Tracking

Vinay P. Namboodiri

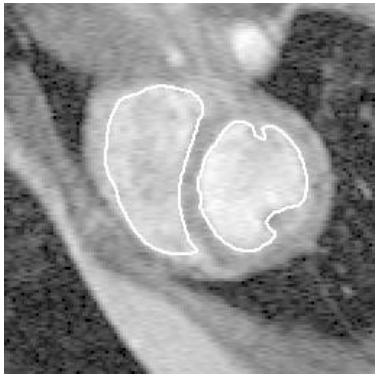
- Slide credit to Kristen Grauman

Tracking: some applications

Body pose tracking, activity recognition

Censusing a bat population

Video-based interfaces



Medical apps

Surveillance

Gmail Motion BETA

A new way to communicate

The mouse and keyboard were invented before the Internet even existed. Since then, countless technological advancements have allowed for much more efficient human computer interaction. Why then do we continue to use outdated technology? Introducing Gmail Motion -- now you can control Gmail with your body.

Try Gmail Motion

Don't have Gmail? Create an account

Easy to learn
Simple and intuitive gestures

Improved productivity
In and out of your email up to 12% faster

Increased physical activity
Get out of that chair and start moving today

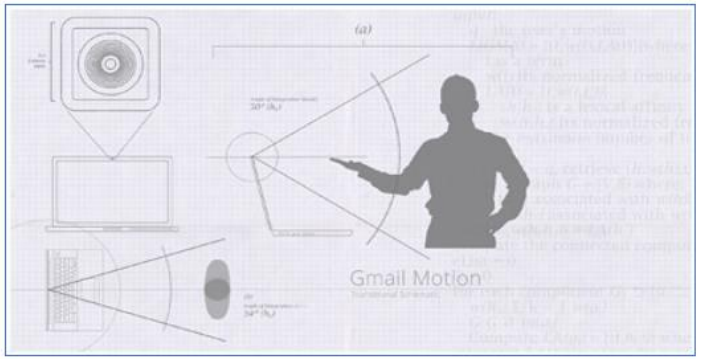
Overview

Paralanguage expert

Movement specialist

How it works

Gmail Motion uses your computer's built-in webcam and Google's patented spatial tracking technology to detect your movements and translate them into meaningful characters and commands. Movements are designed to be simple and intuitive for people of all skill levels.



Google docs Motion BETA

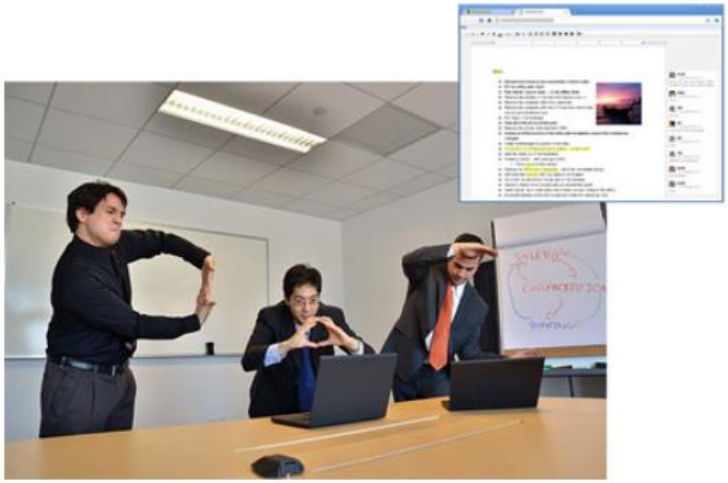
A new way to collaborate

Coming soon! Google Docs Motion BETA will introduce a new way to collaborate -- using your body:

Gesture recognition
Over 10,000 supported gestures

Emotion detection
Writing tone corresponds with emotions

Intuitive
Start being more productive in a few minutes



Work together on business proposals

Documents

Collaborate in real time using Google Docs Motion BETA. Building on the technology developed for Gmail Motion BETA, multiple computers can triangulate position to track each collaborator's inputs. Every feature will be supported, from brand new discussions to formatting, tables and translation.

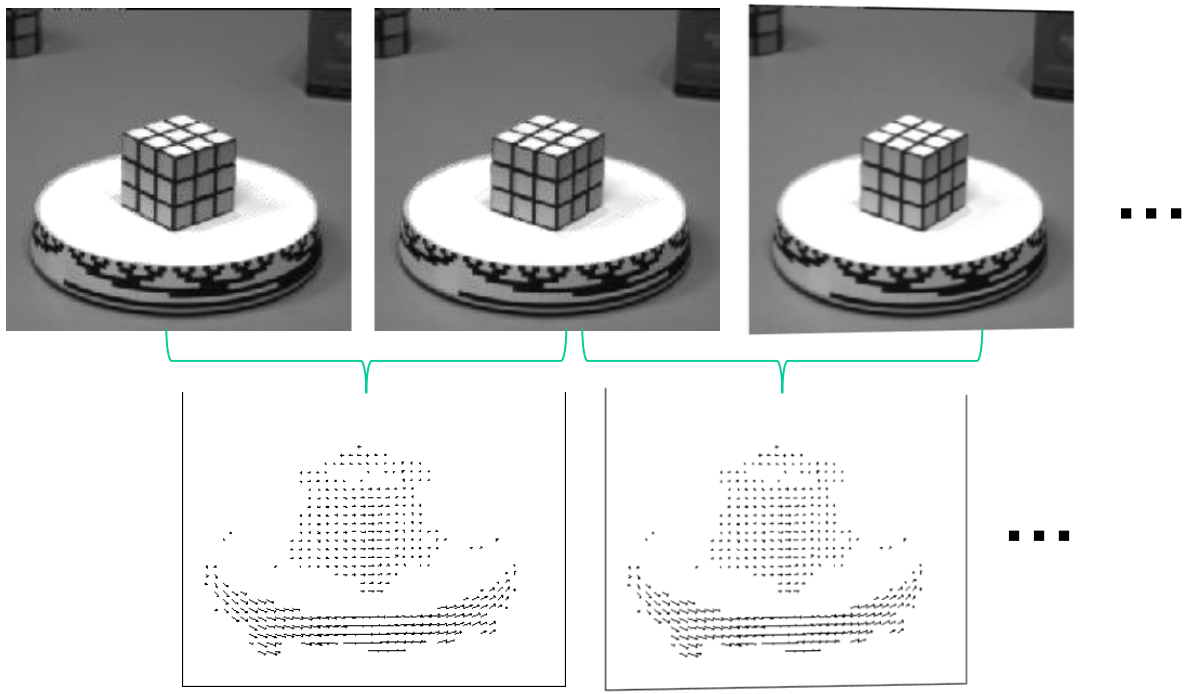
Spreadsheets

What do you get when you combine motion and Google spreadsheets? An incredible new way to create charts, add data, filter and work together. Docs Motion now makes it easy to adjust the look and feel of your charts. Pie charts are easy to make.

Why is tracking challenging?

Optical flow for tracking?

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



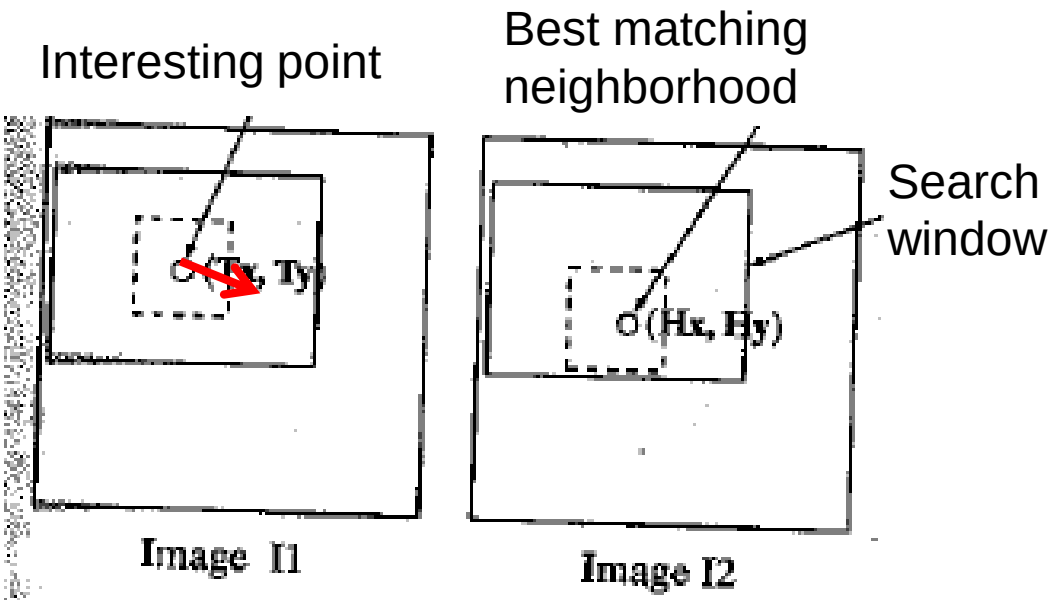
But flow only reliable for small motions, and we may have occlusions, textureless regions that yield bad estimates anyway…

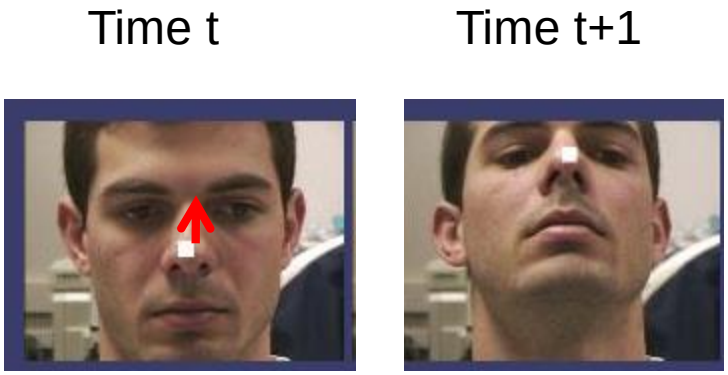
Motion estimation techniques

- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video and when image motion is small

- **Feature-based methods**
  - Extract visual features (corners, textured areas) and track them over multiple frames
  - Sparse motion fields, but more robust tracking
  - Suitable when image motion is large (10s of pixels)

Feature-based matching for motion





Search window is centered at the point where we last saw the feature, in image I1.

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

Example: A Camera Mouse

Video interface: use feature tracking as mouse replacement

- User clicks on the feature to be tracked
- Take the 15x15 pixel square of the feature
- In the next image do a search to find the 15x15 region with the highest correlation
- Move the mouse pointer accordingly
- Repeat in the background every 1/30th of a second

James Gips and Margrit Betke
http://www.bc.edu/schools/csom/eagleeyes/

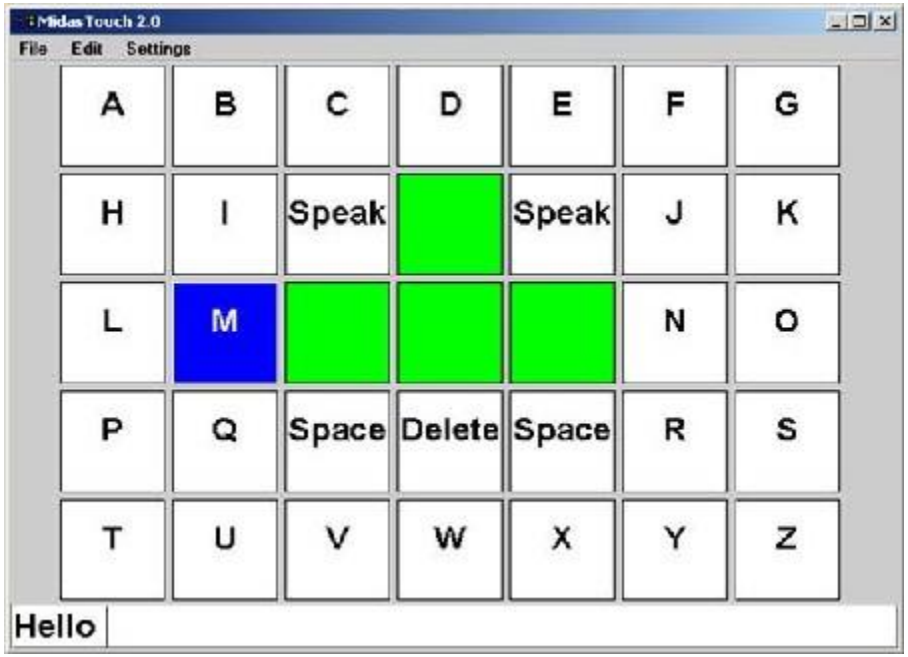
Example: A Camera Mouse

Specialized software for communication, games

James Gips and Margrit Betke
http://www.bc.edu/schools/csom/eagleeyes/

A Camera Mouse

Specialized software for communication, games



James Gips and Margrit Betke
http://www.bc.edu/schools/csom/eagleeyes/

Feature-based matching for motion

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

- Which patches 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**

Kristen Grauman

Detection vs. tracking


t=1


t=2

...


t=20


t=21

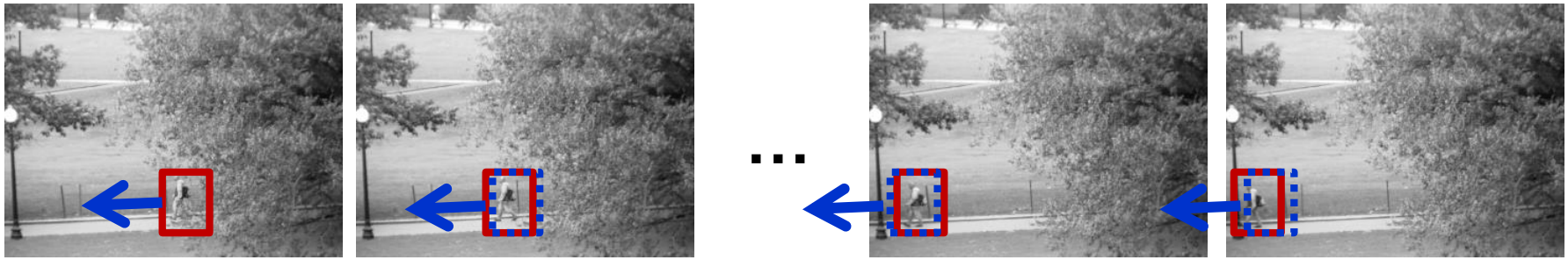
Kristen Grauman

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

Kristen Grauman

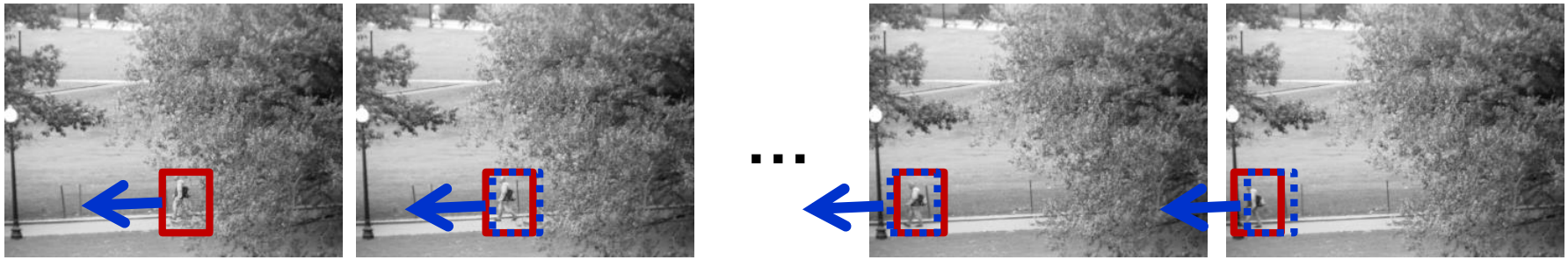
Detection vs. tracking



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

Kristen Grauman

Detection vs. tracking



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

Kristen Grauman

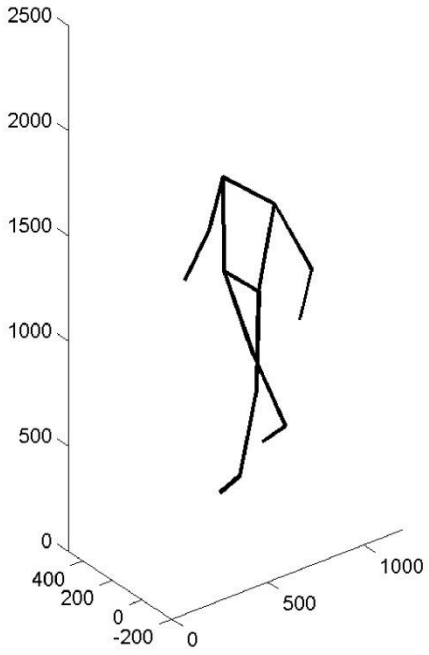
Tracking with dynamics

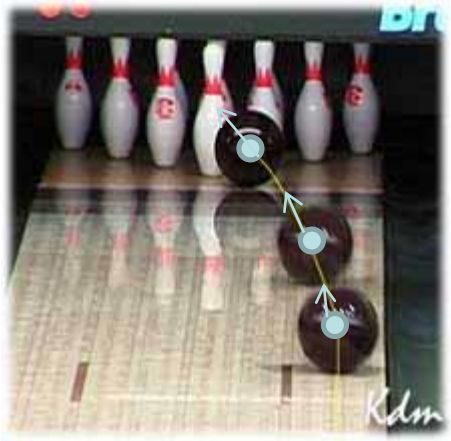
- Use model of expected motion to *predict* where objects will occur in next frame, even before seeing the image.
- **Intent**:
  - Do less work looking for the object, restrict the search.
  - Get improved estimates since measurement noise is tempered by smoothness, dynamics priors.
- **Assumption**: 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

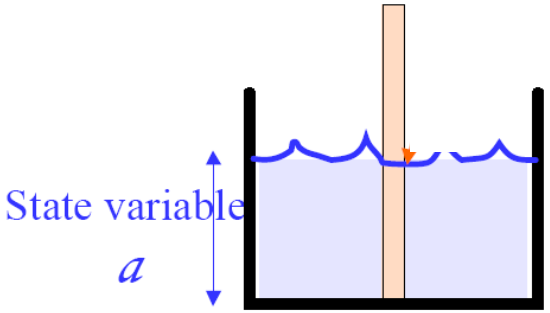
Tracking as inference

- 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.

State vs. observation







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

Time t Time t+1

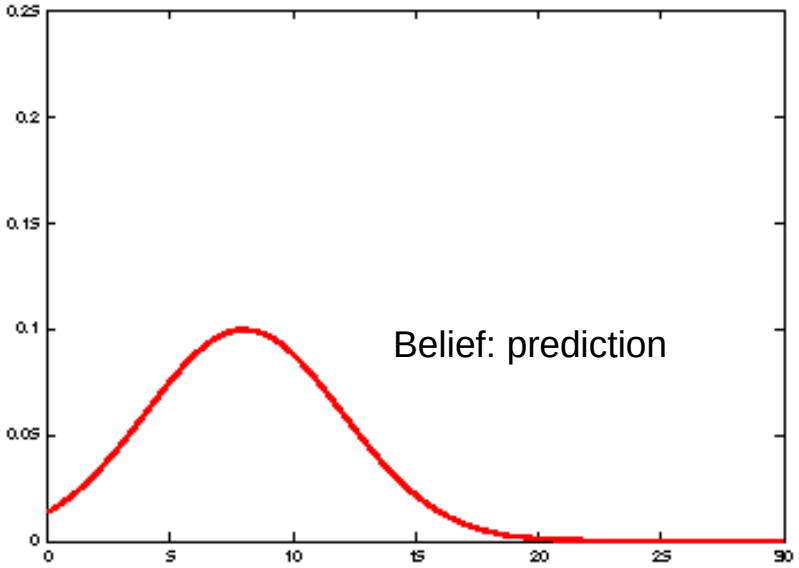
Belief

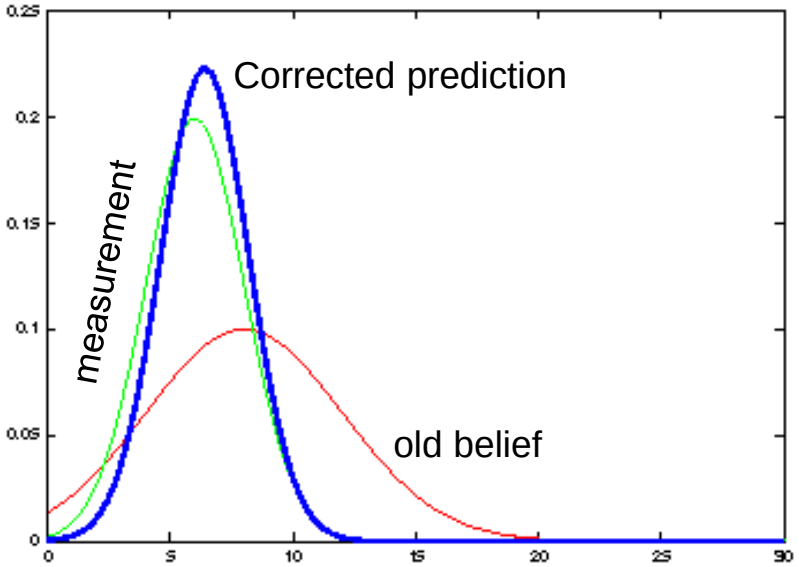
Measurement

Corrected prediction

Kristen Grauman

Tracking as inference: intuition





Time t Time t+1

Kristen Grauman

Independence assumptions

- Only immediate past state influences current state

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

dynamics model

- Measurement at time t depends on current state

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

observation model

Tracking as inference

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

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

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

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

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.

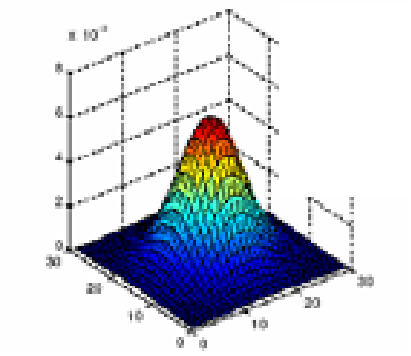
Updates: via the Kalman filter.

Notation reminder

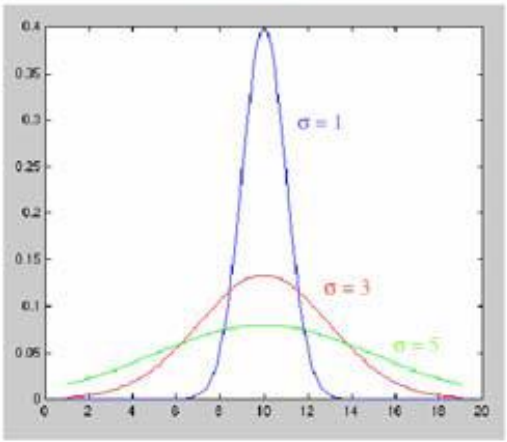
$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- Random variable with Gaussian probability distribution that has the mean vector **μ** and covariance matrix **Σ**.
- **x** and **μ** are *d*-dimensional, **Σ** is *d* x *d*.

d=2



d=1



If x is 1-d, we just have one **Σ** parameter - → the variance: σ^2

Linear dynamic model

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

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

n x 1 n x n n x 1

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

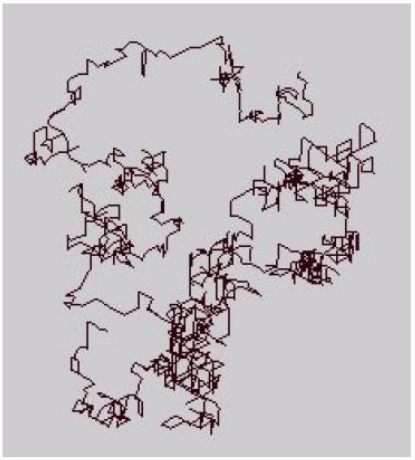
$$\mathbf{y}_t \sim N(\mathbf{M}\mathbf{x}_t; \boldsymbol{\Sigma}_m)$$

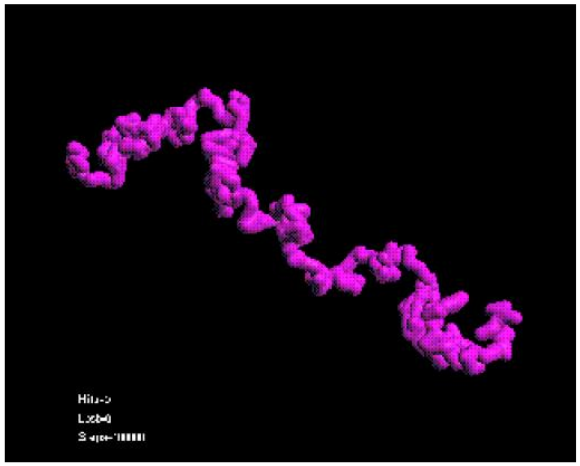
m x 1 m x n n x 1

Example: randomly drifting points

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

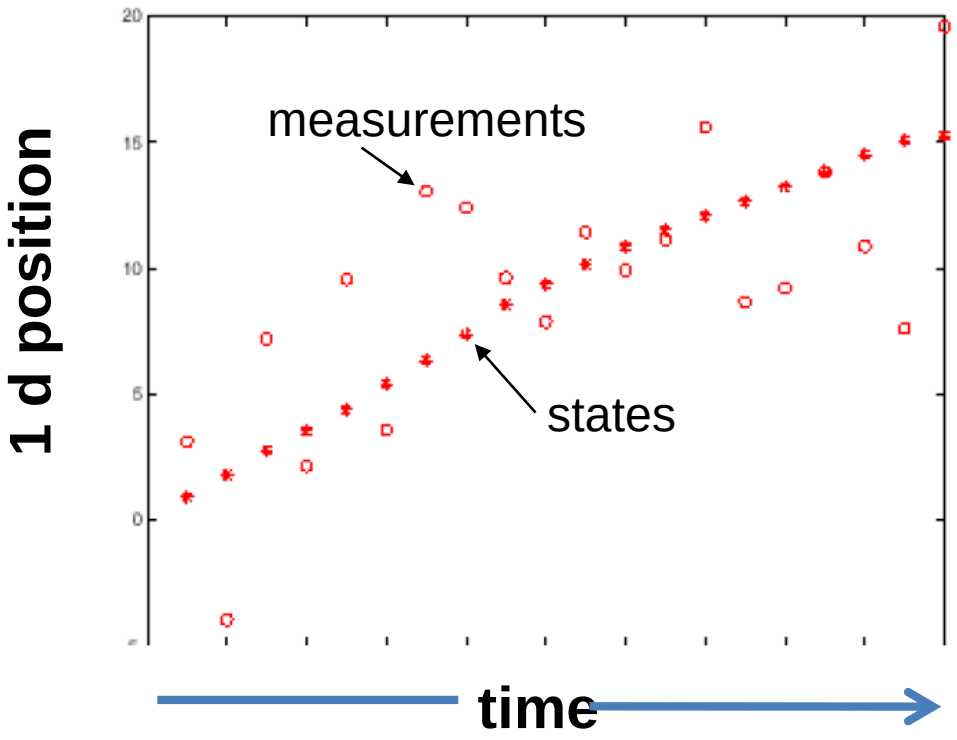
- Consider a stationary object, with state as position
- Position is constant, only motion due to random noise term.
- State evolution is described by identity matrix **D=I**





cic.nist.gov/lipman/sciviz/images/random3.gif
http://www.grunch.net/synergetics/images/rand
3.jpg

Example: Constant velocity (1D points)



Kristen Grauman

Example: Constant velocity (1D points)

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

$$\mathbf{y}_t \sim N(\mathbf{M}\mathbf{x}_t; \boldsymbol{\Sigma}_m)$$

- State vector: position *p* and velocity *v*

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \qquad p_t =$$

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

- Measurement is position only

$$y_t = M x_t + noise =$$

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.

The Kalman filter

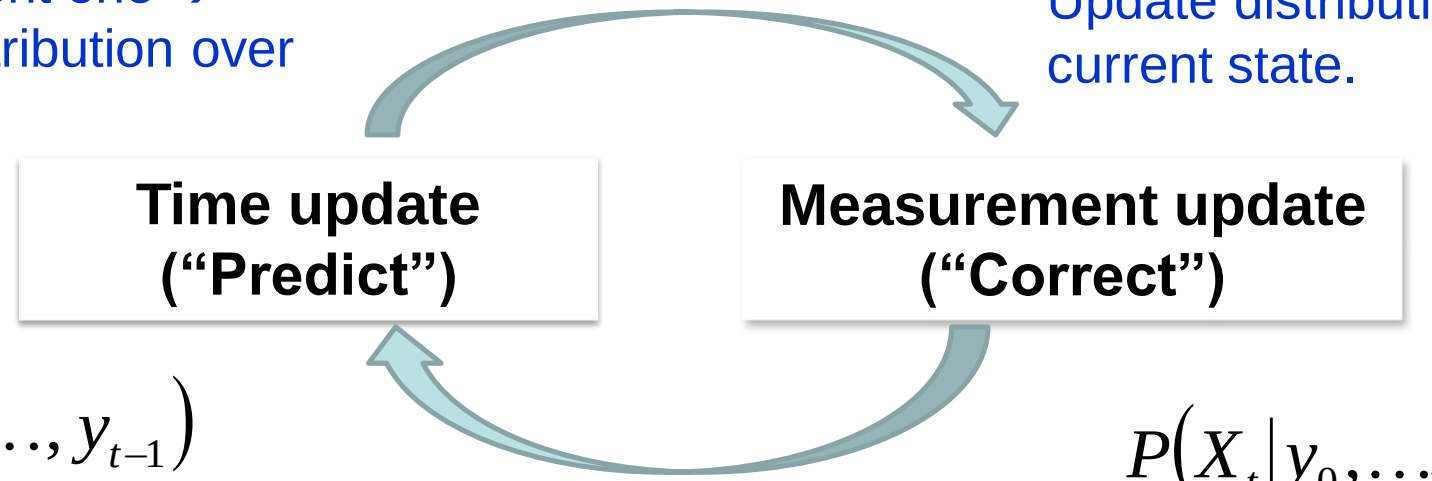
- Method for tracking linear dynamical models in Gaussian noise
- The predicted/corrected state distributions are Gaussian
  - Only need to maintain the mean and covariance
  - The calculations are easy

Kalman filter

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

Receive measurement

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



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

Mean and std. dev. of predicted state:

μ_t^-, σ_t^-

Time advances: t++

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

Mean and std. dev. of corrected state:

μ_t^+, σ_t^+

1D Kalman filter: **Prediction**

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

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

- Want to estimate predicted distribution for next state

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

- 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:

$$Y_t \sim N\left(mx_t, \sigma_m^2\right)$$

- Want to estimate corrected distribution given latest meas.:

$$P\left(X_t \middle| y_0, \ldots, y_t\right) = N\left(\mu_t^+, (\sigma_t^+)^2\right)$$

- 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

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

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

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

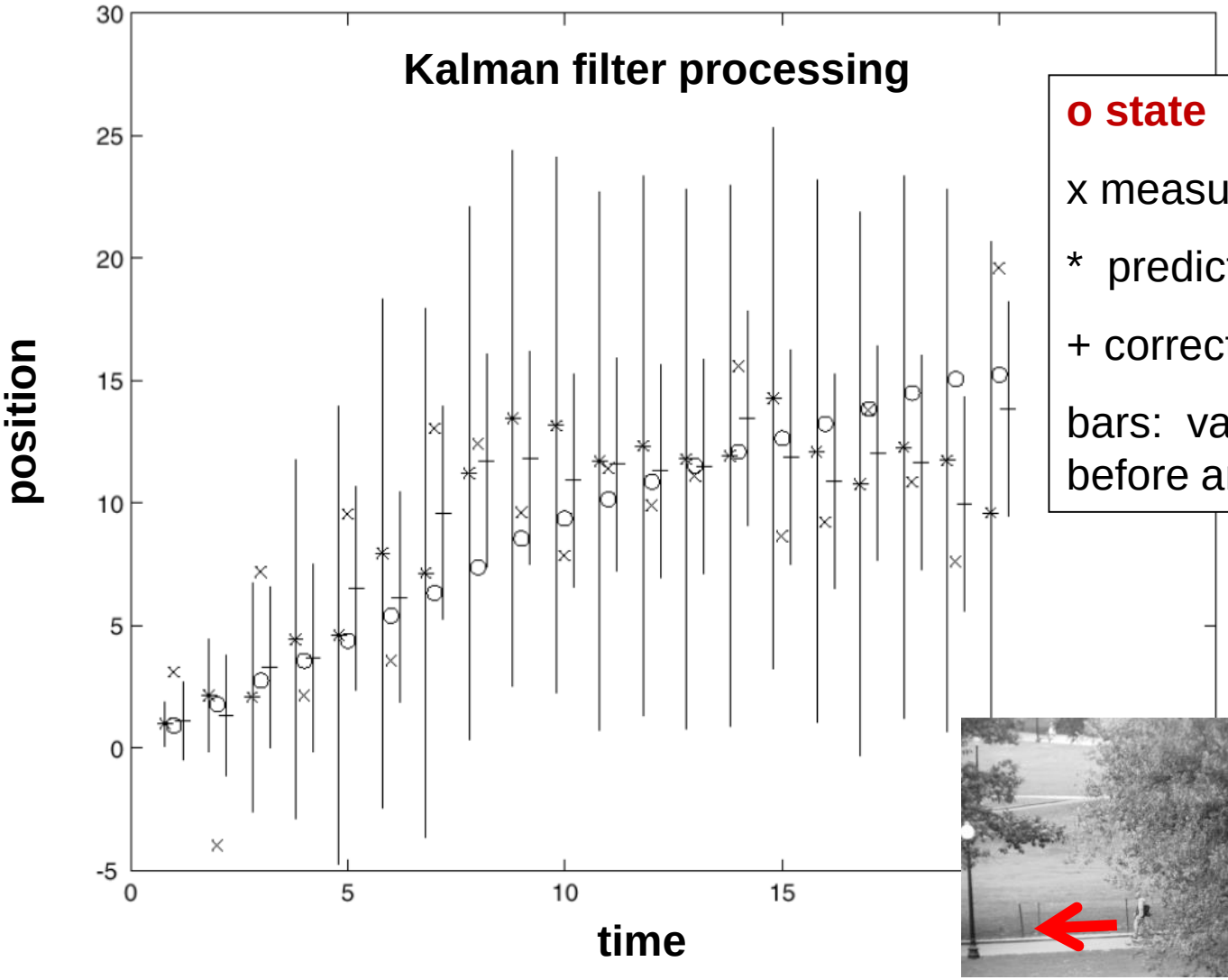
The measurement is ignored!

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

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

The prediction is ignored!

Constant velocity model



o state

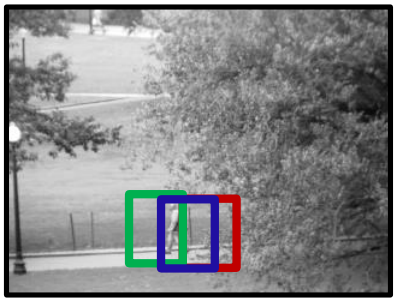
x measurement

* predicted mean estimate

\+ corrected mean estimate

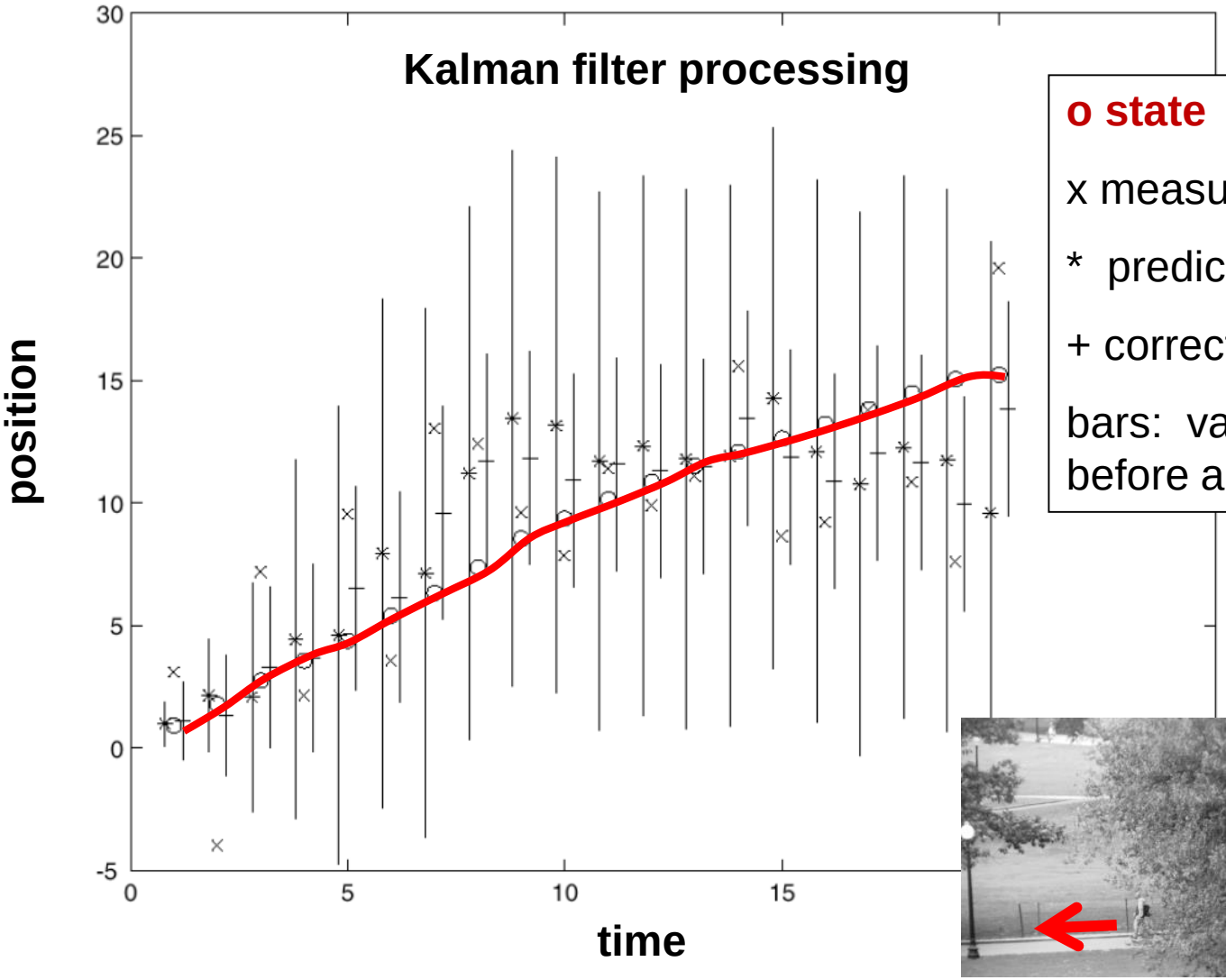
bars: variance estimates before and after measurements

Time t



Time t+1

Constant velocity model



o state

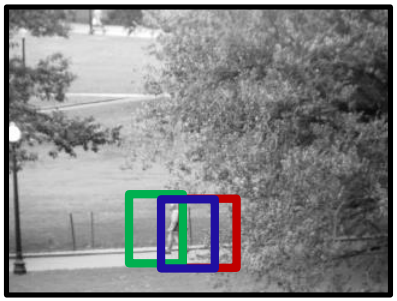
x measurement

* predicted mean estimate

\+ corrected mean estimate

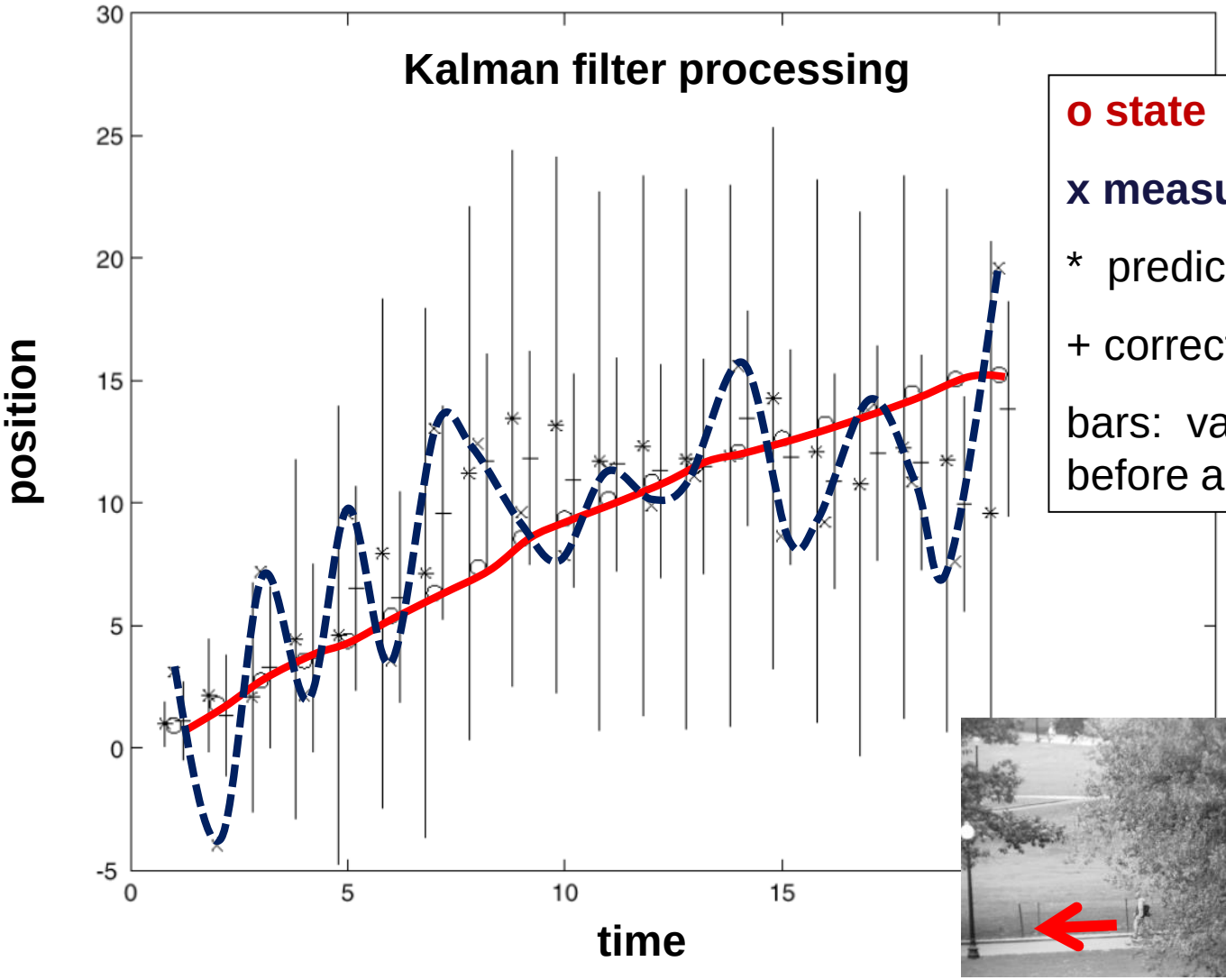
bars: variance estimates before and after measurements

Time t



Time t+1

Constant velocity model



o state

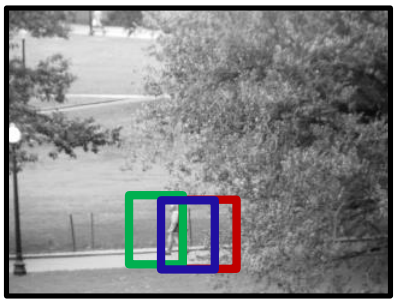
x measurement

* predicted mean estimate

\+ corrected mean estimate

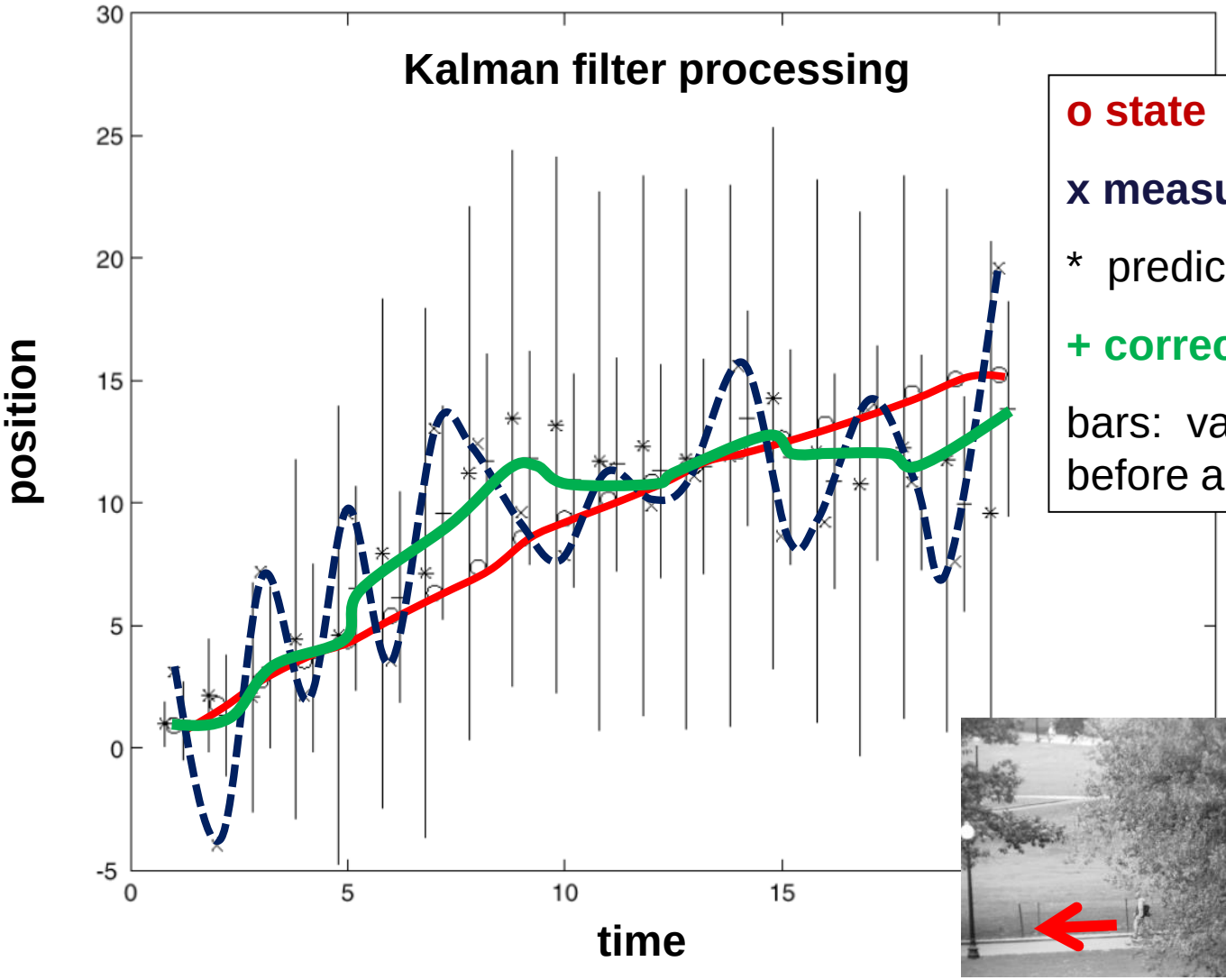
bars: variance estimates before and after measurements

Time t



Time t+1

Constant velocity model



o state

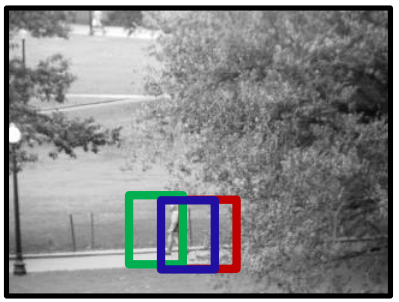
x measurement

* predicted mean estimate

+ corrected mean estimate

bars: variance estimates before and after measurements

Time t



Time t+1

Video synopsis

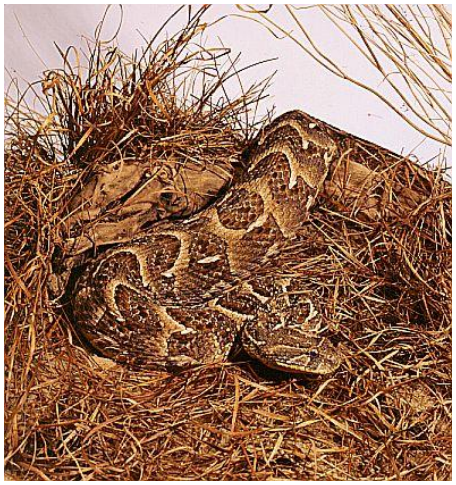
- http://www.vision.huji.ac.il/video-synopsis/

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?



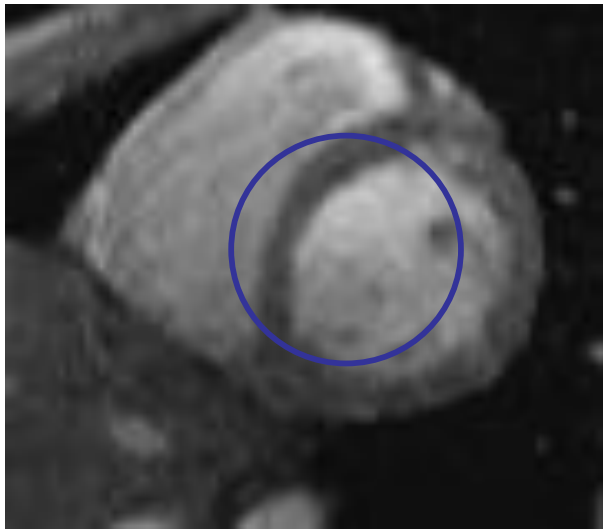
Tracking: issues

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

Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object



[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

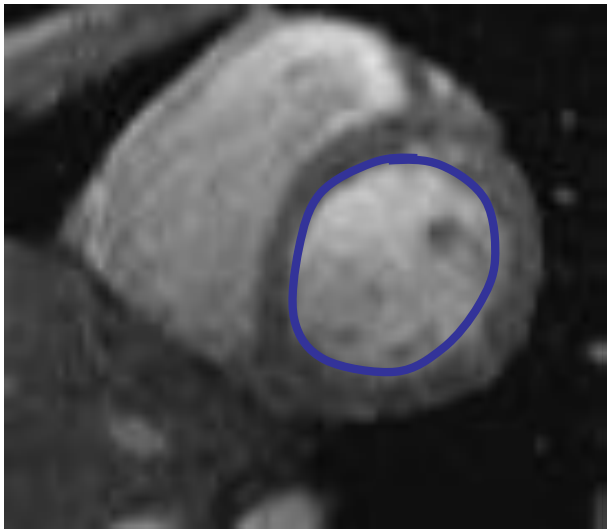
Figure credit: Yuri Boykov

Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



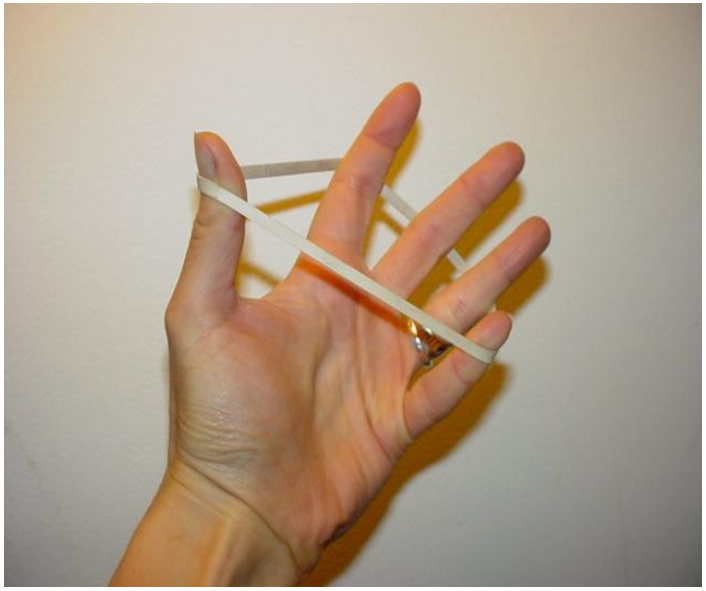
Main idea: elastic band is iteratively adjusted so as to

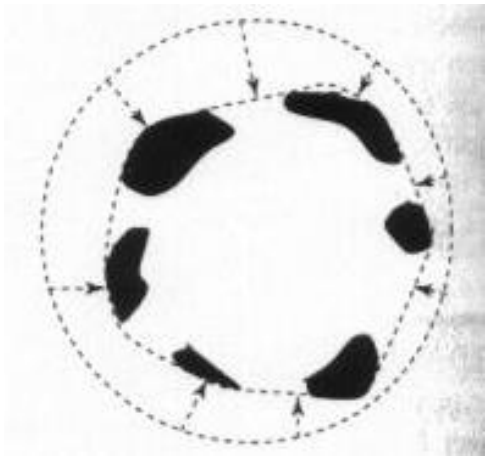
- be near image positions with high gradients, **and**
- satisfy shape “preferences” or contour priors

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Figure credit: Yuri Boykov

Deformable contours: intuition





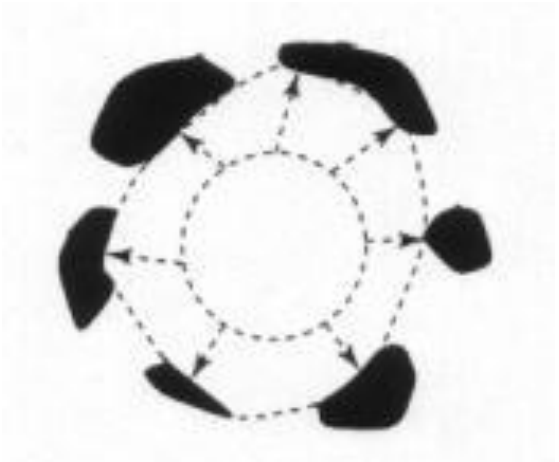
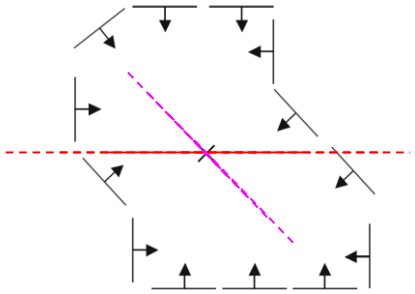


Image from http://www.healthline.com/blogs/exercise_fitness/uploaded_images/HandBand2-795868.JPG

Deformable contours vs. Hough

Like generalized Hough transform, useful for shape fitting; but



<u>Hough</u>

Rigid model shape

Single voting pass can detect multiple instances

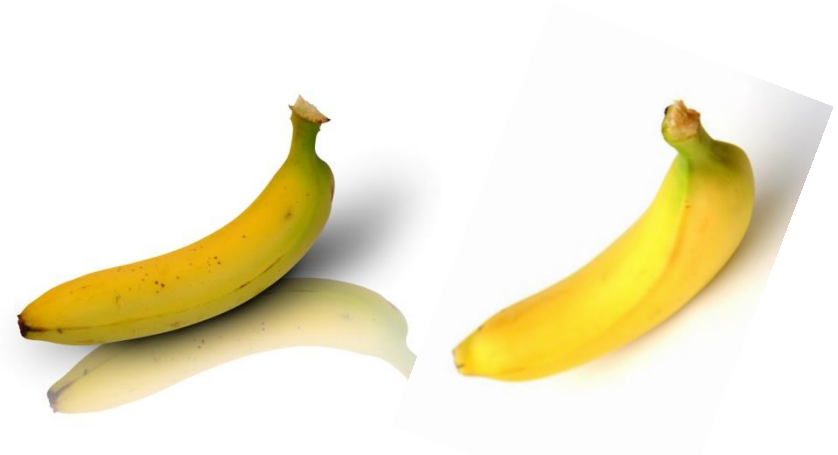
<u>Deformable contours</u>

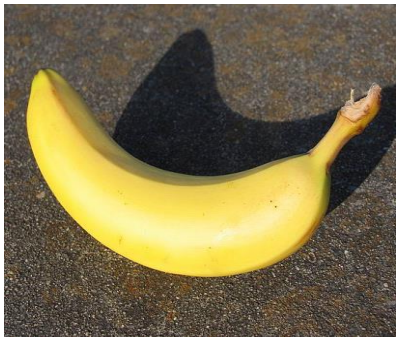
Prior on shape types, but shape iteratively adjusted (*deforms*)

Requires initialization nearby

One optimization “pass” to fit a single contour

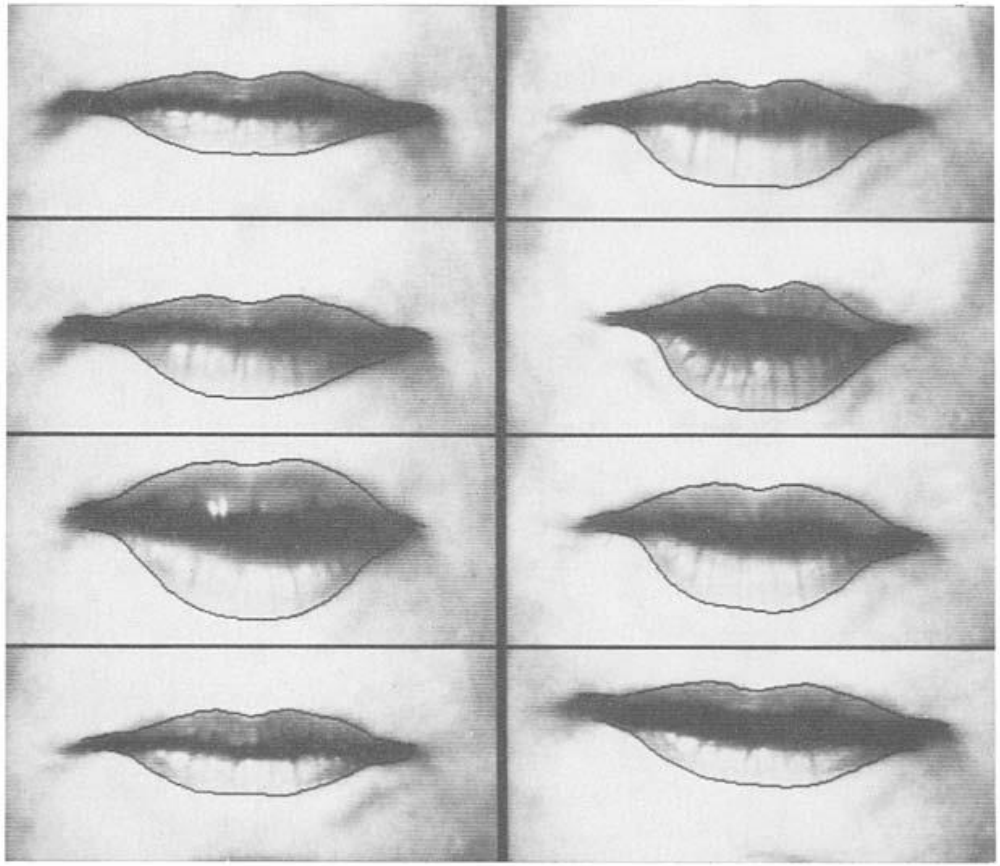
Why do we want to fit deformable shapes?





- Some objects have similar basic form but some variety in the contour shape.

Why do we want to fit deformable shapes?



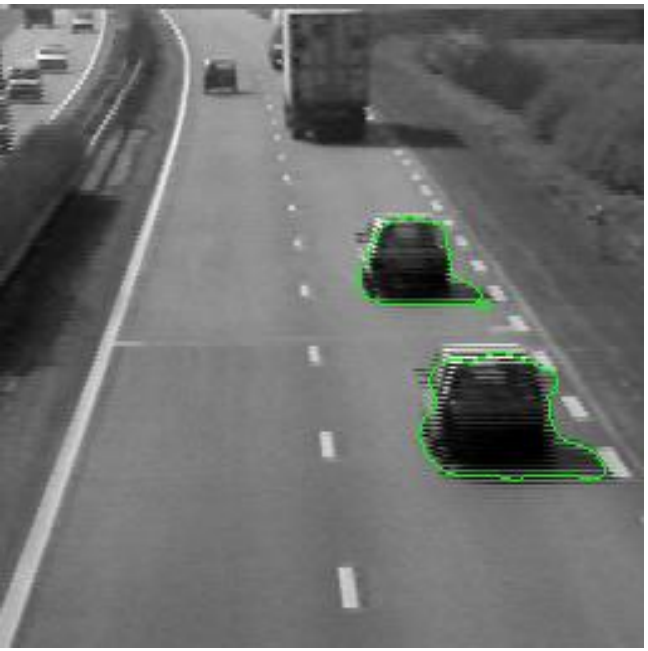
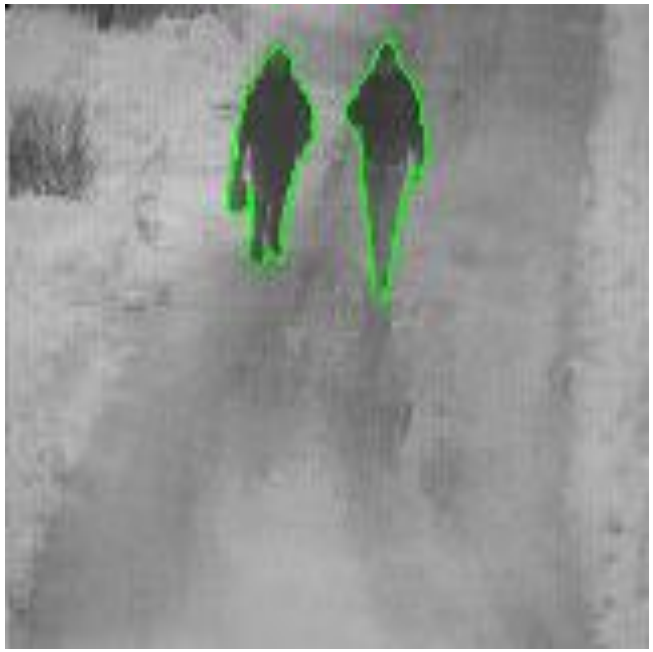
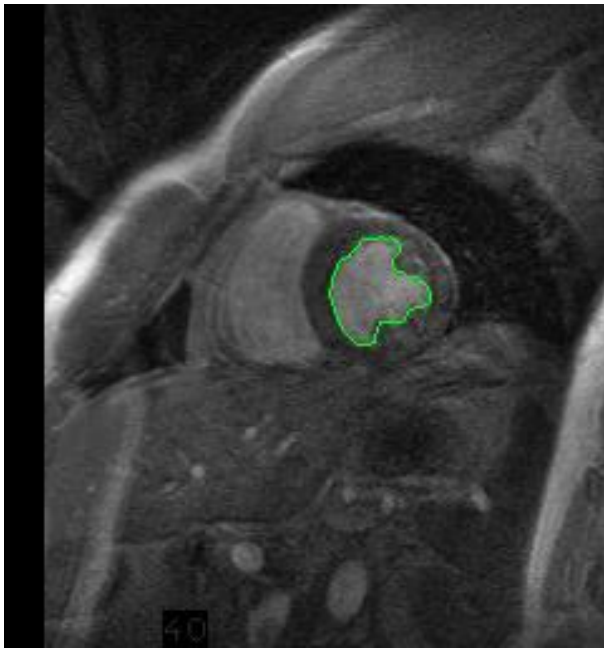
- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands…

Figure from Kass et al. 1987

Why do we want to fit deformable shapes?

- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands…

Why do we want to fit deformable shapes?

- Non-rigid, deformable objects can change their shape over time.

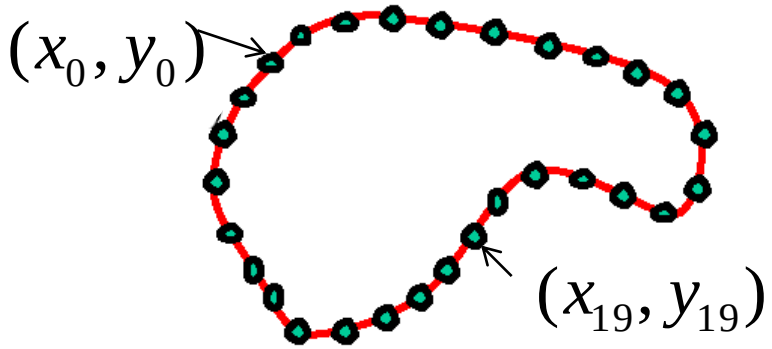
Figure credit: Julien Jomier

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Use:
  - Tracking
  - Interactive segmentation

Representation

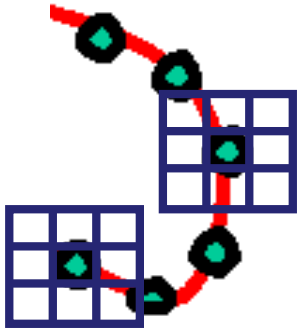
- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions ("vertices").



$$v_i = (x_i, y_i),$$

for $i = 0, 1, \ldots, n-1$

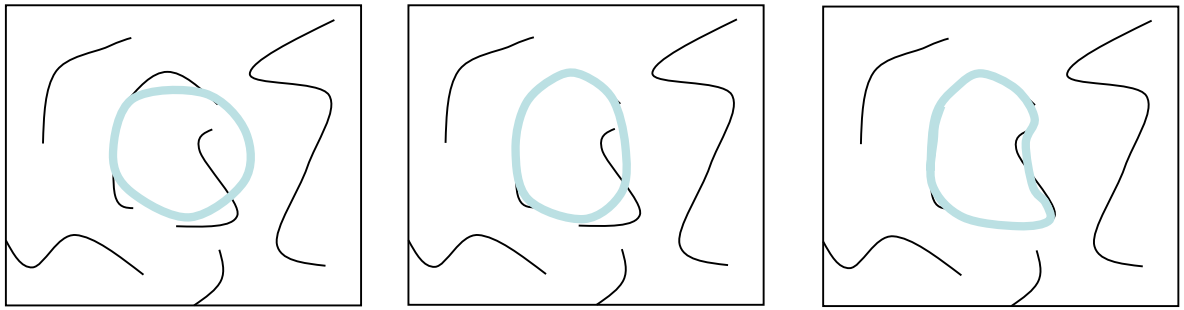
- At each iteration, we'll have the option to move each vertex to another nearby location ("state").



Fitting deformable contours

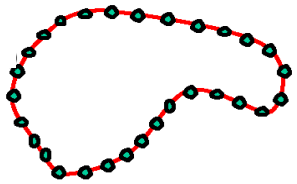
How should we adjust the current contour to form the new contour at each iteration?

- Define a cost function (“energy” function) that says how good a candidate configuration is.
- Seek next configuration that minimizes that cost function.



Energy function

The total energy (cost) of the current snake is defined as:



$$E_{total} = E_{internal} + E_{external}$$

Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

External energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour and the target shape in the image will yield a **low** value for this cost function.

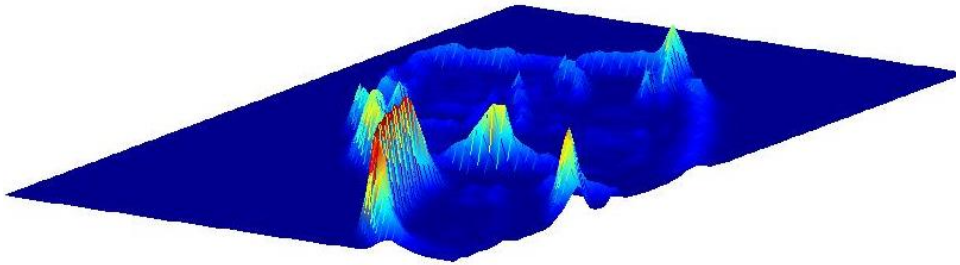
External energy: intuition

- Measure how well the curve matches the image data
- “Attract” the curve toward different image features
  - Edges, lines, texture gradient, etc.

External image energy

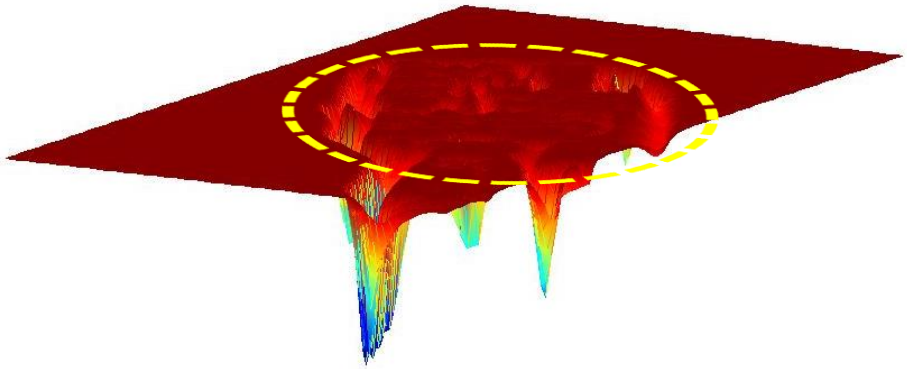
How do edges affect “snap” of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$

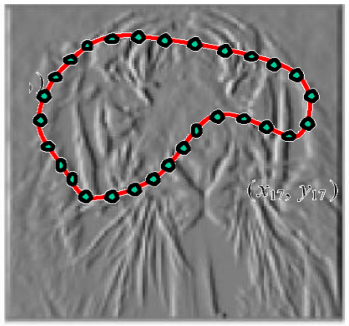


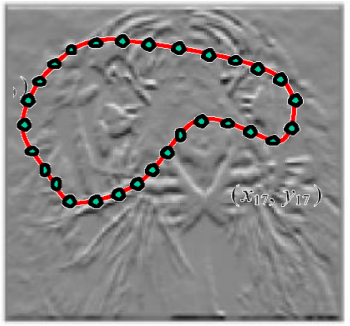
- (Magnitude of gradient)

$$-\left(G_x(I)^2 + G_y(I)^2\right)$$

External image energy

- Gradient images $G_x(x, y)$ and $G_y(x, y)$





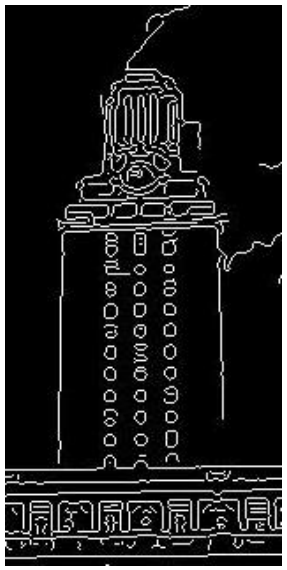
- External energy at a point on the curve is:

$$E_{external}(v) = -(|G_x(v)|^2 + |G_y(v)|^2)$$

- External energy for the whole curve:

$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

Internal energy: intuition

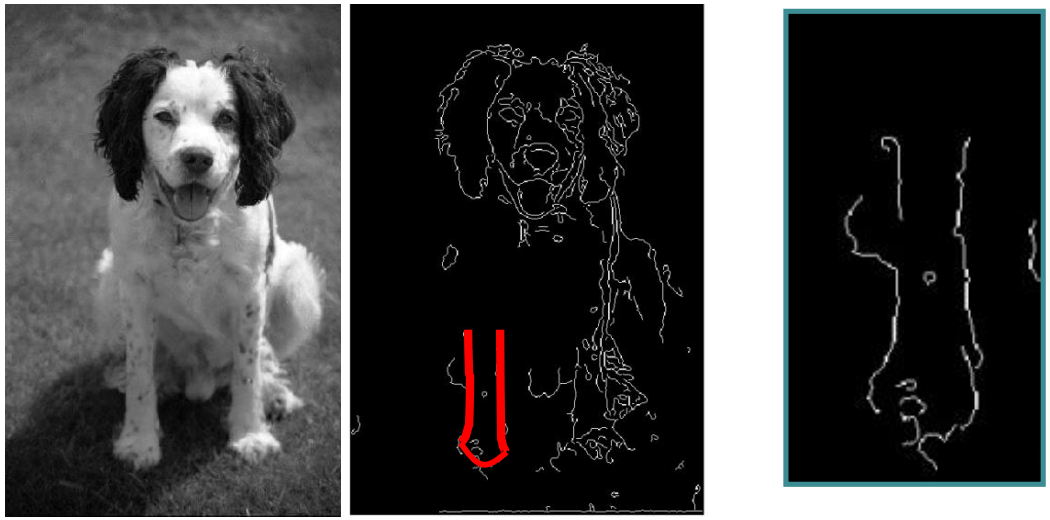


What are the underlying boundaries in this fragmented edge image?

And in this one?

Internal energy: intuition

A priori, we want to favor **smooth** shapes, contours with **low curvature**, contours similar to a **known shape**, etc. to balance what is actually observed (i.e., in the gradient image).



Internal energy

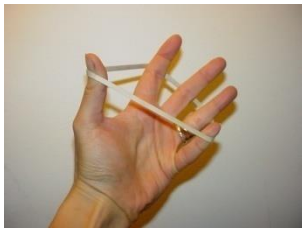
For a *continuous* curve, a common internal energy term is the "bending energy".

At some point *v(s)* on the curve, this is:

$$E_{internal}(v(s)) = \alpha \left| \frac{dv}{ds} \right|^2 + \beta \left| \frac{d^2 v}{d^2 s} \right|^2$$

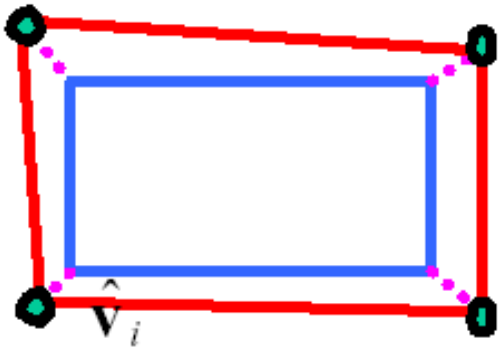
Tension, Elasticity

Stiffness, Curvature



Extending the internal energy: capture shape prior

- If object is some smooth variation on a known shape, we can use a term that will penalize deviation from that shape:



$$E_{internal} += \alpha \cdot \sum_{i=0}^{n-1} (v_i - \hat{v}_i)^2$$

where $\{\hat{v}_i\}$ are the points of the known shape.

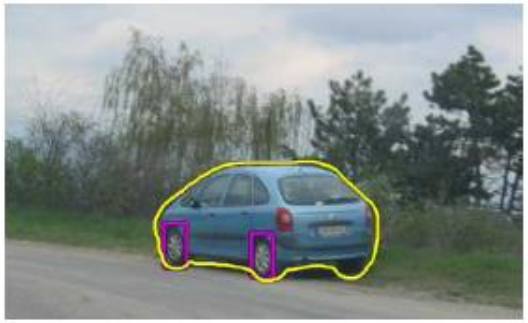


Fig from Y. Boykov

Total energy: function of the weights

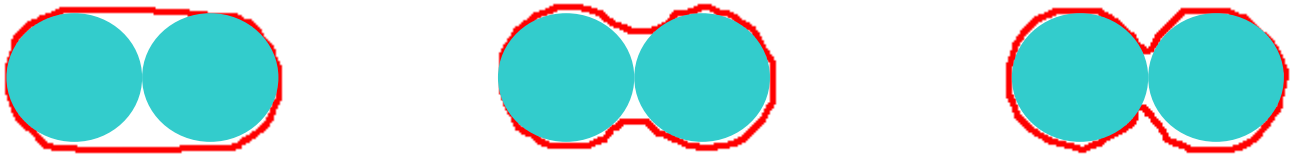
$$E_{total} = E_{internal} + \gamma E_{external}$$

$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left(\bar{d} - \|v_{i+1} - v_i\|\right)^2 + \beta \|v_{i+1} - 2v_i + v_{i-1}\|^2$$

Total energy: function of the weights

- e.g., α weight controls the penalty for internal elasticity

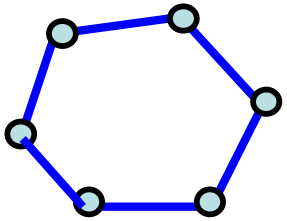


large α medium α small α

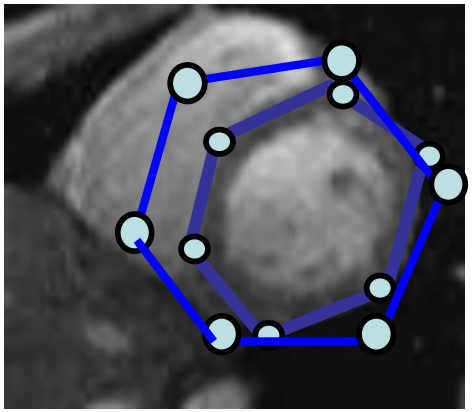
Fig from Y. Boykov

Recap: deformable contour

- A simple elastic snake is defined by:
  - A set of *n* points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)

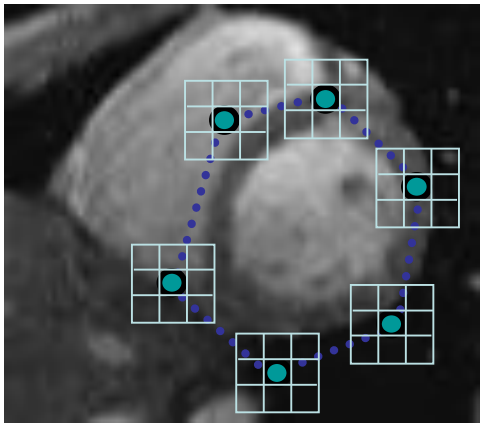


- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy



Energy minimization: greedy

- For each point, search window around it and move to where energy function is minimal
  - Typical window size, e.g., 5 x 5 pixels



- Stop when predefined number of points have not changed in last iteration, or after max number of iterations

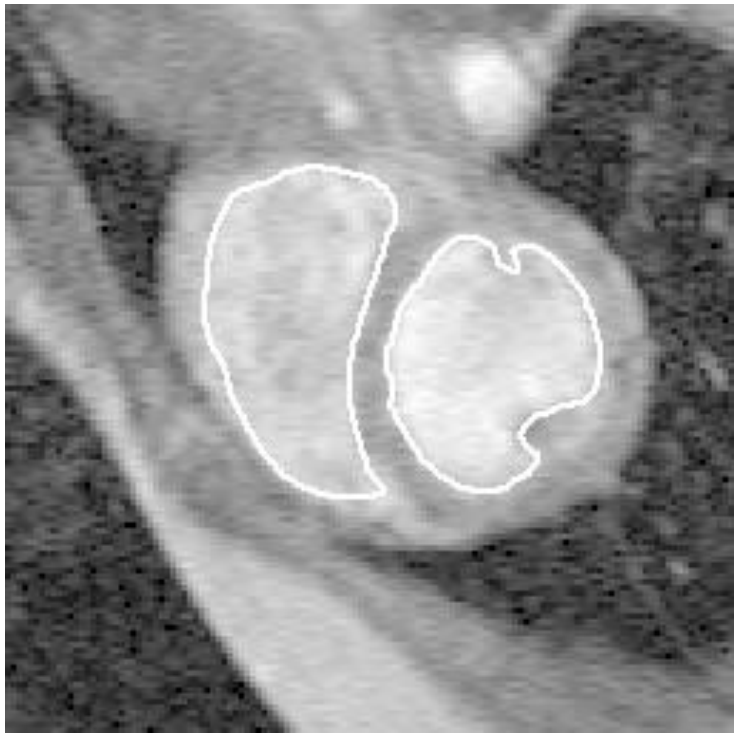
- Note:
  - Convergence not guaranteed
  - Need decent initialization

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Use:
  - Tracking
  - Interactive segmentation

Tracking via deformable contours

1. Use final contour/model extracted at frame *t* as an initial solution for frame *t+1*
2. Evolve initial contour to fit exact object boundary at frame *t+1*
3. Repeat, initializing with most recent frame.



Tracking Heart Ventricles
(multiple frames)

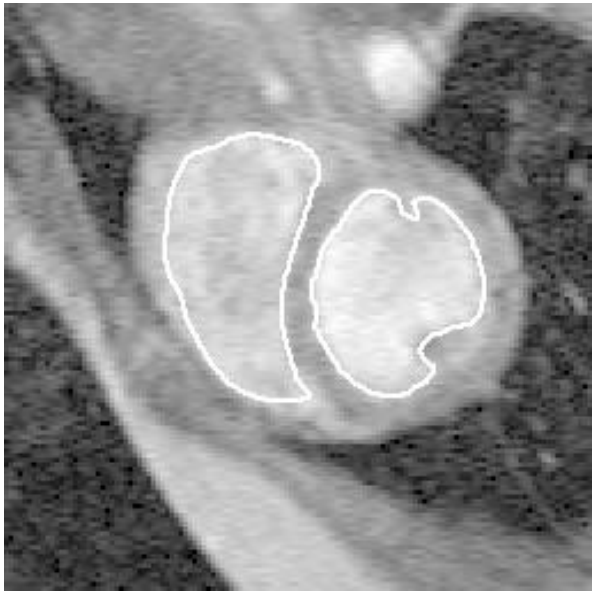
Tracking via deformable contours

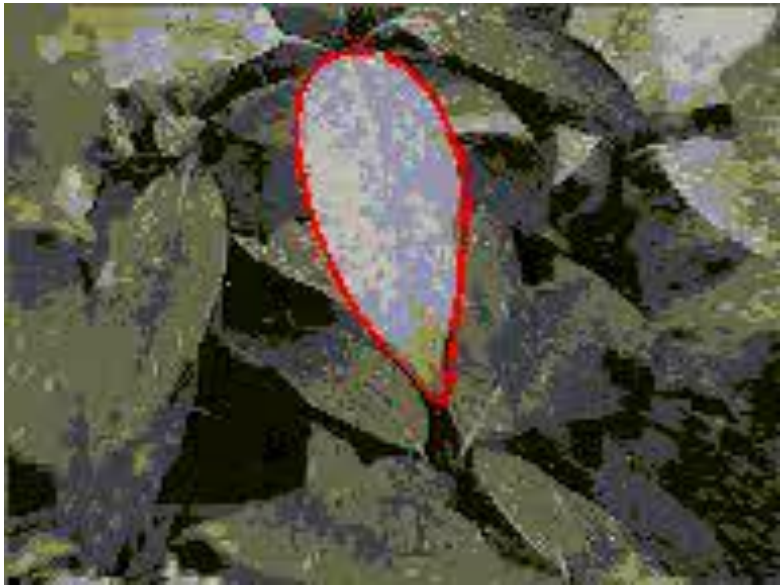
Visual Dynamics Group, Dept. Engineering Science, University of Oxford.

Applications: Traffic monitoring
Human-computer interaction
Animation
Surveillance
Computer assisted diagnosis in medical imaging

Tracking via deformable contours

1. Use final contour/model extracted at frame *t* as an initial solution for frame *t+1*
2. Evolve initial contour to fit exact object boundary at frame *t+1*
3. Repeat, initializing with most recent frame.



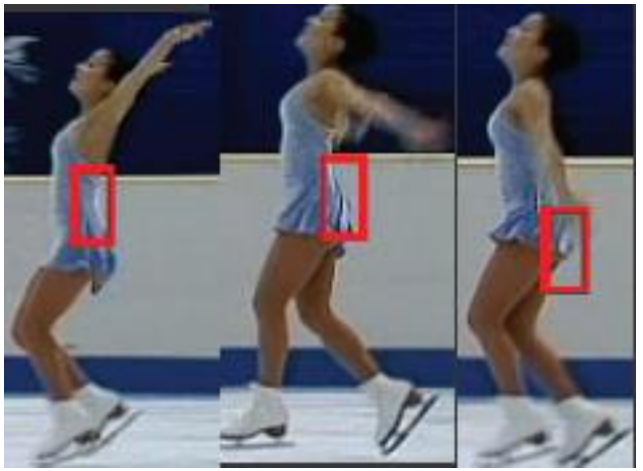


Visual Dynamics Group, Dept. Engineering Science, University of Oxford.

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.

Summary

- Tracking as inference
  - Goal: estimate posterior of object position given measurement
- Linear models of dynamics
  - Represent state evolution and measurement models
- Kalman filters
  - Recursive prediction/correction updates to refine measurement
- Deformable Contours
  - Evolve contour to fit exact deformable object boundary
- General tracking challenges