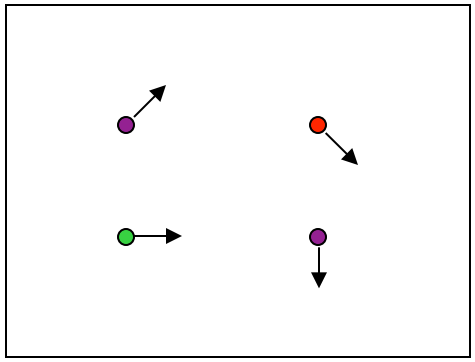
Computer Vision

Bogdan Alexe

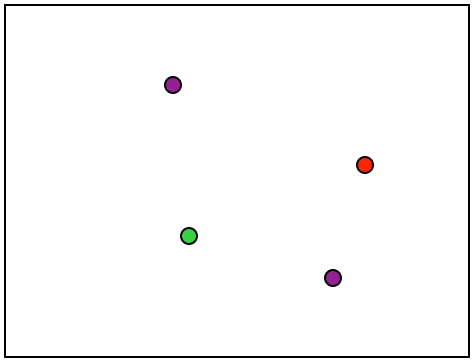
bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2020-2021

Recap - Estimating optical flow



$I(x,y,t\text{-}1)$



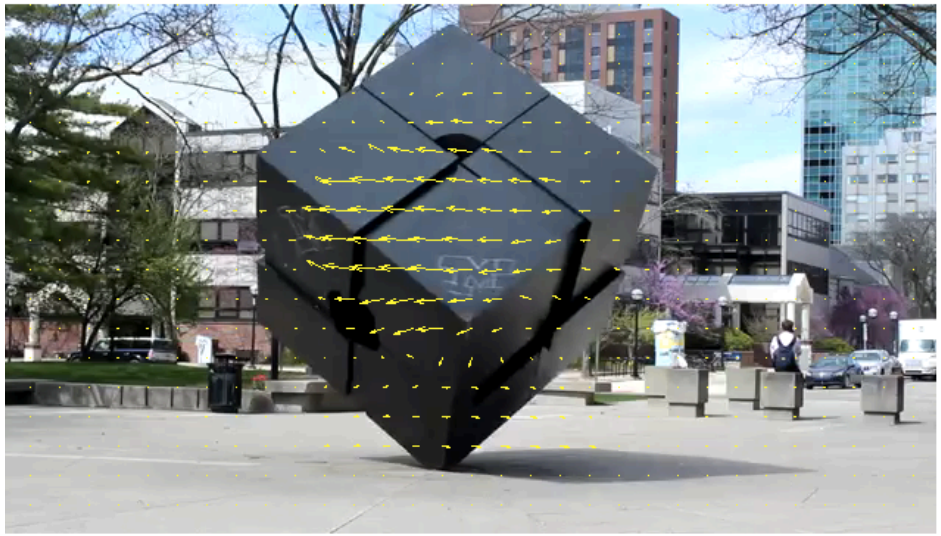
$I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ between them
- Key assumptions
  - Brightness constancy: projection of the same point looks the same in every frame
  - Small motion: points do not move very far
  - Spatial coherence: points move like their neighbors

Recap - Lucas-Kanade flow example

Input frames

Output

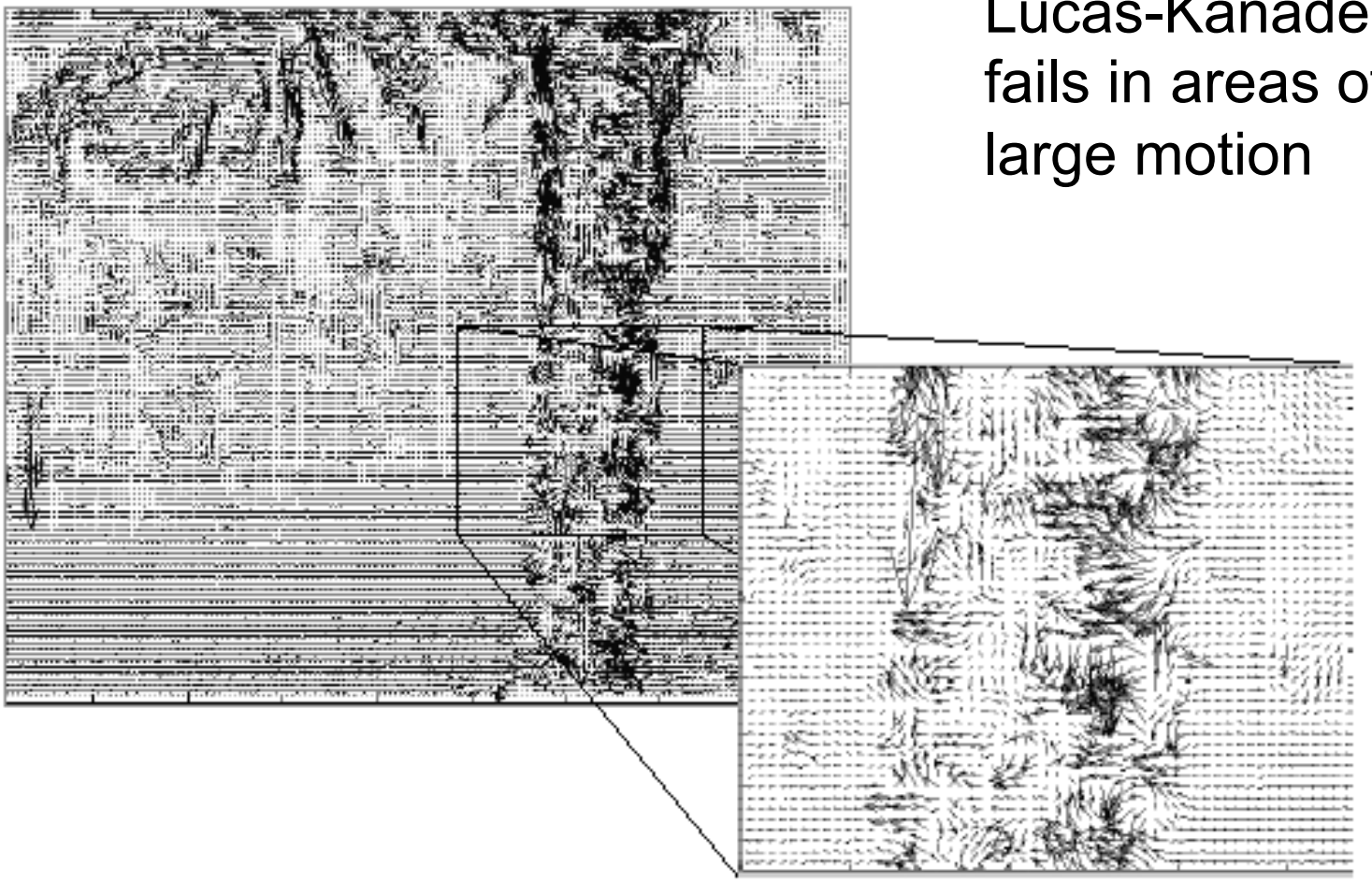


Source: MATLAB Central File Exchange

Recap - “Flower garden” example

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Recap - “Flower garden” example



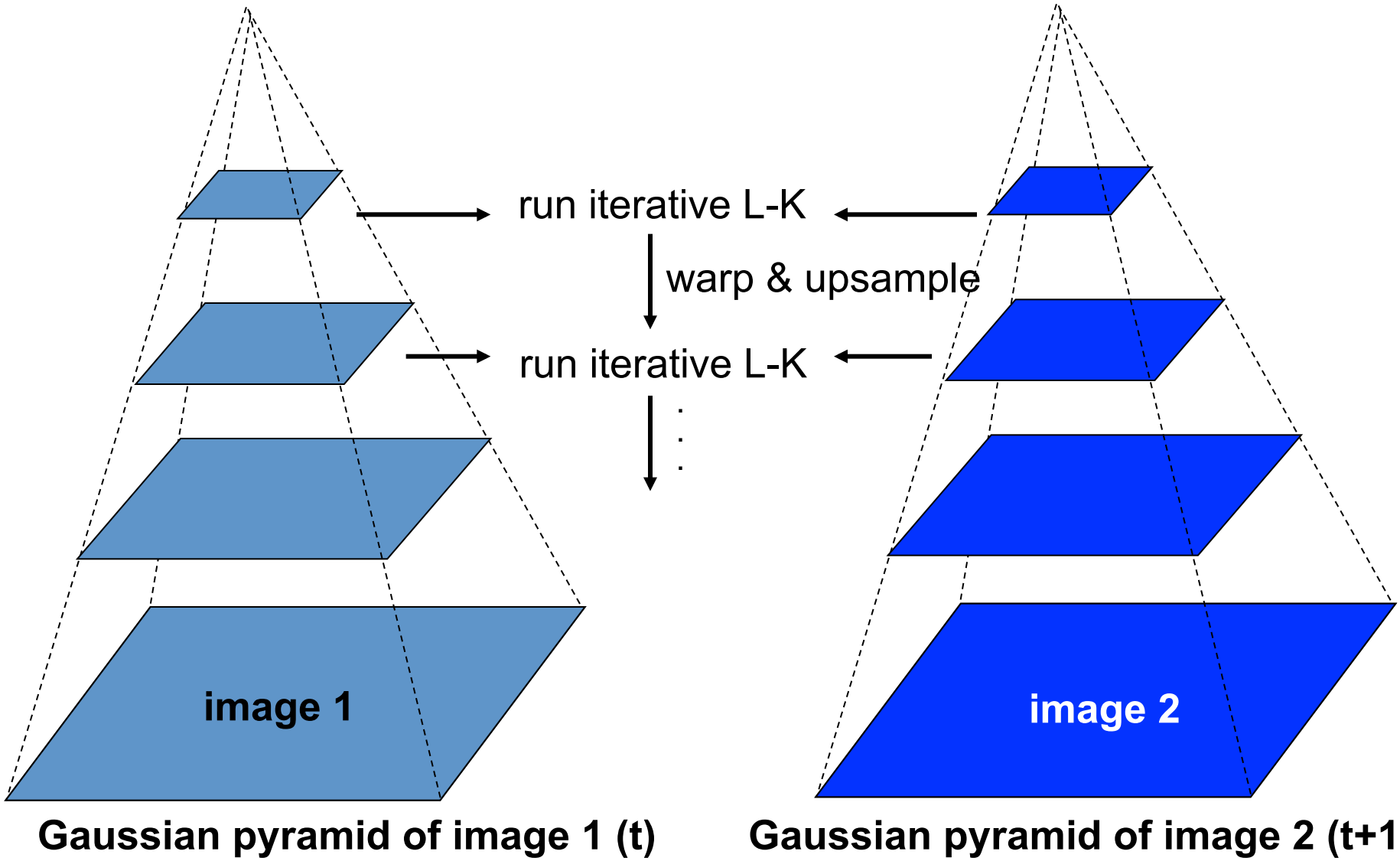
Lucas-Kanade fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

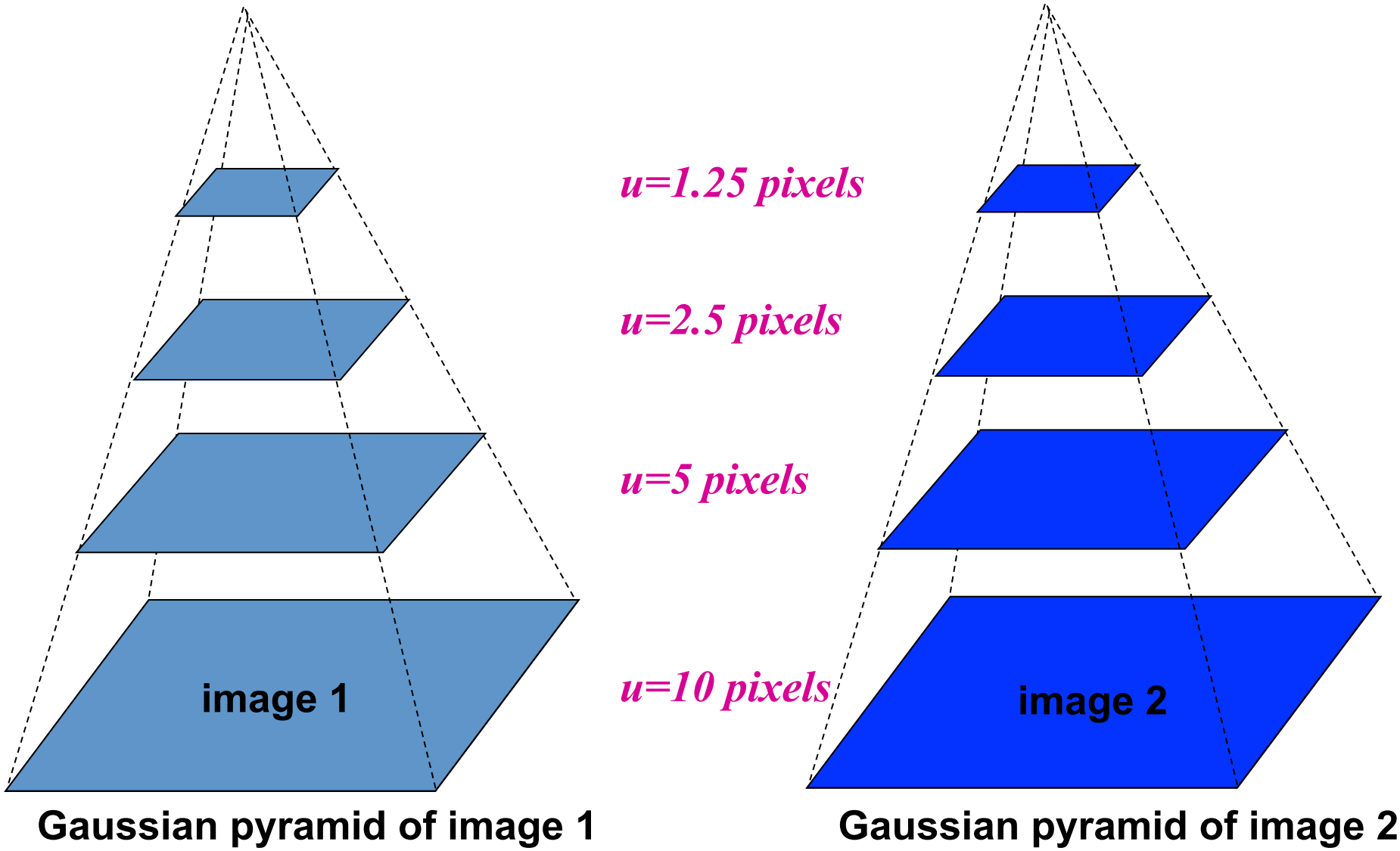
Recap - Multi-resolution Lucas Kanade Algorithm

- Compute 'simple' LK at highest level
- At level i
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution for level i
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.

Recap - Coarse-to-fine optical flow estimation



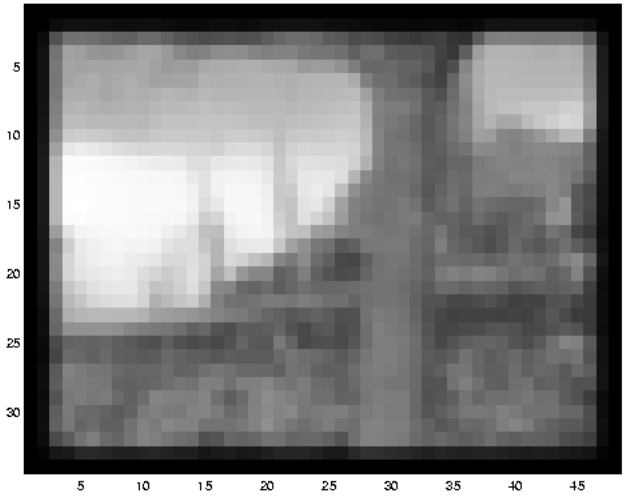
Recap - Coarse-to-fine optical flow estimation

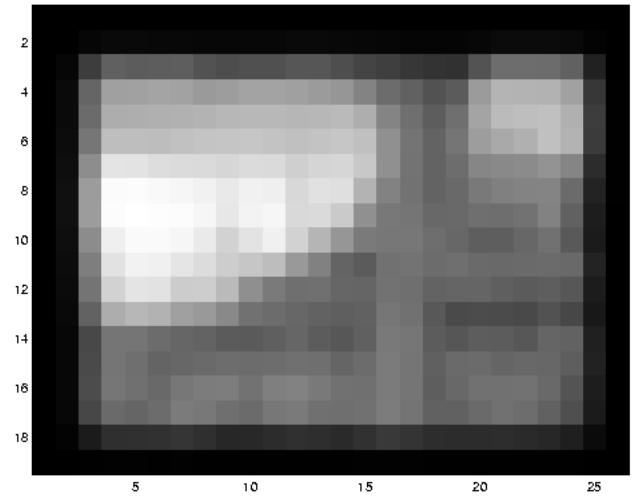


Example

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

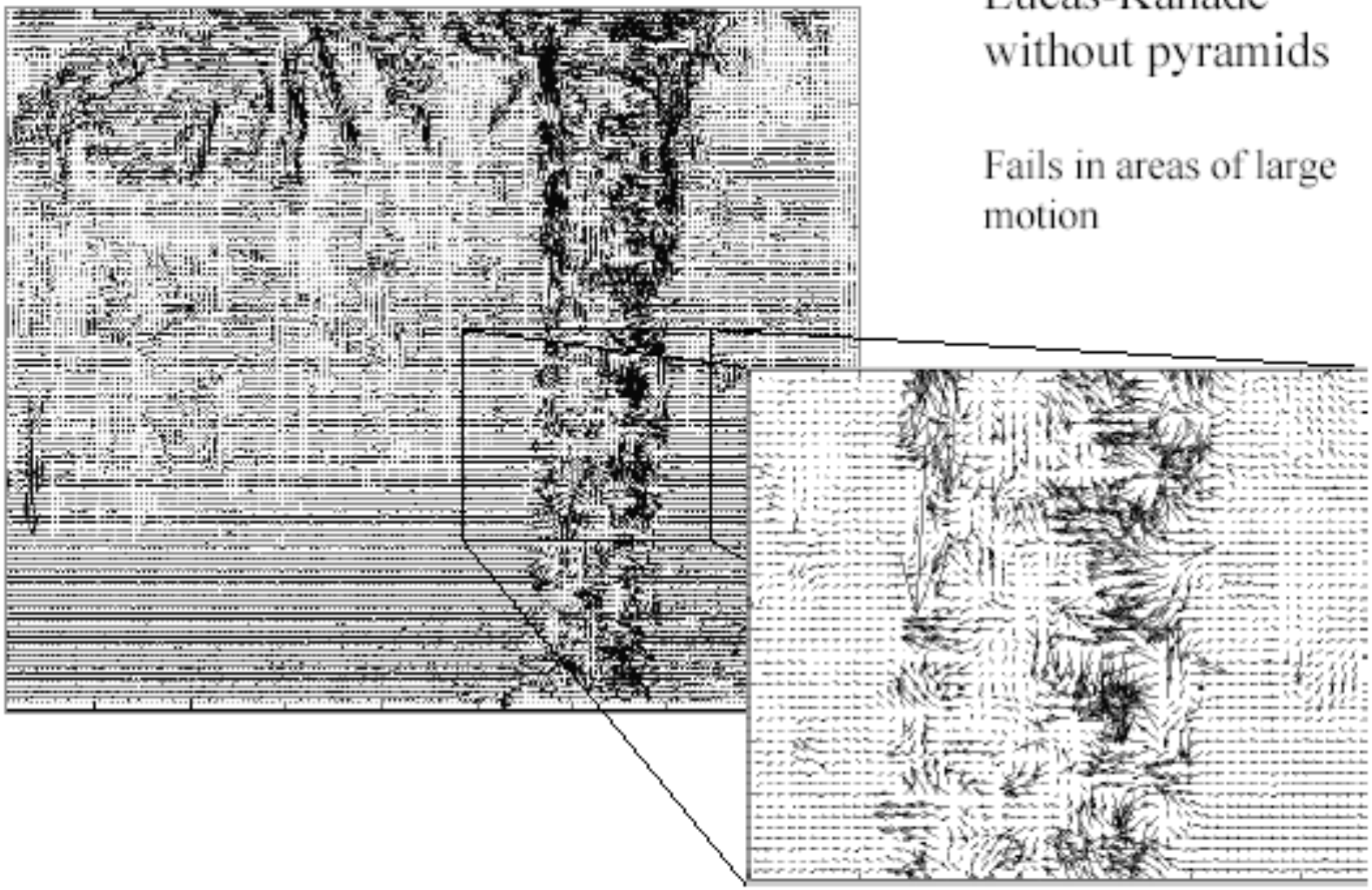
Multi-resolution registration





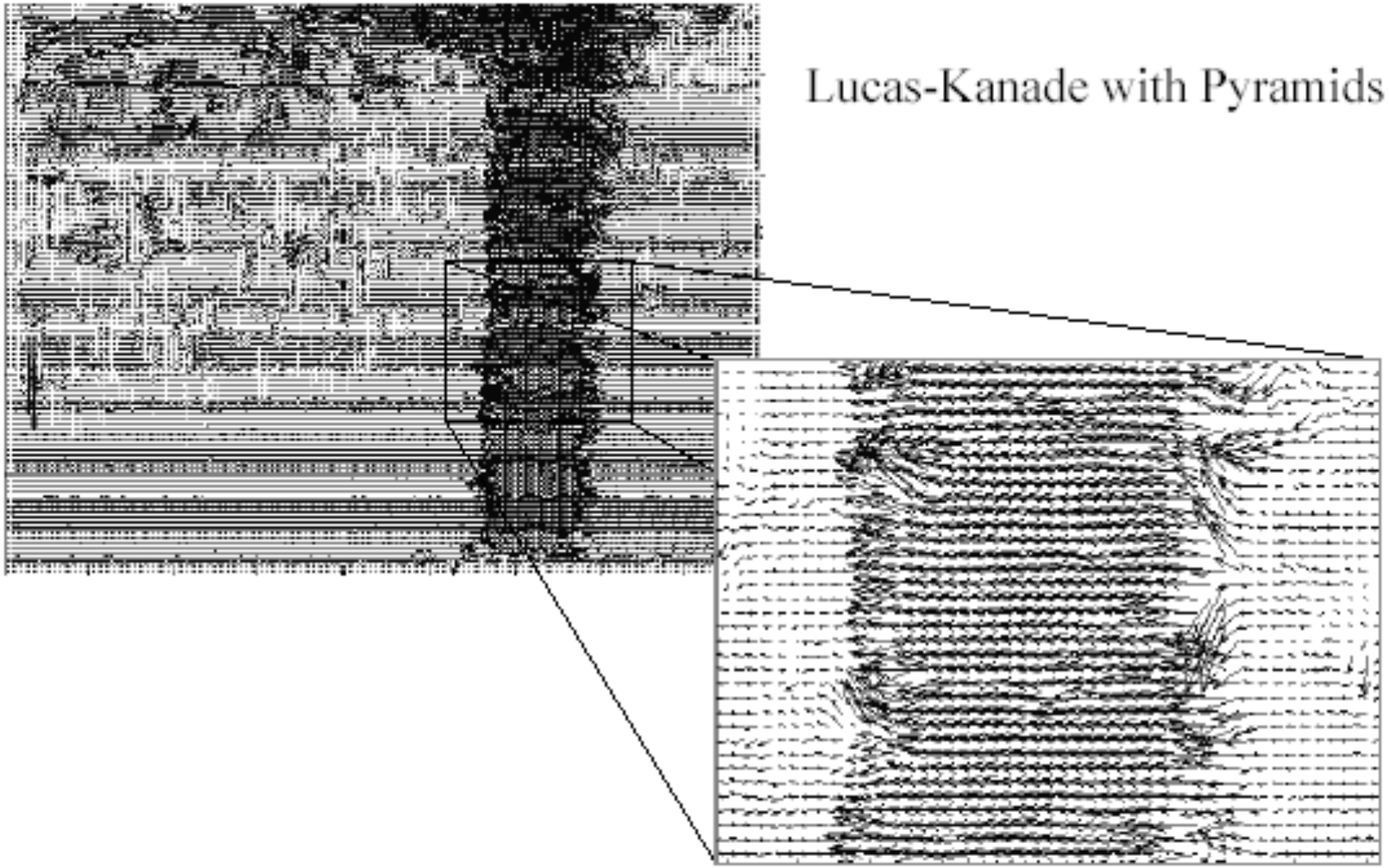
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Fixing the errors in Lucas-Kanade

- The motion is large (larger than a pixel)
  - Multi-resolution estimation, iterative refinement
  - Feature matching
- A point does not move like its neighbors
  - Motion segmentation
- Brightness constancy does not hold
  - Feature matching

Recap: Background Subtraction

Recap: Background Subtraction

-

=

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

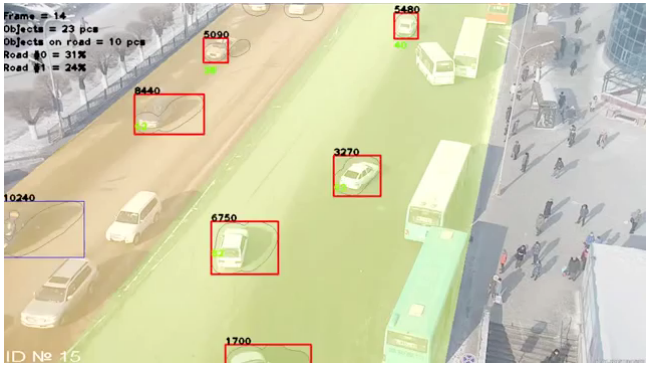
4. **Object Recognition: high – level vision**

   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Object tracking



traffic

body

eyes

soccer

snooker

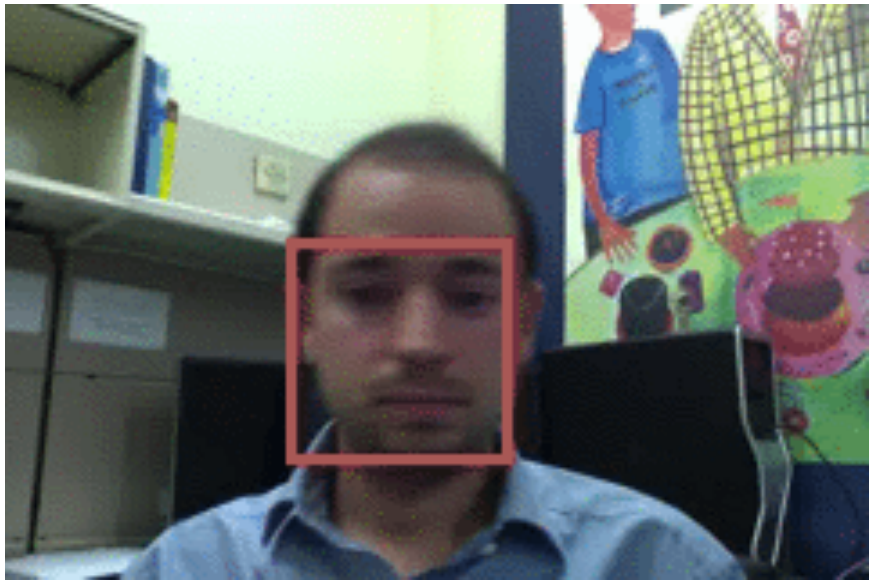
face

Further applications

- Motion capture
- Augmented Reality
- Action Recognition
- Security, traffic monitoring
- Video Compression
- Video Summarization
- Medical Screening
- Sport Analysis

Things that make visual tracking difficult

- Small, few visual features
- Erratic movements, moving very quickly
- Occlusions, leaving and coming back
- Surrounding similar-looking objects



Strategies for tracking

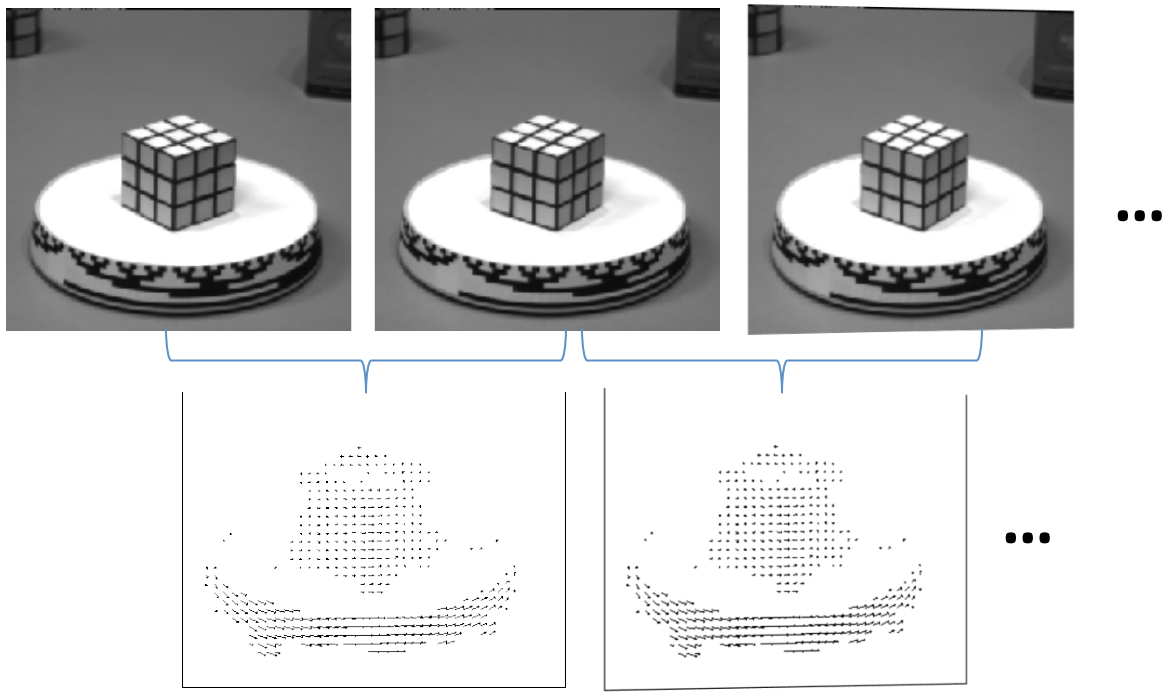
- **Feature-based tracking**
  - extract visual features (corners, textured areas) and track them over multiple frames
  - sparse motion fields, but more robust tracking
  - figure out which features can be tracked
  - some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)

- **Tracking by repeated detection**
  - works well if object is easily detectable (e.g., face, ball or colored glove) and there is only one
  - need some way to link up detections (obtain tracks)
  - best you can do, if you can't predict motion

- **Tracking with dynamics**
  - based on a model of expected motion, predict where objects will occur in next frame, before even seeing the image
  - restrict search for the object
  - measurement noise is reduced by trajectory smoothness

Optical flow for tracking?

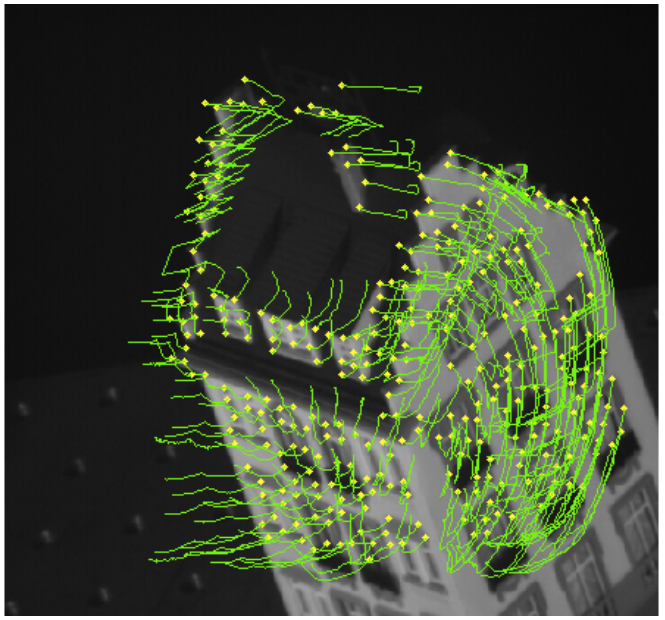
If we have more than just a pair of frames, we could compute flow from one to the next:



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

Feature tracking

- If we have more than two images, we can track a feature from one frame to the next by following the optical flow
- Challenges
  - Finding good features to track
  - Adding and deleting tracks (trajectories of objects in video)

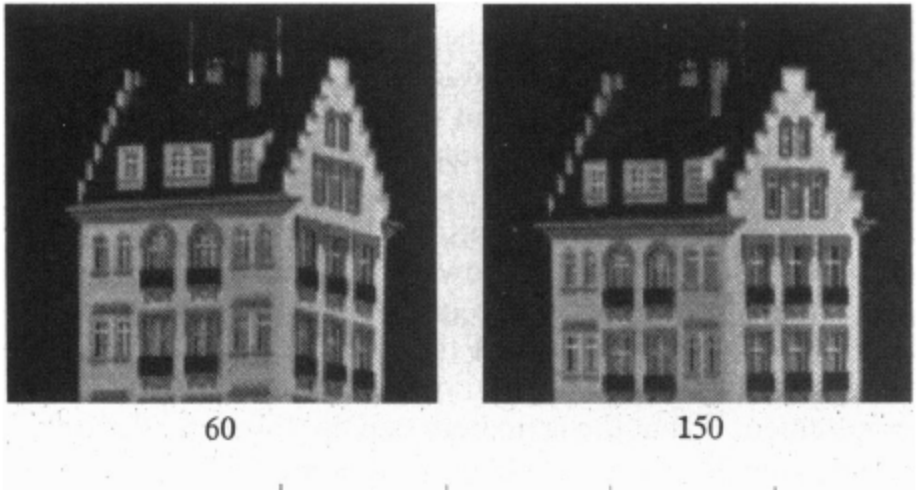


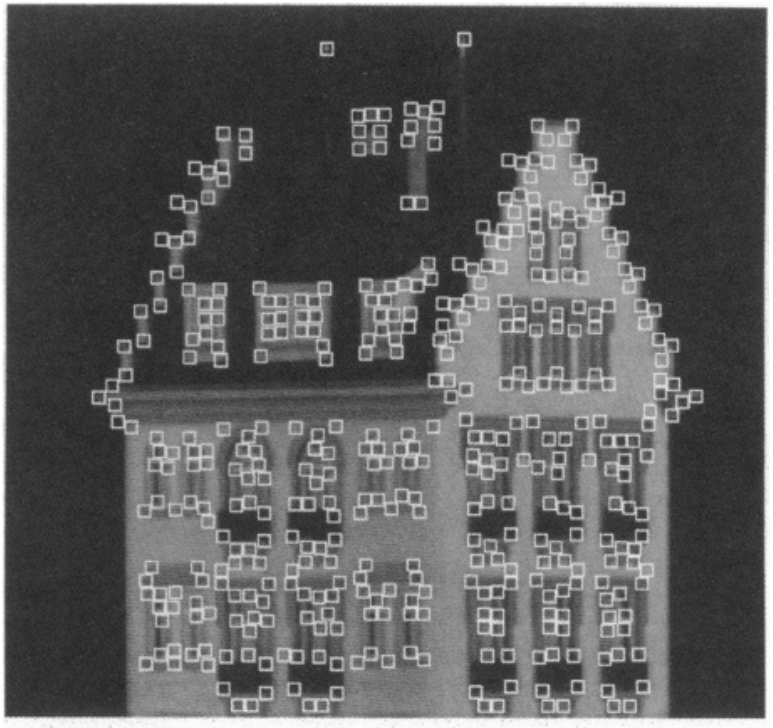
Feature tracking

- Challenges
  - Figure out which features can be tracked
  - Efficiently track across frames
  - Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
  - Drift: small errors can accumulate as appearance model is updated
  - Points may appear or disappear: need to be able to add/delete tracked points

Lucas-Kanade feature tracker

- Same as Lucas-Kanade optical flow, but instead for all pixels just for selected features (e.g. Harris corners)





Shi-Tomasi feature tracker

- Find good features using eigenvalues of second-moment matrix
  - Key idea: "good" features to track are the ones whose motion can be estimated reliably (Harris corner)
- From frame to frame, track with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement
- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Tracking example

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Detection vs. tracking


t=1


t=2

...


t=20


t=21

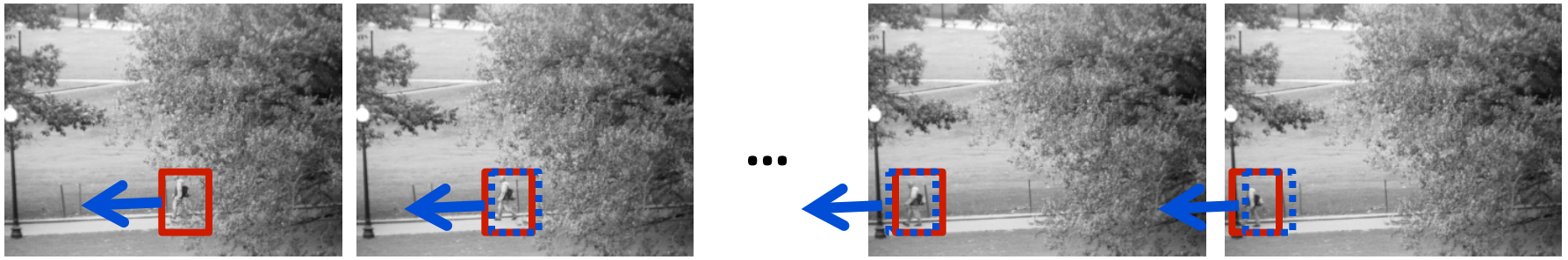
Slide credit: Kristen Grauman

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

Slide credit: 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.

Slide credit: 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

Slide credit: Kristen Grauman

Tracking as inference

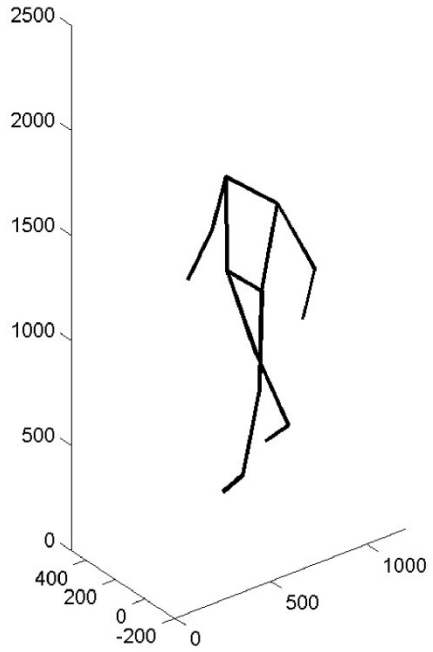
- The *hidden state* consists of the true parameters we care about, denoted *X (*e.g. a vector containing the position, velocity, acceleration).

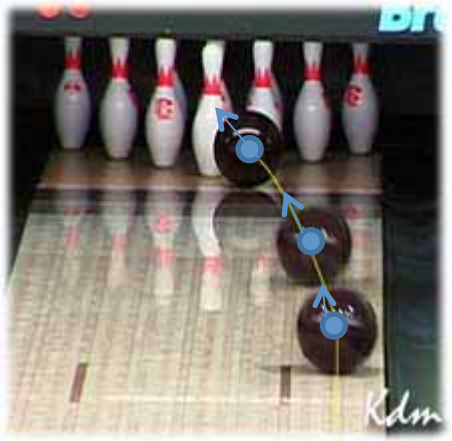
- The *measurement* is our noisy observation that results from the underlying state, denoted *Y*. Could be the position of some image points, the position of some image regions or pretty much anything else.

- At each time step, state changes (from X_{t-1} to X_t) and we get a new observation Y_t.

Slide credit: Kristen Grauman

State vs. observation





Hidden state : parameters of interest

Measurement : what we get to directly observe

Slide credit: Kristen Grauman

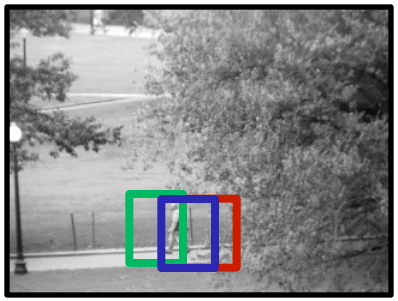
Tracking as inference

- 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.

Slide credit: Kristen Grauman

Tracking as inference: intuition

Time t



Time t+1

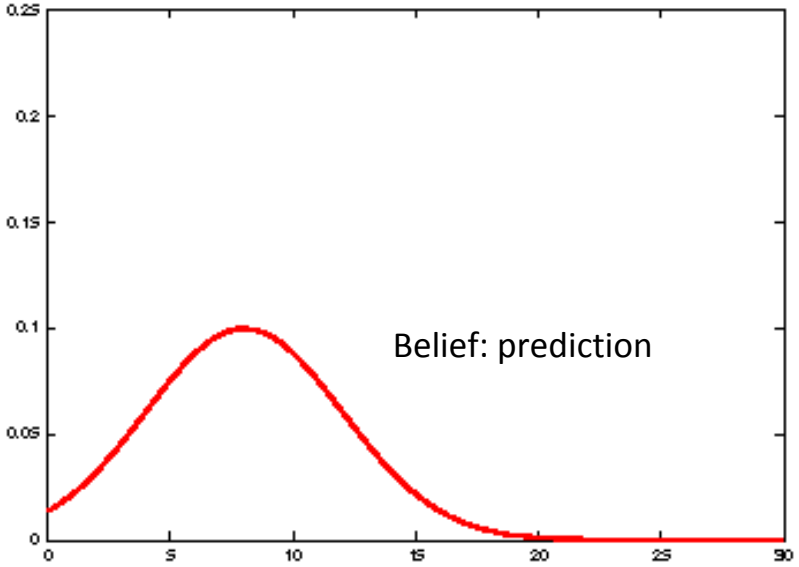
Belief

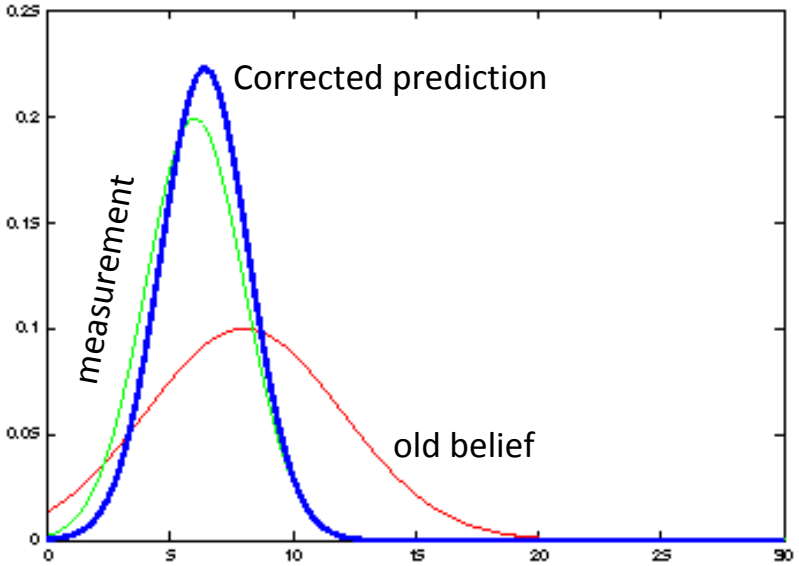
Measurement

Corrected prediction

Slide credit: Kristen Grauman

Tracking as inference: intuition





Time t Time t+1

Slide credit: Kristen Grauman

Independence assumptions

- Only immediate past state influences current state

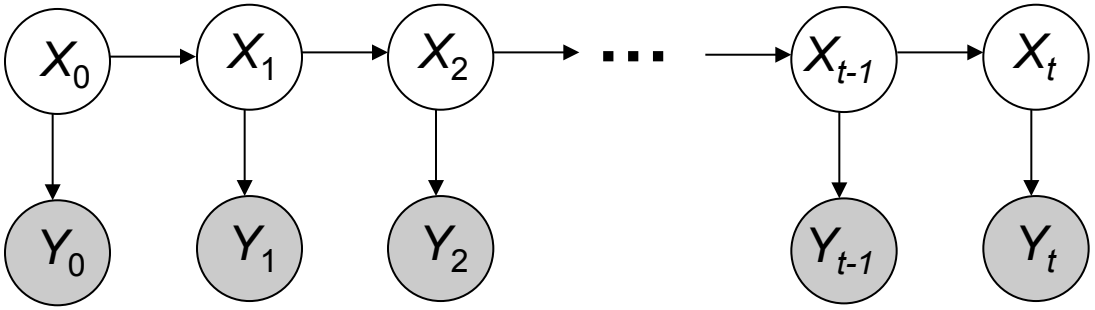
$$P(X_t|X_0,\ldots,X_{t-1}) = P(X_t|X_{t-1})$$

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



Hidden Markov model

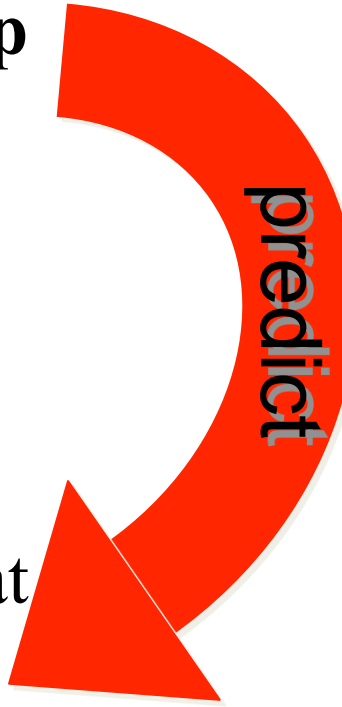
Tracking as inference

- 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?

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



Prediction

- Prediction involves representing $P(X_t|y_0,\ldots,y_{t-1})$ given $P(X_{t-1}|y_0,\ldots,y_{t-1})$

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

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

Law of total probability

Prediction

- Prediction involves representing $P(X_t | y_0, \ldots, y_{t-1})$ given $P(X_{t-1} | y_0, \ldots, y_{t-1})$

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

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

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

Conditioning on X_{t-1}

Prediction

- Prediction involves representing $P(X_t|y_0,\ldots,y_{t-1})$ given $P(X_{t-1}|y_0,\ldots,y_{t-1})$

$$P(X_t|y_0,\ldots,y_{t-1})$$
$$= \int P(X_t, X_{t-1}|y_0,\ldots,y_{t-1})dX_{t-1}$$
$$= \int P(X_t \mid X_{t-1}, y_0,\ldots,y_{t-1})P(X_{t-1} \mid y_0,\ldots,y_{t-1})dX_{t-1}$$
$$= \int P(X_t \mid X_{t-1})P(X_{t-1} \mid y_0,\ldots,y_{t-1})dX_{t-1}$$

Independence assumption

Prediction

- Prediction involves representing $P(X_t | y_0, \ldots, y_{t-1})$ given $P(X_{t-1} | y_0, \ldots, y_{t-1})$

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

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

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

$$= \int \underbrace{P(X_t | X_{t-1})}_{\text{dynamics model}} \underbrace{P(X_{t-1} | y_0, \ldots, y_{t-1})}_{\text{corrected estimate from previous step}} dX_{t-1}$$

Correction

- Correction involves computing $P(X_t | y_0, \ldots, y_t)$ given predicted value $P(X_t | y_0, \ldots, y_{t-1})$

Correction

- Correction involves computing $P(X_t | y_0, \ldots, y_t)$ given predicted value $P(X_t | y_0, \ldots, y_{t-1})$

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

Bayes' Rule

Correction

- Correction involves computing $P(X_t | y_0, \ldots, y_t)$ given predicted value $P(X_t | y_0, \ldots, y_{t-1})$

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

Independence assumption
(observation y_t directly depends only on state X_t)

Correction

- Correction involves computing $P(X_t|y_0,\ldots,y_t)$ given predicted value $P(X_t|y_0,\ldots,y_{t-1})$

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

$$= \frac{P(y_t \mid X_t, y_0,\ldots,y_{t-1})}{P(y_t \mid y_0,\ldots,y_{t-1})} P(X_t \mid y_0,\ldots,y_{t-1})$$

$$= \frac{P(y_t \mid X_t) P(X_t \mid y_0,\ldots,y_{t-1})}{P(y_t \mid y_0,\ldots,y_{t-1})}$$

$$= \frac{P(y_t \mid X_t) P(X_t \mid y_0,\ldots,y_{t-1})}{\int P(y_t \mid X_t) P(X_t \mid y_0,\ldots,y_{t-1}) dX_t}$$

Conditioning on X_t

Correction

- Correction involves computing $P(X_t | y_0, \ldots, y_t)$ given predicted value $P(X_t | y_0, \ldots, y_{t-1})$

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

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

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

observation model → $P(y_t | X_t)$

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

$$= \frac{P(y_t | X_t) P(X_t | y_0, \ldots, y_{t-1})}{\int P(y_t | X_t) P(X_t | y_0, \ldots, y_{t-1}) dX_t}$$

normalization factor

Summary: Prediction and correction

Prediction:

$$P(X_t \mid y_0, \ldots, y_{t-1}) = \int \underbrace{P(X_t \mid X_{t-1})}_{\text{dynamics model}} \underbrace{P(X_{t-1} \mid y_0, \ldots, y_{t-1})}_{\text{corrected estimate from previous step}} dX_{t-1}$$

Correction:

$$P(X_t \mid y_0, \ldots, y_t) = \frac{\overbrace{P(y_t \mid X_t)}^{\text{observation model}} \overbrace{P(X_t \mid y_0, \ldots, y_{t-1})}^{\text{predicted estimate}}}{\int P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1}) dX_t}$$

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.

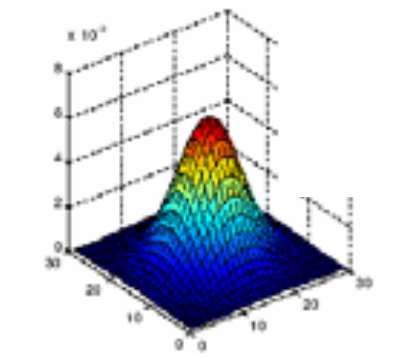
Updates: via the Kalman filter.

Notation

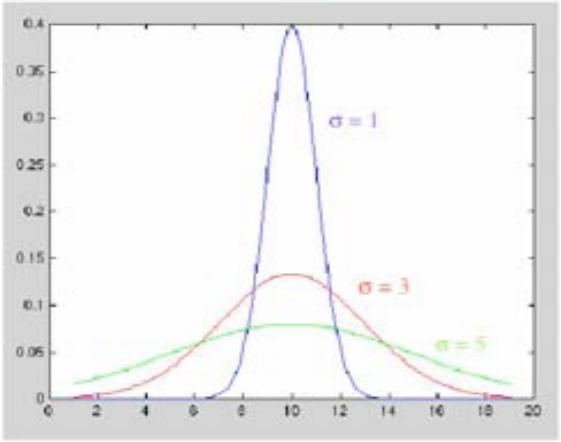
$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- Random variable with Gaussian probability distribution that has the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
- $\mathbf{x}$ and $\boldsymbol{\mu}$ are d-dimensional, $\boldsymbol{\Sigma}$ is d x d.

d=2



d=1



If x is 1-d, we just have one $\boldsymbol{\Sigma}$ parameter → the variance: σ^2

Slide credit: Kristen Grauman

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.

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

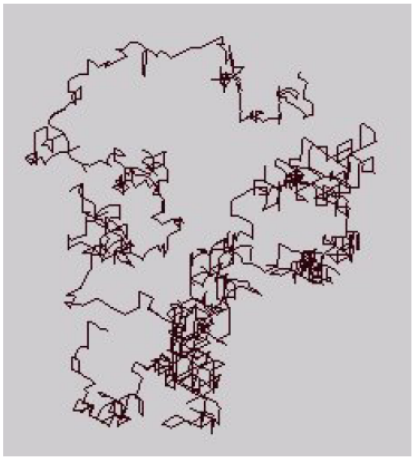
m x 1 m x n n x 1

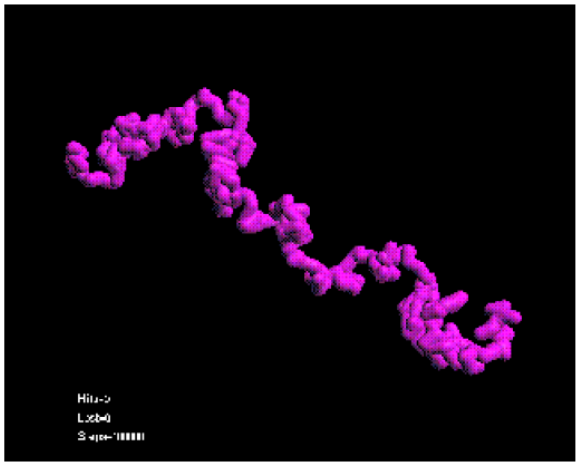
Slide credit: Kristen Grauman

Example: randomly drifting points

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

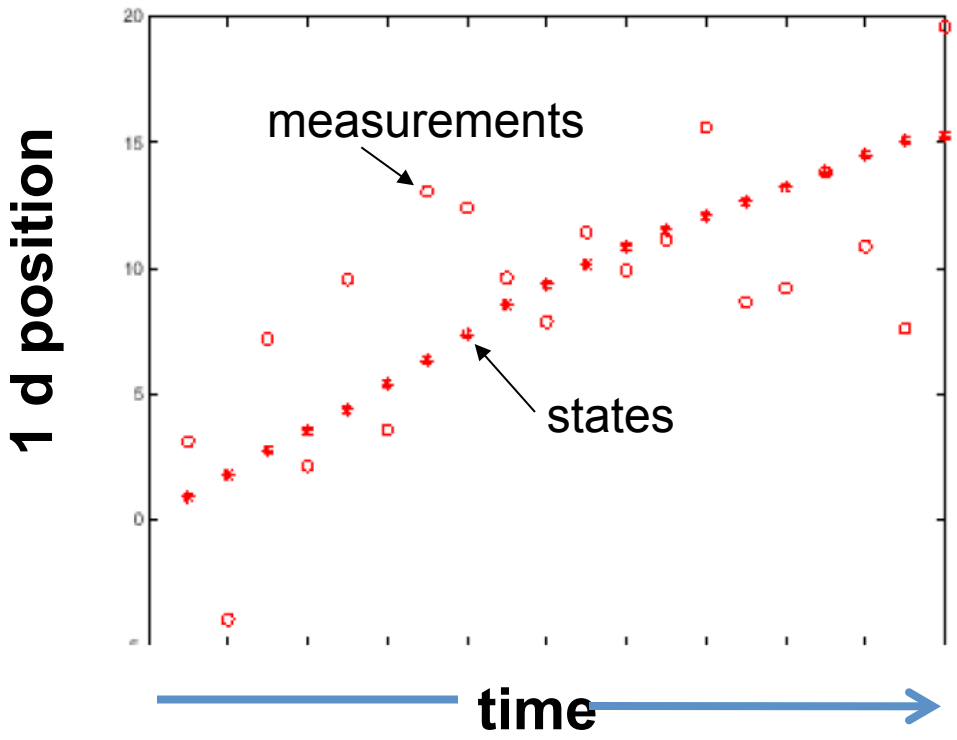
- Consider a stationary object, with state as position
- Position is constant, only motion due to random noise term.
- State evolution is described by identity matrix **D=I**





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

Example: Constant velocity (1D points)



Slide credit: 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

$$y_t = M x_t + noise =$$

Slide credit: 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*

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \qquad p_t = p_{t-1} + (\Delta t)v_{t-1} + \varepsilon$$

$$x_t = D_t x_{t-1} + noise = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + noise$$

- Measurement is position only

$$y_t = Mx_t + noise = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + noise$$

Slide credit: Kristen Grauman

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.

Slide credit: Kristen Grauman

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 in this case (linear dynamics + Gaussian distributions)

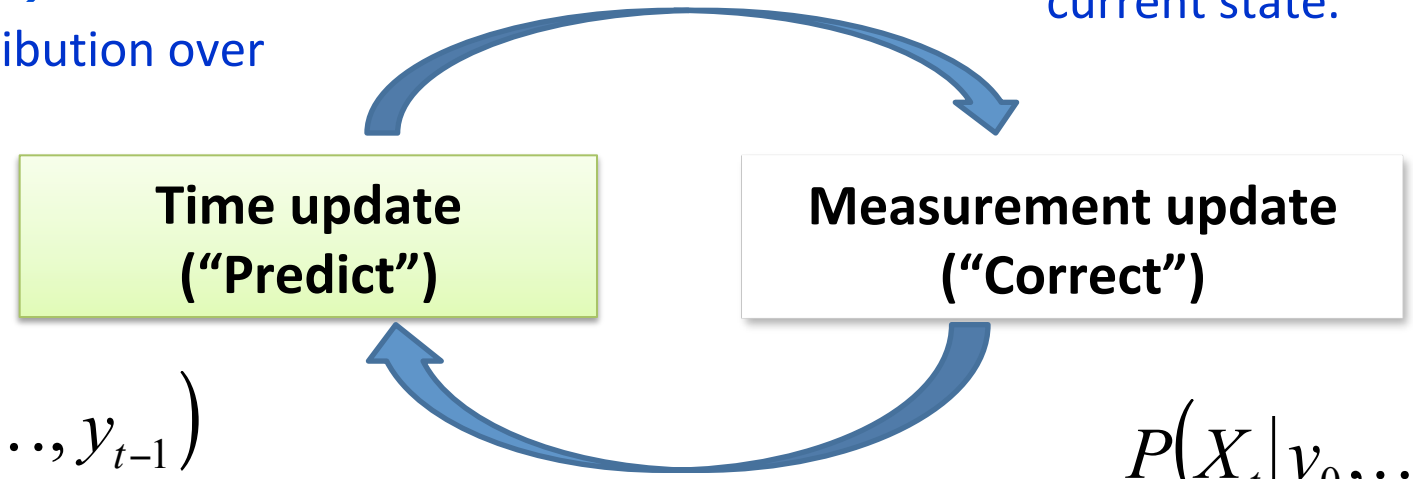
Slide credit: Kristen Grauman

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^+

Slide credit: Kristen Grauman

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:

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

Slide credit: Svetlana Lazebnik

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 measurement:

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

- 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:

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

Slide credit: Svetlana Lazebnik

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!

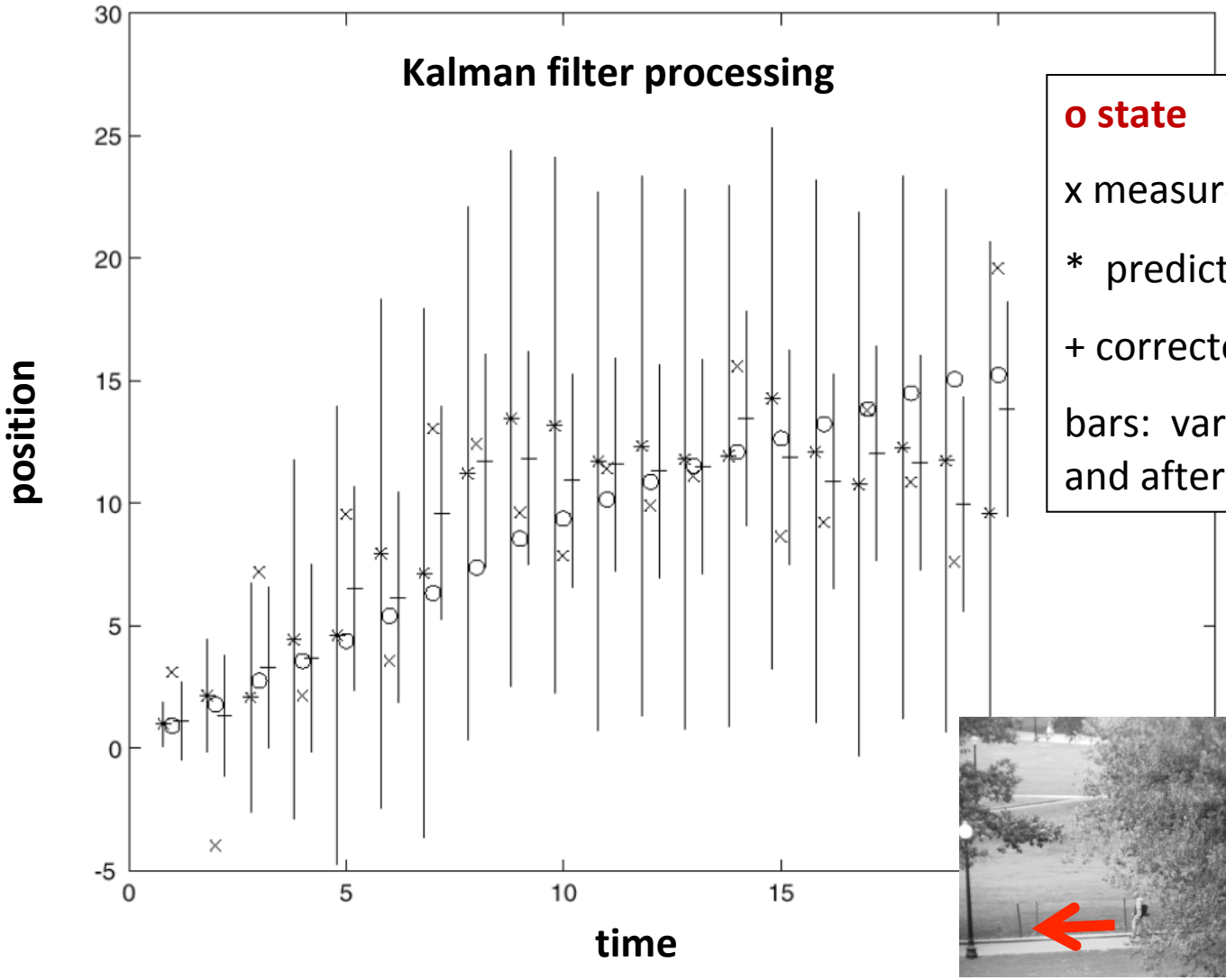
- What if there is no measurement uncertainty $(\sigma_m = 0)$?

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

The prediction is ignored!

Slide credit: Svetlana Lazebnik

Constant velocity model



o state

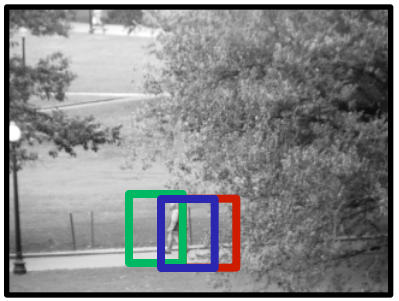
x measurement

* predicted mean estimate

\+ corrected mean estimate

bars: variance estimates before and after measurements

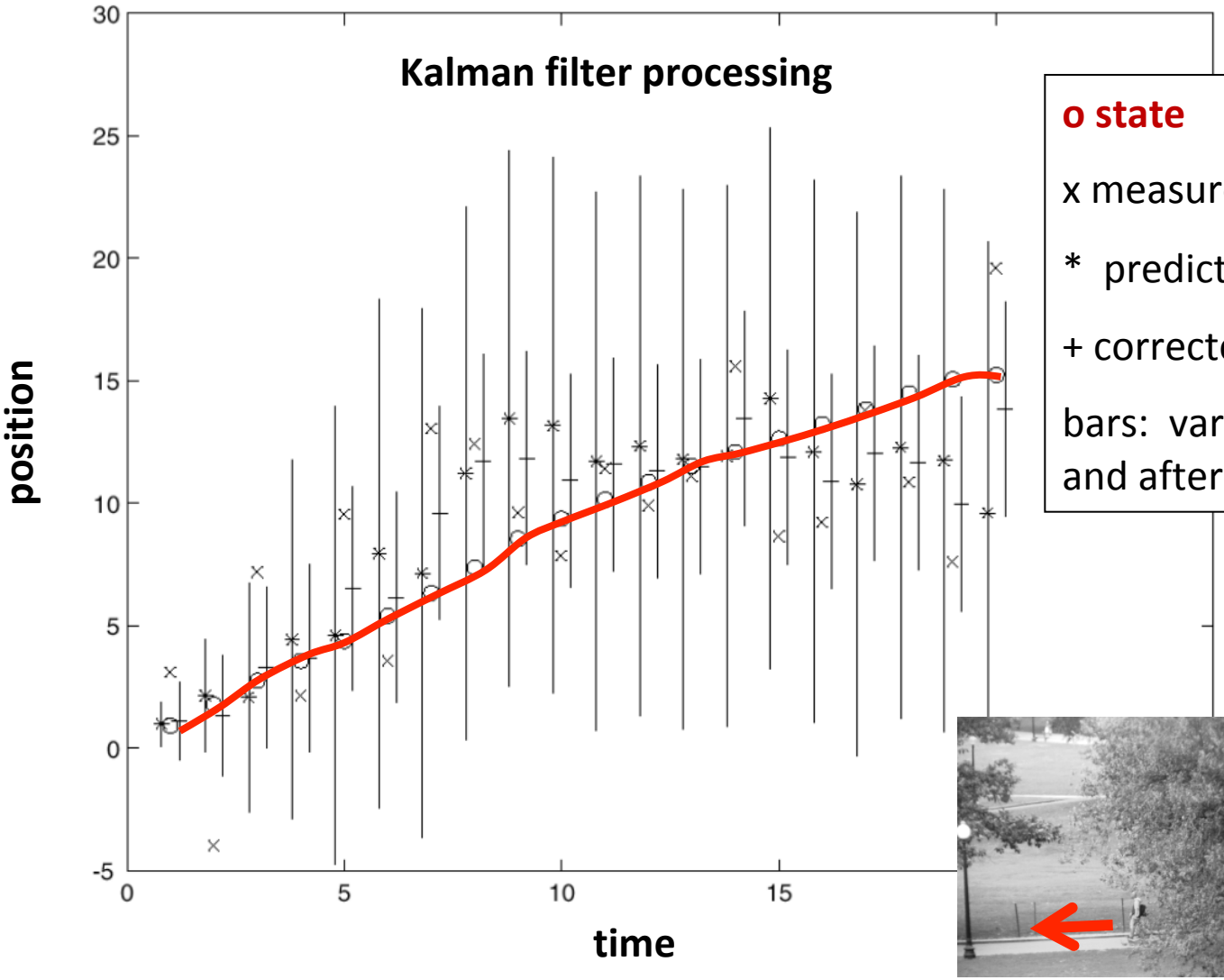
Time t



Time t+1

Slide credit: Svetlana Lazebnik

Constant velocity model



o state

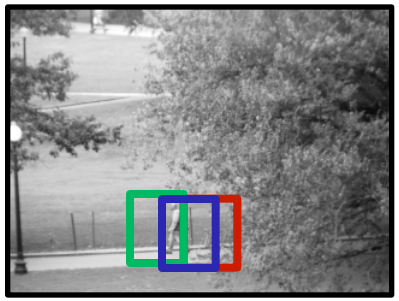
x measurement

* predicted mean estimate

\+ corrected mean estimate

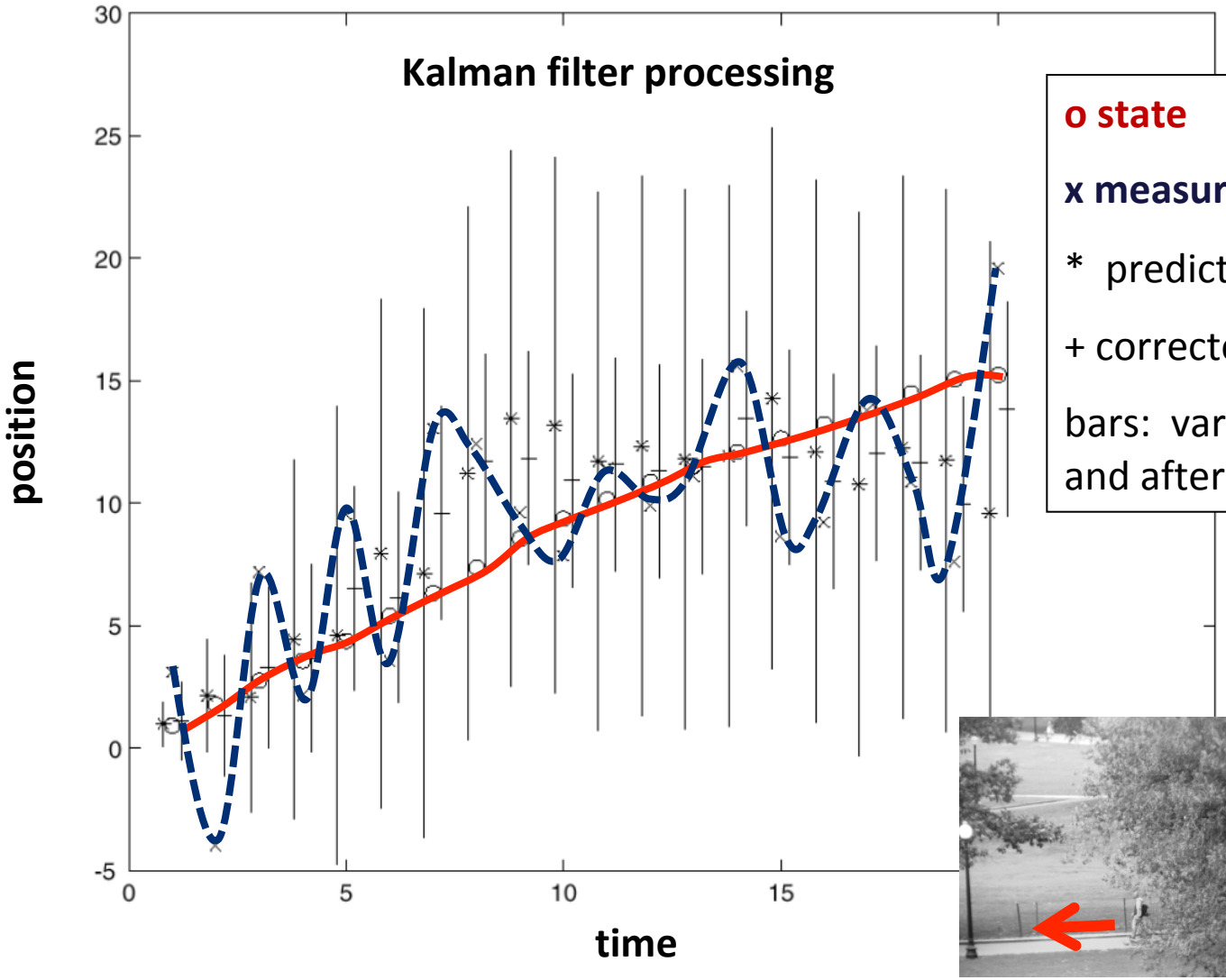
bars: variance estimates before and after measurements

Time t



Time t+1

Constant velocity model



o state

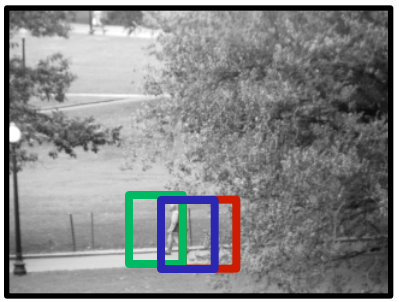
x measurement

* predicted mean estimate

\+ corrected mean estimate

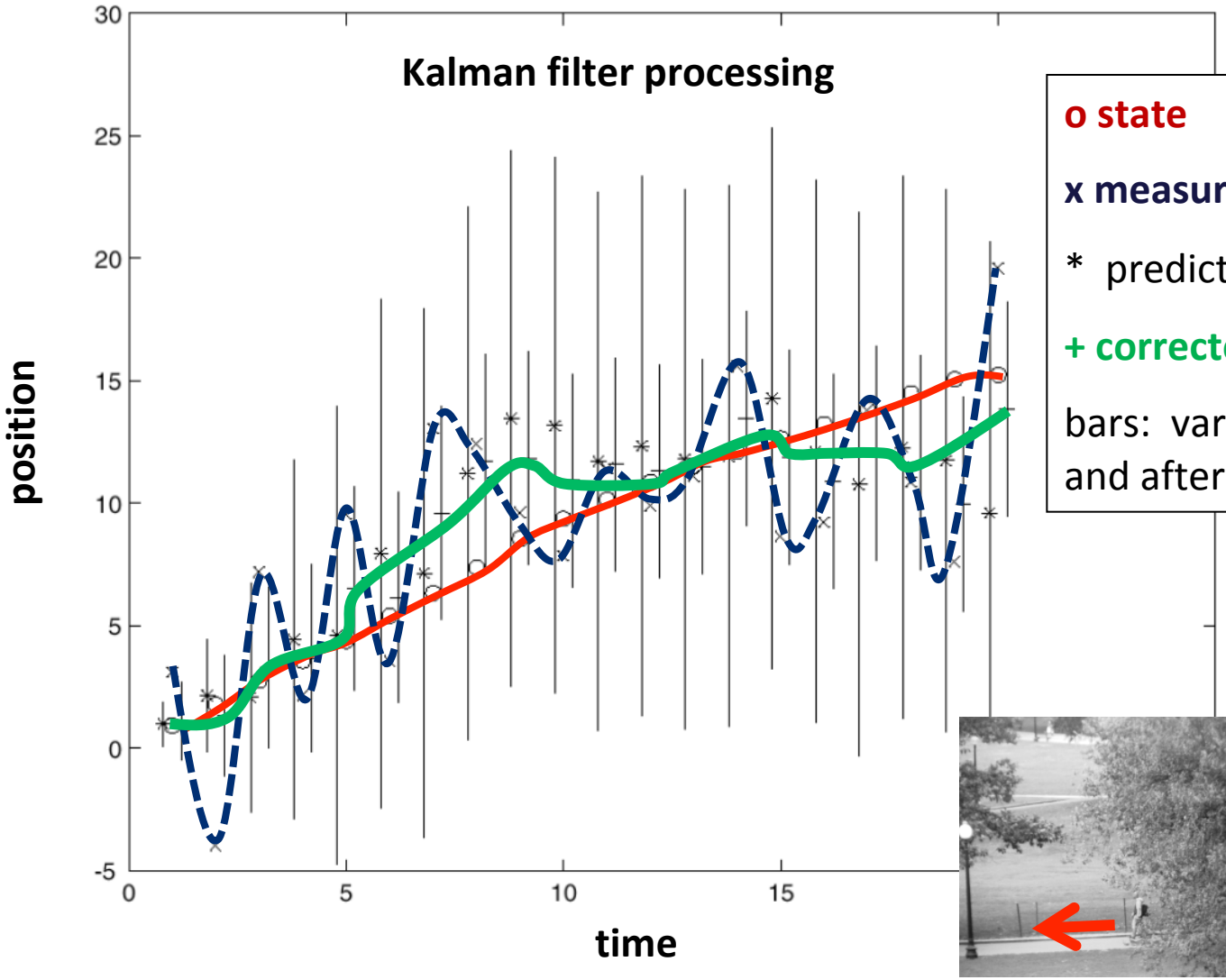
bars: variance estimates before and after measurements

Time t



Time t+1

Constant velocity model



o state

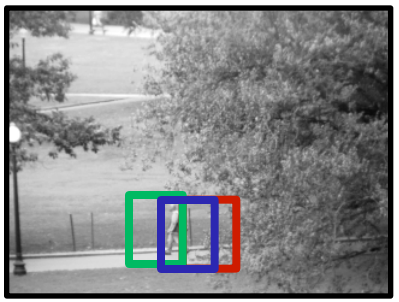
x measurement

* predicted mean estimate

+ corrected mean estimate

bars: variance estimates before and after measurements

Time t



Time t+1

Kalman equations on the general case

Dynamic Model:

$$\boldsymbol{x}_i \sim N(\mathcal{D}_i \boldsymbol{x}_{i-1}, \Sigma_{d_i})$$

$$\boldsymbol{y}_i \sim N(\mathcal{M}_i \boldsymbol{x}_i, \Sigma_{m_i})$$

Start Assumptions: $\overline{\boldsymbol{x}_0}$ and Σ_0^- are known

Update Equations: Prediction

$$\overline{\boldsymbol{x}_i} = \mathcal{D}_i \overline{\boldsymbol{x}}_{i-1}^+$$

$$\Sigma_i^- = \Sigma_{d_i} + \mathcal{D}_i \sigma_{i-1}^+ \mathcal{D}_i$$

Update Equations: Correction

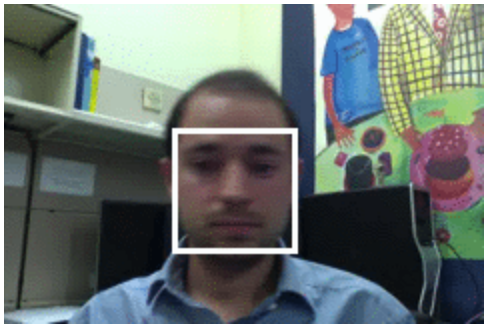
$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i}\right]^{-1}$$

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}_i} + \mathcal{K}_i \left[\boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}_i}\right]$$

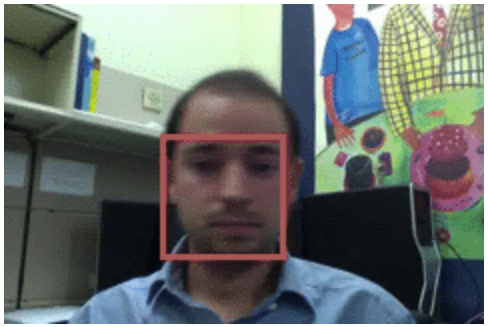
$$\Sigma_i^+ = \left[Id - \mathcal{K}_i \mathcal{M}_i\right] \Sigma_i^-$$

Forsyth and Ponce: Computer Vision: a modern approach, pp 538

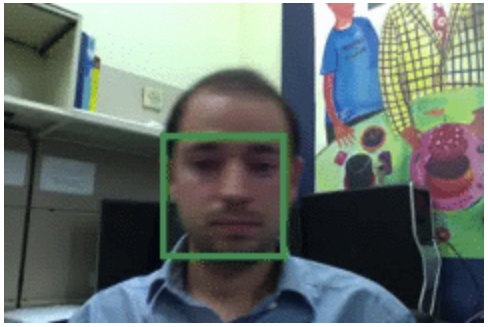
Example: Kalman Filter



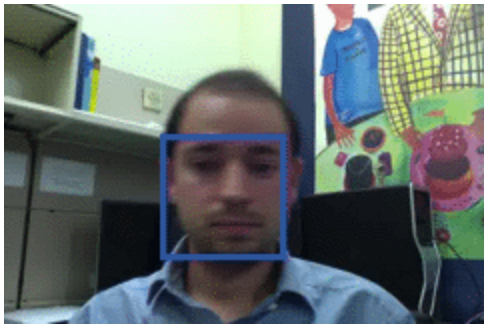
Ground Truth



Observation



Correction

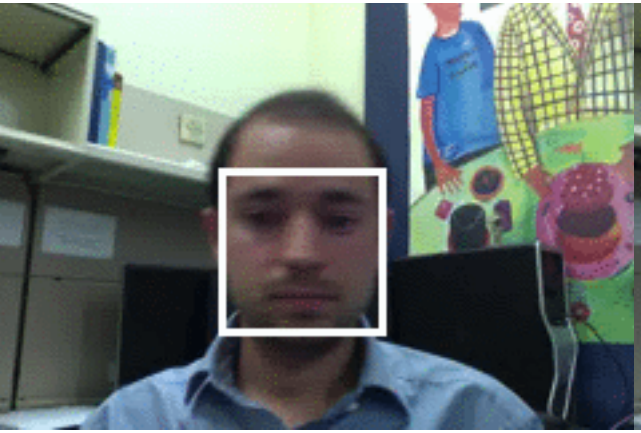


Prediction

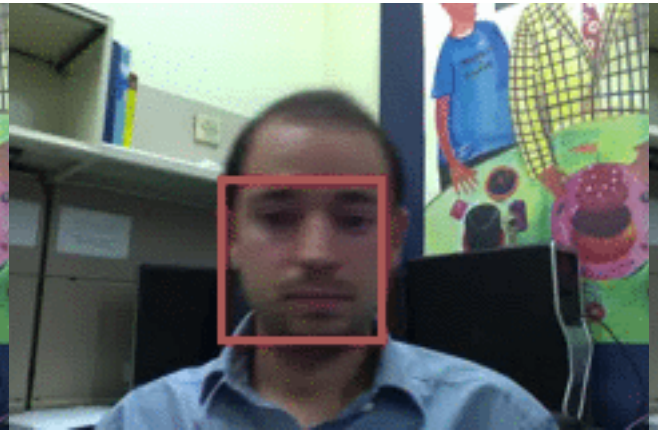
Update Location, Velocity, etc.

Next Frame

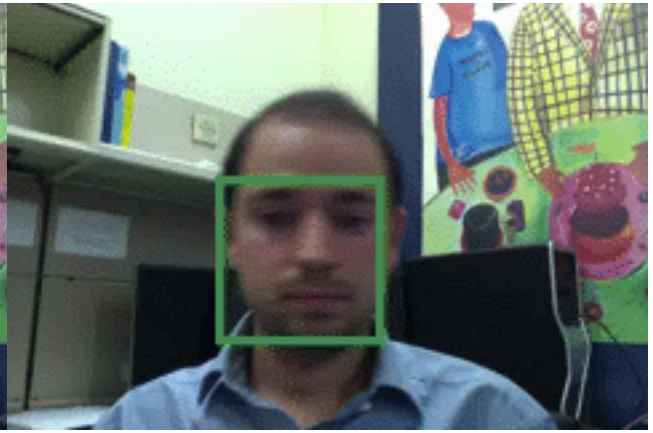
Comparison



Ground Truth

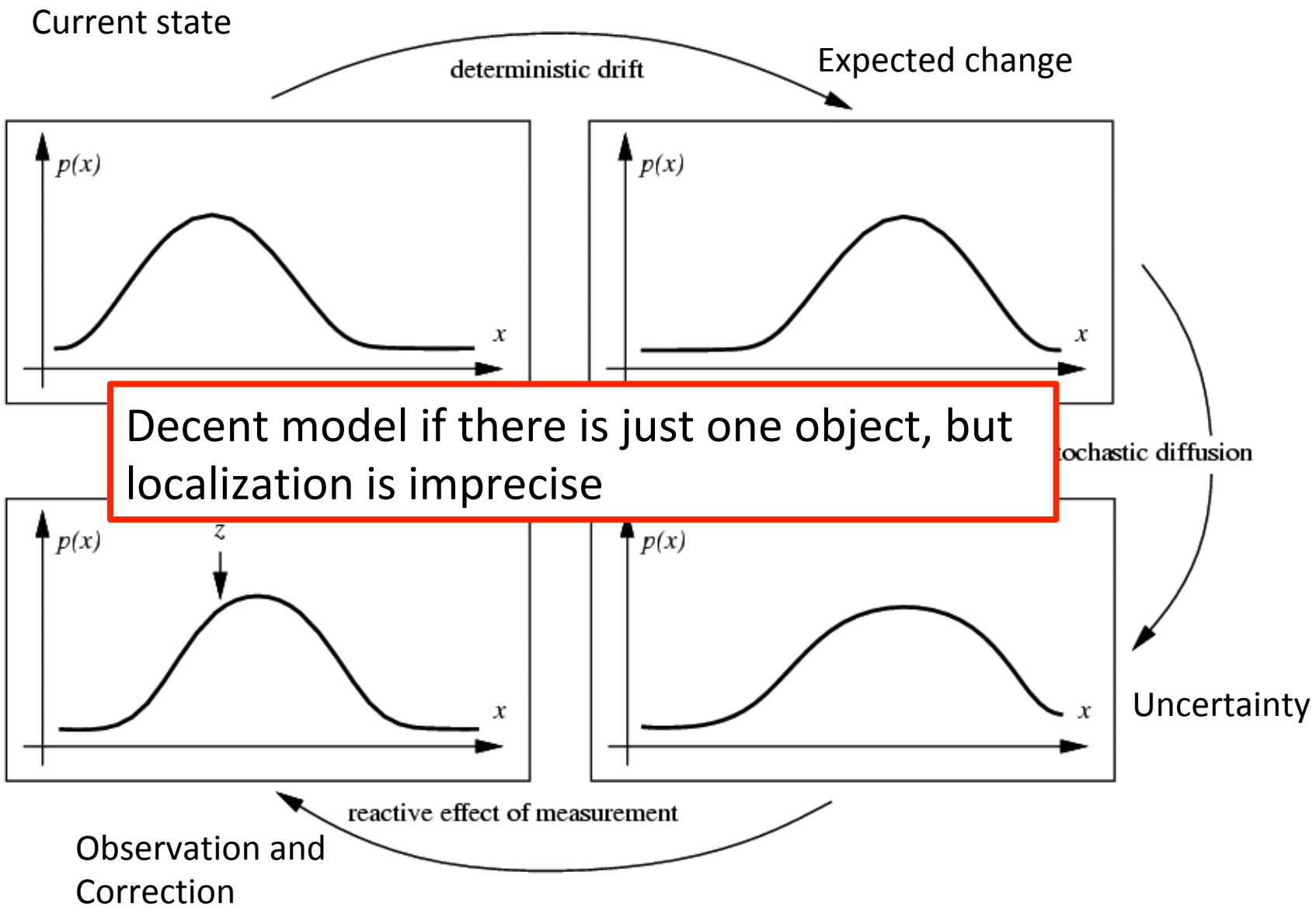


Observation

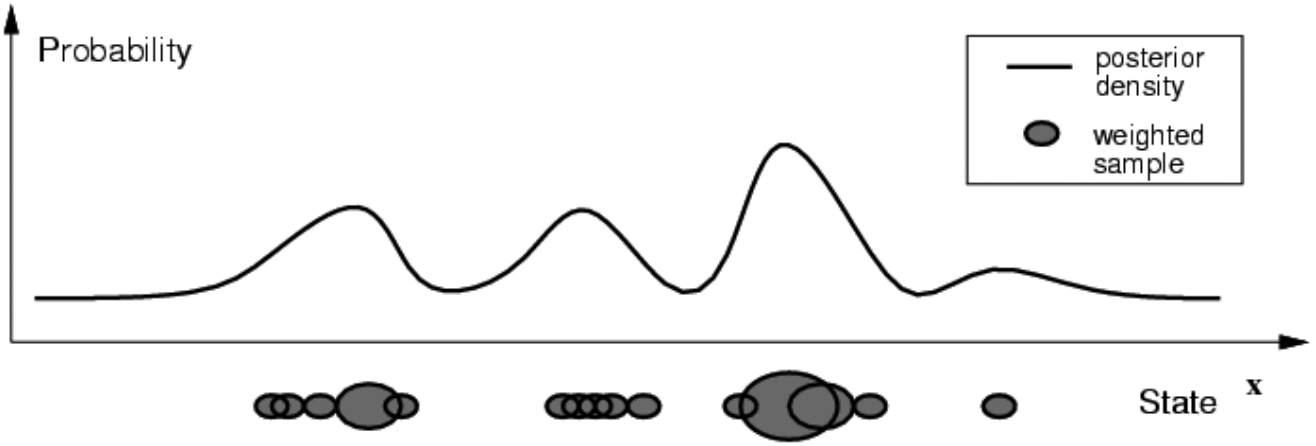


Correction

Propagation of Gaussian densities



Particle filtering

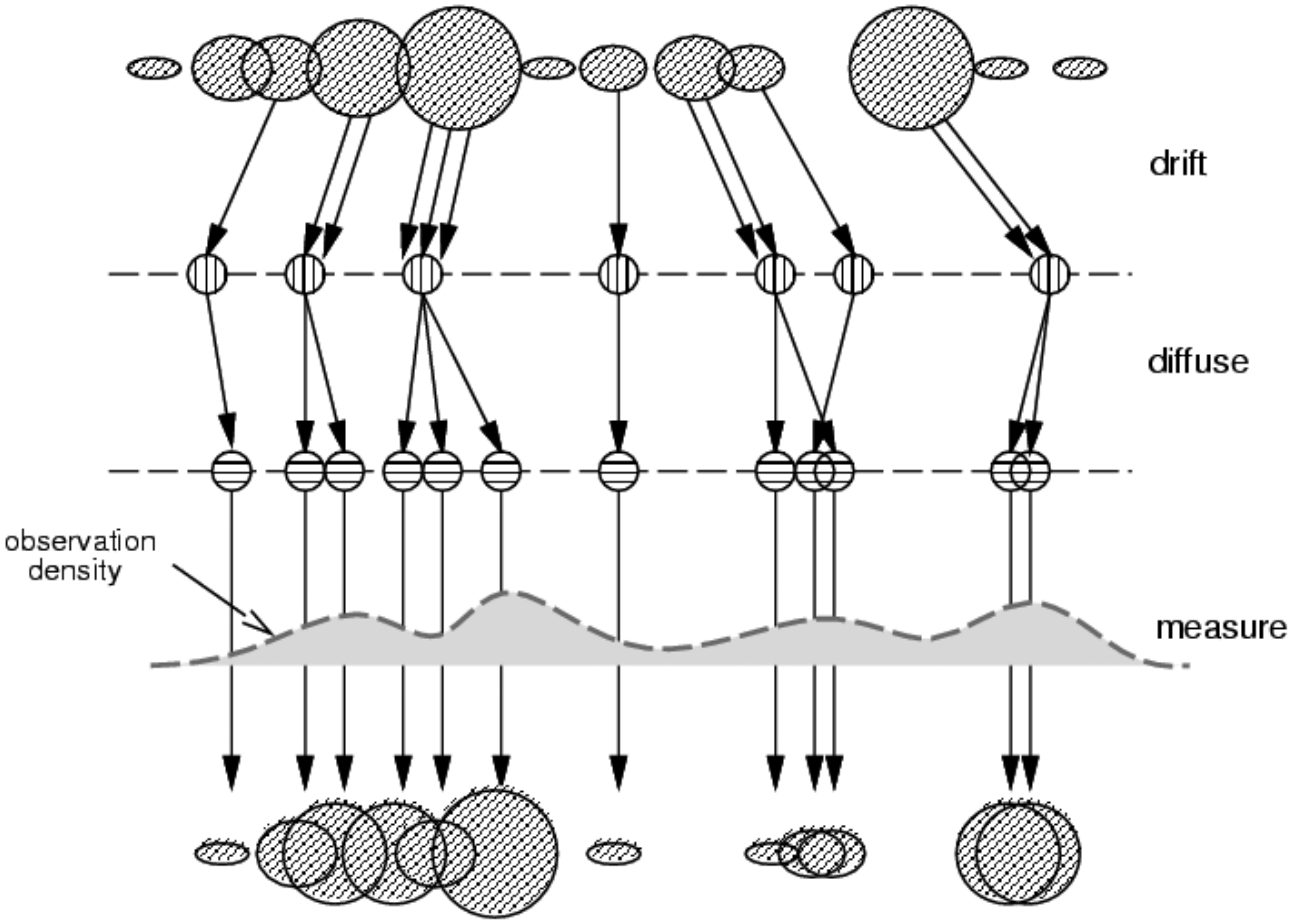


Represent the state distribution non-parametrically

- Prediction: Sample possible values X_{t-1} for the previous state
- Correction: Compute likelihood of X_t based on weighted samples and $P(y_t|X_t)$

M. Isard and A. Blake,
CONDENSATION -- conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998

Particle filtering



Start with weighted samples from previous time step

Sample and shift according to dynamics model

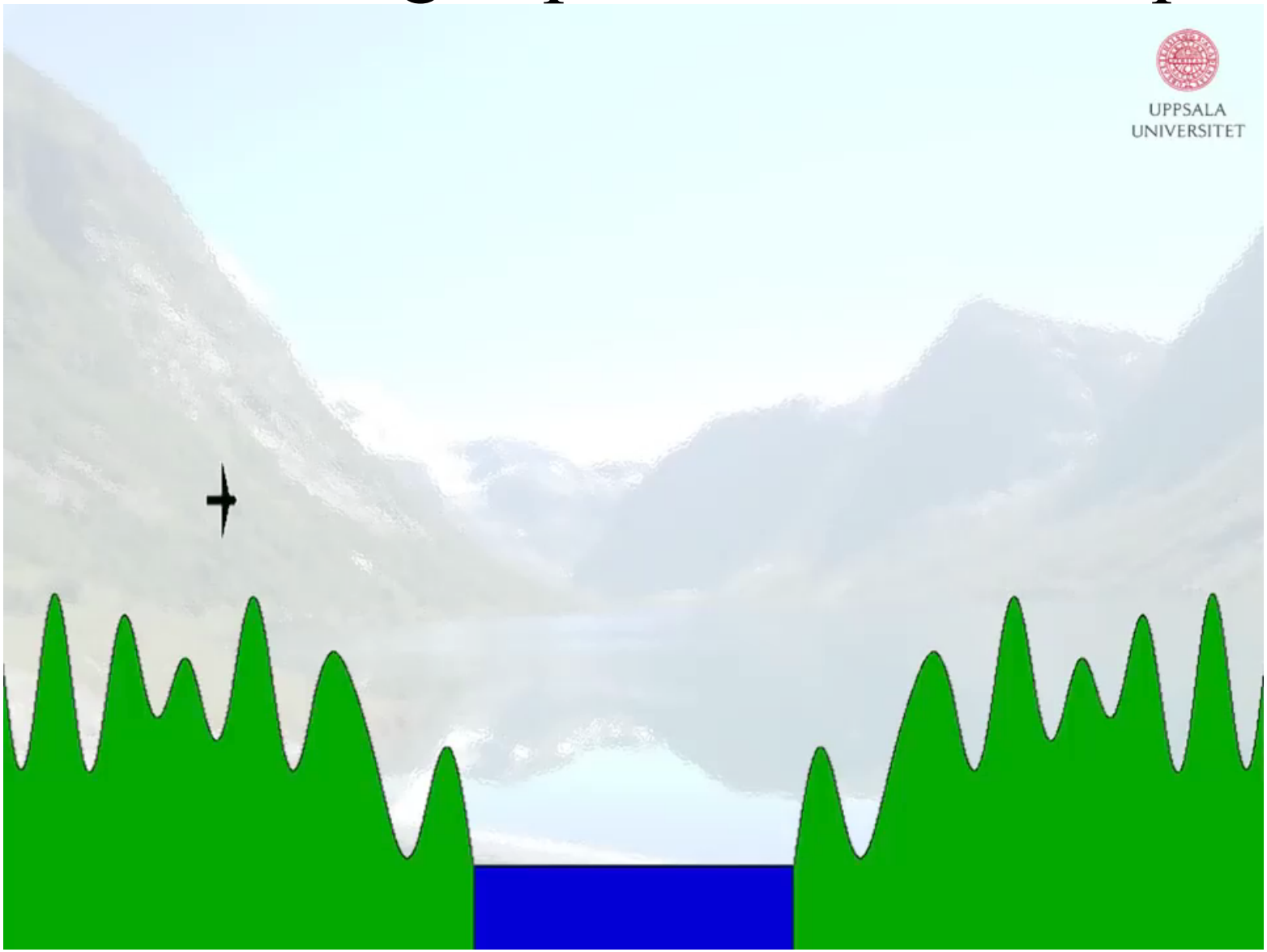
Spread due to randomness; this is predicted density $P(X_t|Y_{t-1})$

Weight the samples according to observation density

Arrive at corrected density estimate $P(X_t|Y_t)$

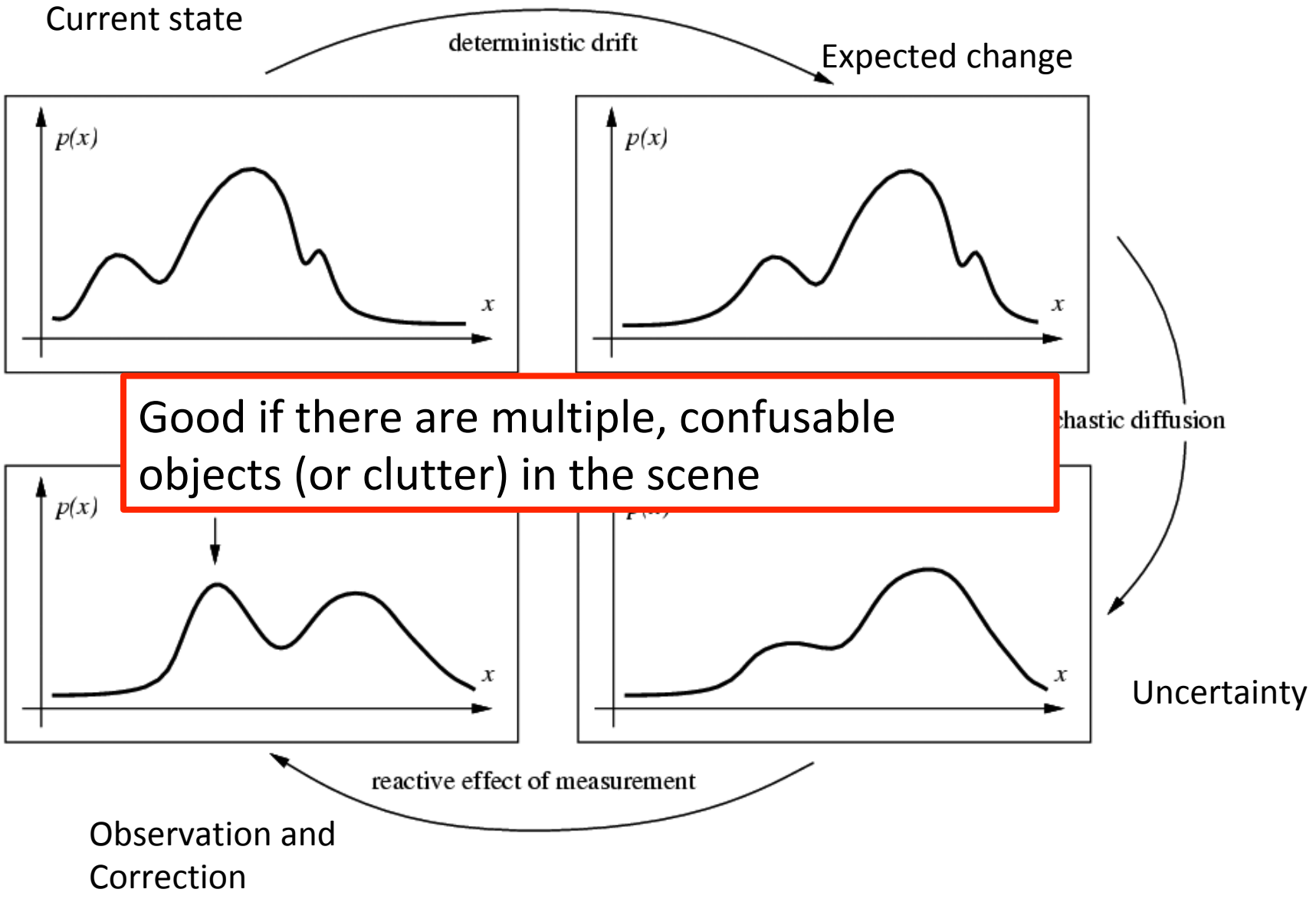
M. Isard and A. Blake,
CONDENSATION -- conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998

Particle filtering explained without equations

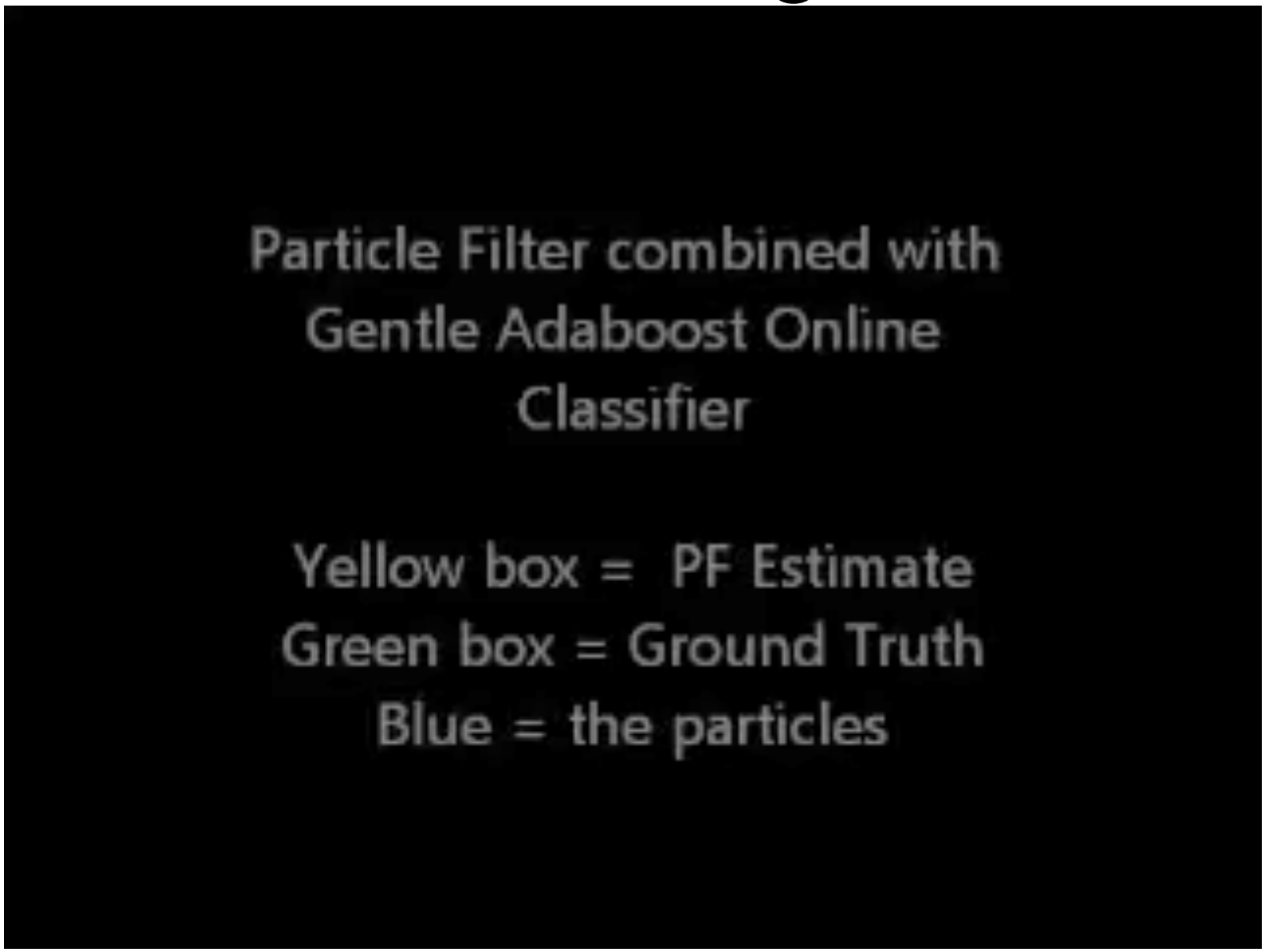


https://www.youtube.com/watch?v=aUkBa1zMKv4

Propagation of non-parametric densities



Particle filtering results



https://www.youtube.com/watch?v=O1FZyWz_yj4

Particle filtering results

https://www.youtube.com/watch?v=tIjufInUqZM

Tracking issues

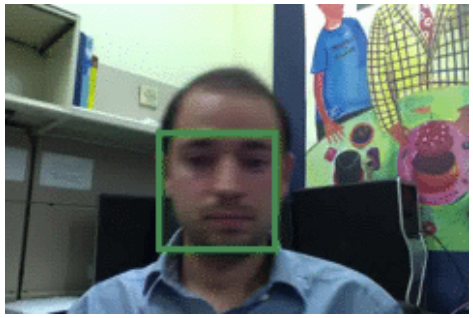
- Initialization
  - Manual
  - Background subtraction
  - Detection

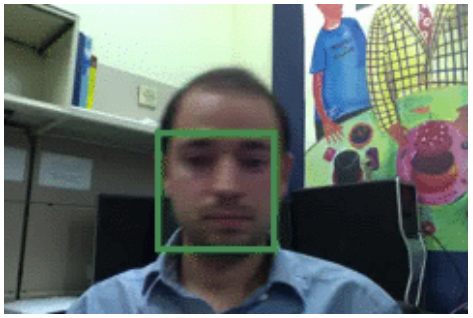
Tracking issues

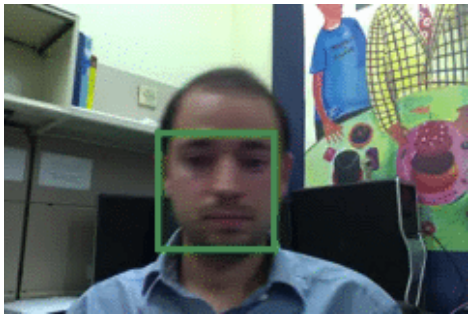
- Initialization
- Getting observation and dynamics models
  - Observation model: match a template or use a trained detector
  - Dynamics model: usually specify using domain knowledge

Tracking issues

- Initialization
- Obtaining observation and dynamics model
- Uncertainty of prediction vs. correction
  - If the dynamics model is too strong, will end up ignoring the data
  - If the observation model is too strong, tracking is reduced to repeated detection





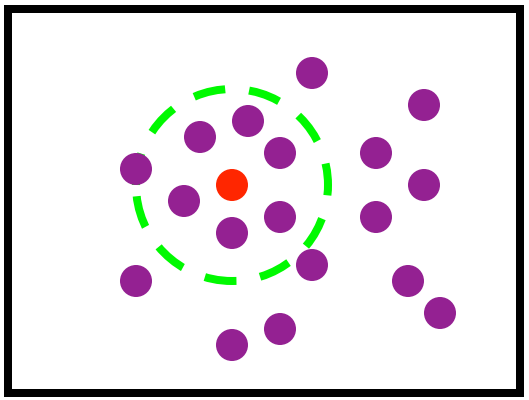


Too strong dynamics model

Too strong observation model

Tracking issues

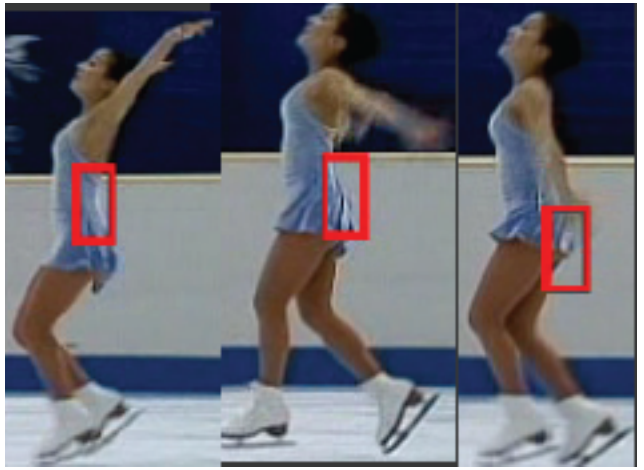
- Initialization
- Getting observation and dynamics models
- Prediction vs. correction
- Data association
  - When tracking multiple objects, need to assign right objects to right tracks (particle filters good for this)



Tracking issues

- Initialization
- Getting observation and dynamics models
- Prediction vs. correction
- Data association
- Drift
  - Errors can accumulate over time

Drift



D. Ramanan, D. Forsyth, and A. Zisserman.
Tracking People by Learning their Appearance. PAMI 2007.

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

4. **Object Recognition: high – level vision**

   Object classification, object detection, part based models, bovw models

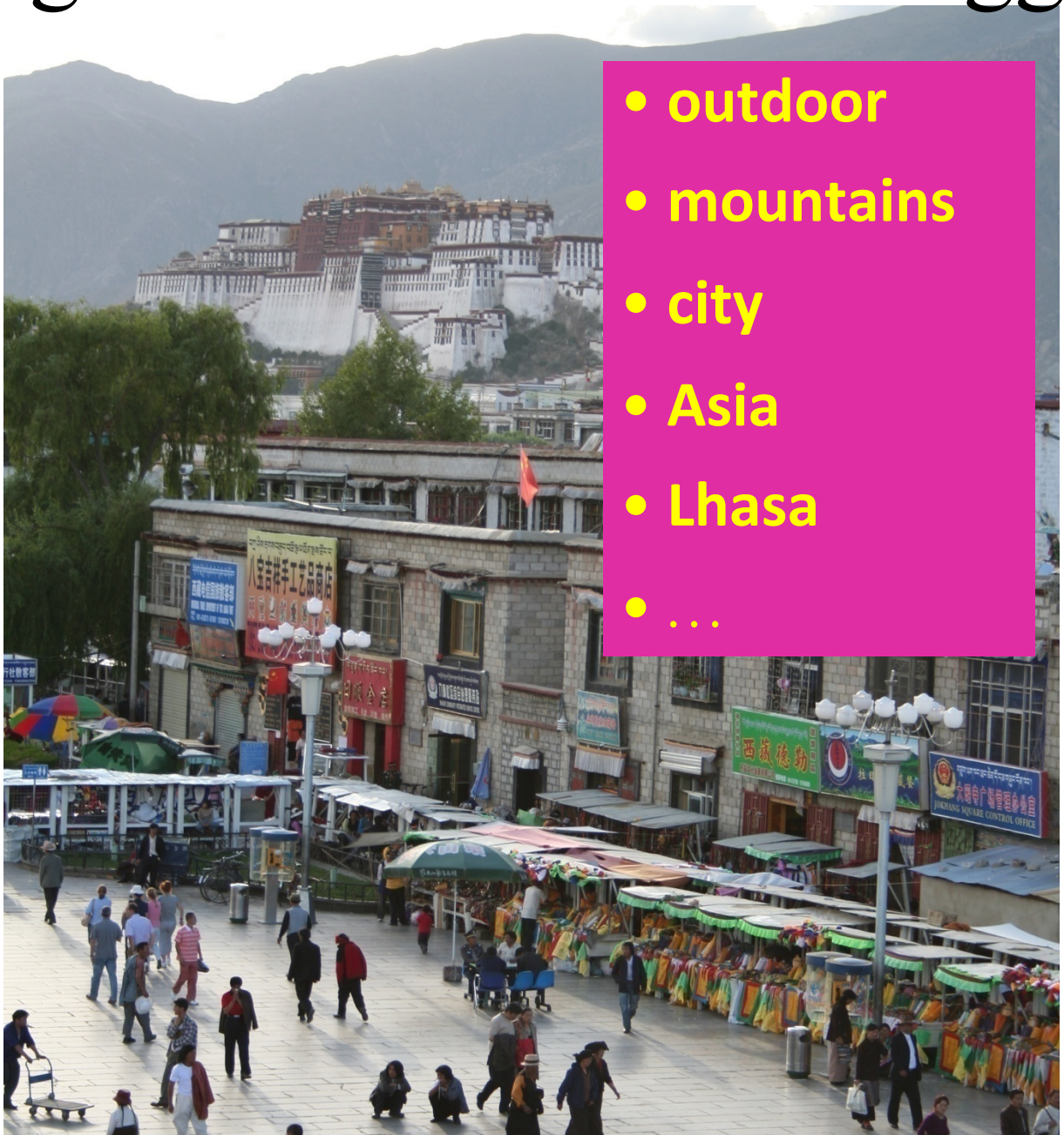
5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Common recognition tasks

Slide credit
Fei-Fei Li

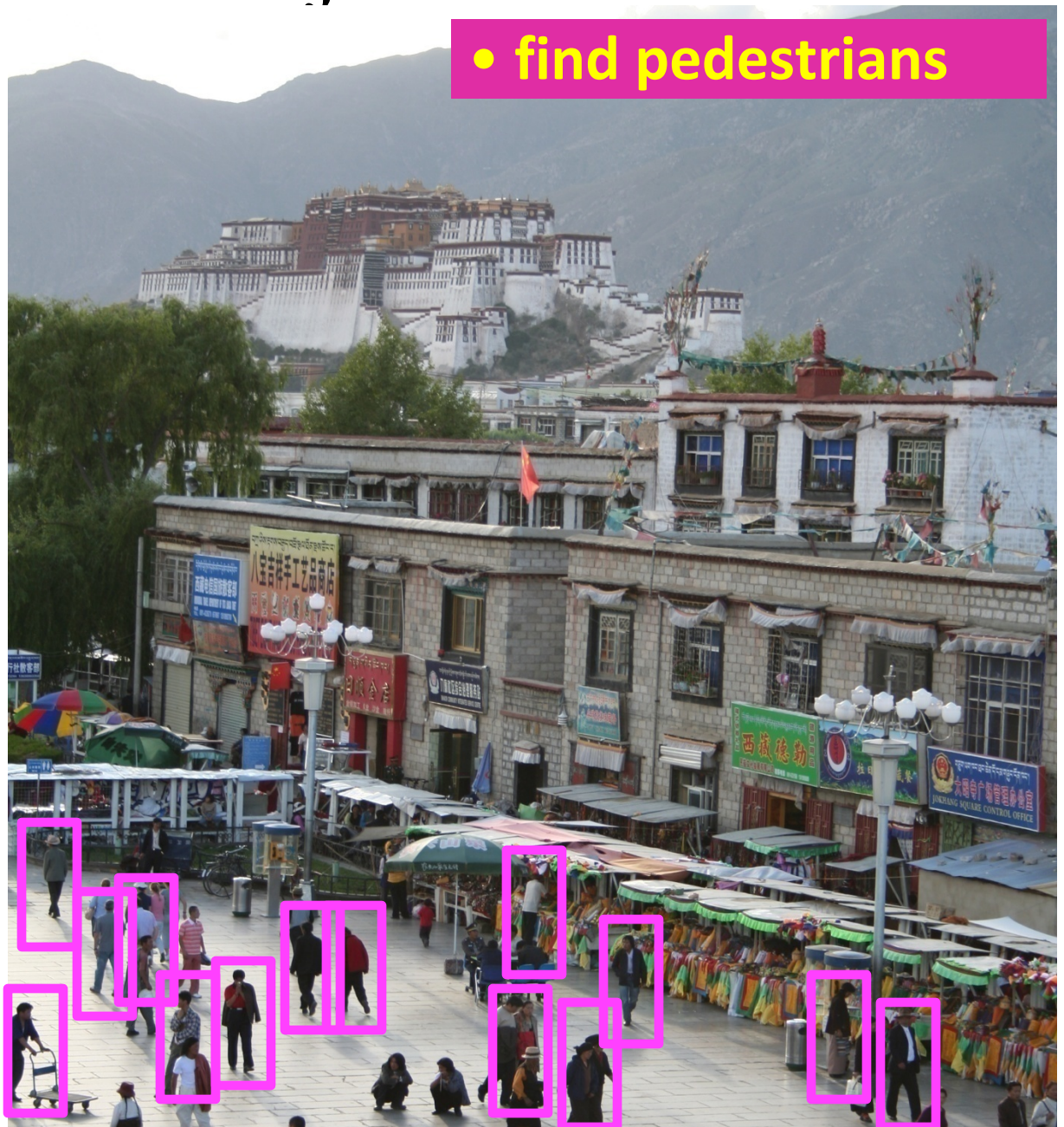
Image classification and tagging



- outdoor
- mountains
- city
- Asia
- Lhasa
- …

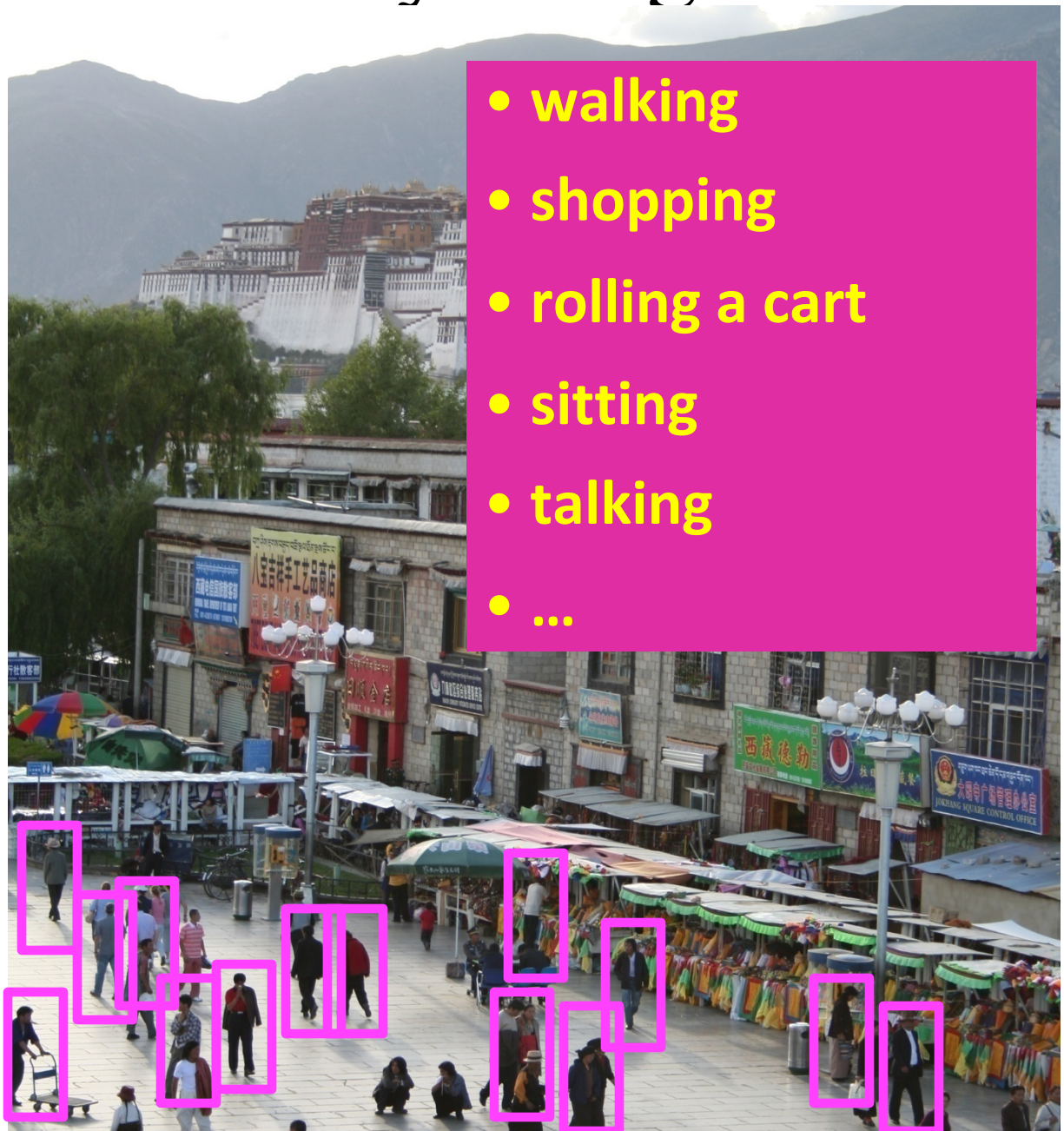
Slide credit
Fei-Fei Li

Object detection



Slide credit
Fei-Fei Li

Activity recognition



Slide credit
Fei-Fei Li

Semantic segmentation

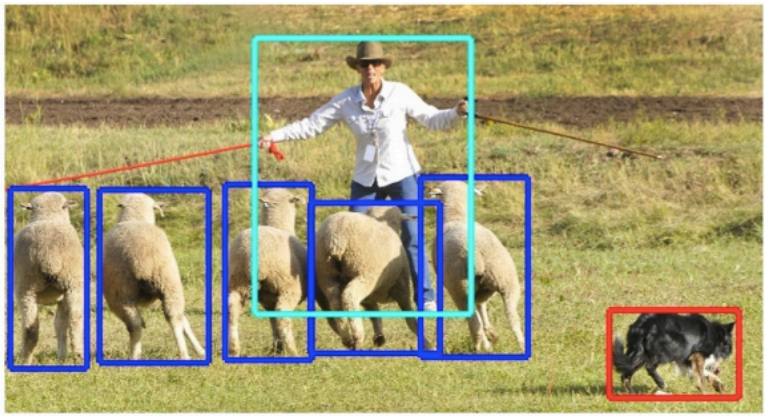
Slide credit
Fei-Fei Li

Semantic segmentation

Slide credit
Fei-Fei Li

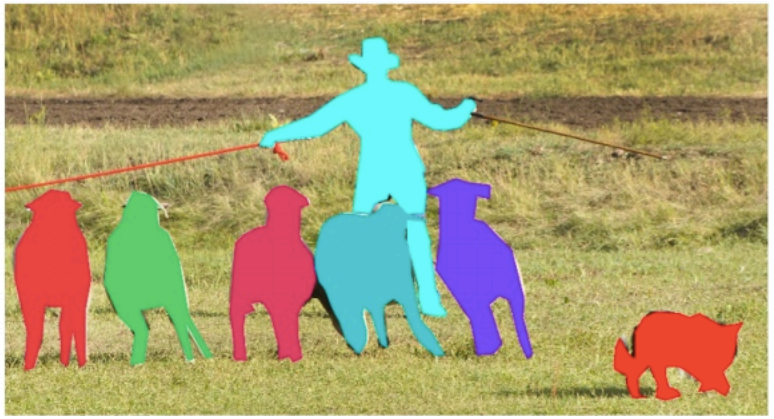
Detection, semantic segmentation, instance segmentation

image classification



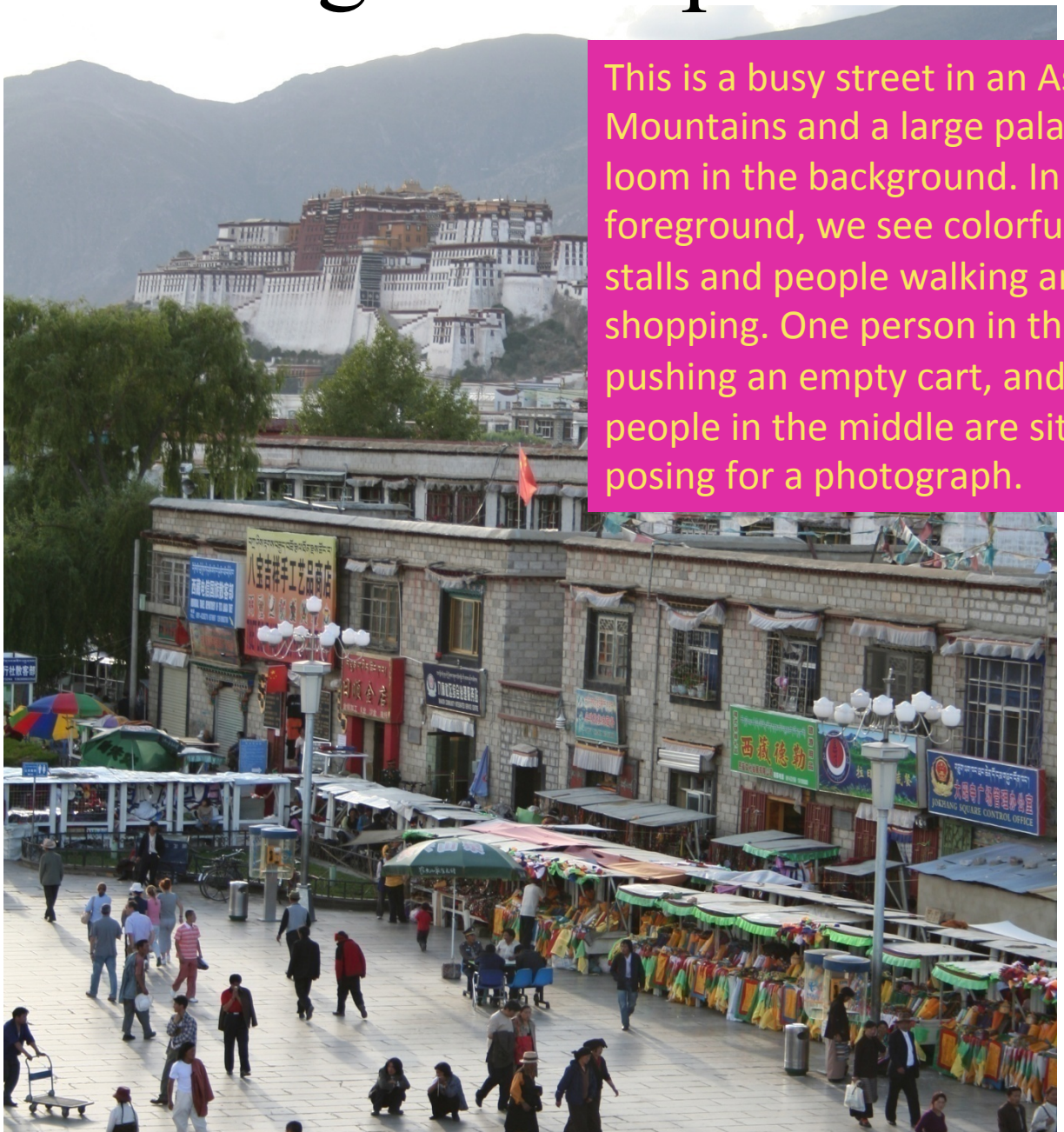
object detection

semantic segmentation



instance segmentation

Image description



This is a busy street in an Asian city. Mountains and a large palace or fortress loom in the background. In the foreground, we see colorful souvenir stalls and people walking around and shopping. One person in the lower left is pushing an empty cart, and a couple of people in the middle are sitting, possibly posing for a photograph.

Slide credit
Fei-Fei Li

Image classification

The statistical learning framework

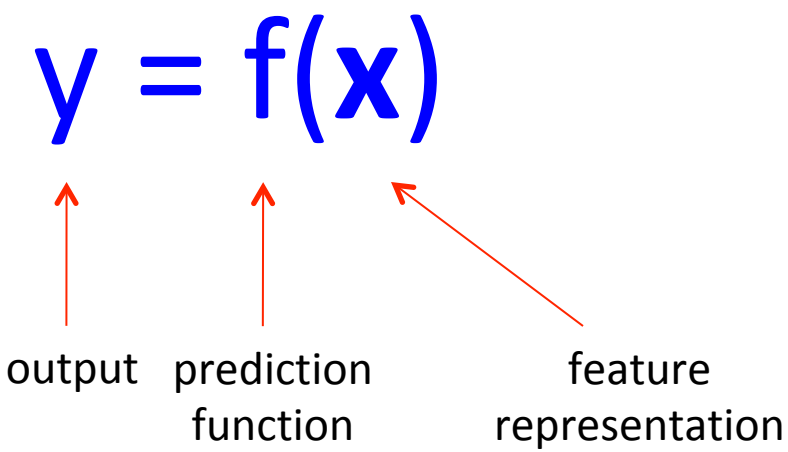
- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

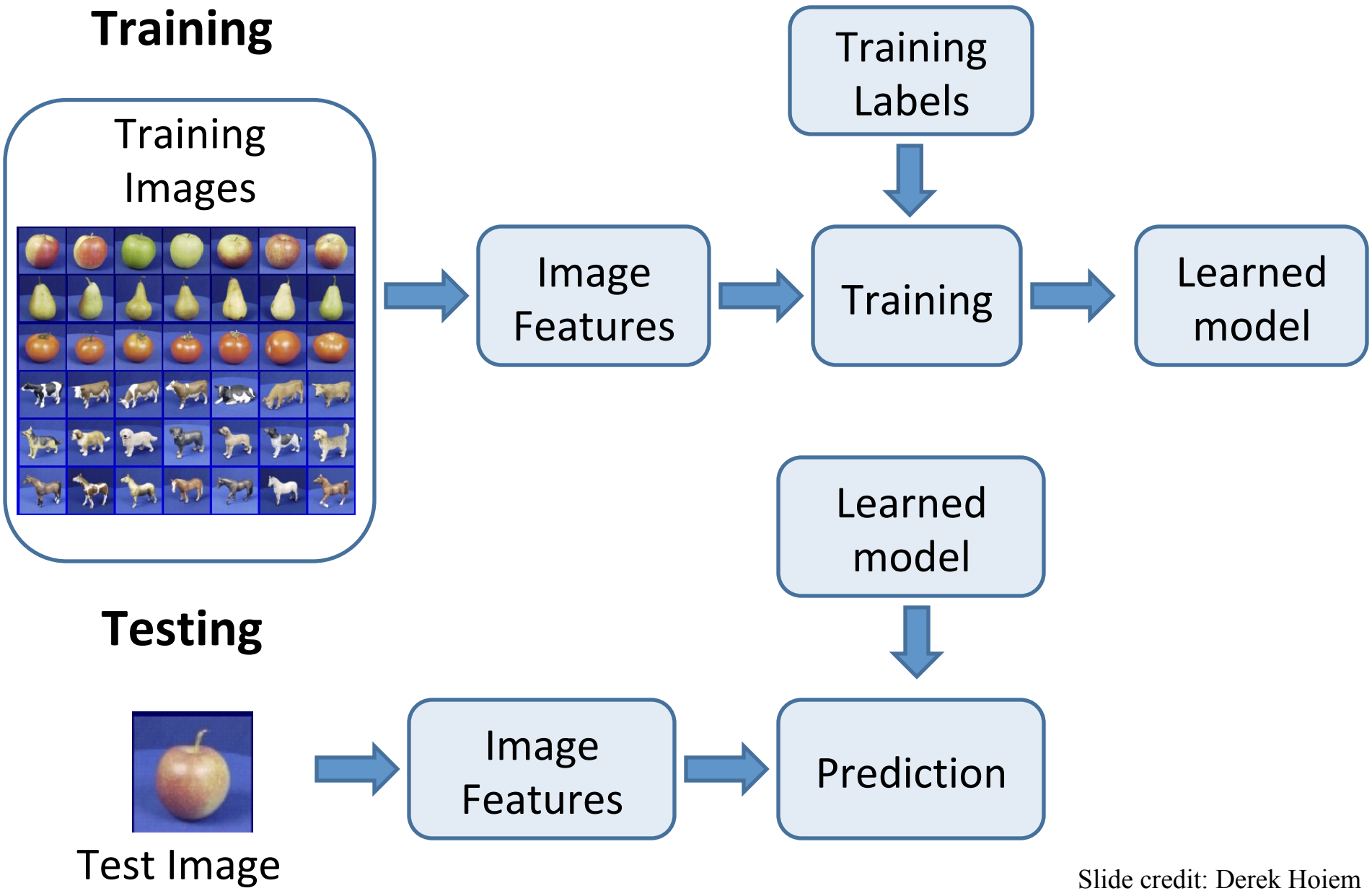
f() = “cow”

The statistical learning framework



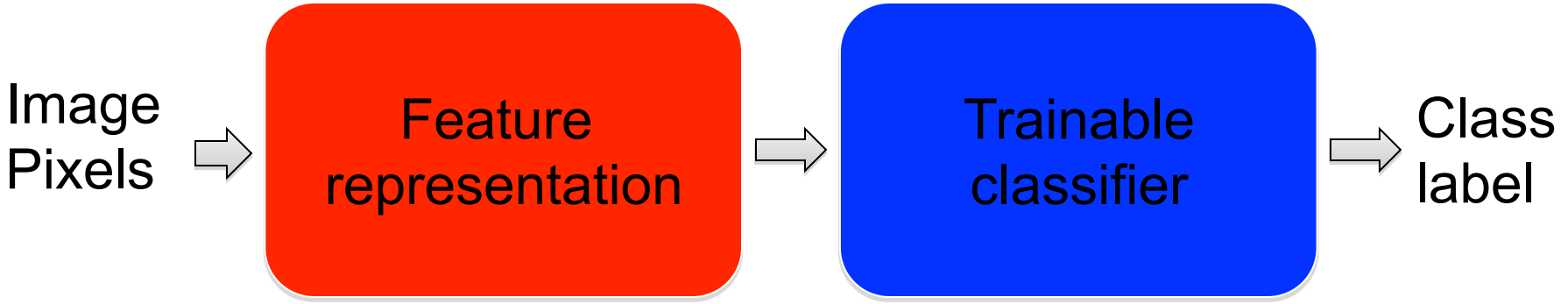
- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Steps



Slide credit: Derek Hoiem

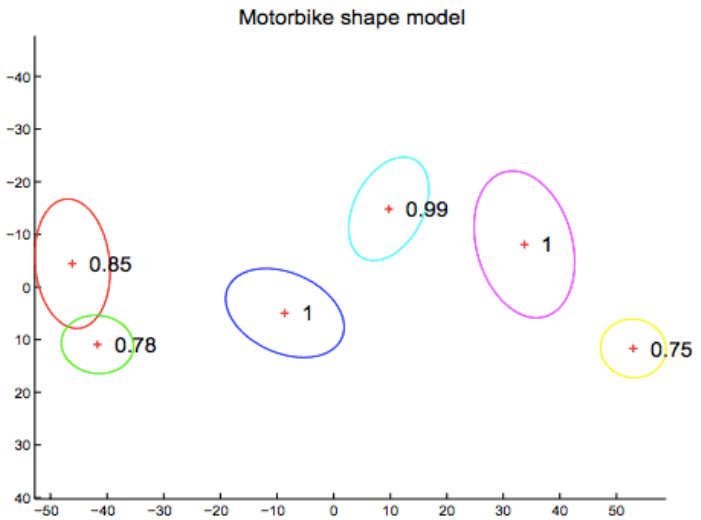
“Classic” recognition pipeline

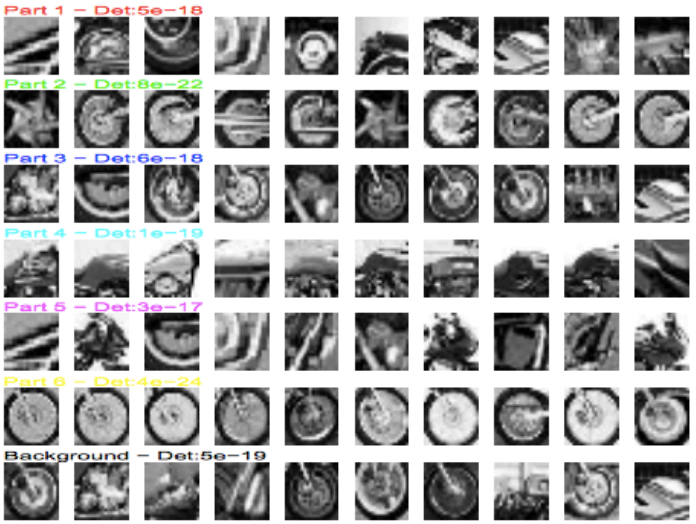


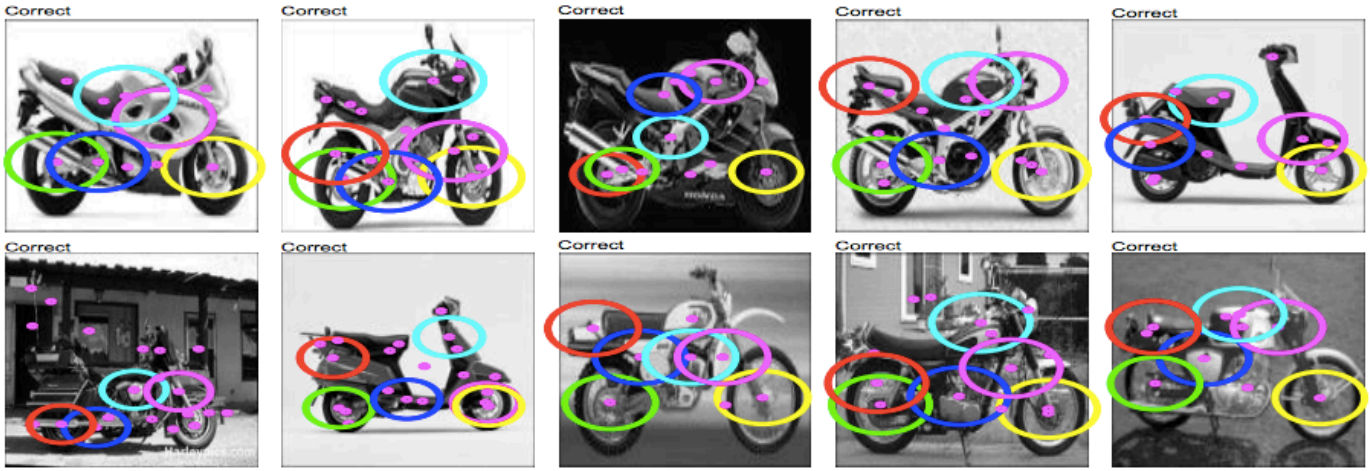
- Hand-crafted feature representation
- Off-the-shelf trainable classifier (SVM, k-NN)

“Classic” representation: Bag of features

Motivation 1: Part-based models

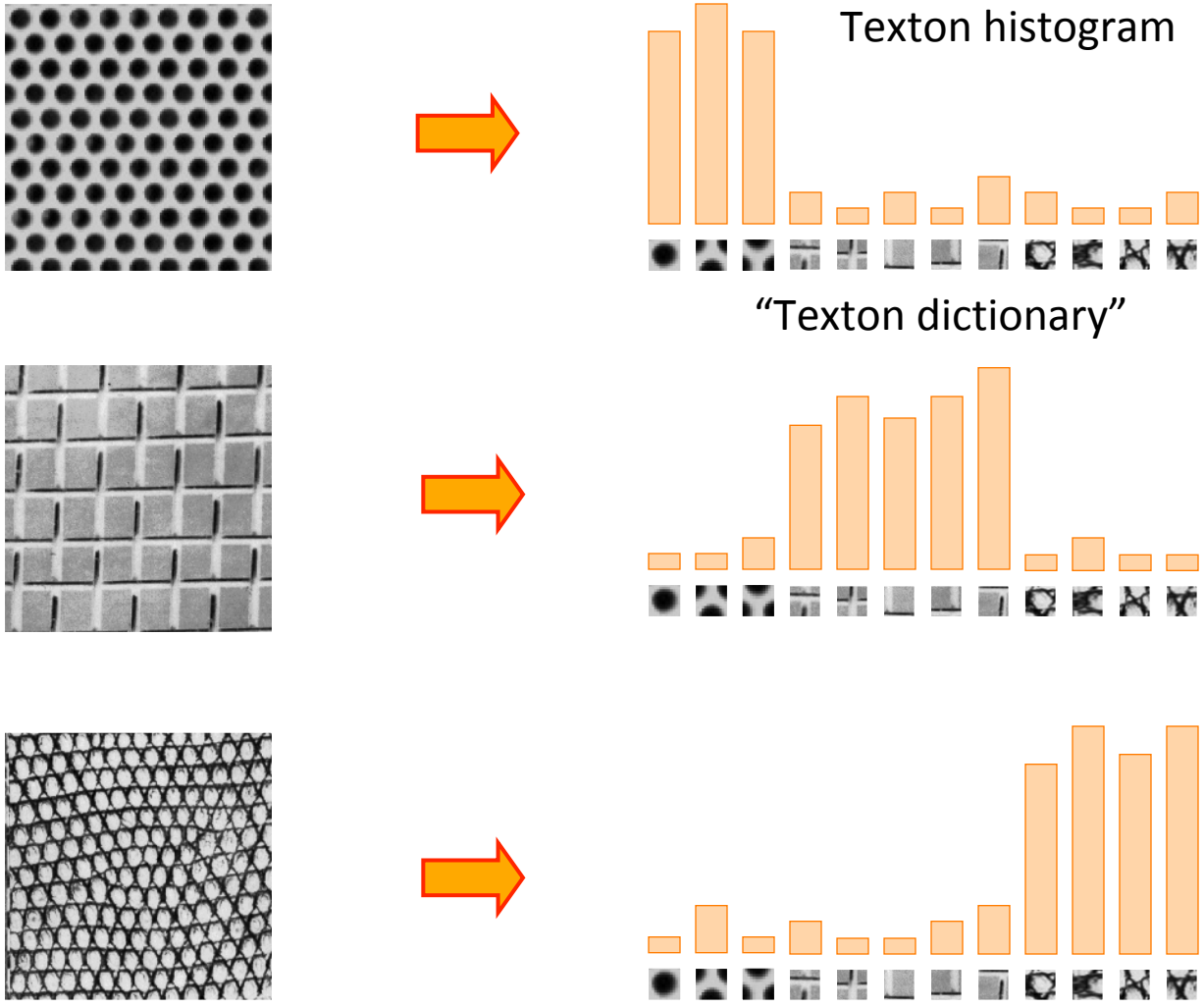






Weber, Welling & Perona (2000), Fergus, Perona & Zisserman (2003)

Motivation 2: Texture models



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Motivation 3: Bags of words

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Motivation 3: Bags of words

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Motivation 3: Bags of words

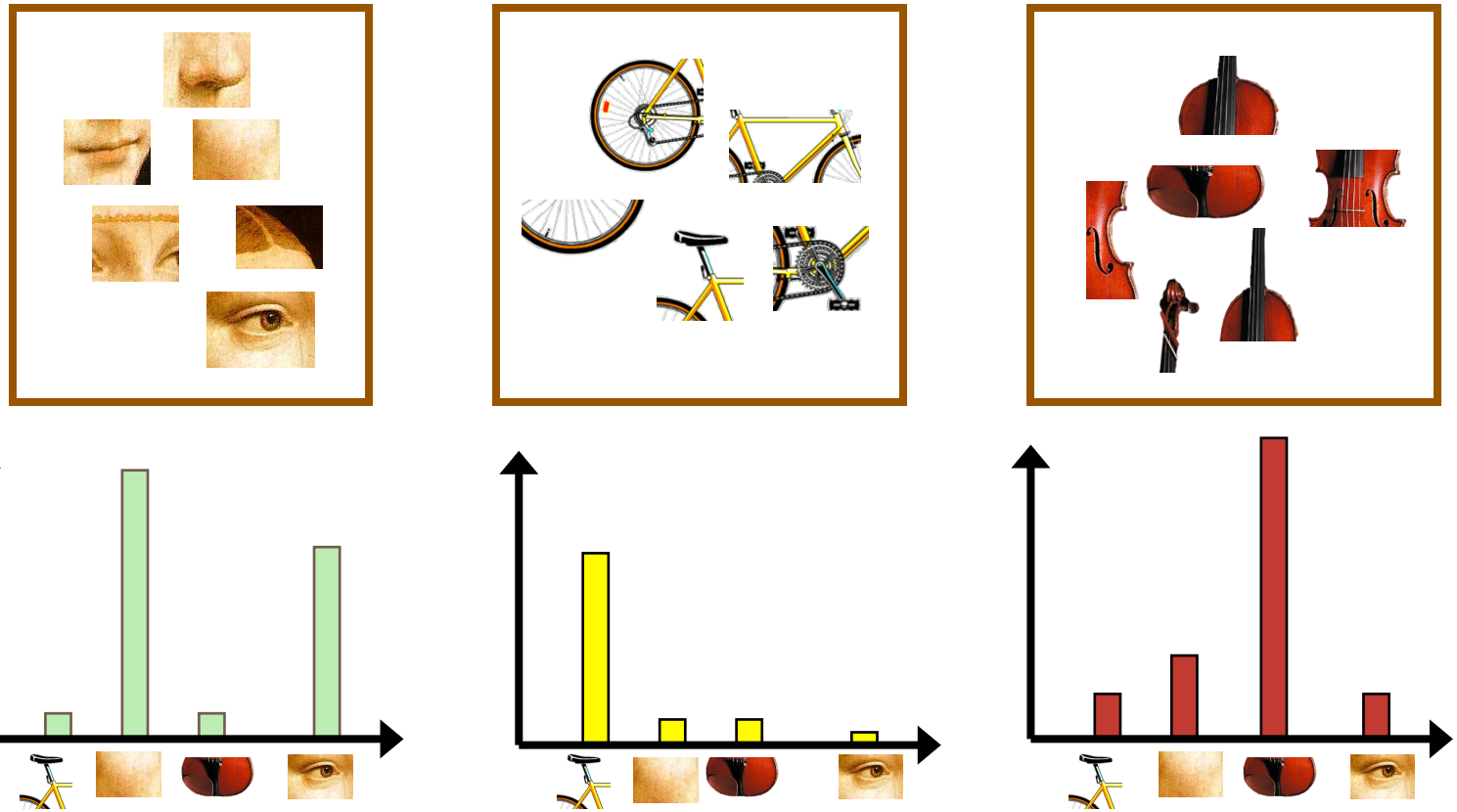
- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Motivation 3: Bags of words

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Bag of features: Outline

1. Extract local features
2. Learn “visual vocabulary”
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of “visual words”



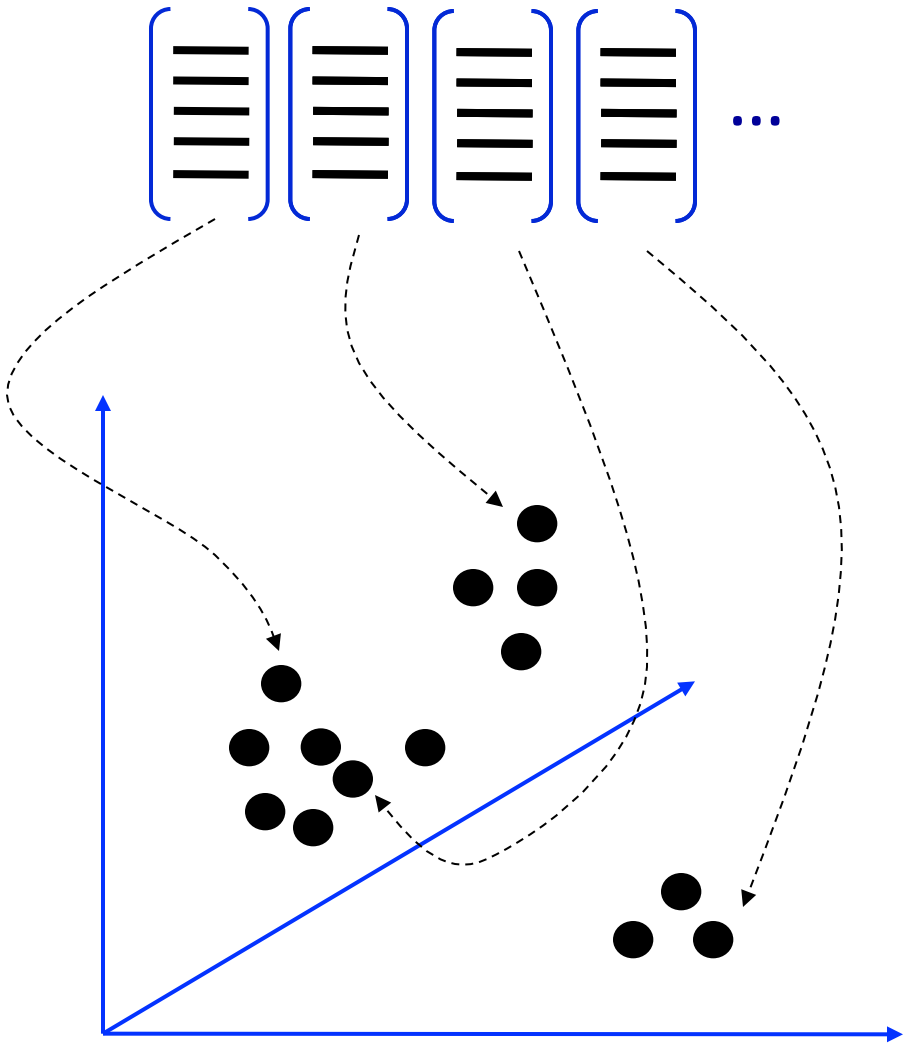
1. Local feature extraction

- Sample patches and extract descriptors

Use interest points

Use a dense grid

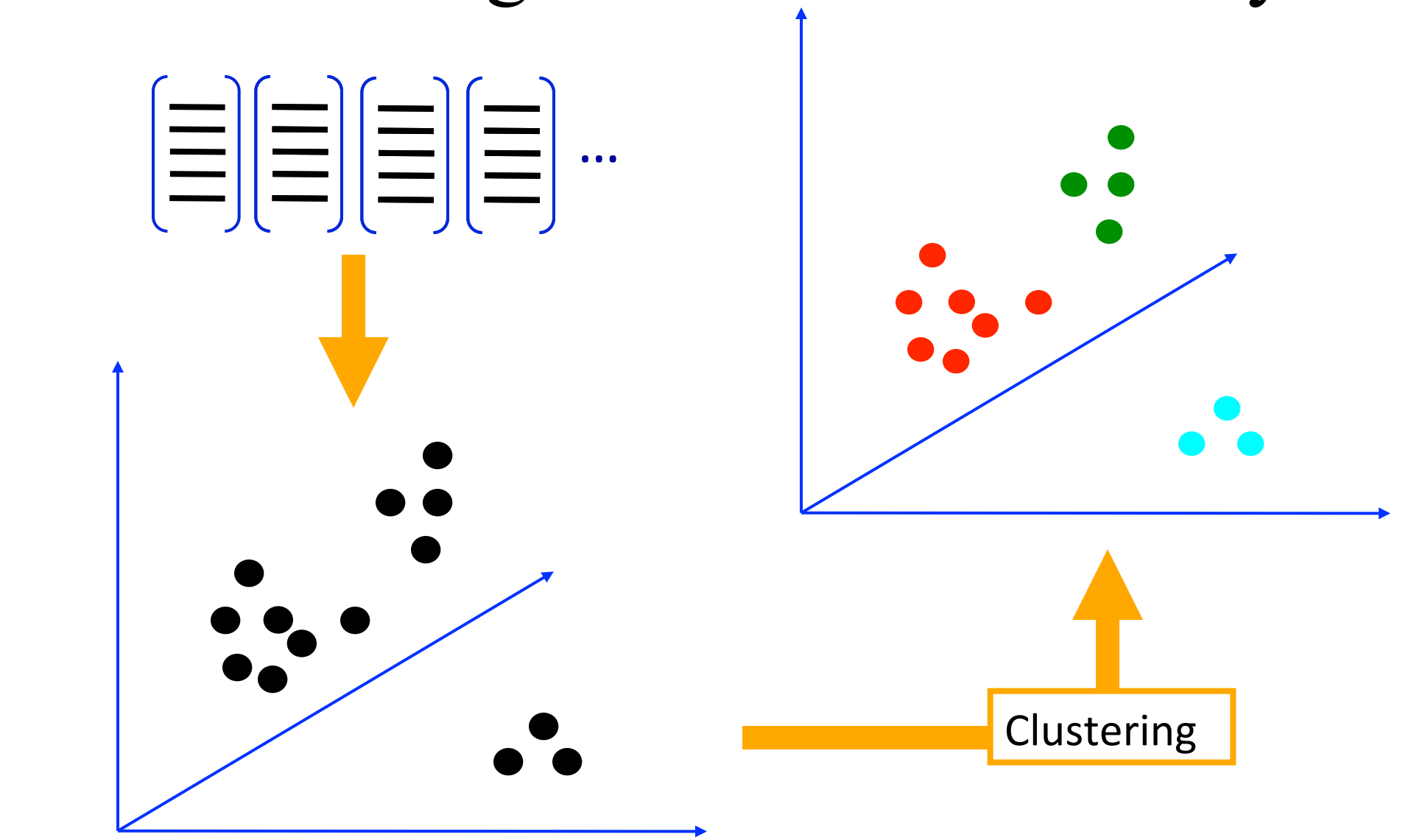
2. Learning the visual vocabulary



Extracted descriptors from the training set

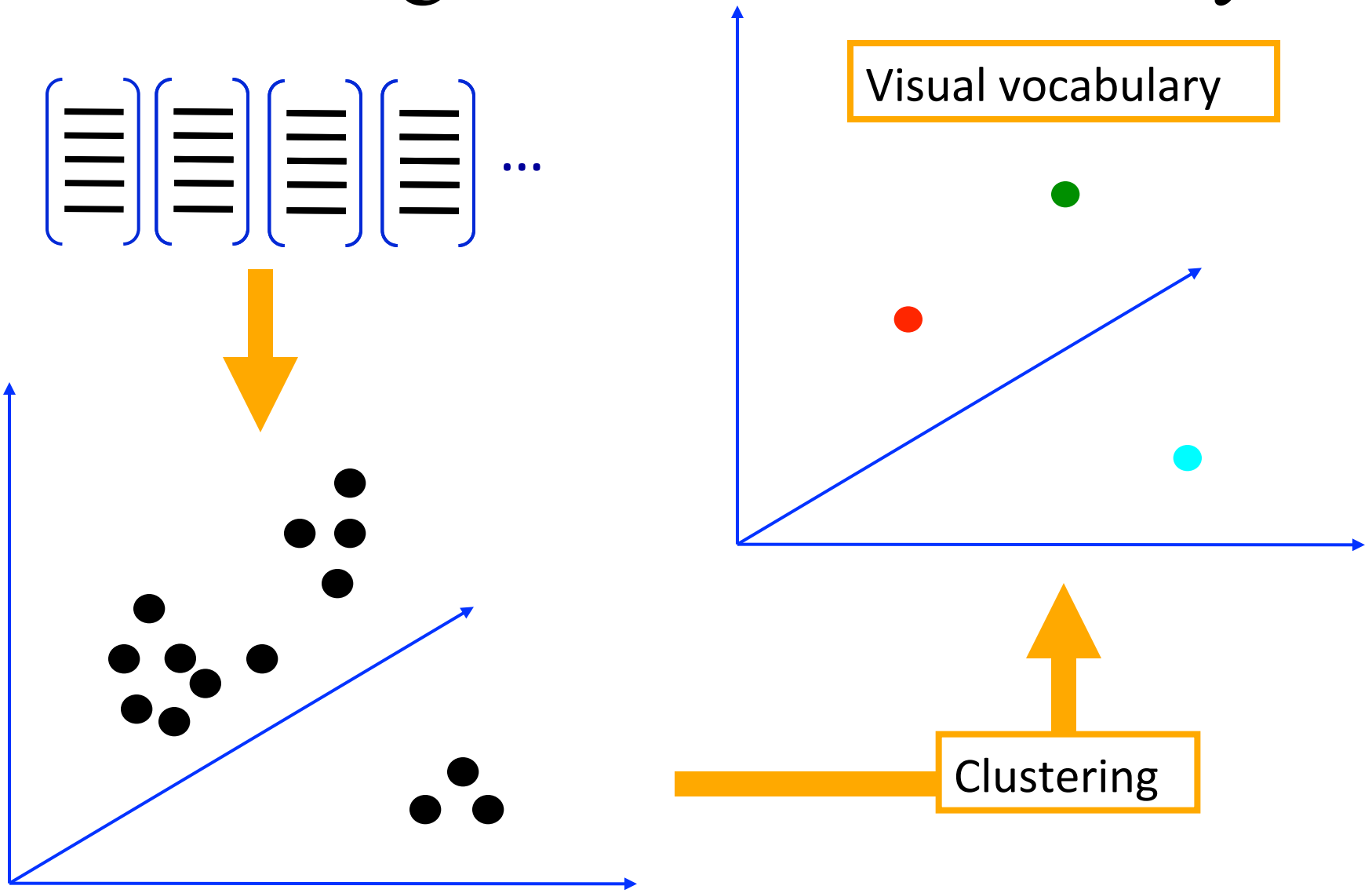
Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

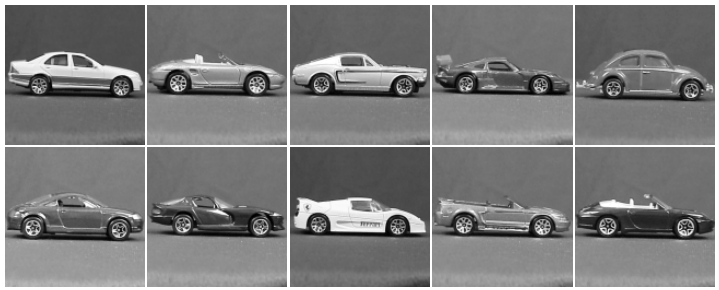
Recall: K-means clustering

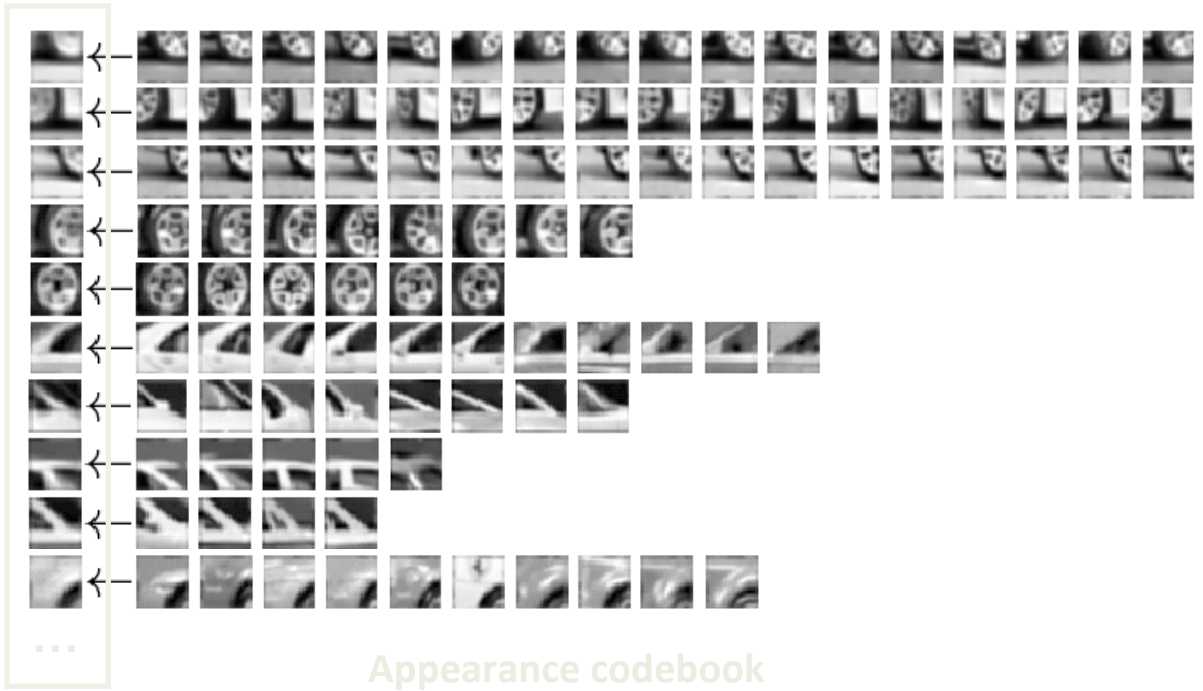
- Want to minimize sum of squared Euclidean distances between features $\mathbf{x}_i$ and their nearest cluster centers $\mathbf{m}_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (\mathbf{x}_i - \mathbf{m}_k)^2$$

- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each feature to the nearest center
  - Recompute each cluster center as the mean of all features assigned to it

Visual vocabularies

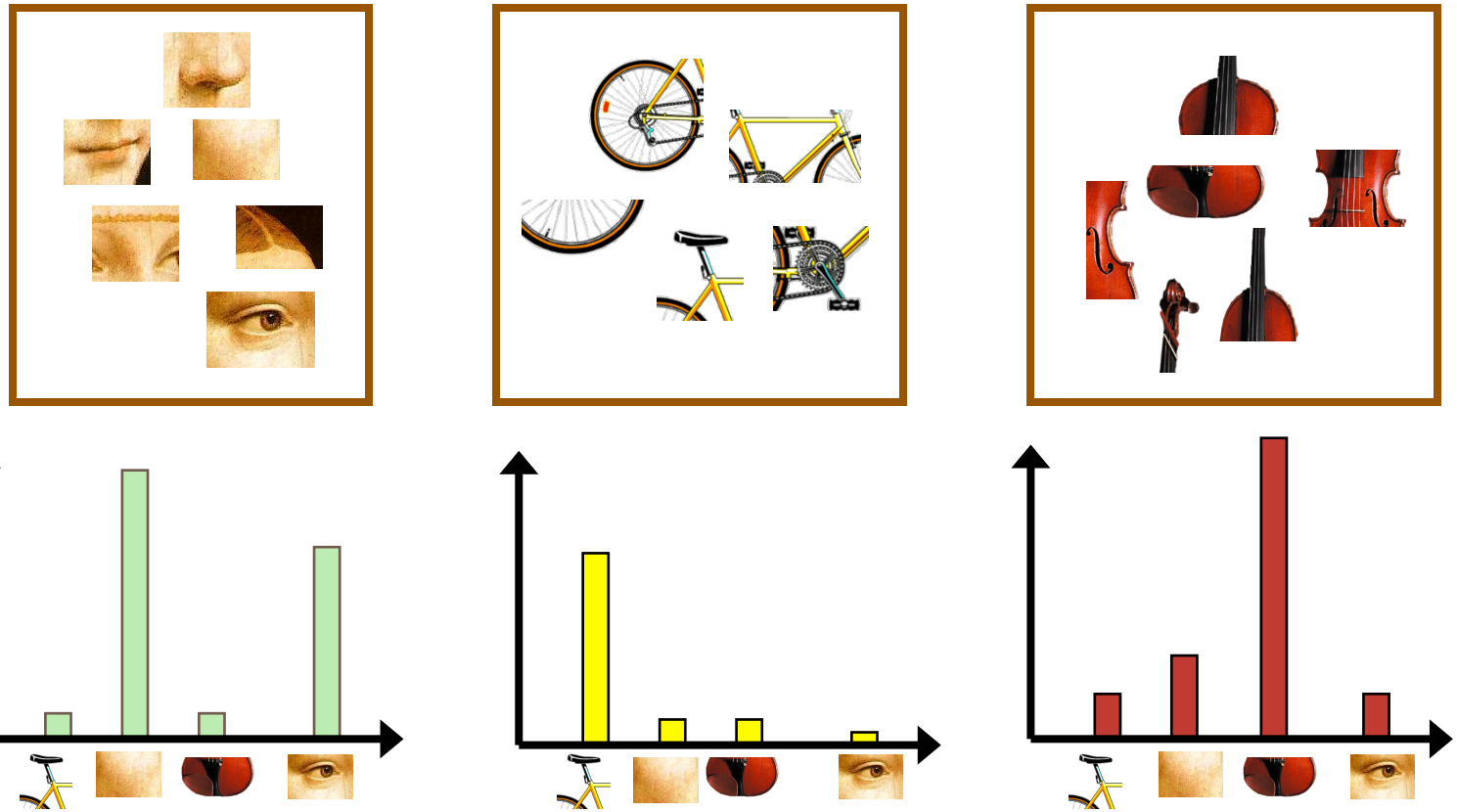




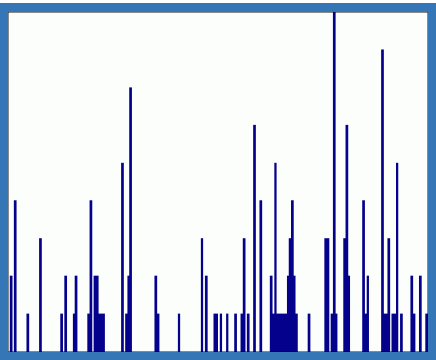
Slide credit: Bastian Leibe

Bag of features: Outline

1. Extract local features
2. Learn “visual vocabulary”
3. **Quantize local features using visual vocabulary**
4. **Represent images by frequencies of “visual words”**



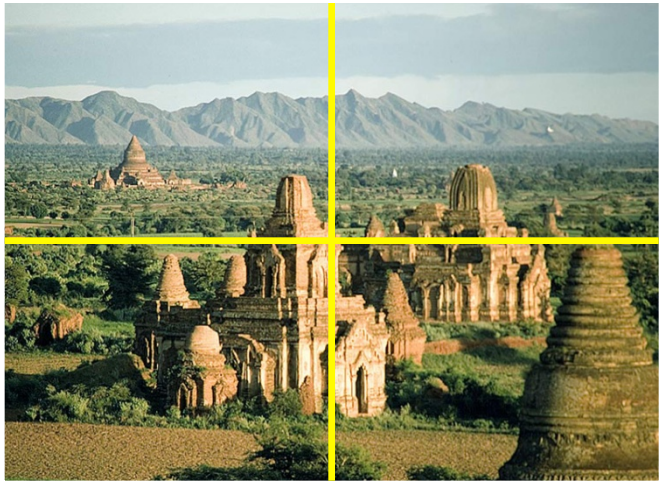
Spatial pyramids

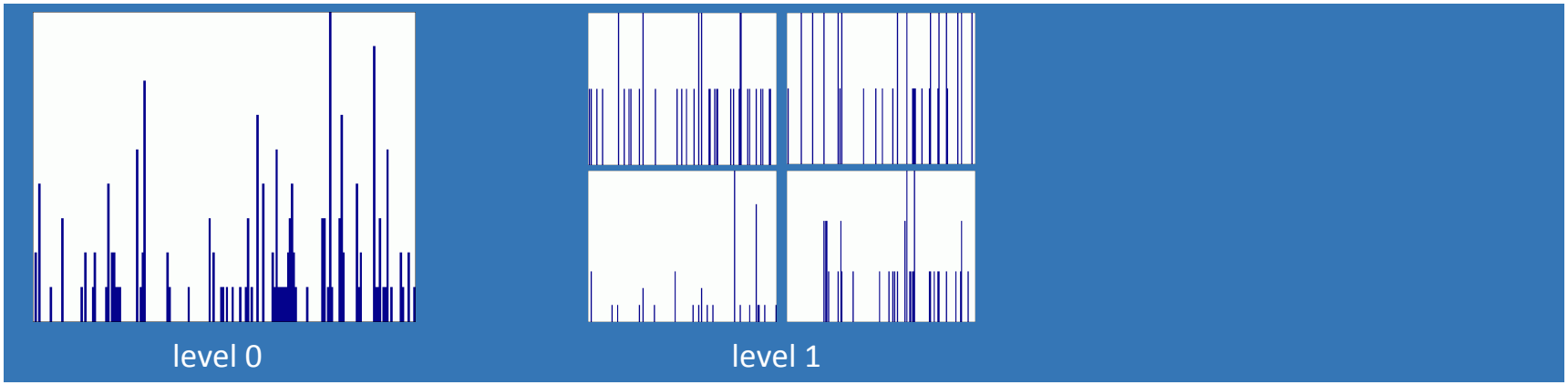


level 0

Lazebnik, Schmid & Ponce (CVPR 2006)

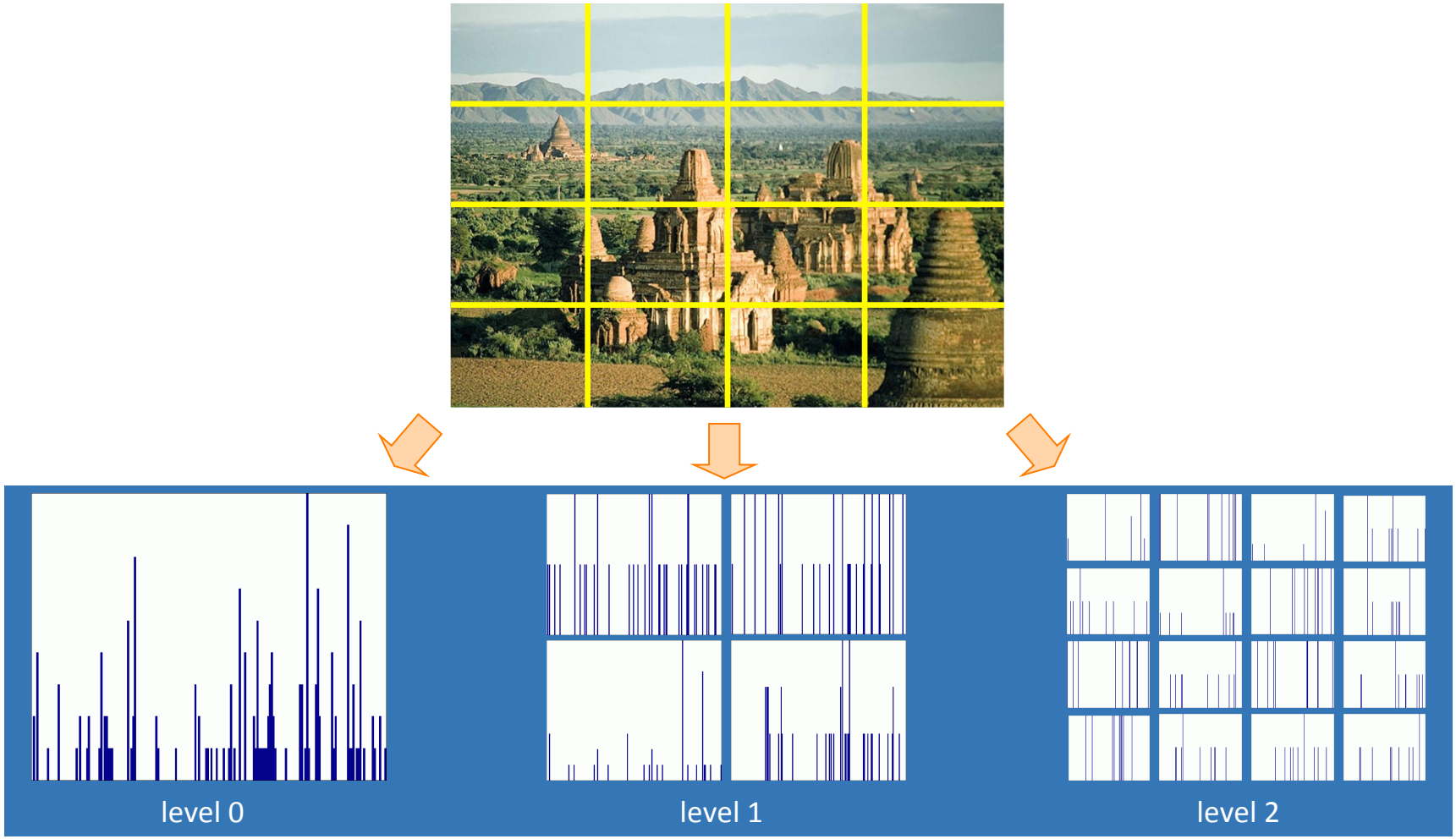
Spatial pyramids





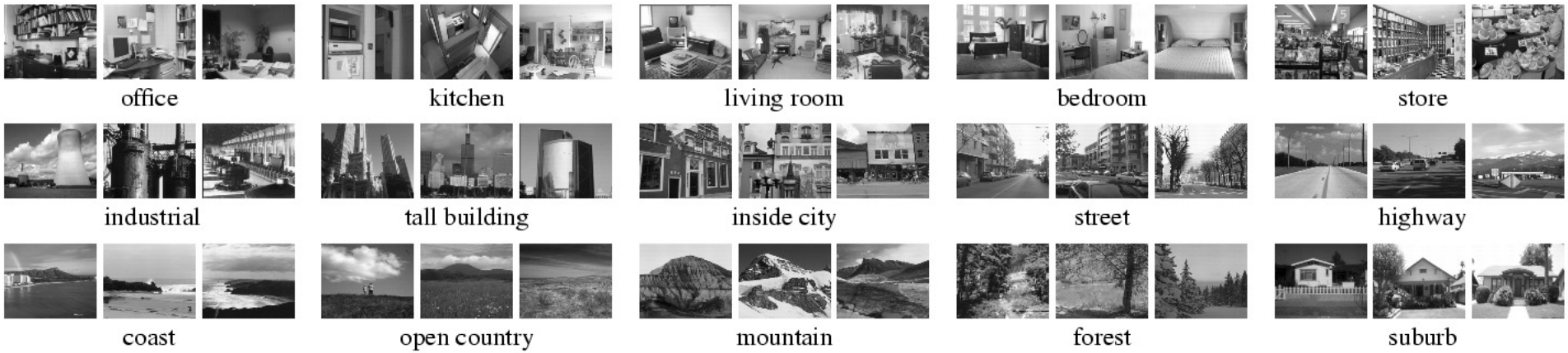
Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids



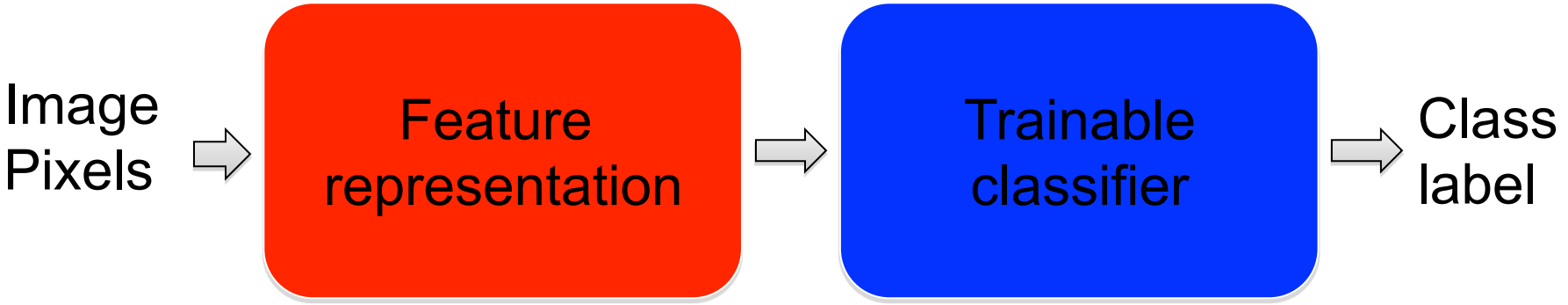
Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramids

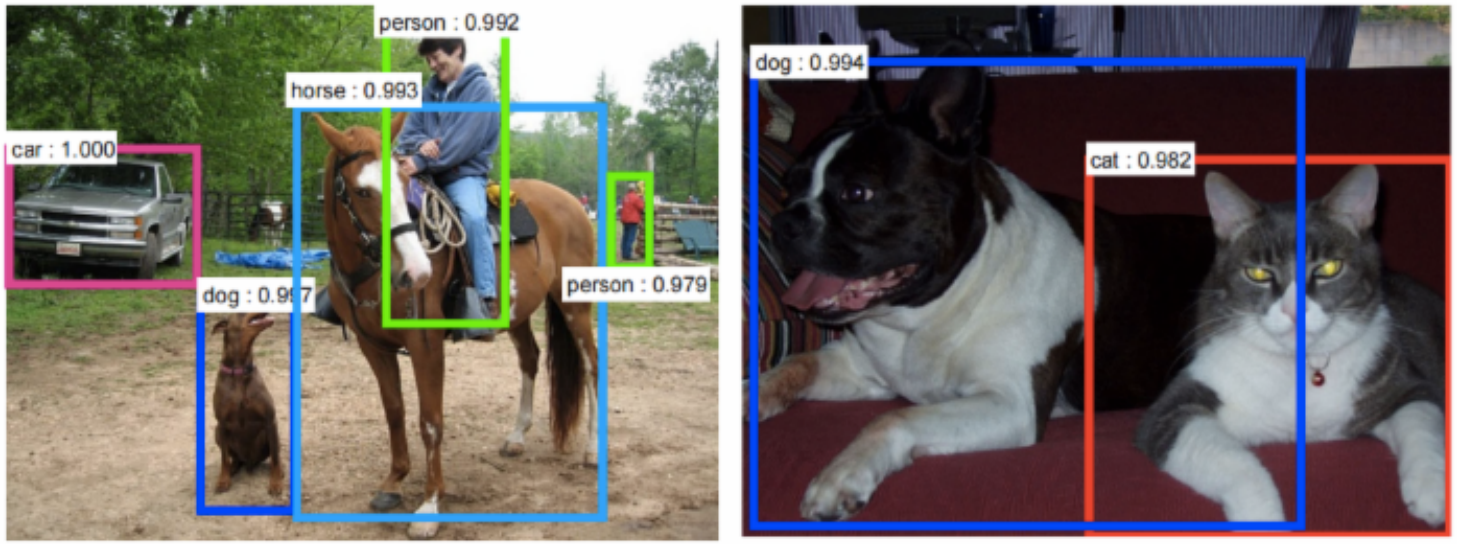


	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1×1)	45.3 ±0.5		72.2 ±0.6	
1 (2×2)	53.6 ±0.3	56.2 ±0.6	77.9 ±0.6	79.0 ±0.5
2 (4×4)	61.7 ±0.6	64.7 ±0.7	79.4 ±0.3	**81.1** ±0.3
3 (8×8)	63.3 ±0.8	**66.8** ±0.6	77.2 ±0.4	80.7 ±0.3

From image classification to object detection

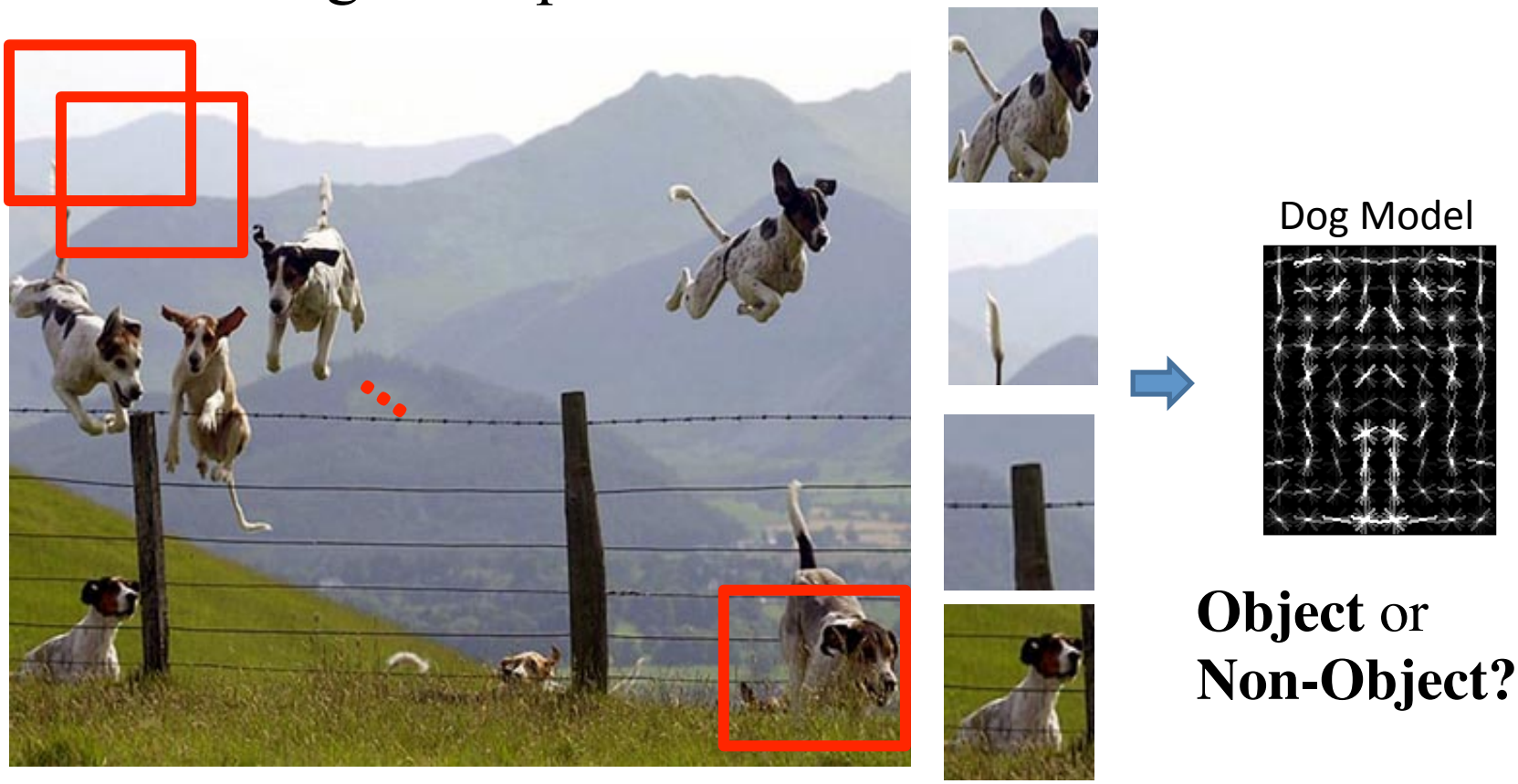


Object detection



Object Category Detection

- Focus on object search: “Where is it?”
- Build templates that quickly differentiate object patch from background patch



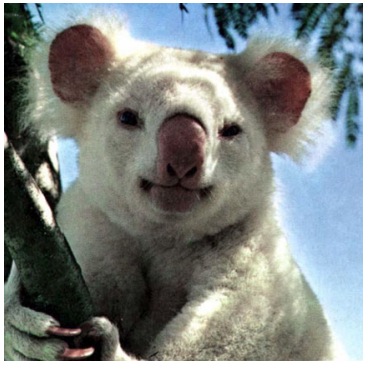
Challenges in modeling the object class

Illumination

Object pose

Clutter

Occlusions



Intra-class appearance

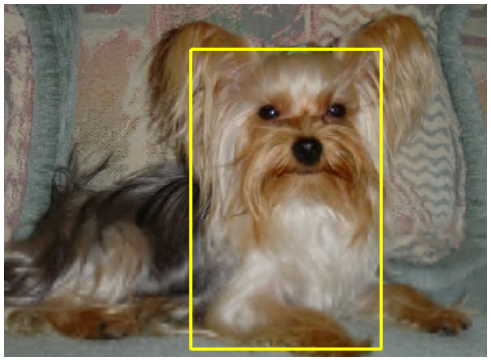
Viewpoint

Slide credit K. Grauman, B. Leibe

Challenges in modeling the object class

True Detections

Bad Localization



Confused with Similar Object

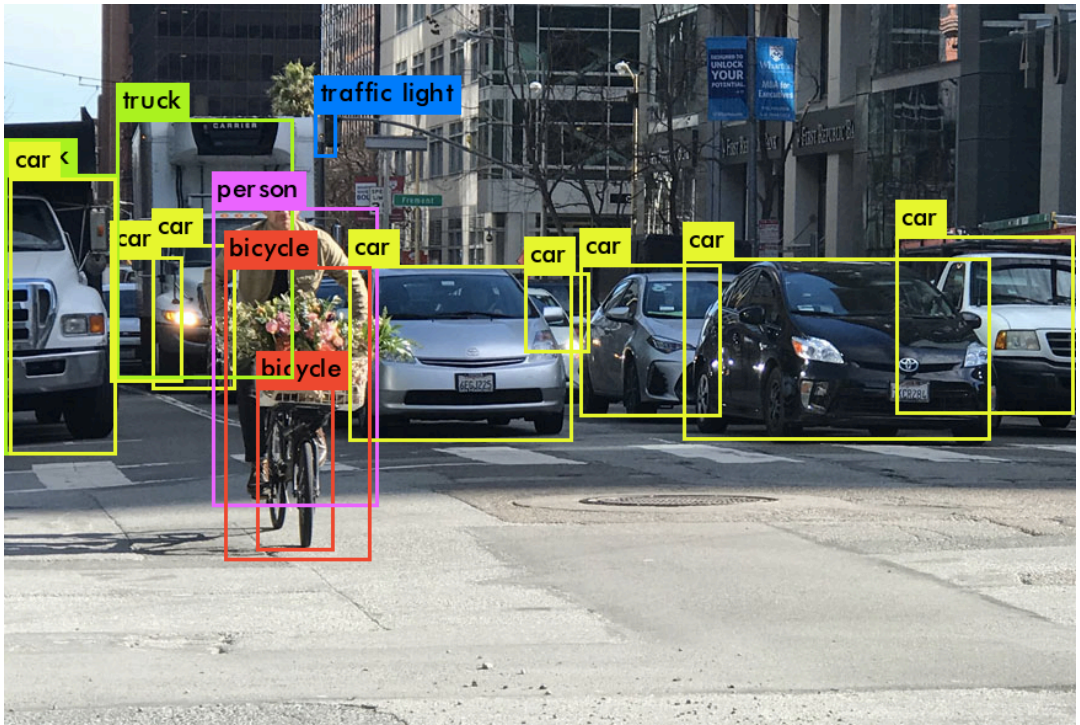
Misc. Background

Confused with Dissimilar Objects

Slide credit K. Grauman, B. Leibe

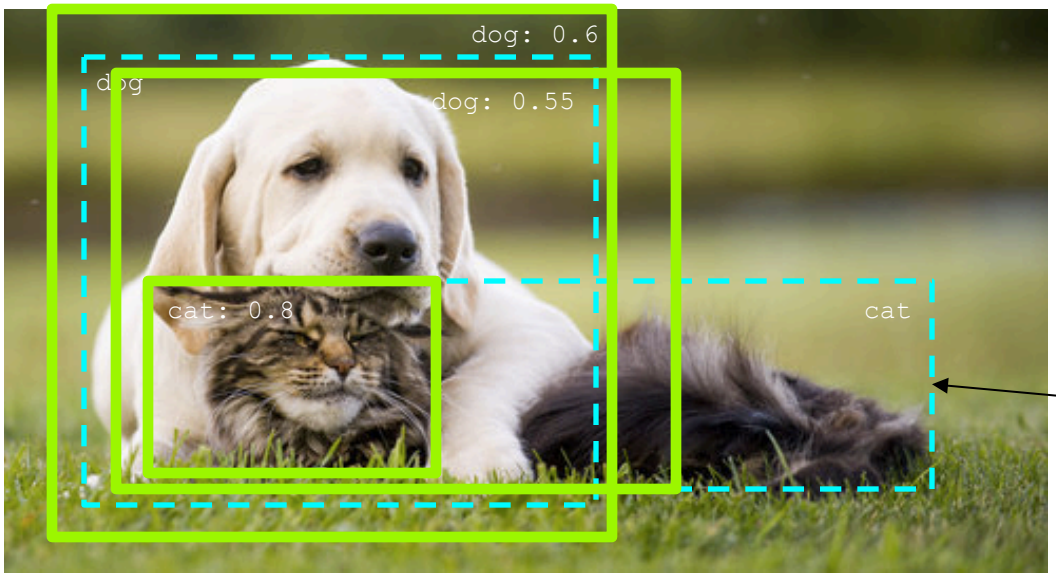
Other challenges of object detection

- Images may contain more than one class, multiple instances from the same class
- Bounding box localization
- Evaluation



Object detection evaluation

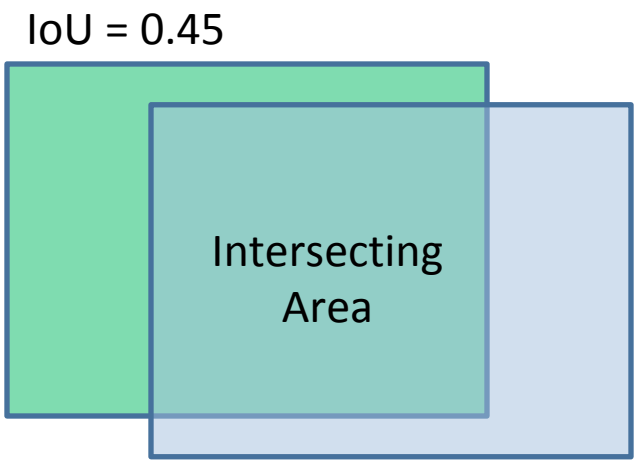
- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive
  - PASCAL criterion: Area(GT ∩ Det) / Area(GT ∪ Det) > 0.5
  - For multiple detections of the same ground truth box, only one considered a true positive

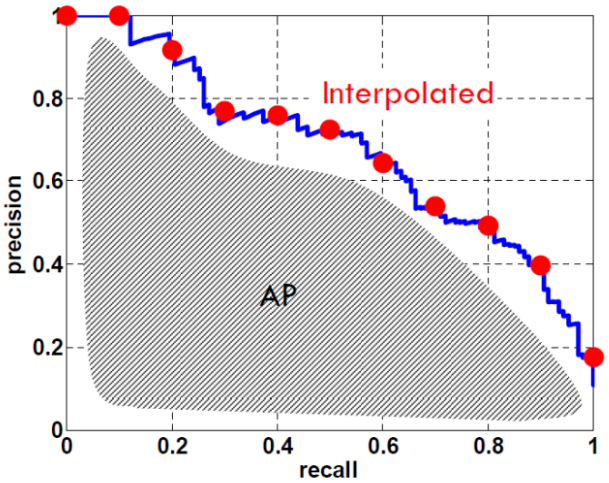


Object detection evaluation

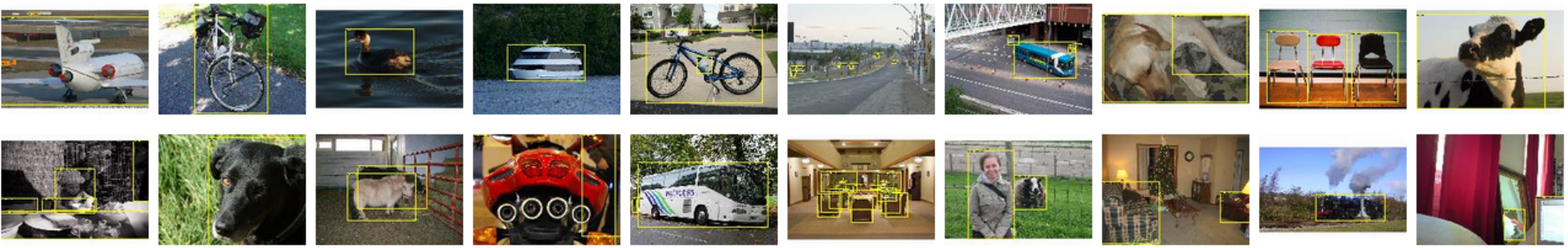
- Datasets
  - PASCAL VOC (2005-2012): 20 classes, ~20,000 images
  - MS COCO (2014-): 60 classes, ~300,000 images

- Evaluation
  - Output: for each class, predict bounding boxes (xmin, ymin, xmax, ymax) with confidences
  - Metric:
    - True detection: >= 0.5 Intersection over Union (IoU), not a duplicate
    - Precision, Recall
    - AP: area under the interpolated curve





PASCAL VOC Challenge (2005-2012)



- 20 challenge classes:
  - *Person*
  - *Animals:* bird, cat, cow, dog, horse, sheep
  - *Vehicles:* aeroplane, bicycle, boat, bus, car, motorbike, train
  - *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor

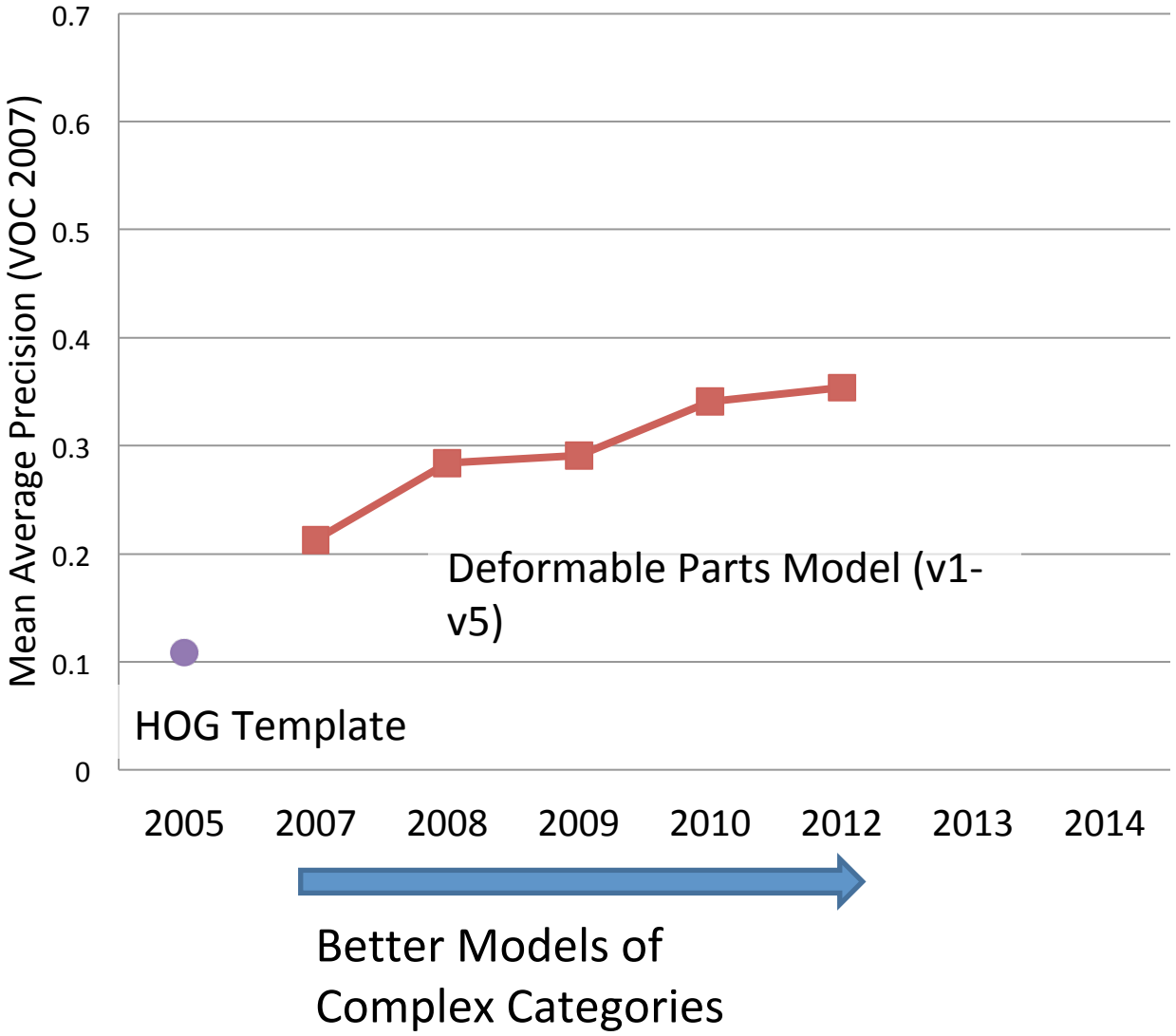
- Dataset size (by 2012): 11.5K training/validation images, 27K bounding boxes, 7K segmentations

http://host.robots.ox.ac.uk/pascal/VOC/

Improvements in object detection

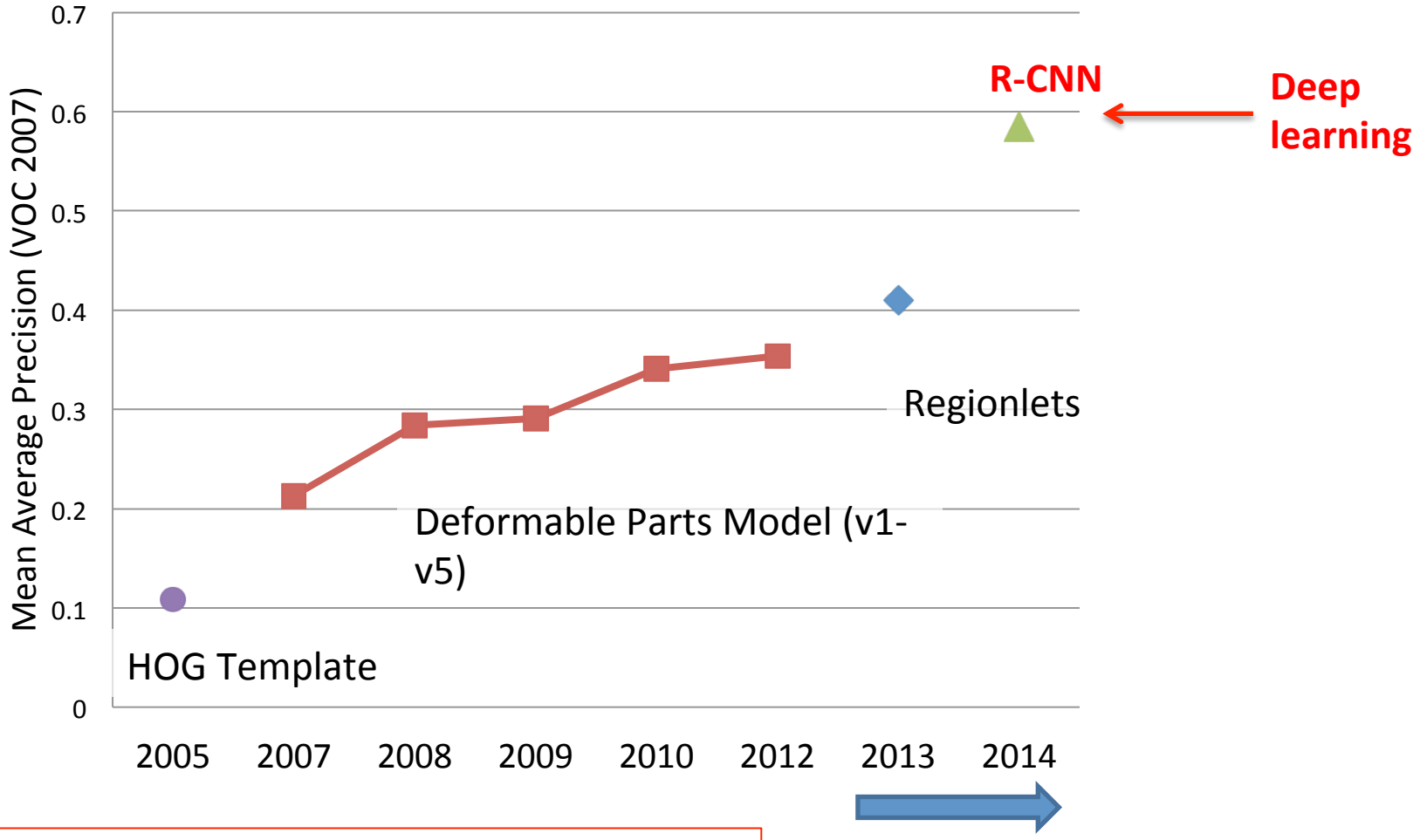
HOG: Dalal-Triggs 2005

Improvements in object detection



HOG: Dalal-Triggs 2005 DPM: Felzenszwalb et al. 2008-2012

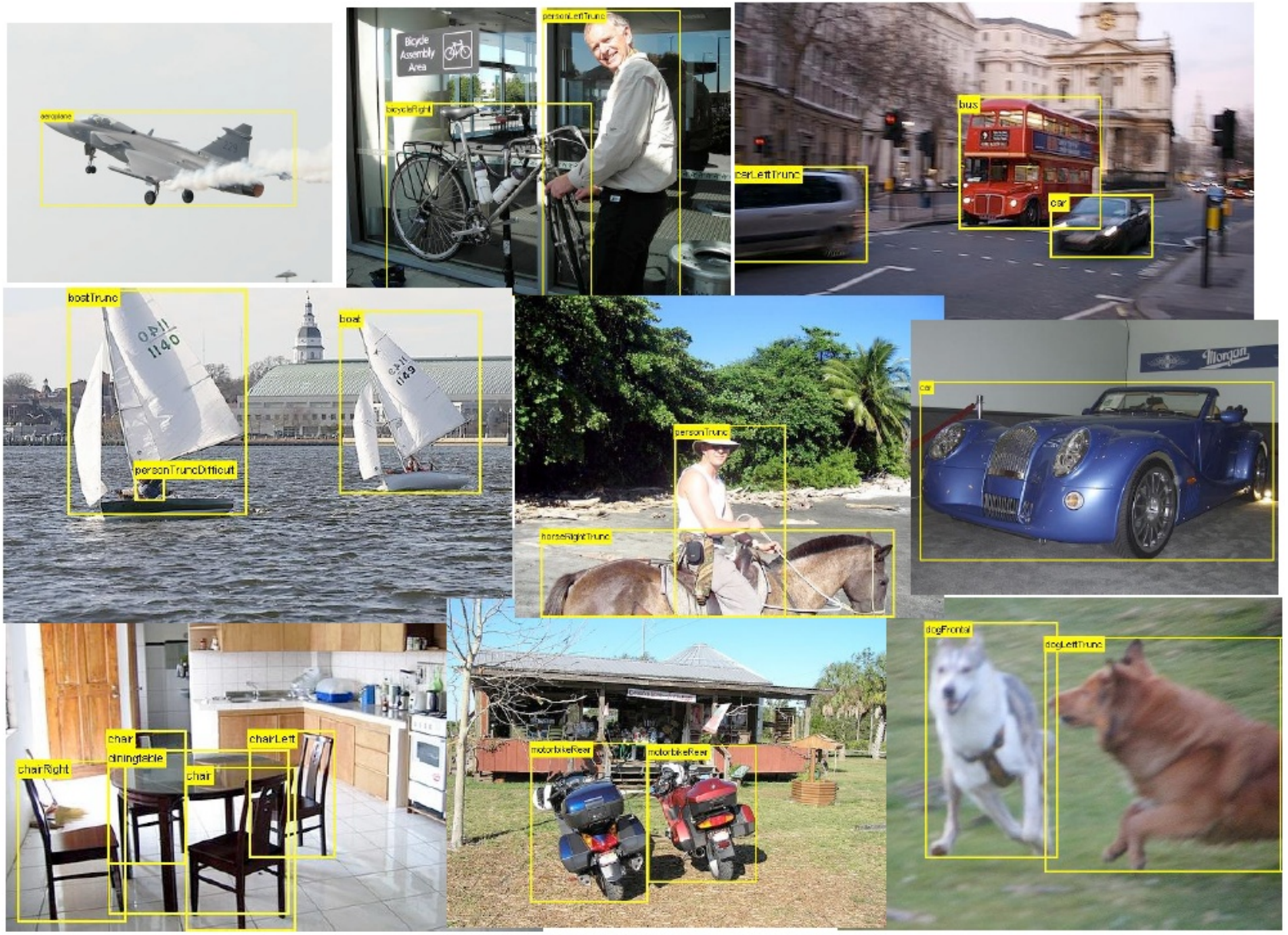
Improvements in object detection



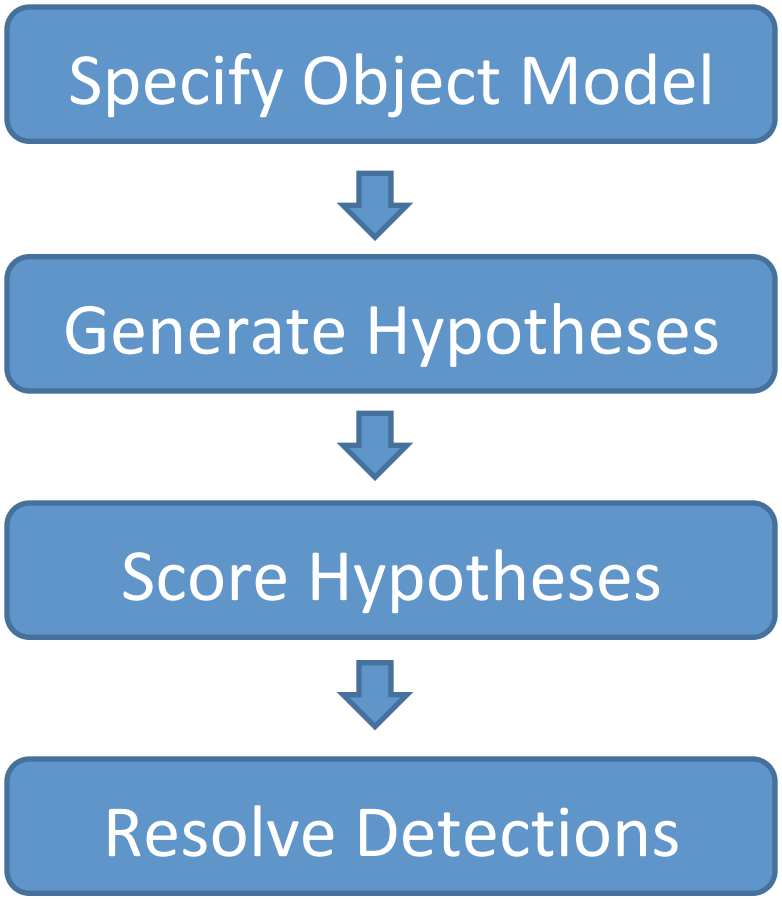
Key Advance: Learn effective features from massive amounts of labeled data *and* adapt to new tasks with less data

HOG: Dalal-Triggs 2005 DPM: Felzenszwalb et al. 2008-2012 Regionlets: Wang et al. 2013 R-CNN: Girshick et al. 2014

Detection before deep learning



General Process of Object Detection



What are the object parameters?