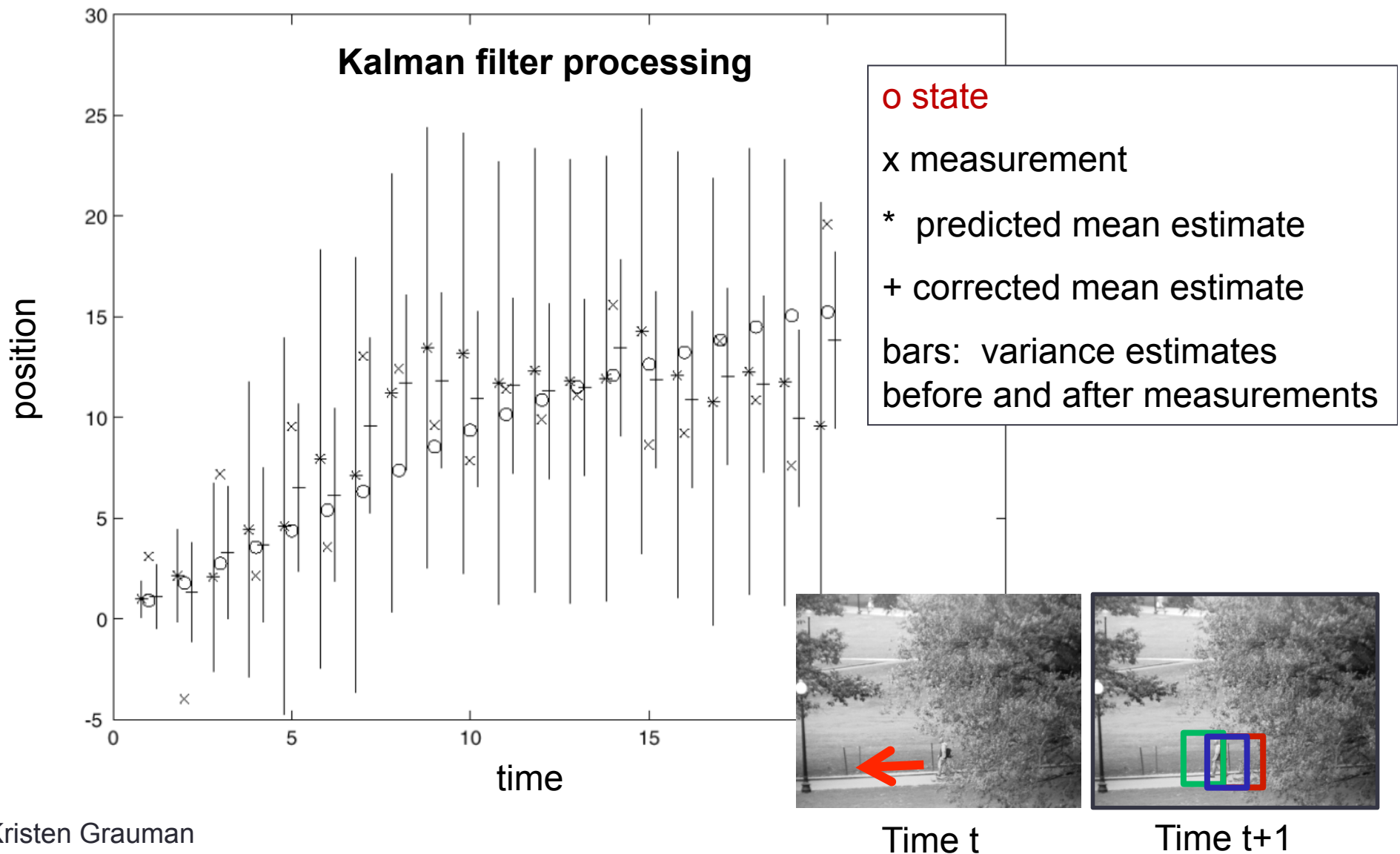
The measurement is ignored!

- What if there is no measurement uncertainty $(\sigma_m = 0)$?

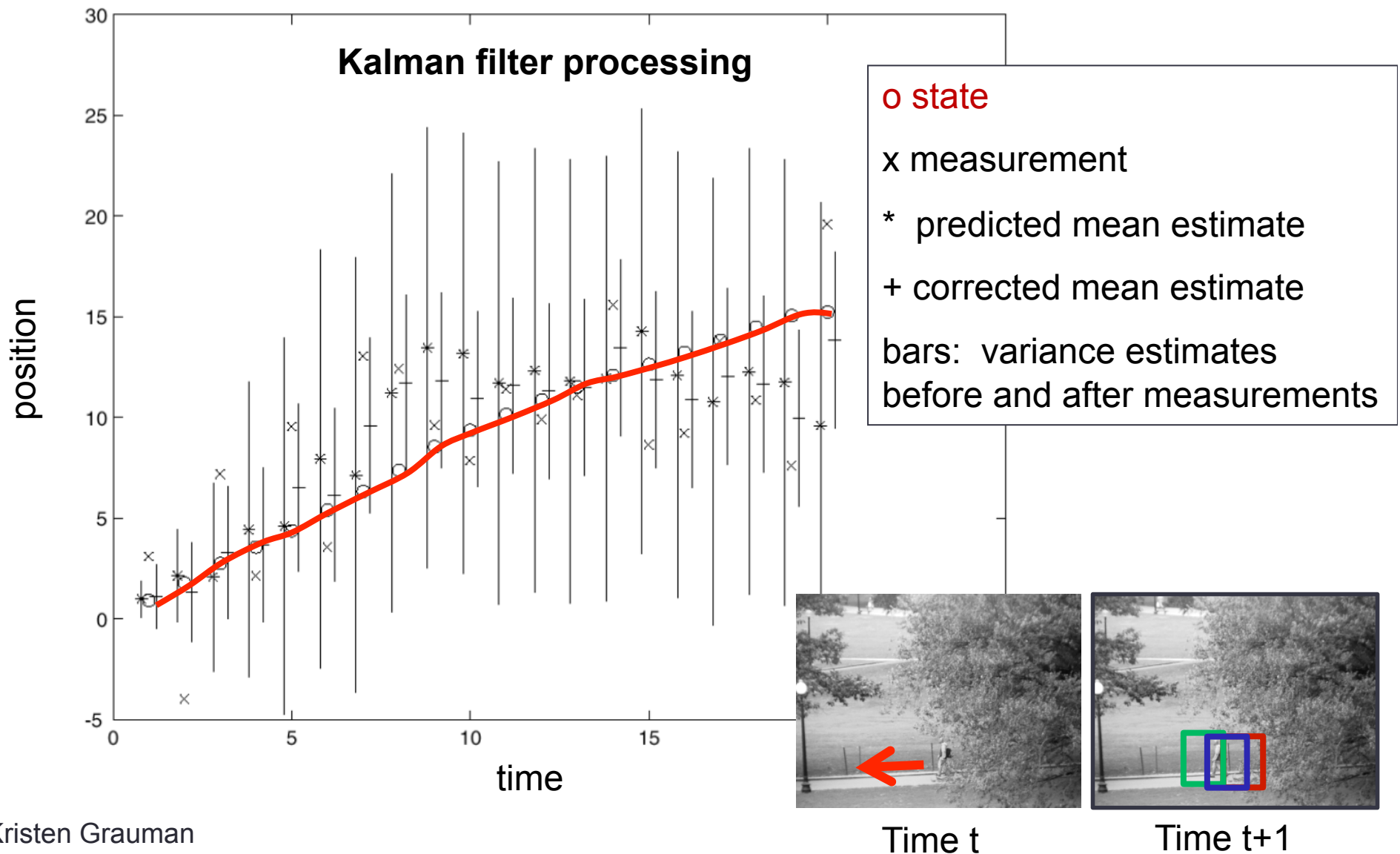
$$\mu_t^+ = \frac{y_t}{m} \quad (\sigma_t^+)^2 = 0$$

The prediction is ignored!

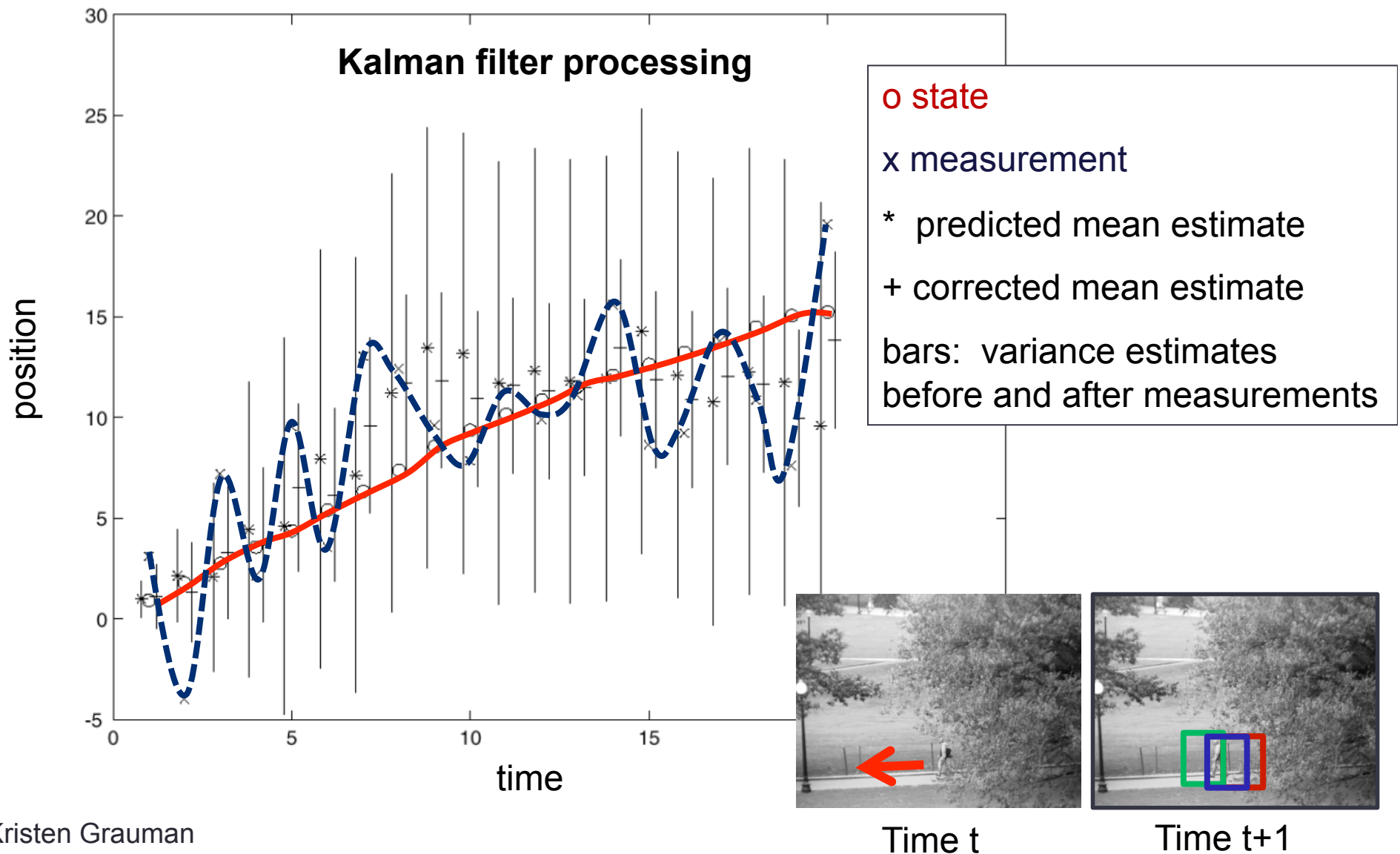
Constant velocity model



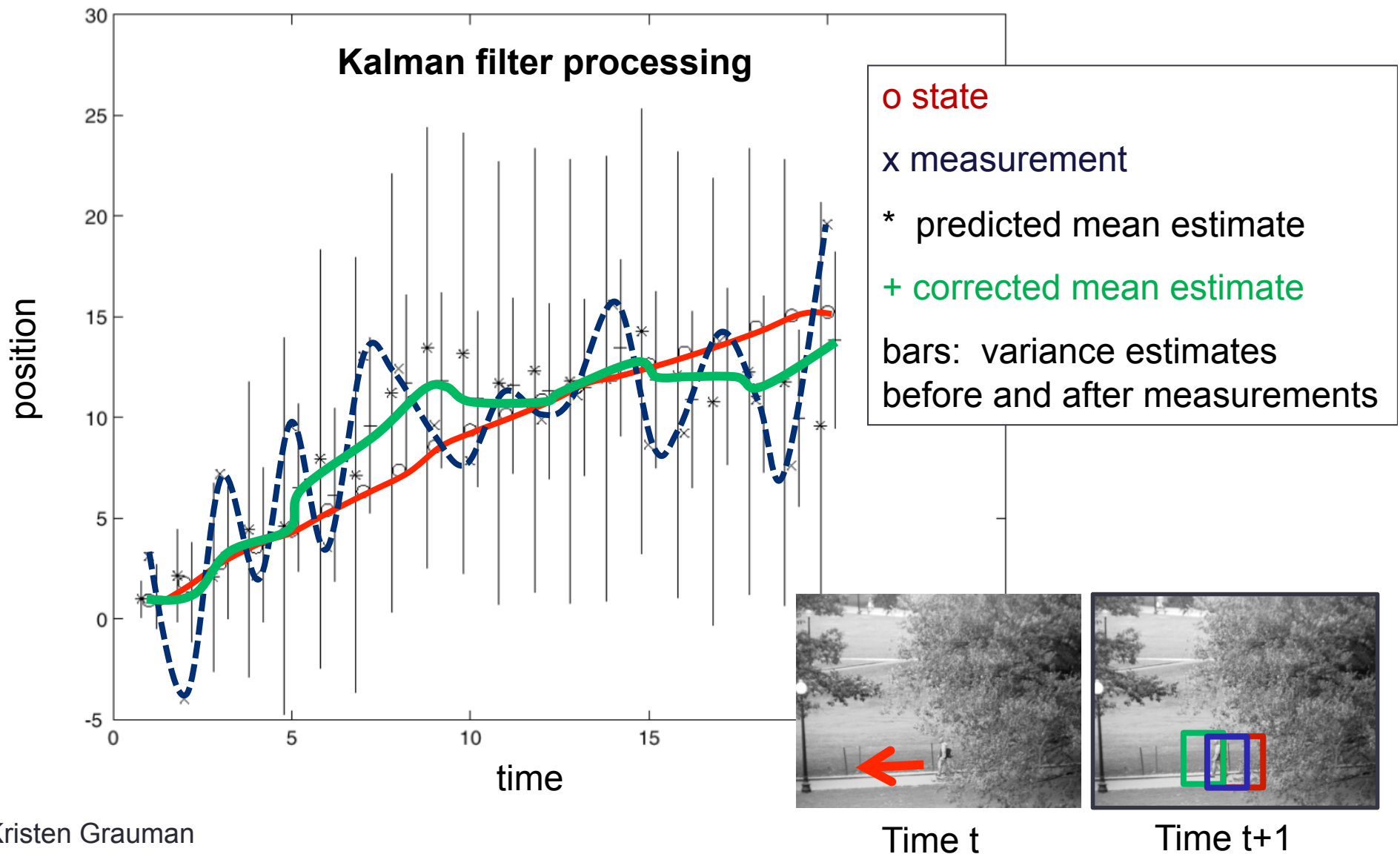
Constant velocity model



Constant velocity model



Constant velocity model



Tracking: issues

- **Initialization**
 - Often done manually
 - Background subtraction, detection can also be used
- **Data association**, multiple tracked objects
 - Occlusions, clutter

Tracking: issues

- **Initialization**

- Often done manually
- Background subtraction, detection can also be used

- **Data association**, multiple tracked objects

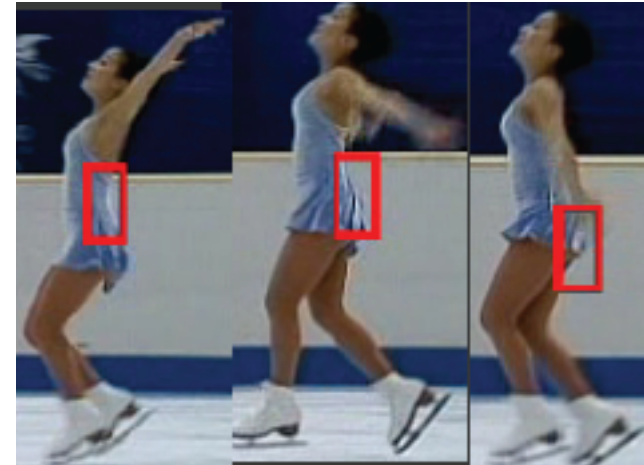
- Occlusions, clutter
- Which measurements go with which tracks?



Tracking: issues

- **Initialization**
 - Often done manually
 - Background subtraction, detection can also be used
- **Data association**, multiple tracked objects
 - Occlusions, clutter
- **Deformable** and articulated objects
- **Constructing accurate models** of dynamics
 - E.g., Fitting parameters for a linear dynamics model
- **Drift**
 - Accumulation of errors over time

Drift



D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](#). PAMI 2007.