Computer Vision II – Lecture 15

Repetition

15.07.2014

Bastian Leibe

RWTH Aachen

http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de

Announcements

- Exams
  - Proposed dates
    - 29./30.07.
    - 22./23.09.
  - Please enter your preferences in the Doodle poll I sent around
  - If none of the dates work for you, please contact me.

- Exam Procedure
  - Oral exams
  - Duration 30min
  - I will give you 4 questions and expect you to answer 3 of them.

Announcements (2)

- **Lecture Evaluation**
  - Please fill out the forms...

Announcements (3)

- Today, I'll summarize the most important points from the lecture.
  - It is an opportunity for you to ask questions…
  - …or get additional explanations about certain topics.
  - *So, please do ask*.
- Today's slides are intended as an index for the lecture.
  - But they are not complete, won't be sufficient as only tool.
  - Also look at the exercises - they often explain algorithms in detail.

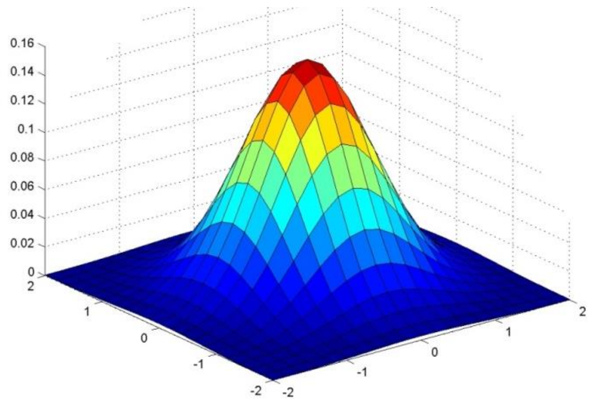
Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

Image source: Tobias Jaeggli

Recap: Gaussian Background Model

- Statistical model
  - Value of a pixel represents a measurement of the radiance of the first object intersected by the pixel's optical ray.
  - With a static background and static lighting, this value will be a constant affected by i.i.d. Gaussian noise.



- Idea
  - Model the background distribution of each pixel by a single Gaussian centered at the mean pixel value:

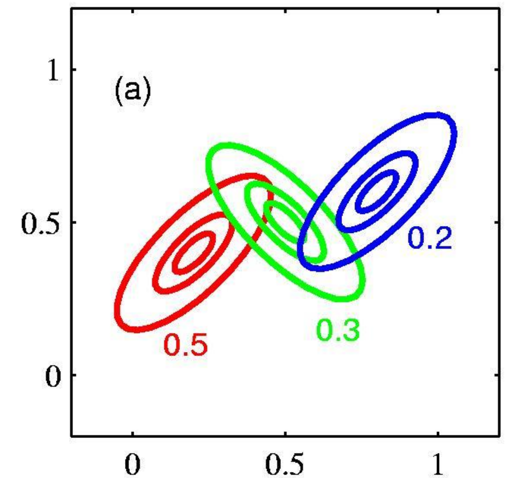
$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

  - Test if a newly observed pixel value has a high likelihood under this Gaussian model.

$\Rightarrow$ Automatic estimation of a sensitivity threshold for each pixel.

Recap: MoG Background Model

- **Improved statistical model**
  - Large jumps between different pixel values because different objects are projected onto the same pixel at different times.
  - While the same object is projected onto the pixel, small local intensity variations due to Gaussian noise.



- **Idea**
  - Model the color distribution of each pixel by a mixture of K Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  - Evaluate likelihoods of observed pixel values under this model.
  - Or let entire Gaussian components adapt to foreground objects and classify components as belonging to object or background.

Image source: Chris Bischop

Recap: Stauffer-Grimson Background Model

- **Idea**
  - Model the distribution of each pixel by a mixture of K Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{where} \quad \boldsymbol{\Sigma}_k = \sigma_k^2 \mathbf{I}$$

  - Check every new pixel value against the existing K components until a match is found (pixel value within $2.5\ \sigma_k$ of $\boldsymbol{\mu}_k$).
  - If a match is found, adapt the corresponding component.
  - Else, replace the least probable component by a distribution with the new value as its mean and an initially high variance and low prior weight.
  - Order the components by the value of w_k/σ_k and select the best B components as the background model, where

$$B = \arg\min_b \left(\sum_{k=1}^{b} \frac{w_k}{\sigma_k} > T \right)$$

[C. Stauffer, W.E.L. Grimson, CVPR'99]

Recap: Stauffer-Grimson Background Model

- Online adaptation
  - Instead of estimating the MoG using EM, use a simpler online adaptation, assigning each new value only to the matching component.
  - Let $M_{k,t} = 1$ iff component k is the model that matched, else 0.

$$\pi_k^{(t+1)} = (1-\alpha)\pi_k^{(t)} + \alpha M_{k,t}$$

  - Adapt only the parameters for the matching component

$$\boldsymbol{\mu}_k^{(t+1)} = (1-\rho)\boldsymbol{\mu}_k^{(t)} + \rho x^{(t+1)}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = (1-\rho)\boldsymbol{\Sigma}_k^{(t)} + \rho(x^{(t+1)} - \boldsymbol{\mu}_k^{(t+1)})(x^{(t+1)} - \boldsymbol{\mu}_k^{(t+1)})^T$$

where

$$\rho = \alpha \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

(i.e., the update is weighted by the component likelihood)

[C. Stauffer, W.E.L. Grimson, CVPR'99]

Recap: Kernel Background Modeling

- **Nonparametric density estimation**
  - **Estimate a pixel's background distribution using the kernel density estimator $K(\cdot)$ as**

$$p(\mathbf{x}^{(t)}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x}^{(t)} - \mathbf{x}^{(i)})$$

  - **Choose K to be a Gaussian $\mathcal{N}(0, \mathbf{\Sigma})$ with $\mathbf{\Sigma} = \mathrm{diag}\{\sigma_j\}$. Then**

$$p(\mathbf{x}^{(t)}) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{1}{2}\frac{(x_j^{(t)}-x_j^{(i)})^2}{\sigma_j^2}}$$

  - **A pixel is considered foreground if $p(\mathbf{x}^{(t)}) < \theta$ for a threshold θ.**
    - **This can be computed very fast using lookup tables for the kernel function values, since all inputs are discrete values.**
    - **Additional speedup: partial evaluation of the sum usually sufficient**

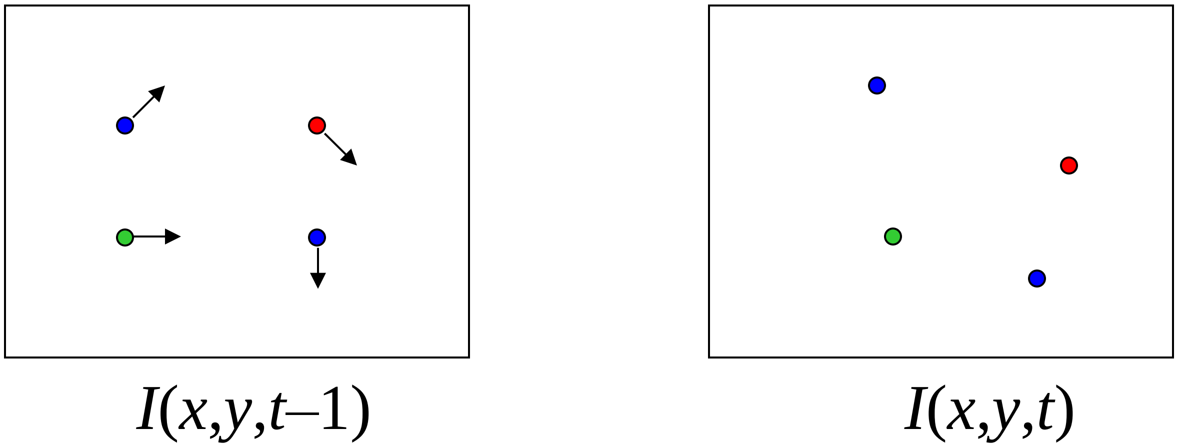
[A. Elgammal, D. Harwood, L. Davis, ECCV'00]

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

Image source: Robert Collins

Recap: Estimating Optical Flow



- **Optical Flow**
  - 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 Optical Flow

- Use all pixels in a $K \times K$ window to get more equations.
- Least squares problem:

$$\begin{bmatrix} I_x(\mathrm{p}_1) & I_y(\mathrm{p}_1) \\ I_x(\mathrm{p}_2) & I_y(\mathrm{p}_2) \\ \vdots & \vdots \\ I_x(\mathrm{p}_{25}) & I_y(\mathrm{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathrm{p}_1) \\ I_t(\mathrm{p}_2) \\ \vdots \\ I_t(\mathrm{p}_{25}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

- Minimum least squares solution given by solution of

$$\underset{2\text{x}2}{(A^T A)} \ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

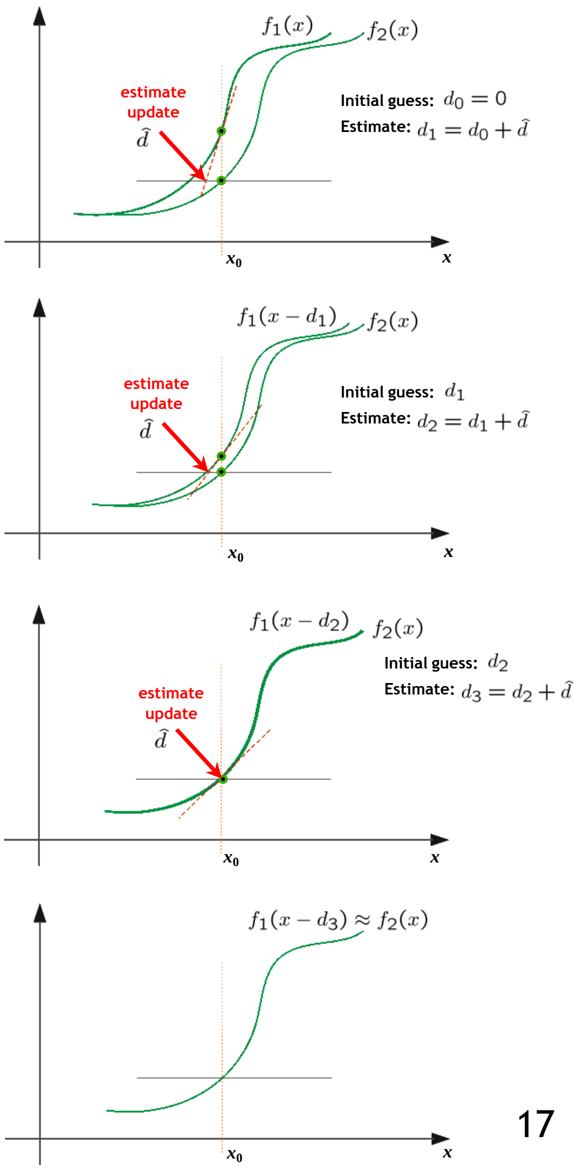
Recall the Harris detector!

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

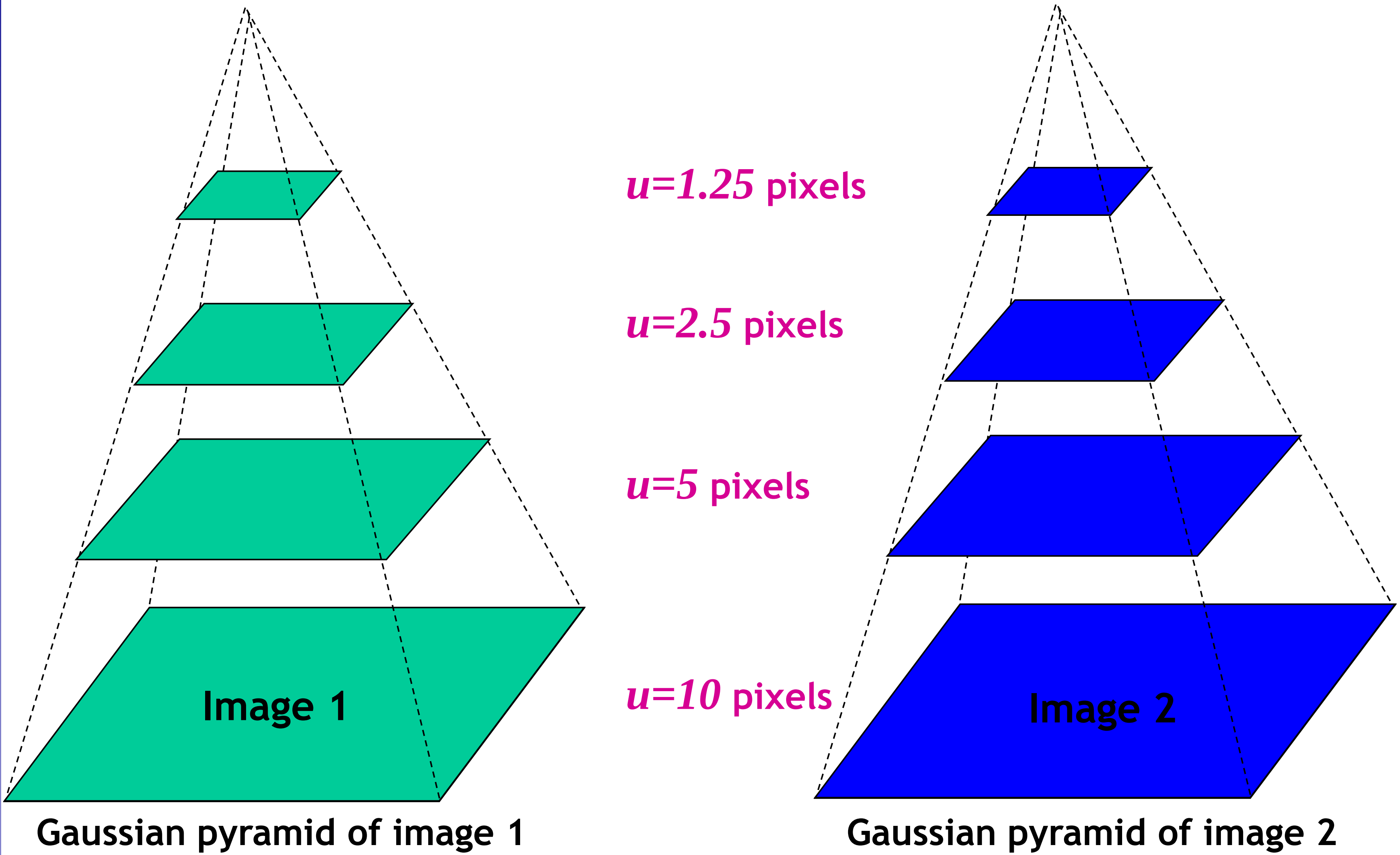
Recap: Iterative Refinement

- Estimate velocity at each pixel using one iteration of LK estimation.
- Warp one image toward the other using the estimated flow field.
- Refine estimate by repeating the process.

- Iterative procedure
  - Results in subpixel accurate localization.
  - Converges for small displacements.

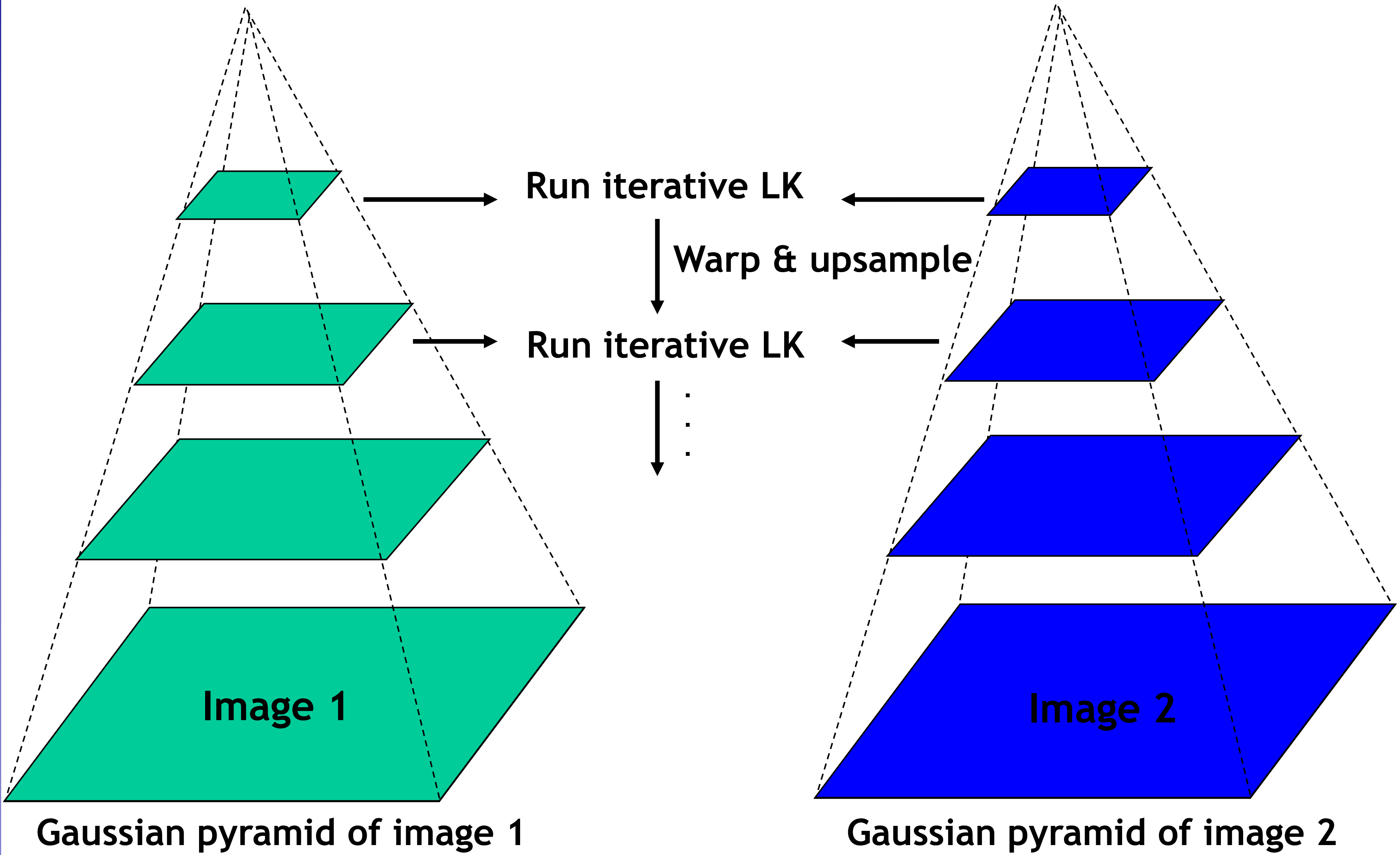


Recap: Coarse-to-fine Optical Flow Estimation



Slide credit: Steve Seitz

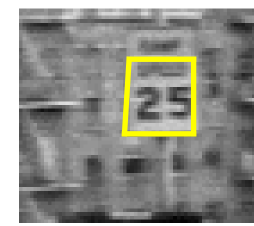
Recap: Coarse-to-fine Optical Flow Estimation



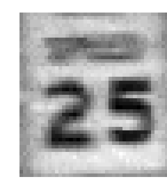
Recap: Shi-Tomasi Feature Tracker (→KLT)

- **Idea**
  - Find good features using eigenvalues of second-moment matrix
  - Key idea: "good" features to track are the ones that can be tracked reliably.

- **Frame-to-frame tracking**
  - Track with LK and a pure *translation* motion model.
  - More robust for small displacements, can be estimated from smaller neighborhoods (e.g., 5×5 pixels).



- **Checking consistency of tracks**
  - *Affine* registration to the first observed feature instance.
  - 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.

Recap: General LK Image Registration

- **Goal**
  - Find the warping parameters $\mathbf{p}$ that minimize the sum-of-squares intensity difference between the template image $T(\mathbf{x})$ and the warped input image $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$.

- **LK formulation**
  - Formulate this as an optimization problem

$$\arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]^2$$

  - We assume that an initial estimate of $\mathbf{p}$ is known and iteratively solve for increments to the parameters $\Delta\mathbf{p}$:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right]^2$$

Recap: Step-by-Step Derivation

- **Key to the derivation**
  - Taylor expansion around $\Delta\mathbf{p}$

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + \mathcal{O}(\Delta\mathbf{p}^2)$$

$$= I(\mathbf{W}([x,y];p_1,\ldots,p_n))$$

$$+ \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{bmatrix} \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_n \end{bmatrix}$$

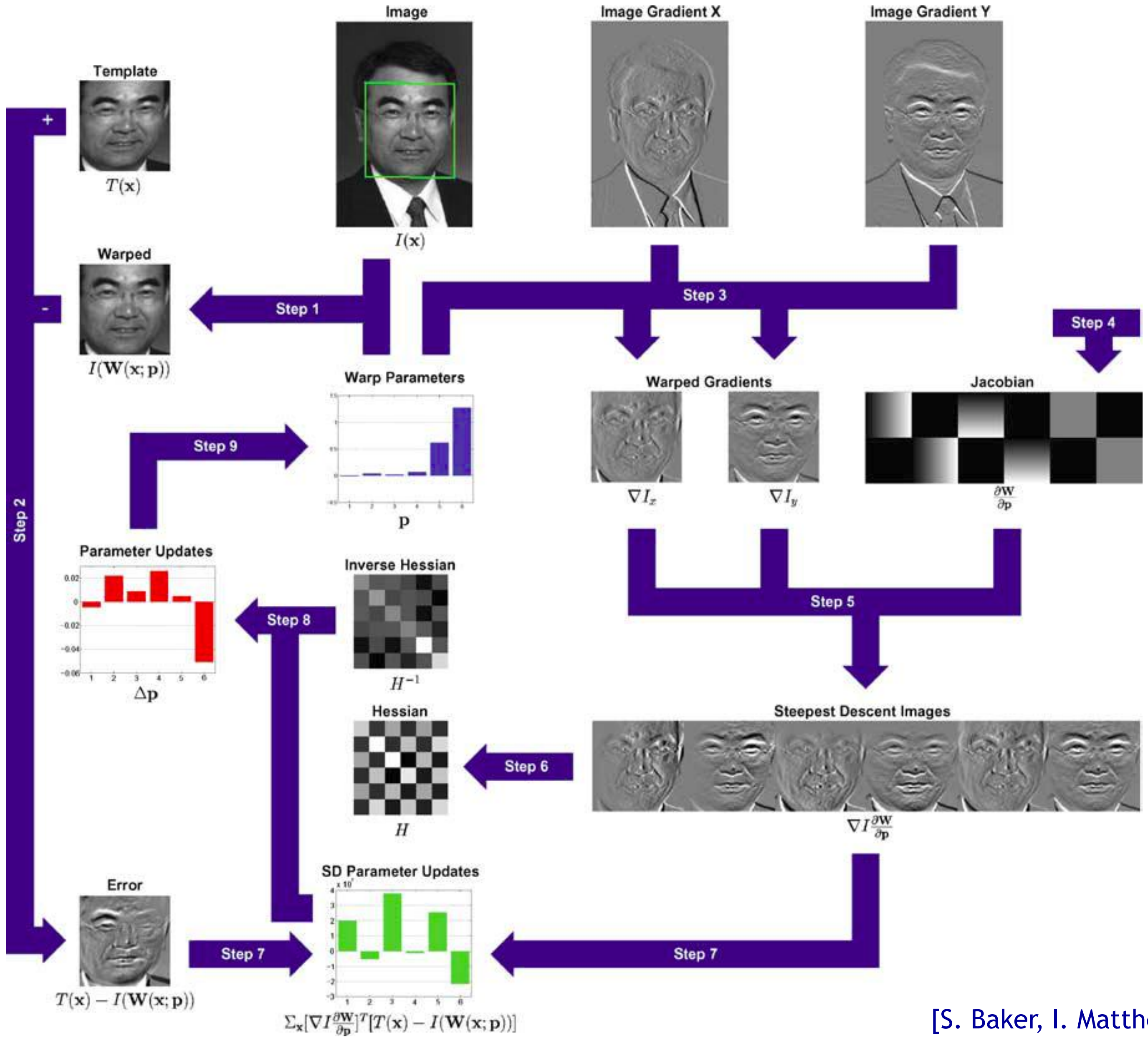
Gradient	Jacobian	Increment parameters to solve for
∇I	$\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$	$\Delta\mathbf{p}$

Recap: General LK Algorithm

- Iterate
  - Warp I to obtain $I(\mathbf{W}([x, y]; \mathbf{p}))$
  - Compute the error image $T([x, y]) - I(\mathbf{W}([x, y]; \mathbf{p}))$
  - Warp the gradient ∇I with $\mathbf{W}([x, y]; \mathbf{p})$
  - Evaluate $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $([x, y]; \mathbf{p})$ (**Jacobian**)
  - Compute steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
  - Compute Hessian matrix $\mathbf{H} = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$
  - Compute $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [T([x,y]) - I(\mathbf{W}([x,y];\mathbf{p}))]$
  - Compute $\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [T([x,y]) - I(\mathbf{W}([x,y];\mathbf{p}))]$
  - Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$
- Until $\Delta \mathrm{p}$ magnitude is negligible

[S. Baker, I. Matthews, IJCV'04]

Recap: General LK Algorithm Visualization



[S. Baker, I. Matthews, IJCV'04]

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

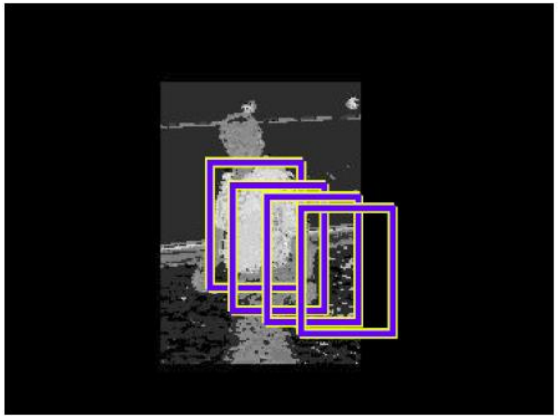
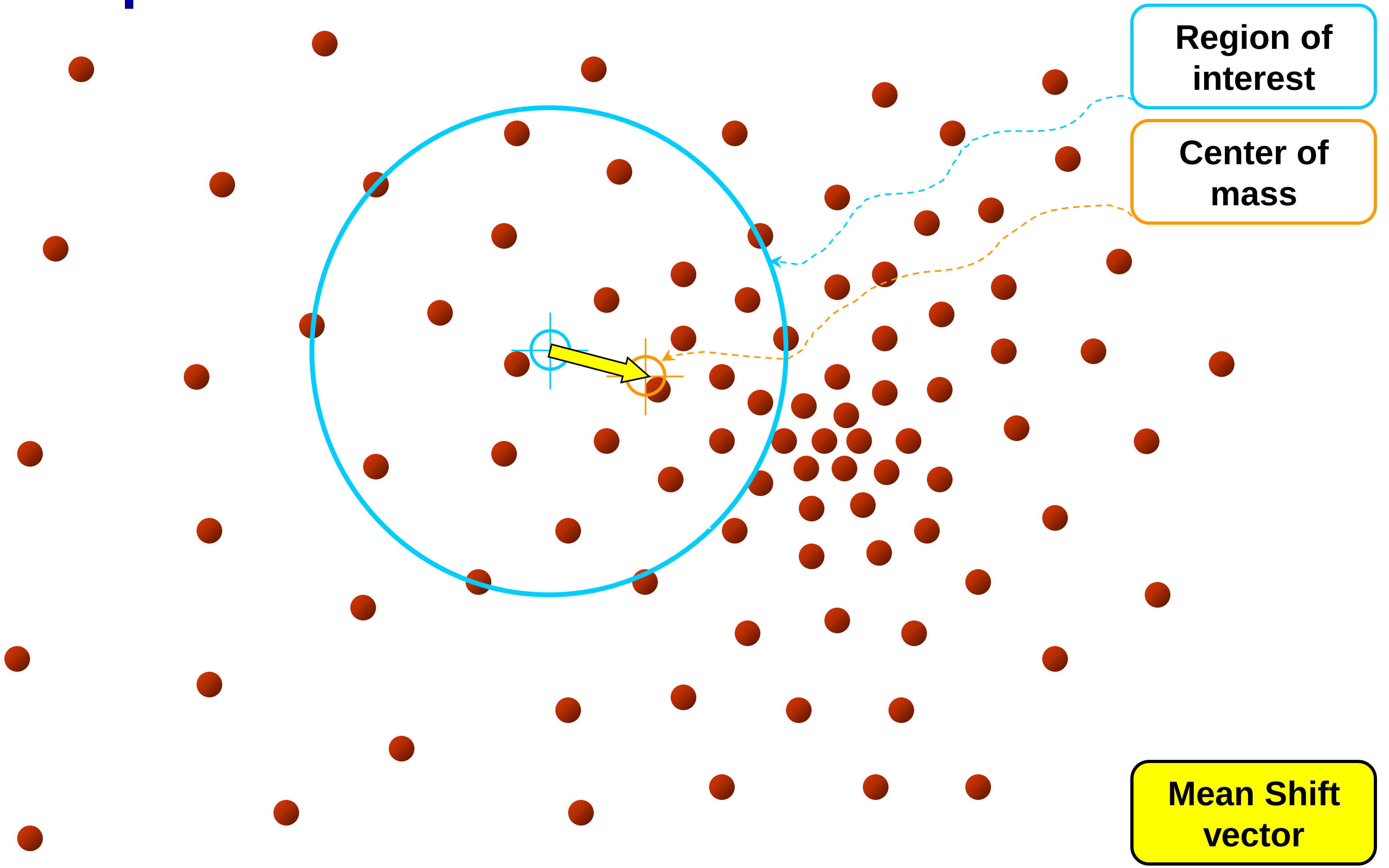


Image source: Robert Collins

Recap: Mean-Shift



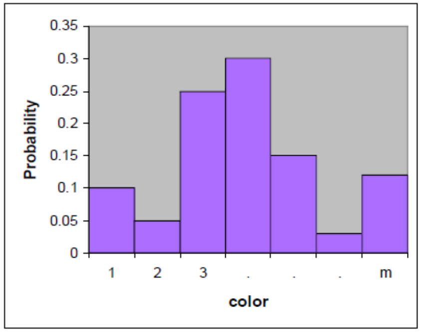
Objective: Find the densest region

Slide by Y. Ukrainitz & B. Sarel

Recap: Using Mean-Shift on Color Models

- Two main approaches
  1. Explicit weight images
     - Create a color likelihood image, with pixels weighted by the similarity to the desired color (best for unicolored objects).
     - Use mean-shift to find spatial modes of the likelihood.

  2. Implicit weight images
     - Represent color distribution by a histogram.
     - Use mean-shift to find the region that has the most similar color distribution.



Mean-Shift on Weight Images

- Ideal case
  - Want an indicator function that returns 1 for pixels on the tracked object and 0 for all other pixels.
- Instead
  - Compute likelihood maps
  - Value at a pixel is proportional to the likelihood that the pixel comes from the tracked object.

- Likelihood can be based on
  - Color
  - Texture
  - Shape (boundary)
  - Predicted location

Recap: Mean-Shift Tracking

- Mean-Shift finds the mode of an explicit likelihood image

Kernel weight evaluated at offset $(\mathbf{a}-\mathbf{x})$

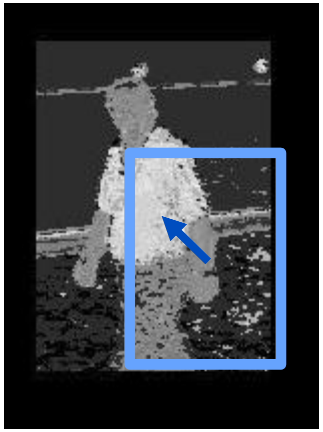
Weight from the likelihood image at pixel $\mathbf{a}$

Offset of pixel $\mathbf{a}$ to kernel center $\mathbf{x}$

$$\Delta\mathbf{x} = \frac{\sum_{\mathbf{a}} K(\mathbf{a}-\mathbf{x})w(\mathbf{a})(\mathbf{a}-\mathbf{x})}{\sum_{\mathbf{a}} K(\mathbf{a}-\mathbf{x})w(\mathbf{a})}$$

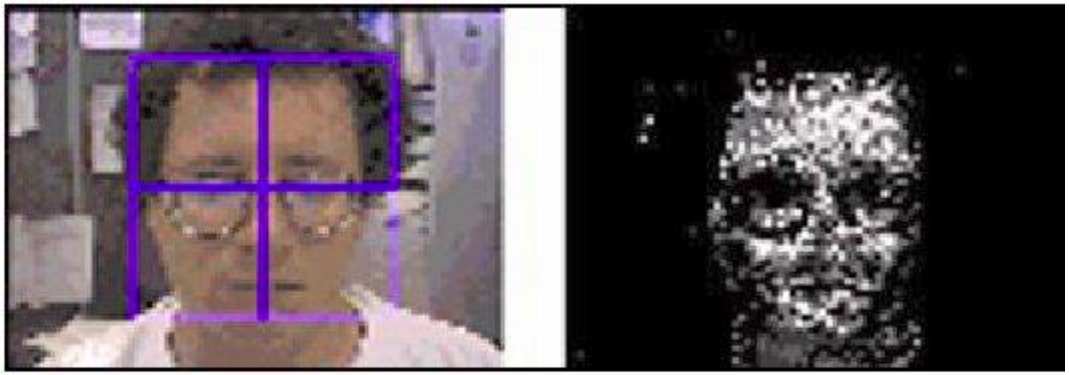
Sum over all pixels $\mathbf{a}$ under kernel K

Normalization term



$\Rightarrow$ *Mean-shift computes the weighted mean of all shifts (offsets), weighted by the point likelihood and the kernel function centered at* $\mathbf{x}$.

Recap: Explicit Weight Images



- **Histogram backprojection**
  - **Histogram is an empirical estimate of** $p(color \mid object) = p(c \mid o)$
  - **Bayes' rule says:** $p(o|c) = \frac{p(c|o)p(o)}{p(c)}$
  - **Simplistic approximation: assume** $p(o)/p(c)$ **is constant.**

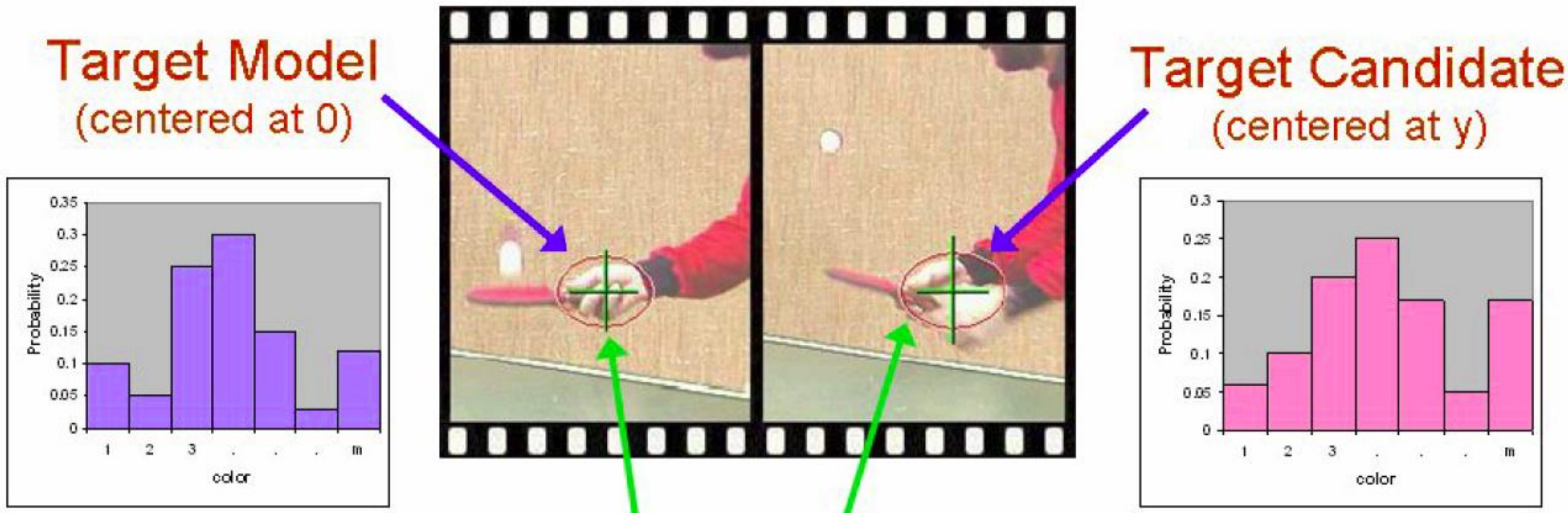
  $\Rightarrow$ **Use histogram** h **as a lookup table to set pixel values in the weight image.**
  - **If pixel maps to histogram bucket** i**, set weight for pixel to** $h(i)$**.**

Recap: Scale Adaptation in CAMshift

Mean shift window initialization

Image source: http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_meanshift/py_meanshift.html

Recap: Tracking with Implicit Weight Images



$$\vec{q} = \{q_u\}_{u=1..m} \qquad \sum_{u=1}^{m} q_u = 1$$

$$\vec{p}(y) = \{p_u(y)\}_{u=1..m} \qquad \sum_{u=1}^{m} p_u = 1$$

Similarity Function: $f(y) = f\left[\vec{q}, \vec{p}(y)\right]$

Recap: Comaniciu's Mean-Shift

- **Color histogram representation**

target model: $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m}$ $\quad \sum_{u=1}^{m} \hat{q}_u = 1$

target candidate: $\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\ldots m}$ $\quad \sum_{u=1}^{m} \hat{p}_u = 1 .$

- **Measuring distances between histograms**
  - Distance as a function of window location $\mathbf{y}$

$$d(\mathbf{y}) = \sqrt{1 - \rho\left[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}\right]} ,$$

  - where $\hat{\rho}(\mathbf{y})$ is the Bhattacharyya coefficient

$$\hat{\rho}(\mathbf{y}) \equiv \rho\left[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}\right] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} ,$$

Slide credit: Robert Collins

Recap: Comaniciu's Mean-Shift

- Compute histograms via Parzen estimation

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^\star\|^2)\delta\left[b(\mathbf{x}_i^\star) - u\right] ,$$

$$\hat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right)\delta\left[b(\mathbf{x}_i) - u\right] ,$$

  - where $k(\cdot)$ is some radially symmetric smoothing kernel profile, $\mathbf{x}_i$ is the pixel at location i, and $b(\mathbf{x}_i)$ is the index of its bin in the quantized feature space.

- Consequence of this formulation
  - Gathers a histogram over a neighborhood
  - Also allows interpolation of histograms centered around an off-lattice location.

Recap: Result of Taylor Expansion

- Simple update procedure: At each iteration, perform

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)} \quad \text{where } g(x) = -k'(x),$$

  - which is just standard mean-shift on (implicit) weight image w_i.
  - Let's look at the weight image more closely. For each pixel $\mathbf{x}_i$

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \boxed{\delta\left[b(\mathbf{x}_i) - u\right]}.$$

This is only 1 once in the summation

⇒ If pixel $\mathbf{x}_i$'s value maps to histogram bucket B, then

$$w_i = \sqrt{q_B / p_B(\mathbf{y}_0)}$$

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

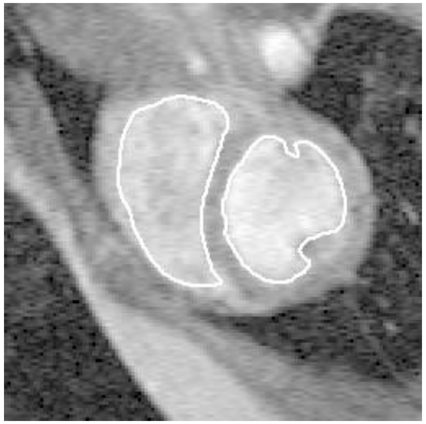
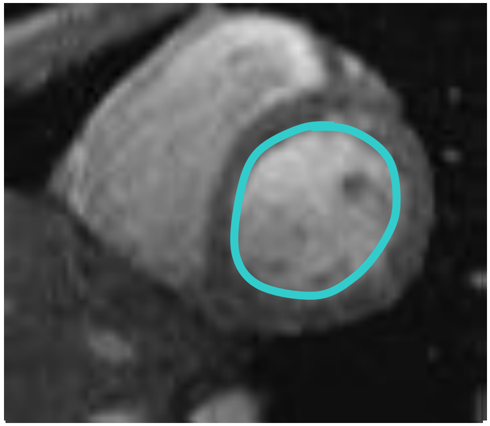


Image source: Yuri Boykov

Recap: Deformable Contours

- Given
  - Initial contour (model) near desired object
- Goal
  - Evolve the contour to fit the exact object boundary



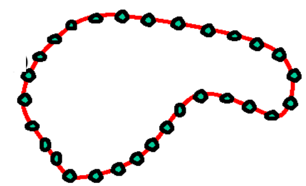
- Main ideas
  - Iteratively adjust the elastic band so as to be near image positions with high gradients, and
  - Satisfy shape “preferences” or contour priors
  - Formulation as energy minimization problem.

M. Kass, A. Witkin, D. Terzopoulos. Snakes: Active Contour Models, IJCV1988.

Image source: Yuri Boykov

Recap: Energy Function

- Definition
  - Total energy (cost) of the current snake



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

- Internal energy
  - Encourage prior shape preferences: e.g., smoothness, elasticity, particular known shape.
- External energy
  - Encourage contour to fit on places where image structures exist, e.g., edges.

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

Recap: Energy Formulation

- **Total energy**

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

  - **with the component terms**

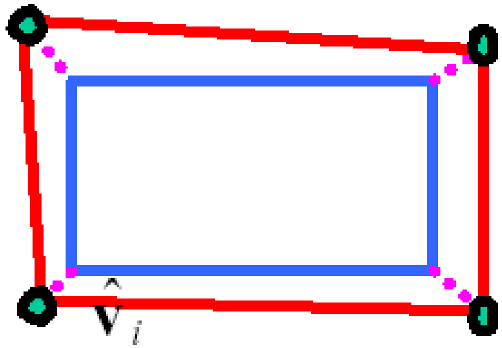
$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

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

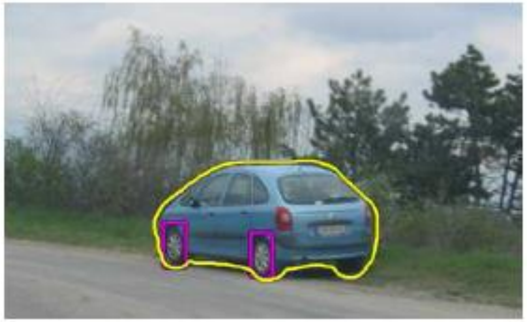
Behavior can be controlled by adapting the weights α, β, γ.

Recap: Extension with Shape Priors

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

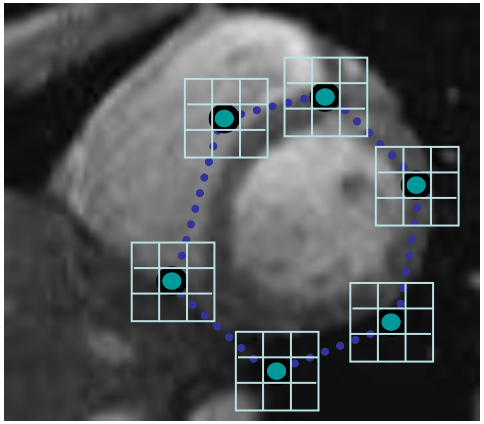


$$E_{internal} += \alpha \cdot \sum_{i=0}^{n-1} (v_i - \hat{v}_i)^2$$



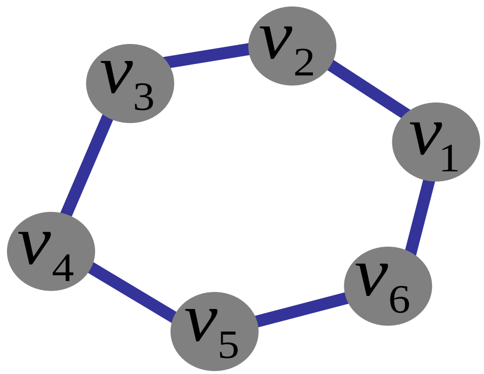
where $\{\hat{v}_i\}$ are the points of the known shape.

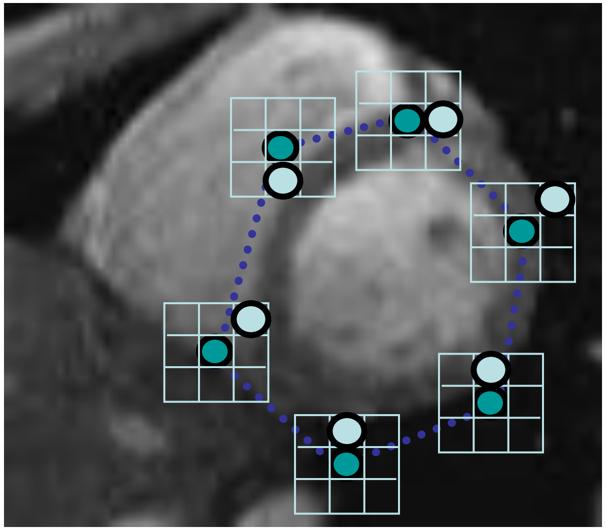
Recap: Greedy Energy Minimization



- Greedy optimization
  - For each point, search window around it and move to where energy function is minimal.
  - Typical window size, e.g., 5×5 pixels
- Stopping criterion
  - Stop when predefined number of points have not changed in last iteration, or after max number of iterations.
- Note:
  - Local optimization - need decent initialization!
  - Convergence not guaranteed

Recap: Energy Min. by Dynamic Programming





- **Dynamic Programming solution**
  - Limit possible moves to neighboring pixels (discrete states).
  - Find the best joint move of all points using Viterbi algorithm.
  - Iterate until optimal position for each point is the center of the box, *i.e.*, the snake is optimal in the local search space constrained by boxes.

Slide credit: Kristen Grauman

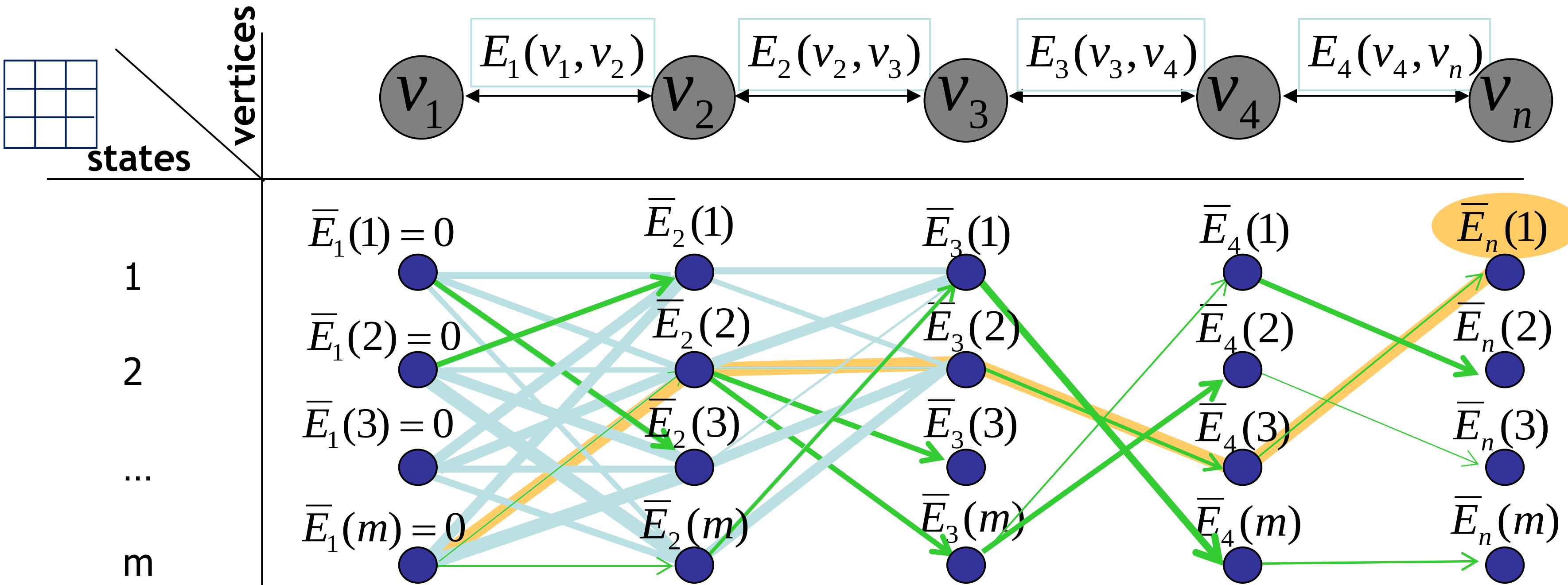
[Amini, Weymouth, Jain, 1990]

Figure source: Yuri Boykov

Recap: Viterbi Algorithm

- **Main idea:**
  - Determine optimal state of predecessor, for each possible state
  - Then backtrack from best state for last vertex

$$E_{total} = E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$$

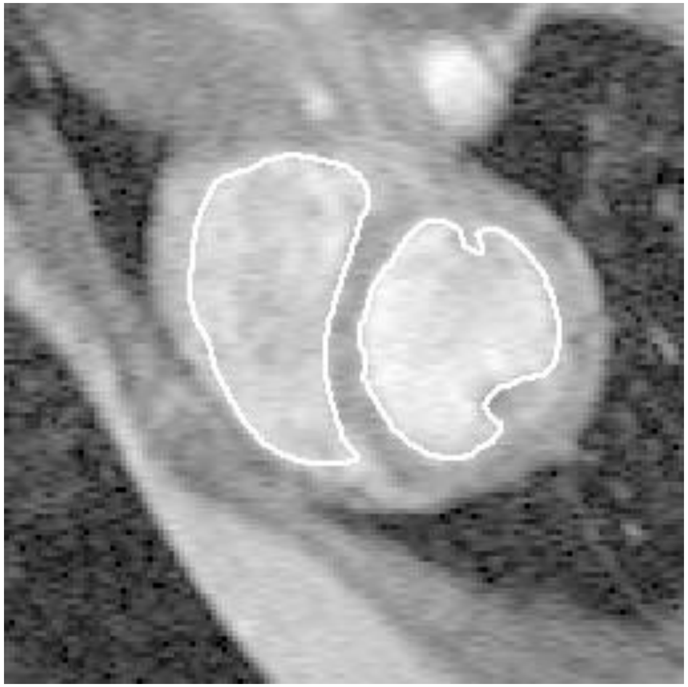


Complexity: $\mathcal{O}(nm^2)$ **vs. brute force search ______?**

Slide credit: Kristen Grauman, adapted from Yuri Boykov

Recap: Tracking via Deformable Contours

- **Idea**
  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)

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

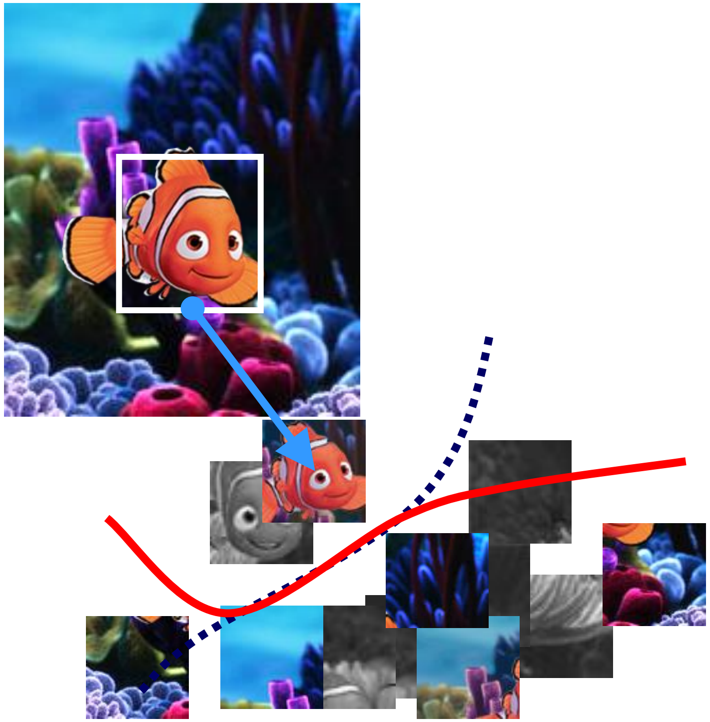
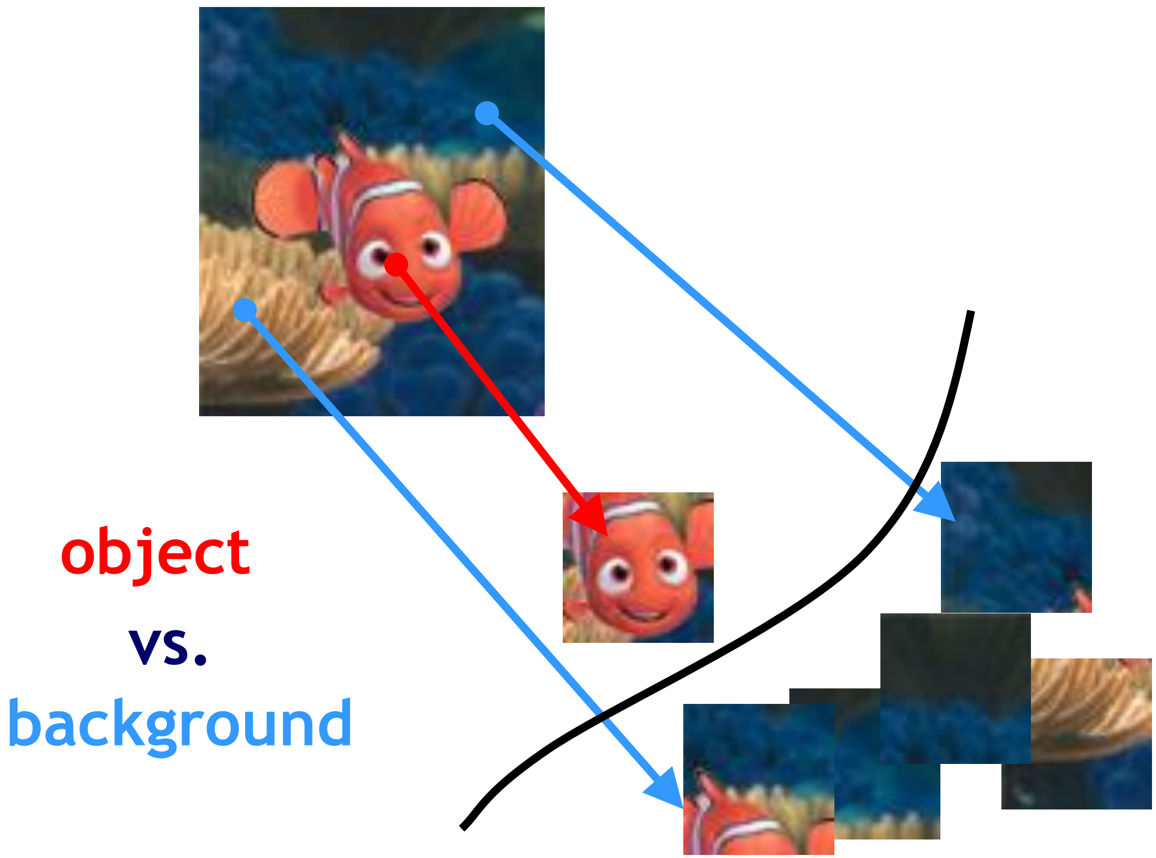


Image source: Helmut Grabner, Disney/Pixar

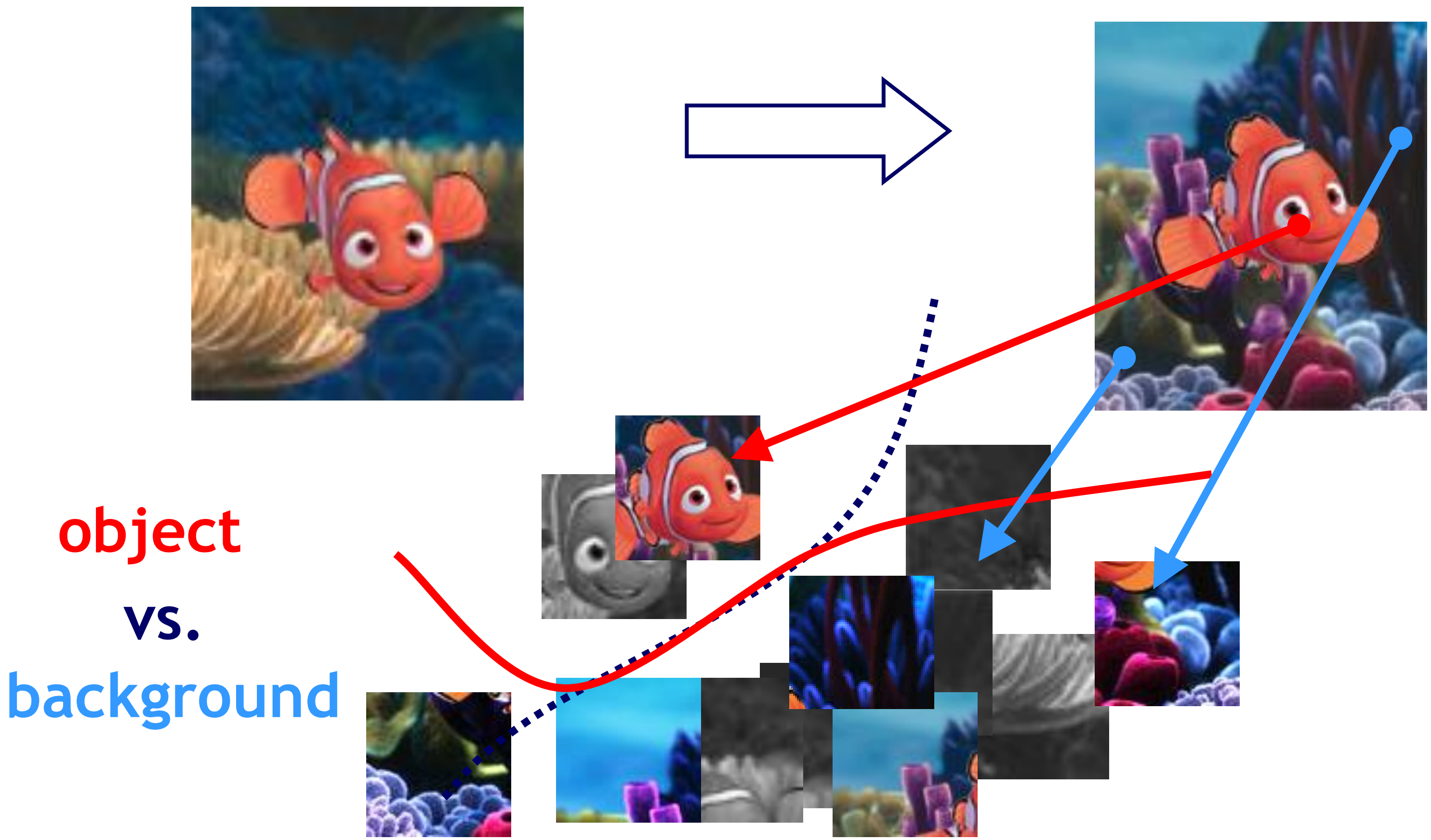
Recap: Tracking as Online Classification

- **Tracking as binary classification problem**



Recap: Tracking as Online Classification

- Tracking as binary classification problem



  - Handle object <u>and</u> background changes by **online updating**

Recap: AdaBoost – "Adaptive Boosting"

- Main idea [Freund & Schapire, 1996]
  - Iteratively select an ensemble of classifiers
  - Reweight misclassified training examples after each iteration to focus training on difficult cases.
- Components
  - $h_m(\mathbf{x})$: "weak" or base classifier
    - Condition: <50% training error over any distribution
  - $H(\mathbf{x})$: "strong" or final classifier
- AdaBoost:
  - Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

$$H(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$$

Recap: AdaBoost – Algorithm

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1,\ldots,N$.
2. For $m = 1,\ldots,M$ iterations
   a) Train a new weak classifier $h_m(\mathbf{x})$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function

   $$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)$$

   $$I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$$

   b) Estimate the weighted error of this classifier on $\mathbf{X}$:

   $$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

   c) Calculate a weighting coefficient for $h_m(\mathbf{x})$:

   $$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$

   d) Update the weighting coefficients:

   $$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\right\}$$

Recap: From Offline to Online Boosting

- Main issue
  - Computing the weight distribution for the samples.
  - We do not know a priori the difficulty of a sample! (Could already have seen the same sample before...)
- Idea of Online Boosting
  - Estimate the importance of a sample by propagating it through a set of weak classifiers.
  - This can be thought of as modeling the information gain w.r.t. the first n classifiers and code it by the importance weight λ for the $n+1$ classifier.
  - Proven [Oza]: Given the same training set, Online Boosting converges to the same weak classifiers as Offline Boosting in the limit of $N \rightarrow \infty$ iterations.

N. Oza and S. Russell. Online Bagging and Boosting. Artificial Intelligence and Statistics, 2001.

Recap: From Offline to Online Boosting

off-line

Given:

- set of labeled training samples
  $\mathcal{X} = \{\langle \mathbf{x_1}, y_1 \rangle, ..., \langle \mathbf{x_L}, y_L \rangle \mid y_i \pm 1\}$
- weight distribution over them
  $D_0 = 1/L$

```
for n = 1 to N
```

- train a weak classifier using samples and weight dist.
  $$h_n^{weak}(\mathbf{x}) = \mathcal{L}(\mathcal{X}, D_{n-1})$$
- calculate error e_n
- calculate weight $\alpha_n = f(e_n)$
- update weight dist. D_n

```
next
```

$$h^{strong}(\mathbf{x}) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x}))$$

on-line

Given:

- ONE labeled training sample
  $\langle \mathbf{x}, y \rangle \mid y \pm 1$
- strong classifier to update

- initial importance $\lambda = 1$

```
for n = 1 to N
```

- update the weak classifier using samples and importance
  $$h_n^{weak}(\mathbf{x}) = \mathcal{L}(h_n^{weak}, \langle x, y \rangle, \lambda)$$
- update error estimation $\widehat{e_n}$
- update weight $\alpha_n = f(\widehat{e}_n)$
- update importance weight λ

```
next
```

$$h^{strong}(\mathbf{x}) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x}))$$

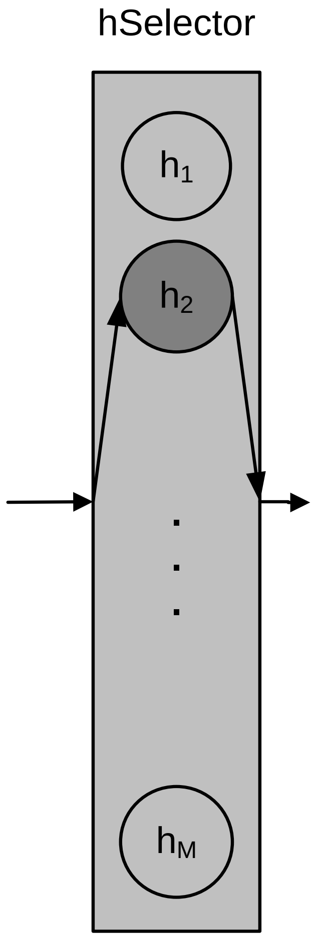
Recap: Online Boosting for Feature Selection

- **Introducing "Selector"**
  - **Selects one feature from its local feature pool**

$$\mathcal{H}^{weak} = \{h_1^{weak}, ..., h_M^{weak}\}$$
$$\mathcal{F} = \{f_1, ..., f_M\}$$

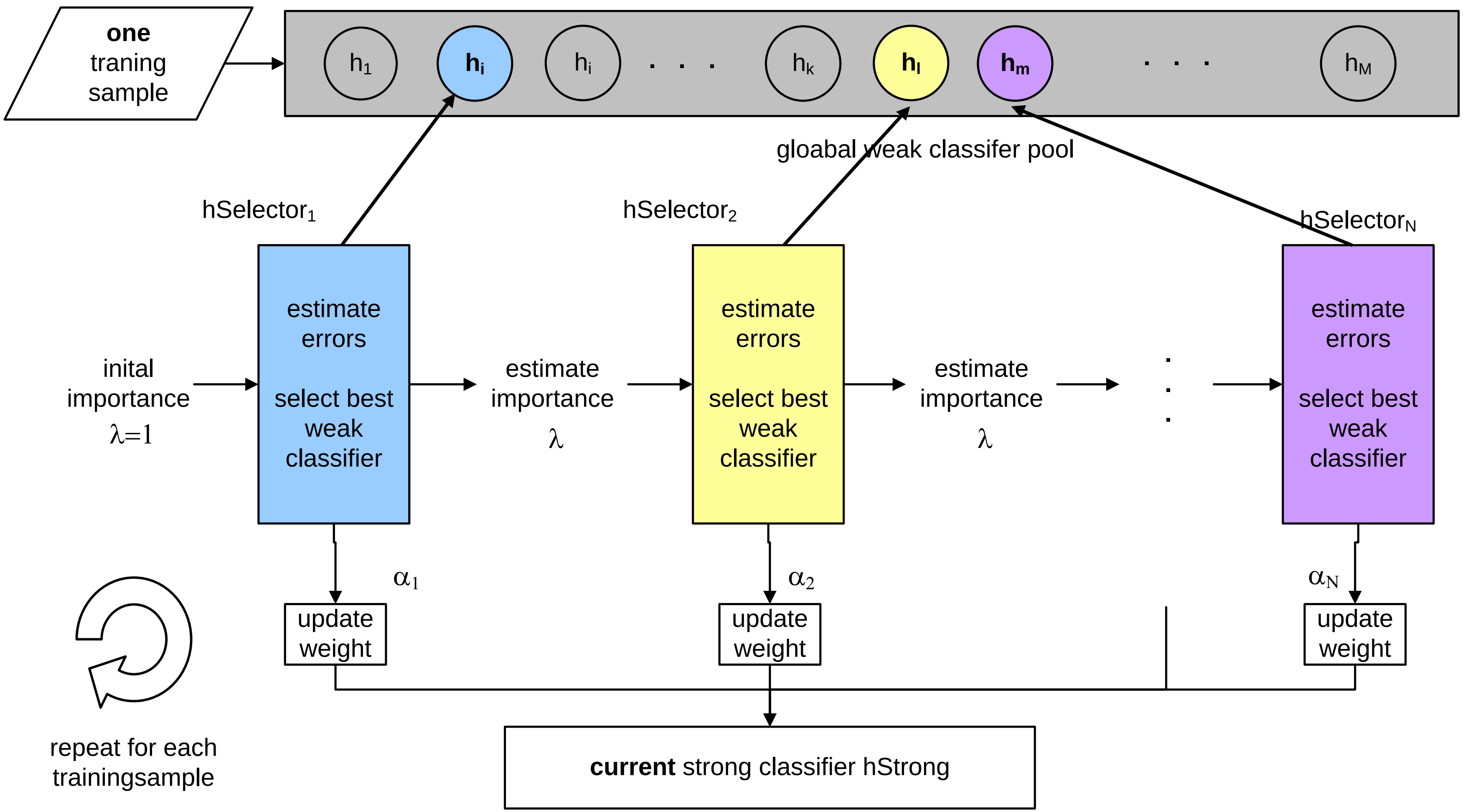
$$h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x})$$
$$m = \arg\min_i e_i$$

On-line boosting is performed on the Selectors and not on the weak classifiers directly.



H. Grabner and H. Bischof. On-line boosting and vision. CVPR, 2006.

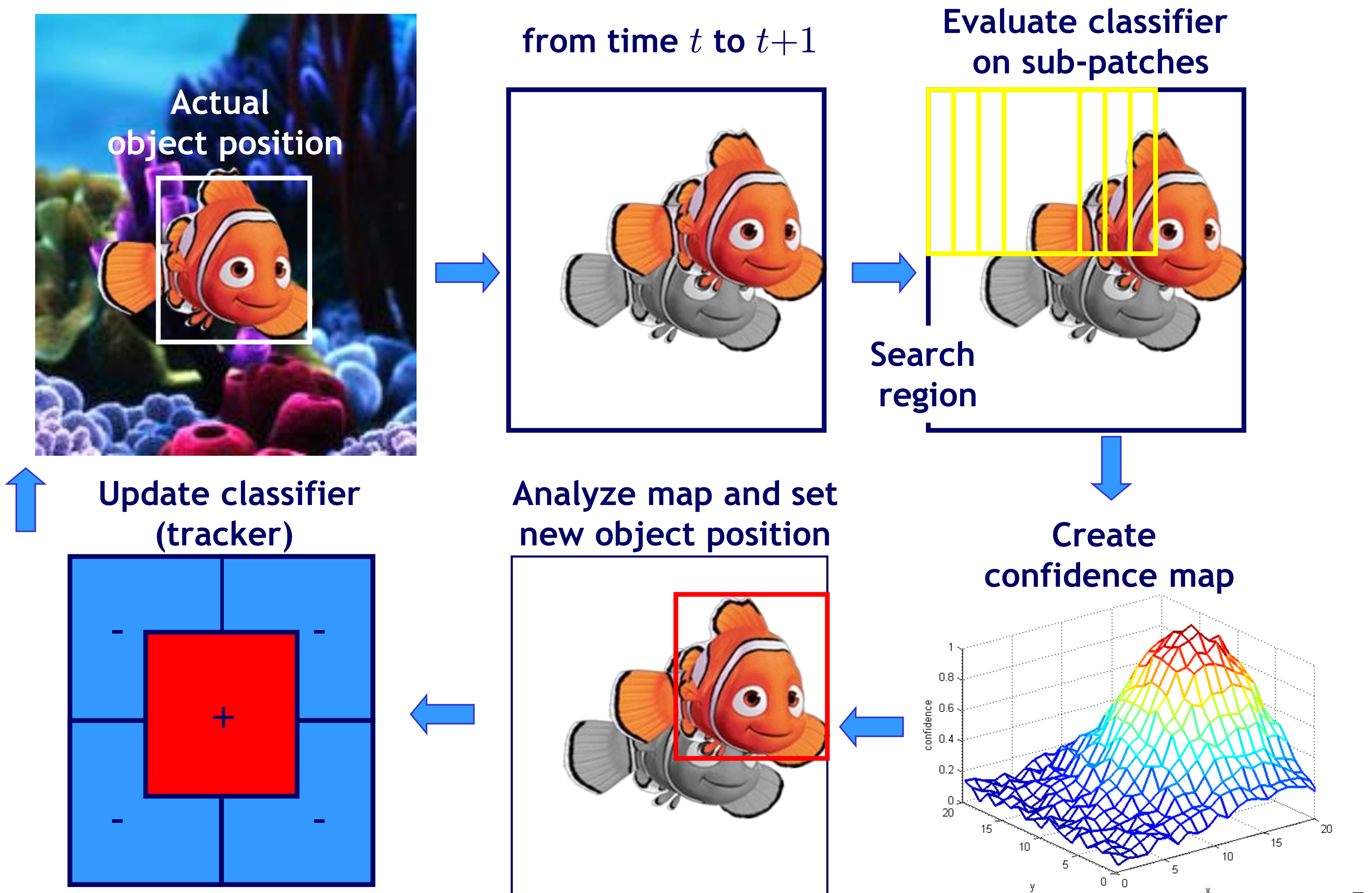
Recap: Direct Feature Selection



- **Shared feature pool for all selectors to save computation**

Slide credit: Helmut Grabner

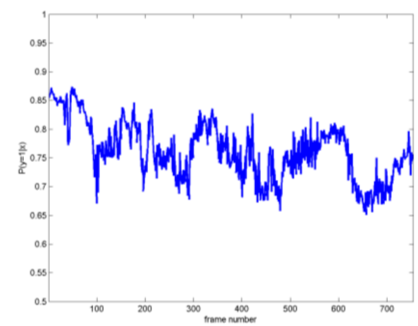
Recap: Tracking by Online Classification



Recap: Self-Learning and Drift

- Drift
  - Major problem in all adaptive or self-learning trackers.
  - Difficulty: distinguish "allowed" appearance changes due to lighting or viewpoint variation from "unwanted" appearance change due to drifting.
  - Cannot be decided based on the tracker confidence!
- Several approaches to address this
  - Comparison with initialization
  - Semi-supervised learning (additional data)
  - Additional information sources

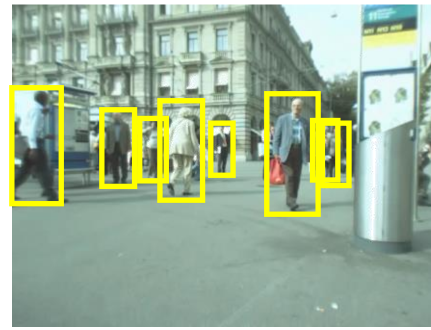
Tracked Patches



Confidence

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking



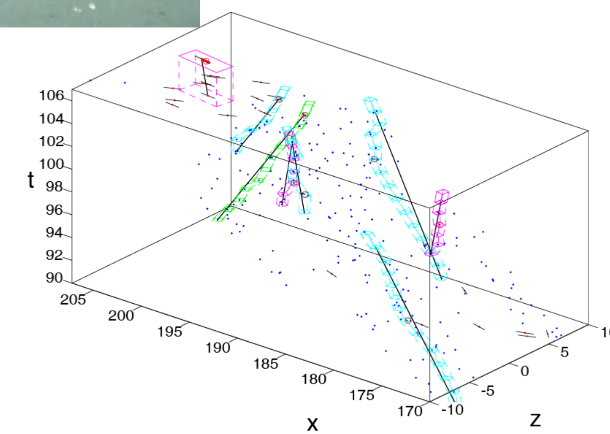
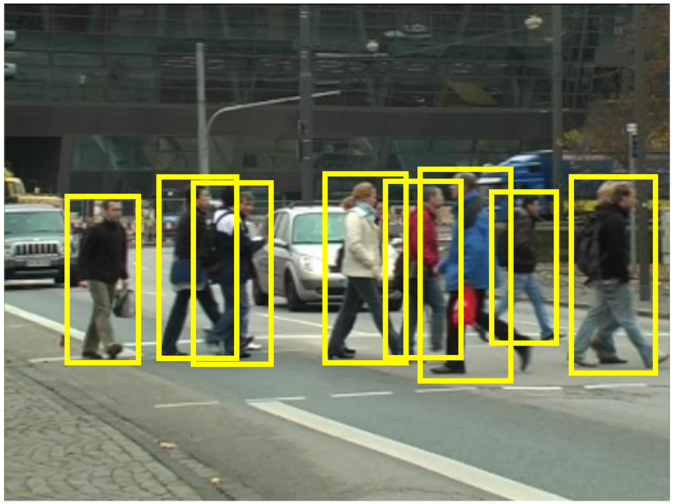
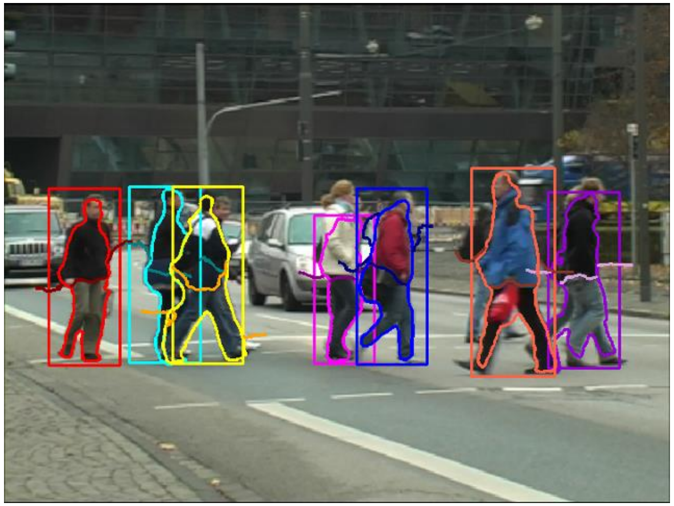


Image source: Helmut Grabner, Disney/Pixar

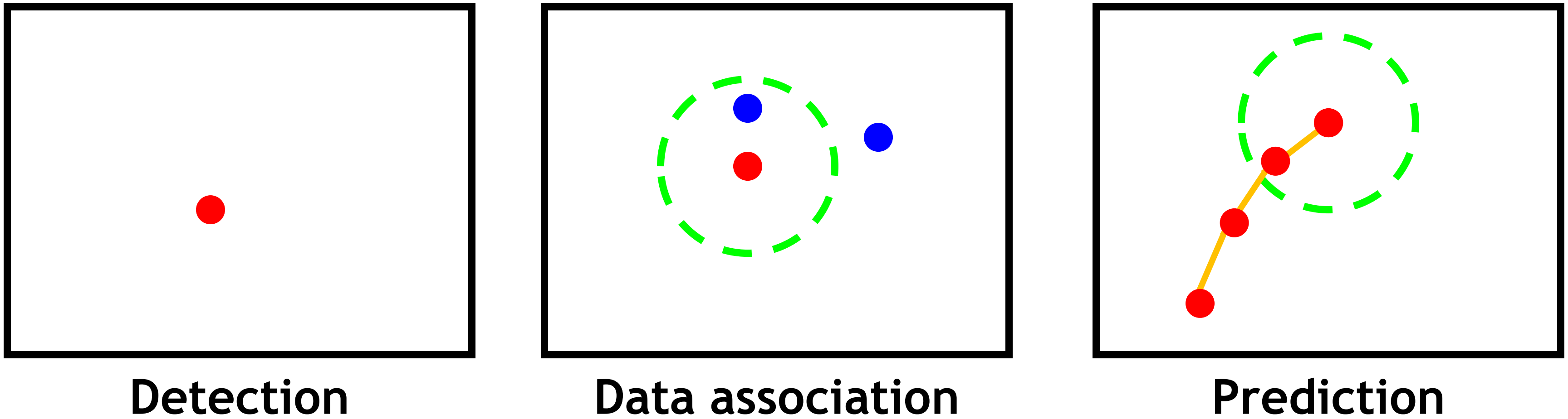
Recap: Tracking-by-Detection





- **Main ideas**
  - Apply a generic object detector to find objects of a certain class
  - Based on the detections, extract object appearance models
  - Link detections into trajectories

Elements of Tracking

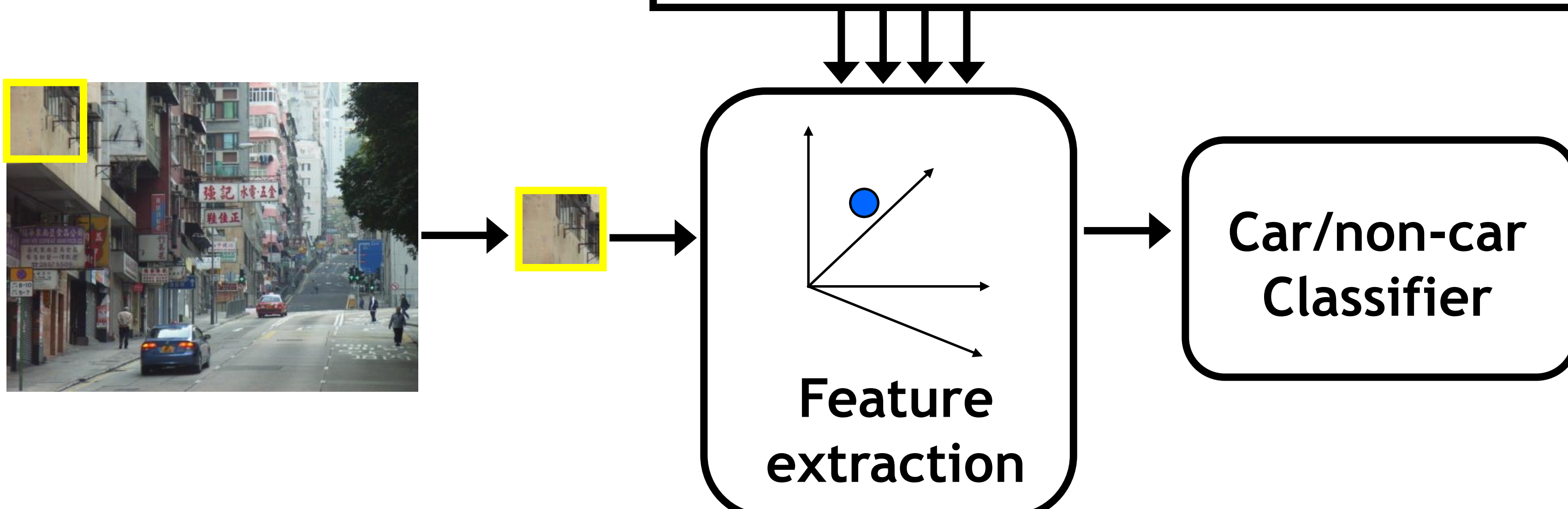


- **Detection**
  - *Where are candidate objects?*
- **Data association**
  - *Which detection corresponds to which object?*
- **Prediction**
  - *Where will the tracked object be in the next time step?*

Recap: Sliding-Window Object Detection

Fleshing out this pipeline a bit more, we need to:

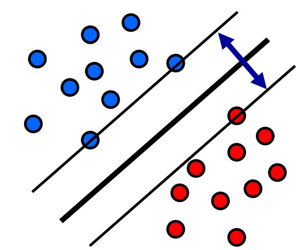
1. Obtain training data
2. Define features
3. Define classifier



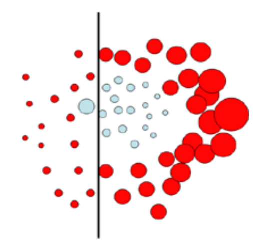
Recap: Object Detector Design

- In practice, the classifier often determines the design.
  - Types of features
  - Speedup strategies

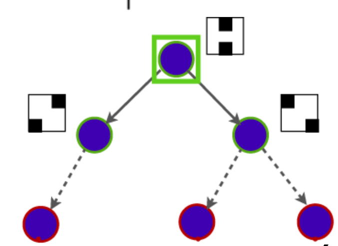
- We've looked at 2 state-of-the-art detector designs
  - Based on SVMs
    → HOG, DPM detectors



  - Based on Boosting
    → Viola-Jones, VeryFast, Roerei detectors



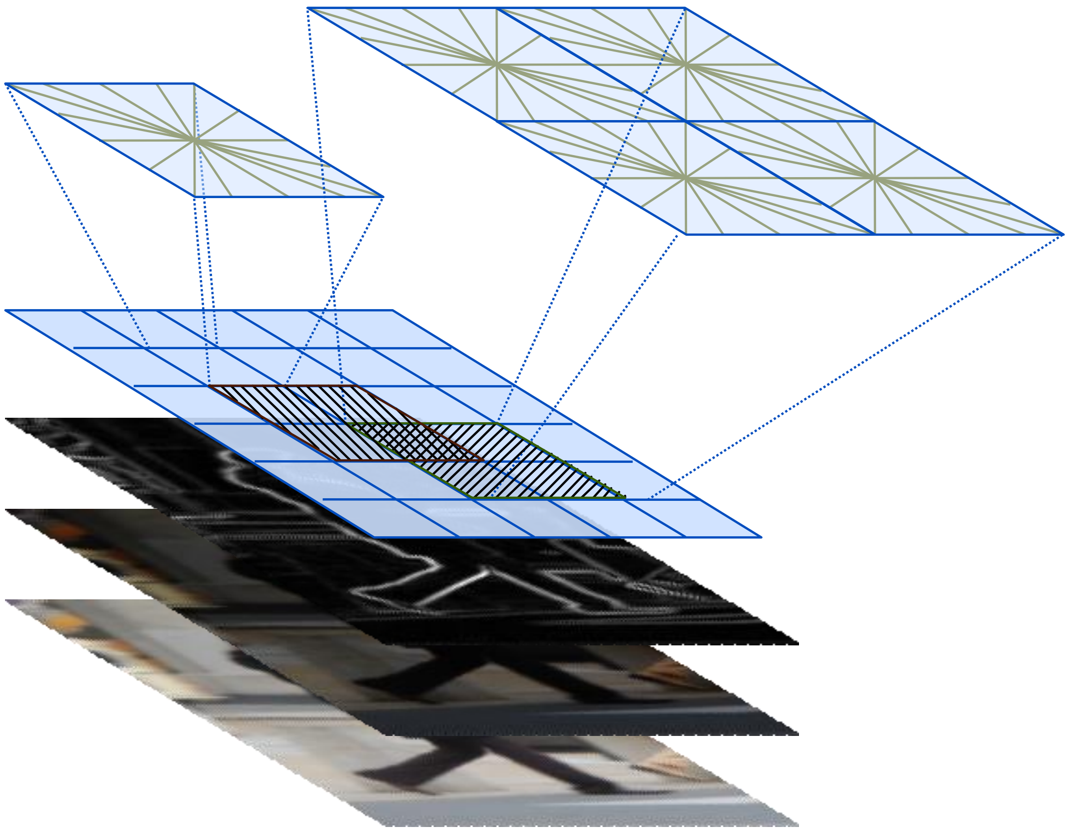
  - Based on Random Forests
    → (Cut due to time constraints...)



Recap: Histograms of Oriented Gradients (HOG)

- **Holistic object representation**
  - **Localized gradient orientations**

[…, …, …, …]



Object/Non-object

↑

Linear SVM

↑

Collect HOGs over detection window

↑

Contrast normalize over overlapping spatial cells

↑

Weighted vote in spatial & orientation cells

↑

Compute gradients

↑

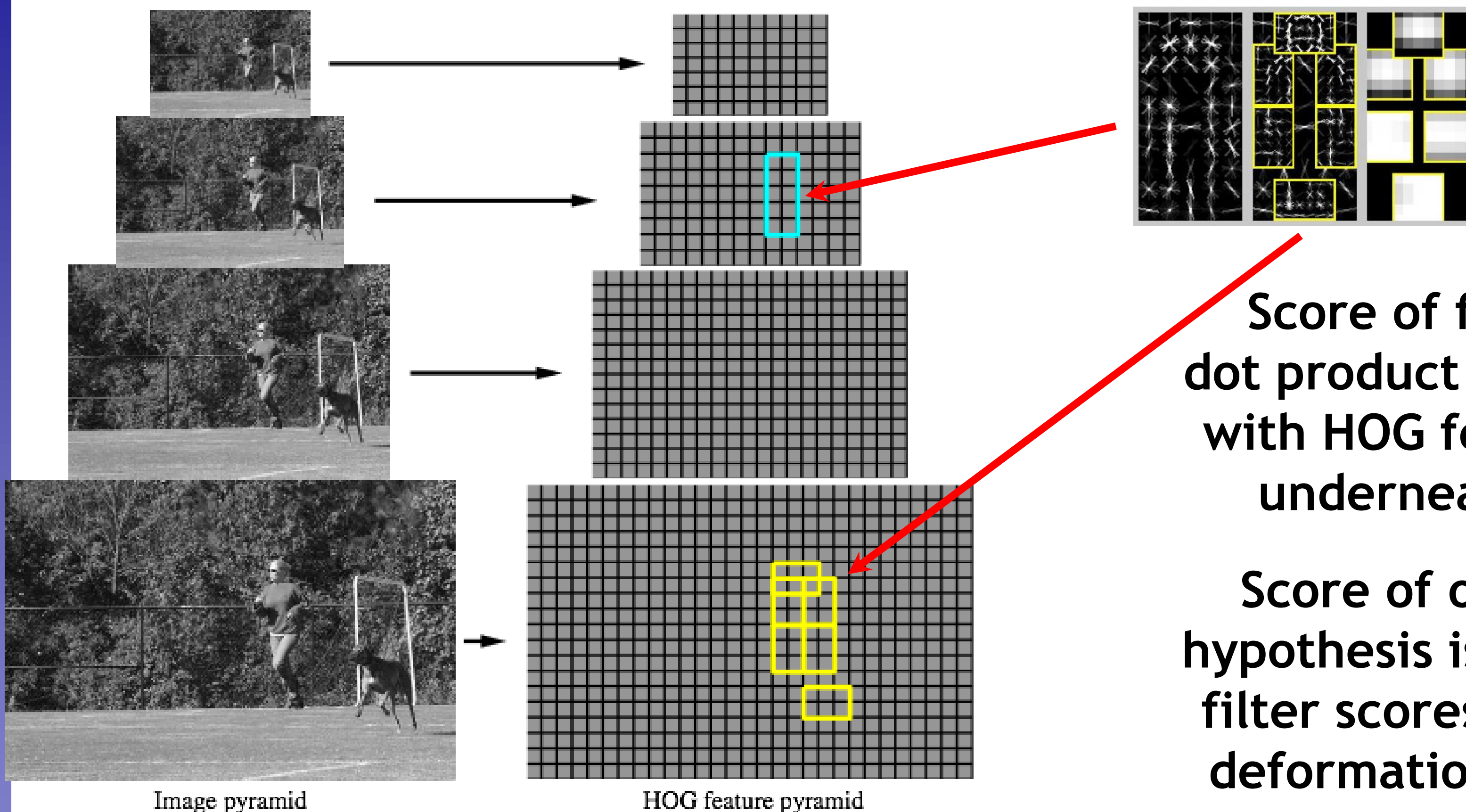
Gamma compression

↑

Image Window

Slide adapted from Navneet Dalal

Recap: Deformable Part-based Model (DPM)



Score of filter: dot product of filter with HOG features underneath it

Score of object hypothesis is sum of filter scores minus deformation costs

- **Multiscale model captures features at two resolutions**

[Felzenszwalb, McAllister, Ramanan, CVPR'08]

Recap: DPM Hypothesis Score

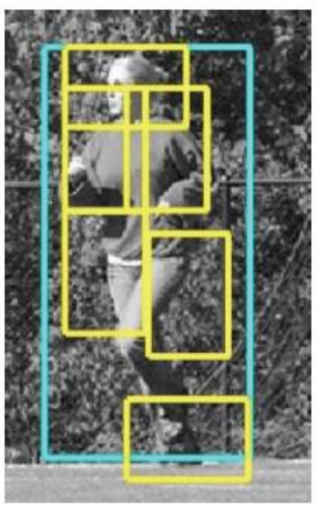
"data term" "spatial prior"

$$\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)$$

filters

displacements

deformation parameters



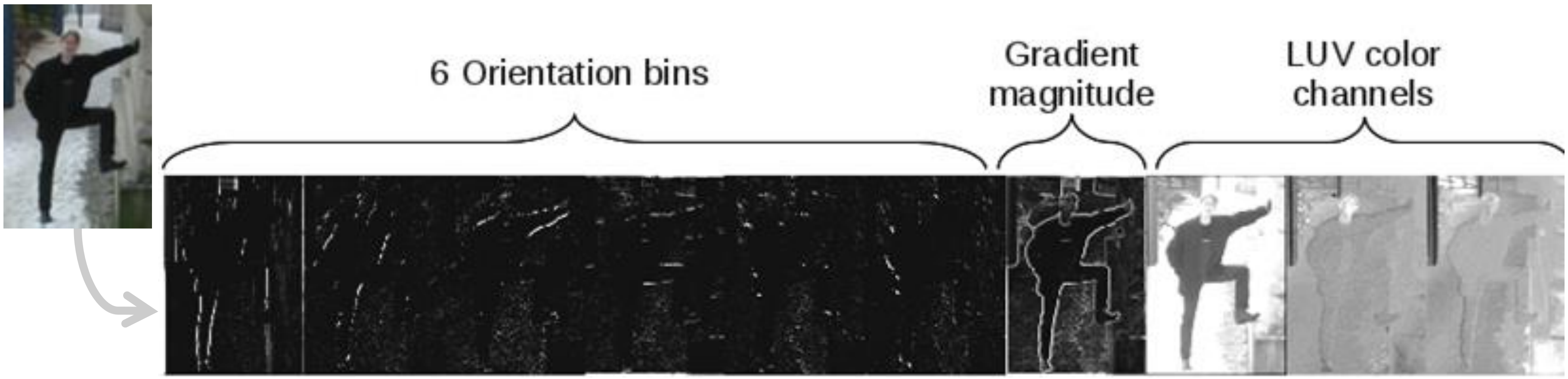
$$\text{score}(z) = \beta \cdot \Psi(H, z)$$

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features

[Felzenszwalb, McAllister, Ramanan, CVPR'08]

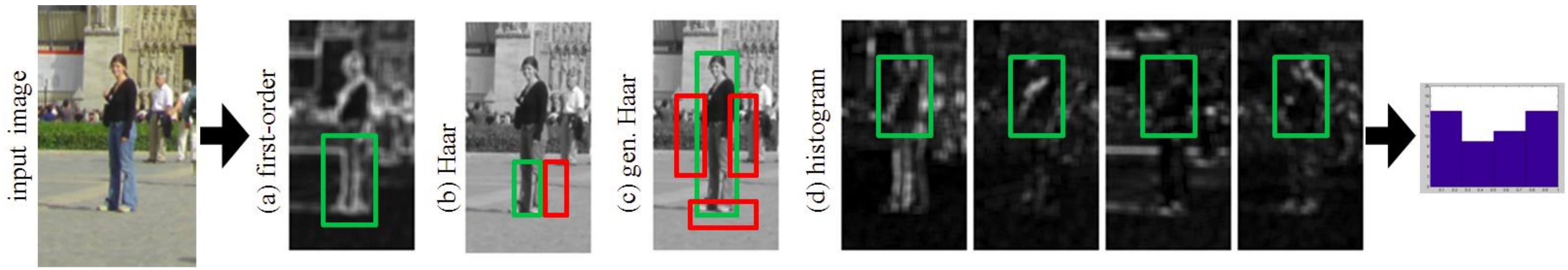
Recap: Integral Channel Features



- Generalization of Haar Wavelet idea from Viola-Jones
  - Instead of only considering intensities, also take into account other feature channels (gradient orientations, color, texture).
  - Still efficiently represented as integral images.

P. Dollar, Z. Tu, P. Perona, S. Belongie. Integral Channel Features, BMVC'09.

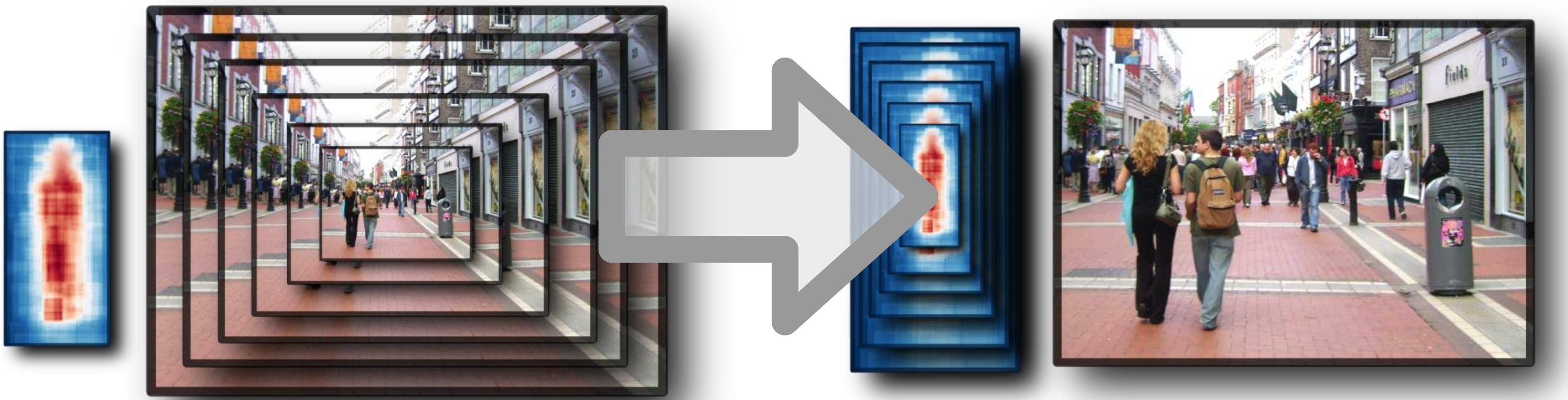
Recap: Integral Channel Features



- Generalize also block computation
  - 1st order features:
    - Sum of pixels in rectangular region.
  - 2nd-order features:
    - Haar-like difference of sum-over-blocks
  - Generalized Haar:
    - More complex combinations of weighted rectangles
  - Histograms
    - Computed by evaluating local sums on quantized images.

Recap: VeryFast Detector

- **Idea 1: Invert the template scale vs. image scale relation**



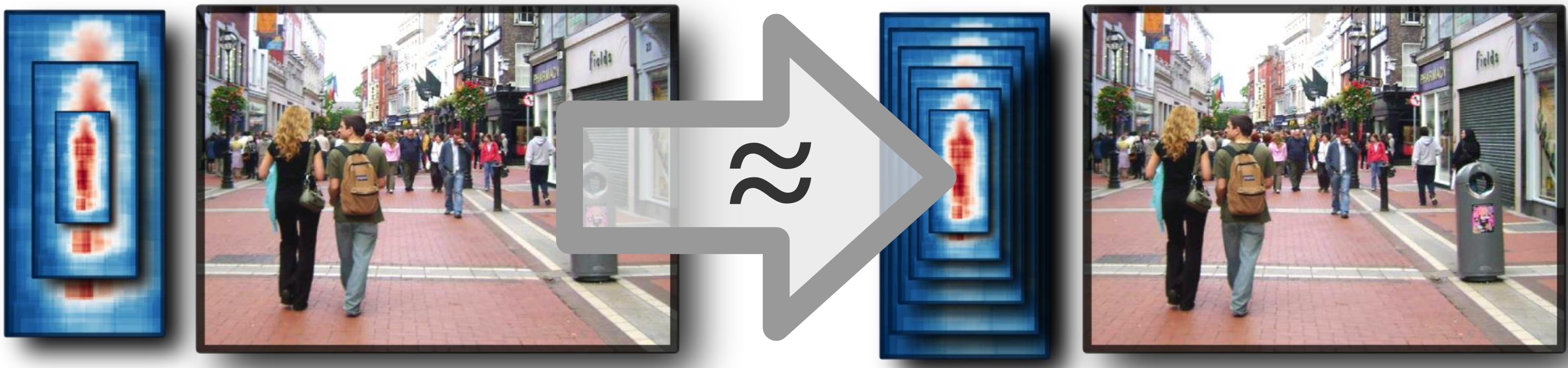
1 model,
50 image scales

50 models,
1 image scale

R. Benenson, M. Mathias, R. Timofte, L. Van Gool. [Pedestrian Detection at 100 Frames per Second](), CVPR'12.

Recap: VeryFast Detector

- **Idea 2: Reduce training time by feature interpolation**

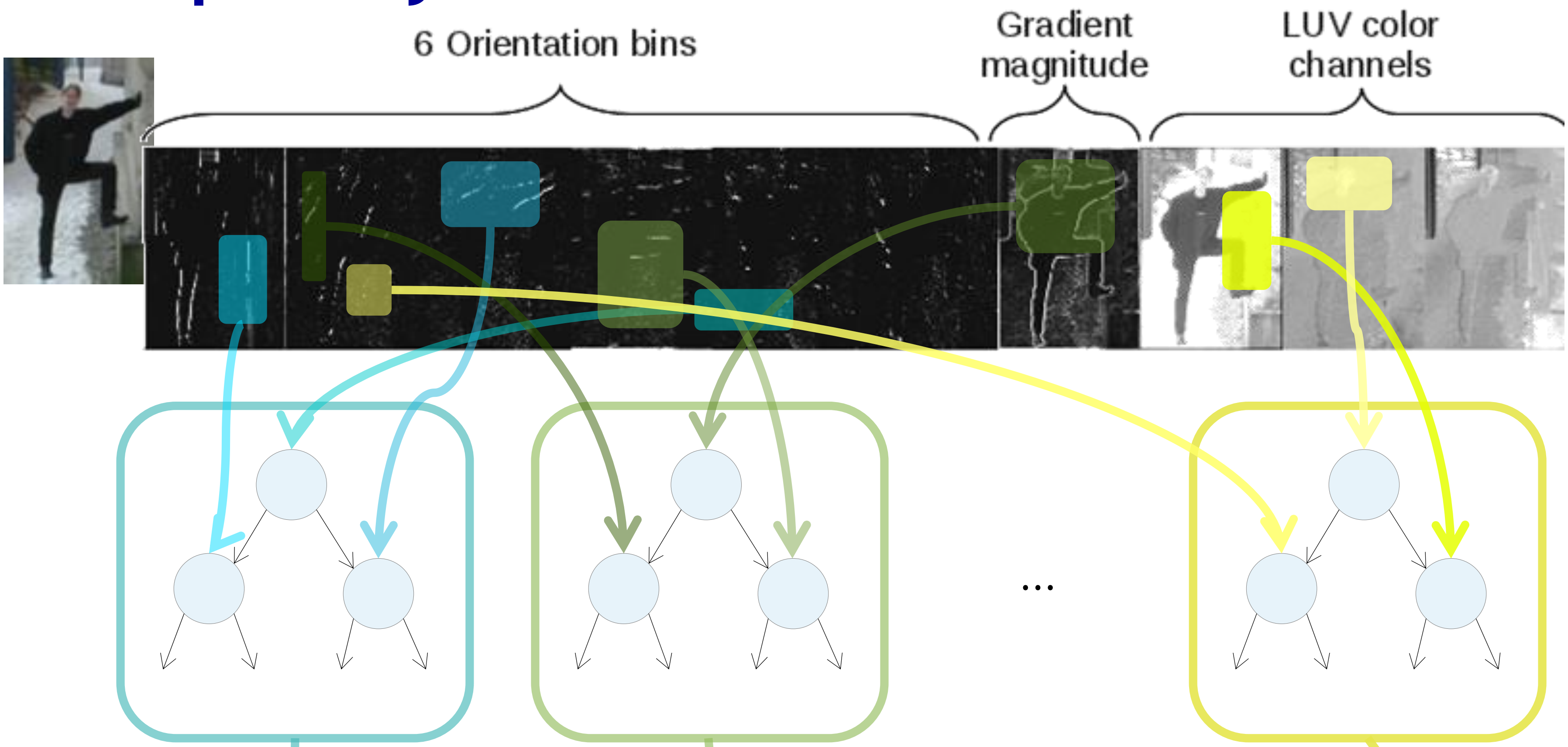


5 models,
1 image scale

50 models,
1 image scale

- **Shown to be possible for Integral Channel features**
  - P. Dollár, S. Belongie, Perona. [The Fastest Pedestrian Detector in the West](), BMVC 2010.

Recap: VeryFast Classifier Construction



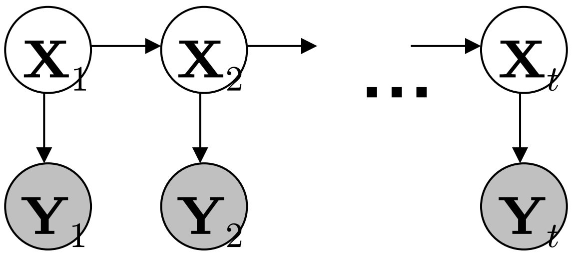
$$score = \; w_1 \cdot h_1 + \quad w_2 \cdot h_2 + \quad \ldots \quad + w_N \cdot h_N$$

- **Ensemble of short trees, learned by AdaBoost**

Slide credit: Rodrigo Benenson

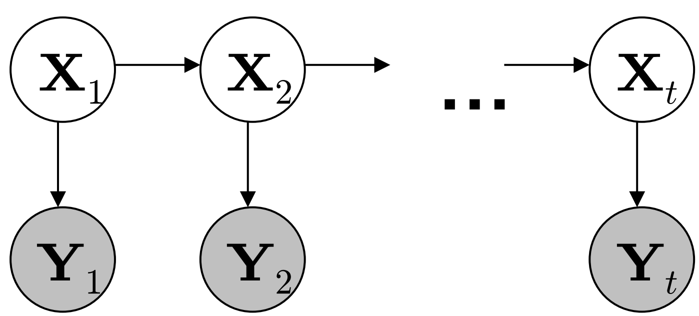
Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
  - Kalman filter
  - Particle filter
- Multi-Object Tracking
- Articulated Tracking



Recap: Tracking as Inference

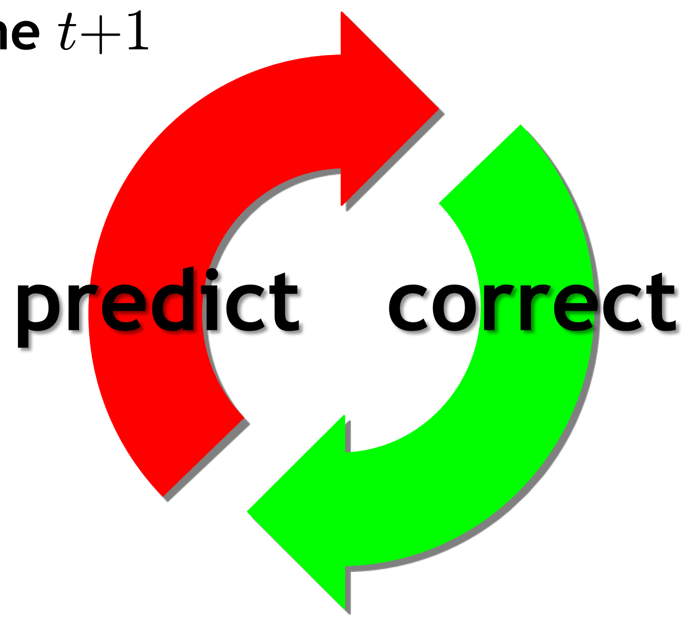
- Inference problem
  - The hidden state consists of the true parameters we care about, denoted $\mathbf{X}$.
  - The measurement is our noisy observation that results from the underlying state, denoted $\mathbf{Y}$.
  - At each time step, state changes (from $\mathbf{X}_{t\text{-}1}$ to $\mathbf{X}_t$)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



Slide credit: Kristen Grauman

Recap: Tracking as Induction

- Base case:
  - Assume we have initial prior that predicts state in absence of any evidence: $P(\mathbf{X}_0)$
  - At the first frame, *correct* this given the value of $\mathbf{Y}_0 = \mathbf{y}_0$
- Given corrected estimate for frame t:
  - Predict for frame $t+1$
  - Correct for frame $t+1$



Recap: 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:**

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

Recap: Linear Dynamic Models

- **Dynamics model**
  - State undergoes linear tranformation D_t plus Gaussian noise

$$\boldsymbol{x}_t \sim N\left(\boldsymbol{D}_t \boldsymbol{x}_{t-1}, \Sigma_{d_t}\right)$$

- **Observation model**
  - Measurement is linearly transformed state plus Gaussian noise

$$\boldsymbol{y}_t \sim N\left(\boldsymbol{M}_t \boldsymbol{x}_t, \Sigma_{m_t}\right)$$

Slide credit: S. Lazebnik, K. Grauman

Recap: Constant Velocity Model (1D)

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

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

(greek letters denote noise terms)

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

- **Measurement is position only**

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

Recap: Constant Acceleration Model (1D)

- State vector: position p, velocity v, and acceleration a.

$$x_t = \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} \qquad \begin{aligned} p_t &= p_{t-1} + (\Delta t) v_{t-1} + \varepsilon \\ v_t &= v_{t-1} + (\Delta t) a_{t-1} + \xi \\ a_t &= a_{t-1} + \zeta \end{aligned}$$

(greek letters denote noise terms)

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

- Measurement is position only

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

Recap: General Motion Models

- Assuming we have differential equations for the motion
  - E.g. for (undampened) periodic motion of a spring

$$\frac{d^2 p}{dt^2} = -p$$

- Substitute variables to transform this into linear system

$$p_1 = p \qquad p_2 = \frac{dp}{dt} \qquad p_3 = \frac{d^2 p}{dt^2}$$

- Then we have

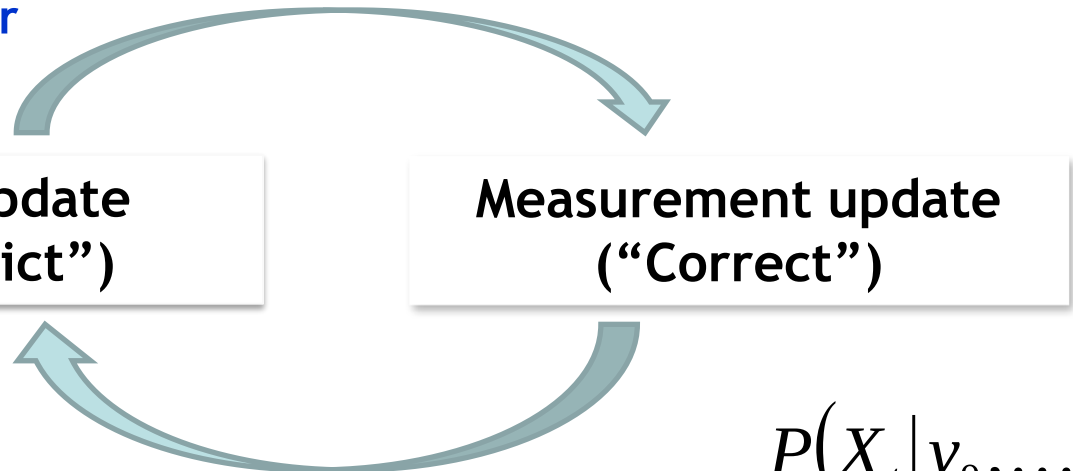
$$x_t = \begin{bmatrix} p_{1,t} \\ p_{2,t} \\ p_{3,t} \end{bmatrix} \qquad \begin{aligned} p_{1,t} &= p_{1,t-1} + (\Delta t) p_{2,t-1} + \varepsilon \\ p_{2,t} &= p_{2,t-1} + (\Delta t) p_{3,t-1} + \xi \\ p_{3,t} &= -p_{1,t-1} + \zeta \end{aligned} \qquad D_t = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ -1 & 0 & 0 \end{bmatrix}$$

Recap: The 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.



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

Mean and std. dev. of predicted state:

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

Time advances: t++

$$P\left(X_t \middle| y_0, \ldots, y_t\right)$$

Mean and std. dev. of corrected state:

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

Recap: General Kalman Filter (>1dim)

- What if state vectors have more than one dimension?

PREDICT

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

$$\Sigma_t^- = D_t \Sigma_{t-1}^+ D_t^T + \Sigma_{d_t}$$

CORRECT

$$K_t = \Sigma_t^- M_t^T \left(M_t \Sigma_t^- M_t^T + \Sigma_{m_t} \right)^{-1}$$

"Kalman gain" "residual"

$$x_t^+ = x_t^- + K_t \left(y_t - M_t x_t^- \right)$$

$$\Sigma_t^+ = \left(I - K_t M_t \right) \Sigma_t^-$$

More weight on residual when measurement error covariance approaches 0.

Less weight on residual as a priori estimate error covariance approaches 0.

for derivations, see F&P Chapter 17.3

Recap: Kalman Filter

- Algorithm summary
  - Assumption: linear model

$$\begin{aligned}\mathbf{x}_t &= \mathbf{D}_t \mathbf{x}_{t-1} + \varepsilon_t \\ \mathbf{y}_t &= \mathbf{M}_t \mathbf{x}_t + \delta_t\end{aligned}$$

  - Prediction step

$$\begin{aligned}\mathbf{x}_t^- &= \mathbf{D}_t \mathbf{x}_{t-1}^+ \\ \mathbf{\Sigma}_t^- &= \mathbf{D}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{D}_t^T + \mathbf{\Sigma}_{d_t}\end{aligned}$$

  - Correction step

$$\begin{aligned}\mathbf{K}_t &= \mathbf{\Sigma}_t^- \mathbf{M}_t^T \left(\mathbf{M}_t \mathbf{\Sigma}_t^- \mathbf{M}_t^T + \mathbf{\Sigma}_{m_t}\right)^{-1} \\ \mathbf{x}_t^+ &= \mathbf{x}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{M}_t \mathbf{x}_t^-\right) \\ \mathbf{\Sigma}_t^+ &= \left(\mathbf{I} - \mathbf{K}_t \mathbf{M}_t\right) \mathbf{\Sigma}_t^-\end{aligned}$$

Recap: Extended Kalman Filter (EKF)

- Algorithm summary
  - Nonlinear model

$$\mathbf{x}_t = \mathbf{g}\left(\mathbf{x}_{t-1}\right) + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{h}\left(\mathbf{x}_t\right) + \delta_t$$

  - Prediction step

$$\mathbf{x}_t^- = \mathbf{g}\left(\mathbf{x}_{t-1}^+\right)$$
$$\mathbf{\Sigma}_t^- = \mathbf{G}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{G}_t^T + \mathbf{\Sigma}_{d_t}$$

  - Correction step

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{H}_t^T \left(\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^T + \mathbf{\Sigma}_{m_t}\right)^{-1}$$
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{h}\left(\mathbf{x}_t^-\right)\right)$$
$$\mathbf{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \mathbf{\Sigma}_t^-$$

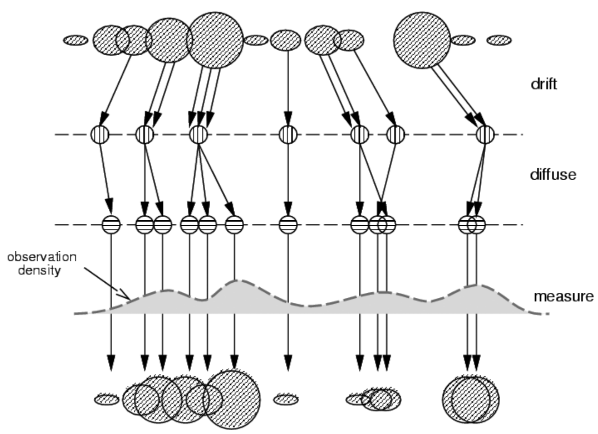
with the Jacobians

$$\mathbf{G}_t = \left.\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_{t-1}^+}$$

$$\mathbf{H}_t = \left.\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_t^-}$$

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
  - Kalman filters
  - Particle filters
- Multi-Object Tracking
- Articulated Tracking



Recap: Propagation of General Densities

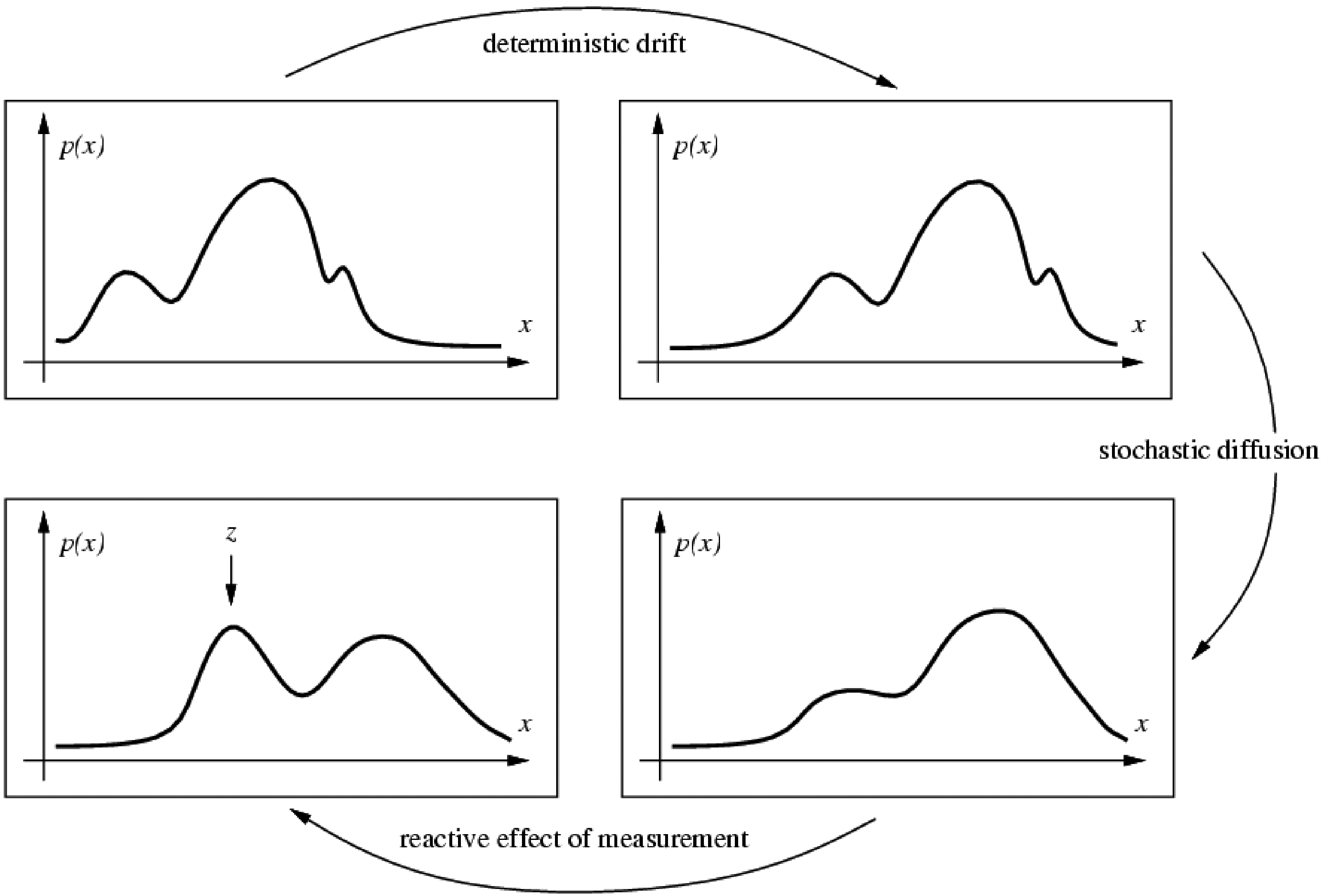
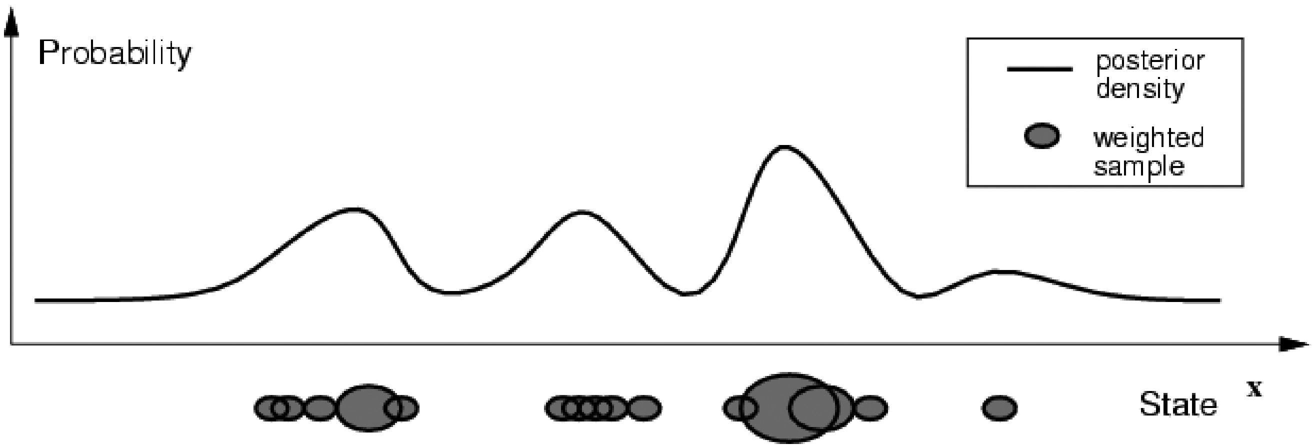


Figure from Isard & Blake

Recap: Factored Sampling

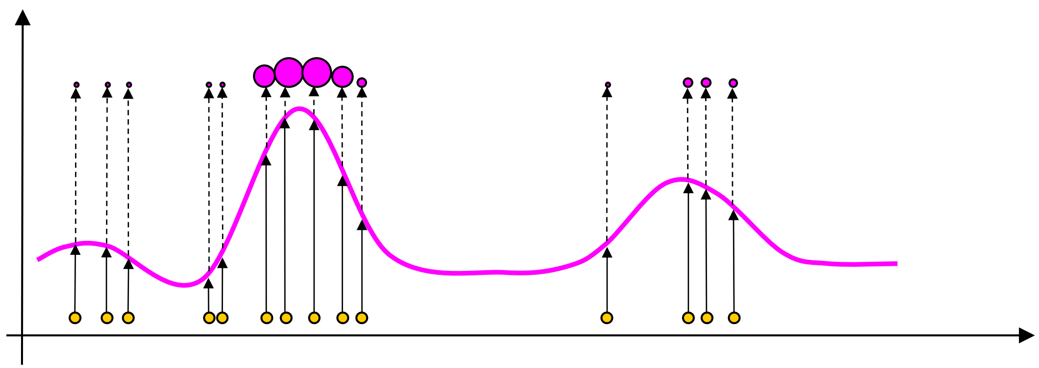


- **Idea: Represent state distribution non-parametrically**
  - **Prediction: Sample points from prior density for the state,** $P(X)$
  - **Correction: Weight the samples according to** $P(Y|X)$

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

Recap: Particle Filtering

- **Many variations, one general concept:**
  - ***Represent the posterior pdf by a set of randomly chosen weighted samples (particles)***



  - **Randomly Chosen = Monte Carlo (MC)**
  - **As the number of samples become very large - the characterization becomes an equivalent representation of the true pdf.**

Recap: Sequential Importance Sampling

$$\text{function } \left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$$

$\eta = 0$ — **Initialize**

for $i = 1{:}N$

$$\mathbf{x}_t^i \sim q(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{y}_t)$$ **Sample from proposal pdf**

$$w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$$ **Update weights**

$$\eta = \eta + w_t^i$$ **Update norm. factor**

end

for $i = 1{:}N$

$$w_t^i = w_t^i / \eta$$ **Normalize weights**

end

Recap: Sequential Importance Sampling

$$\textbf{function} \; \left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$$

$\eta = 0$ — **Initialize**

for $i = 1{:}N$

$\mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — **Sample from proposal pdf**

$w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — **Update weights**

$\eta = \eta + w_t^i$ — **Update norm. factor**

end

for $i = 1{:}N$

$w_t^i = w_t^i / \eta$ — **Normalize weights**

end

For a concrete algorithm, we need to define the importance density $q(.|.)$!

Recap: SIS Algorithm with Transitional Prior

$$\textbf{function} \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$\eta = 0$ — **Initialize**

for $i = 1{:}N$

$\quad \mathbf{x}_t^i \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$ — **Sample from proposal pdf**

$\quad w_t^i = w_{t-1}^i p(\mathbf{y}_t | \mathbf{x}_t^i)$ — **Update weights**

$\quad \eta = \eta + w_t^i$ — **Update norm. factor**

end

for $i = 1{:}N$

$\quad w_t^i = w_t^i / \eta$ — **Normalize weights**

end

Transitional prior

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{y}_t) = p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$$

Recap: Resampling

- **Degeneracy problem with SIS**
  - After a few iterations, most particles have negligible weights.
  - Large computational effort for updating particles with very small contribution to $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.

- **Idea: Resampling**
  - Eliminate particles with low importance weights and increase the number of particles with high importance weight.

$$\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N \rightarrow \left\{\mathbf{x}_t^{i*}, \frac{1}{N}\right\}_{i=1}^N$$

  - The new set is generated by sampling with replacement from the discrete representation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ such that

$$Pr\left\{\mathbf{x}_t^{i*} = \mathbf{x}_t^j\right\} = w_t^j$$

Recap: Efficient Resampling Approach

- **From Arulampalam paper:**

Algorithm 2: Resampling Algorithm

$[\{\mathbf{x}_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}] = \text{RESAMPLE } [\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$

- Initialize the CDF: $c_1 = 0$
- FOR $i = 2 : N_s$
  - Construct CDF: $c_i = c_{i-1} + w_k^i$
- END FOR
- Start at the bottom of the CDF: $i = 1$
- Draw a starting point: $u_1 \sim \mathbb{U}[0, N_s^{-1}]$
- FOR $j = 1 : N_s$
  - Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  - WHILE $u_j > c_i$
  * $i = i + 1$
  - END WHILE
  - Assign sample: $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$
  - Assign weight: $w_k^j = N_s^{-1}$
  - Assign parent: $i^j = i$
- END FOR

Basic idea: choose one initial small random number; deterministically sample the rest by "crawling" up the cdf. This is $\mathcal{O}(N)$!

Recap: Generic Particle Filter

$$\textbf{function} \quad \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = PF\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$$\textit{Apply SIS filtering} \quad \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$$\textit{Calculate } N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^N (w_t^i)^2}$$

$$\textbf{if} \quad N_{eff} < N_{thr}$$

$$\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = RESAMPLE\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right]$$

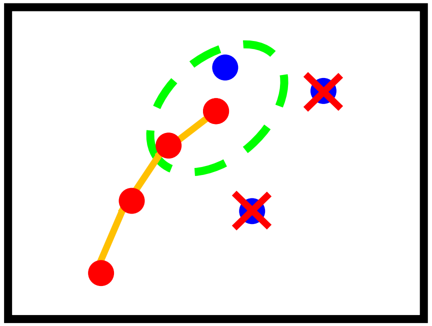
$$\textbf{end}$$

- **We can also apply resampling selectively**
  - **Only resample when it is needed, i.e., N_{eff} is too low.**

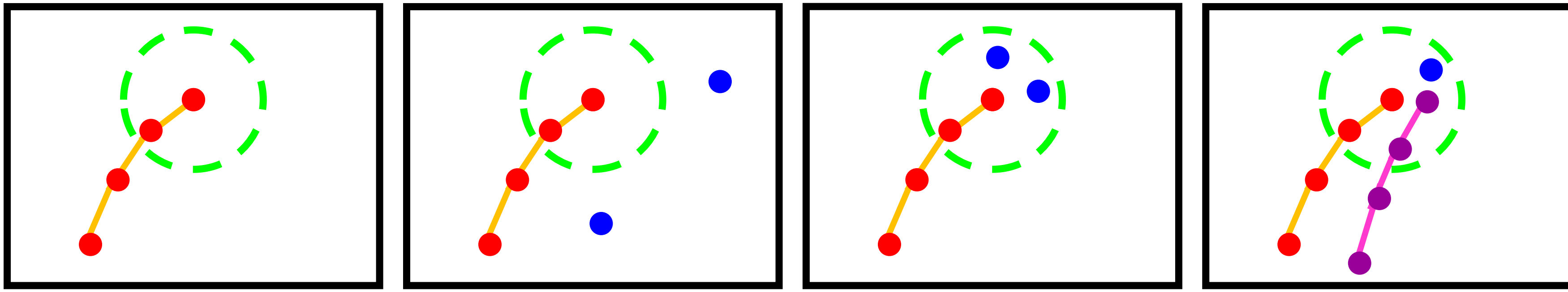
  $\Rightarrow$ **Avoids drift when there the tracked state is stationary.**

Outline of This Lecture

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
  - Data association
  - MHT
  - Network flow optimization
- Articulated Tracking
  - GP body pose estimation
  - Pictorial Structures



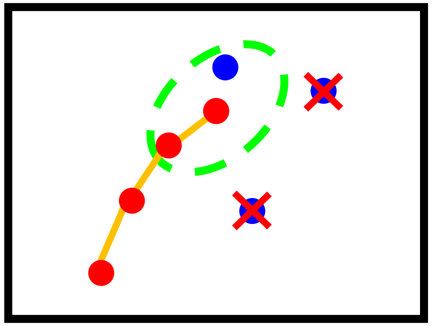
Recap: Motion Correspondence Ambiguities



1. **Predictions may not be supported by measurements**
   - Have the objects ceased to exist, or are they simply occluded?
2. **There may be unexpected measurements**
   - Newly visible objects, or just noise?
3. **More than one measurement may match a prediction**
   - Which measurement is the correct one (what about the others)?
4. **A measurement may match to multiple predictions**
   - Which object shall the measurement be assigned to?

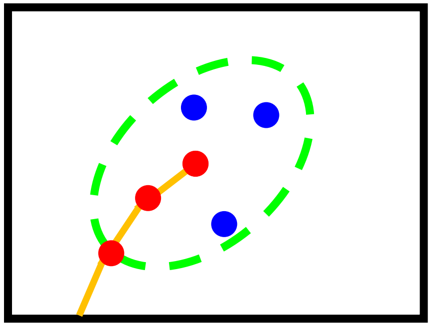
Recap: Reducing Ambiguities

- **Gating**
  - Only consider measurements within a certain area around the predicted location.
  - $\Rightarrow$ Large gain in efficiency, since only a small region needs to be searched



- **Nearest-Neighbor Filter**
  - Among the candidates in the gating region, only take the one closest to the prediction $\mathbf{x}_p$

$$z_l^{(k)} = \arg\min_j (\mathbf{x}_{p,l}^{(k)} - \mathbf{y}_j^{(k)})^T (\mathbf{x}_{p,l}^{(k)} - \mathbf{y}_j^{(k)})$$



  - Better: the one most likely under a Gaussian prediction model

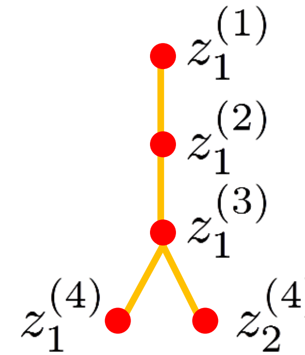
$$z_l^{(k)} = \arg\max_j \mathcal{N}(\mathbf{y}_j^{(k)}; \mathbf{x}_{p,l}^{(k)}, \mathbf{\Sigma}_{p,l}^{(k)})$$

which is equivalent to taking the Mahalanobis distance

$$z_l = \arg\min_j (\mathbf{x}_{p,l} - \mathbf{y}_j)^T \mathbf{\Sigma}_{p,l}^{-1} (\mathbf{x}_{p,l} - \mathbf{y}_j)$$

Recap: Track-Splitting Filter

- **Idea**
  - Instead of assigning the measurement that is currently closest, as in the NN algorithm, select the *sequence* of measurements that minimizes the *total* Mahalanobis distance over some interval!



  - Form a track tree for the different association decisions
  - Modified log-likelihood provides the merit of a particular node in the track tree.
  - Cost of calculating this is low, since most terms are needed anyway for the Kalman filter.
- **Problem**
  - The track tree grows exponentially, may generate a very large number of possible tracks that need to be maintained.

Recap: Pruning Strategies

- In order to keep this feasible, need to apply pruning
  - Deleting unlikely tracks
    - May be accomplished by comparing the modified log-likelihood $\lambda(k)$, which has a χ^2 distribution with kn_z degrees of freedom, with a threshold α (set according to χ^2 distribution tables).
    - Problem for long tracks: modified log-likelihood gets dominated by old terms and responds very slowly to new ones.

    $\Rightarrow$ Use sliding window or exponential decay term.
  - Merging track nodes
    - If the state estimates of two track nodes are similar, merge them.
    - E.g., if both tracks validate identical subsequent measurements.
  - Only keeping the most likely N tracks
    - Rank tracks based on their modified log-likelihood.

Outline of This Lecture

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
  - Data association
  - MHT
  - Network flow optimization
- Articulated Tracking
  - GP body pose estimation
  - Pictorial Structures

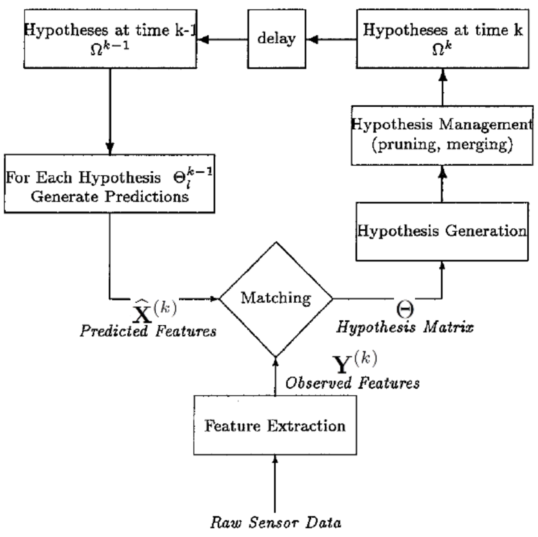
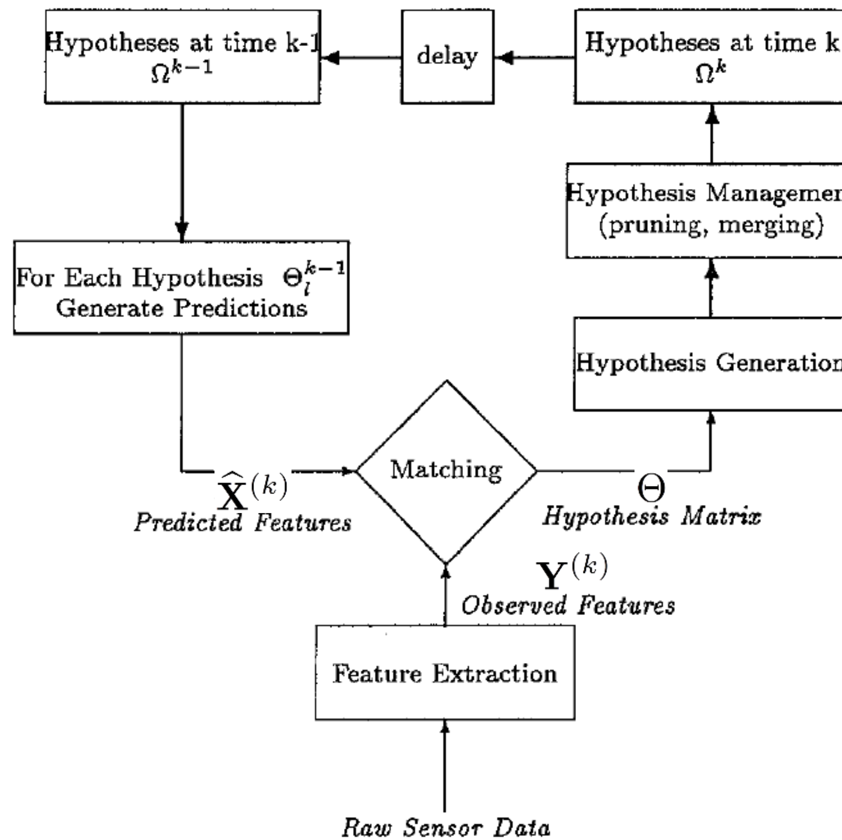


Image source: [Cox, IJCV'93]

Recap: Multi-Hypothesis Tracking (MHT)

- Ideas
  - Instead of forming a track tree, keep a set of hypotheses that generate child hypotheses based on the associations.
  - Enforce exclusion constraints between tracks and measure-ments in the assignment.
  - Integrate track generation into the assignment process.
  - After hypothesis generation, merge and prune the current hypothesis set.



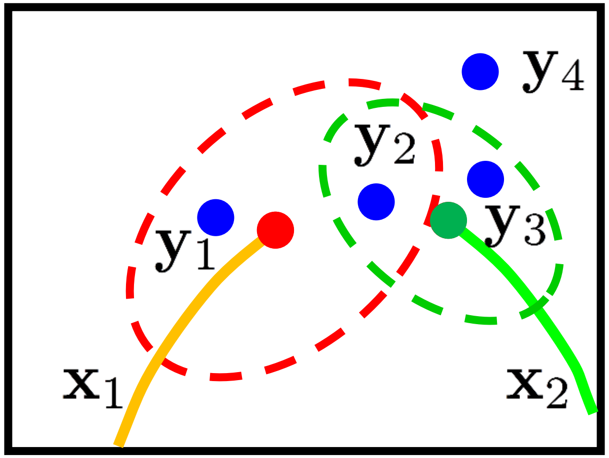
D. Reid, An Algorithm for Tracking Multiple Targets, IEEE Trans. Automatic Control, Vol. 24(6), pp. 843-854, 1979.

Image source: [Cox, IJCV'93]

Recap: Hypothesis Generation

- Create hypothesis matrix of the **feasible associations**

$$\Theta = \begin{array}{c} \mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \mathbf{x}_{fa} \;\; \mathbf{x}_{nt} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{array}{c} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \mathbf{y}_4 \end{array}$$



- Interpretation
  - Columns represent tracked objects, rows encode measurements
  - A non-zero element at matrix position (i,j) denotes that measurement $\mathbf{y}_i$ is contained in the validation region of track $\mathbf{x}_j$.
  - Extra column $\mathbf{x}_{fa}$ for association as *false alarm*.
  - Extra column $\mathbf{x}_{nt}$ for association as *new track*.
  - Turn this hypothesis matrix

Recap: Creating Assignments

Z_j	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_{fa}$	$\mathbf{x}_{nt}$
$\mathbf{y}_1$	0	0	1	0
$\mathbf{y}_2$	1	0	0	0
$\mathbf{y}_3$	0	1	0	0
$\mathbf{y}_4$	0	0	0	1

- **Impose constraints**
  - A measurement can originate from only one object.

  $\Rightarrow$ Any row has only a single non-zero value.

  - An object can have at most one associated measurement per time step.

  $\Rightarrow$ Any column has only a single non-zero value, except for $\mathbf{x}_{fa}$, $\mathbf{x}_{nt}$

Recap: Calculating Hypothesis Probabilities

- **Probabilistic formulation**
  - It is straightforward to enumerate all possible assignments.
  - However, we also need to calculate the probability of each child hypothesis.
  - This is done recursively:

$$
\begin{aligned}
p(\Omega_j^{(k)}|\mathbf{Y}^{(k)}) &= p(Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}|\mathbf{Y}^{(k)}) \\
&\overset{Bayes}{=} \eta p(\mathbf{Y}^{(k)}|Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) p(Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) \\
&= \eta p(\mathbf{Y}^{(k)}|Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) p(Z_j^{(k)}|\Omega_{p(j)}^{(k-1)}) p(\Omega_{p(j)}^{(k-1)})
\end{aligned}
$$

Normalization factor (η) | Measurement likelihood | Prob. of assignment set | Prob. of parent

Recap: Measurement Likelihood

- Use KF prediction
  - Assume that a measurement $\mathbf{y}_i^{(k)}$ associated to a track $\mathbf{x}_j$ has a Gaussian pdf centered around the measurement prediction $\hat{\mathbf{x}}_j^{(k)}$ with innovation covariance $\widehat{\boldsymbol{\Sigma}}_j^{(k)}$.
  - Further assume that the pdf of a measurement belonging to a new track or false alarm is uniform in the observation volume W (the sensor's field-of-view) with probability W^{-1}.
  - Thus, the measurement likelihood can be expressed as

$$p\left(\mathbf{Y}^{(k)} | Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}\right) = \prod_{i=1}^{M_k} \mathcal{N}\left(\mathbf{y}_i^{(k)}; \hat{\mathbf{x}}_j, \widehat{\boldsymbol{\Sigma}}_j^{(k)}\right)^{\delta_i} W^{-(1-\delta_i)}$$

$$= W^{-(N_{fal}+N_{new})} \prod_{i=1}^{M_k} \mathcal{N}\left(\mathbf{y}_i^{(k)}; \hat{\mathbf{x}}_j, \widehat{\boldsymbol{\Sigma}}_j^{(k)}\right)^{\delta_i}$$

Recap: Probability of an Assignment Set

$$p(Z_j^{(k)}|\Omega_{p(j)}^{(k-1)})$$

- Composed of three terms
  1. Probability of the number of tracks N_{det}, N_{fal}, N_{new}
     - Assumption 1: N_{det} follows a binomial distribution

$$p(N_{det}|\Omega_{p(j)}^{(k-1)}) = \binom{N}{N_{det}} p_{det}^{N_{det}}(1-p_{det})^{(N-N_{det})}$$

where N is the number of tracks in the parent hypothesis

     - Assumption 2: N_{fal} and N_{new} both follow a Poisson distribution with expected number of events $\lambda_{fal}W$ and $\lambda_{new}W$

$$p(N_{det}, N_{fal}, N_{new}|\Omega_{p(j)}^{(k-1)}) = \binom{N}{N_{det}} p_{det}^{N_{det}}(1-p_{det})^{(N-N_{det})} \cdot\mu(N_{fal};\lambda_{fal}W)\cdot\mu(N_{new};\lambda_{new}W)$$

Recap: Probability of an Assignment Set

2. Probability of a specific assignment of measurements
   - Such that $M_k = N_{det} + N_{fal} + N_{new}$ holds.
   - This is determined as 1 over the number of combinations

$$\binom{M_k}{N_{det}} \binom{M_k - N_{det}}{N_{fal}} \binom{M_k - N_{det} - N_{fal}}{N_{new}}$$

3. Probability of a specific assignment of tracks
   - Given that a track can be either *detected* or not *detected*.
   - This is determined as 1 over the number of assignments

$$\frac{N!}{(N - N_{det})!} \binom{N - N_{det}}{N_{det}}$$

$\Rightarrow$ When combining the different parts, many terms cancel out!

Outline of This Lecture

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
  - Data association
  - MHT
  - Network flow optimization
- Articulated Tracking
  - GP body pose estimation
  - Pictorial Structures

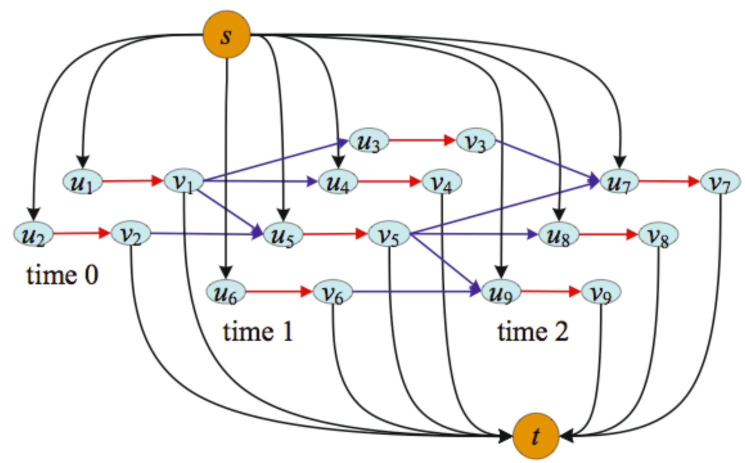
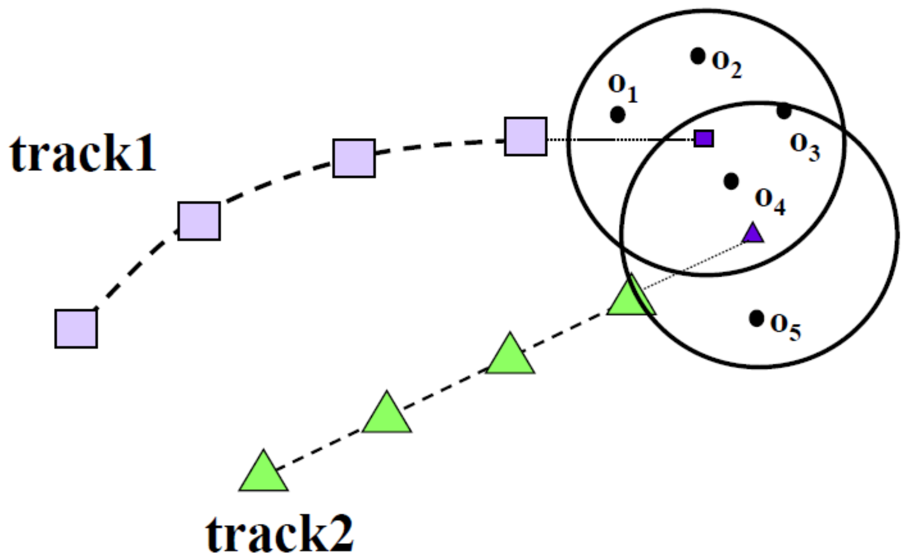


Image source: [Zhang, Li, Nevatia, CVPR'08]

Recap: Linear Assignment Formulation

- Form a matrix of pairwise similarity scores
- Example: Similarity based on motion prediction
  - Predict motion for each trajectory and assign scores for each measurement based on inverse (Mahalanobis) distance, such that closer measurements get higher scores.



	ai1	ai2
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

  - Choose at most one match in each row and column to maximize sum of scores

Recap: Linear Assignment Problem

- Formal definition
  - Maximize $\sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij} z_{ij}$

    subject to
    $$\left.\begin{aligned} &\textstyle\sum_{j=1} z_{ij} = 1;\ \ i = 1, 2, \ldots, N \\ &\textstyle\sum_{i=1} z_{ij} = 1;\ \ j = 1, 2, \ldots, M \\ &z_{ij} \in \{0, 1\} \end{aligned}\right\}$$

    Those constraints ensure that Z is a permutation matrix

  - The permutation matrix constraint ensures that we can only match up one object from each row and column.
  - Note: Alternatively, we can minimize cost rather than maximizing weights.

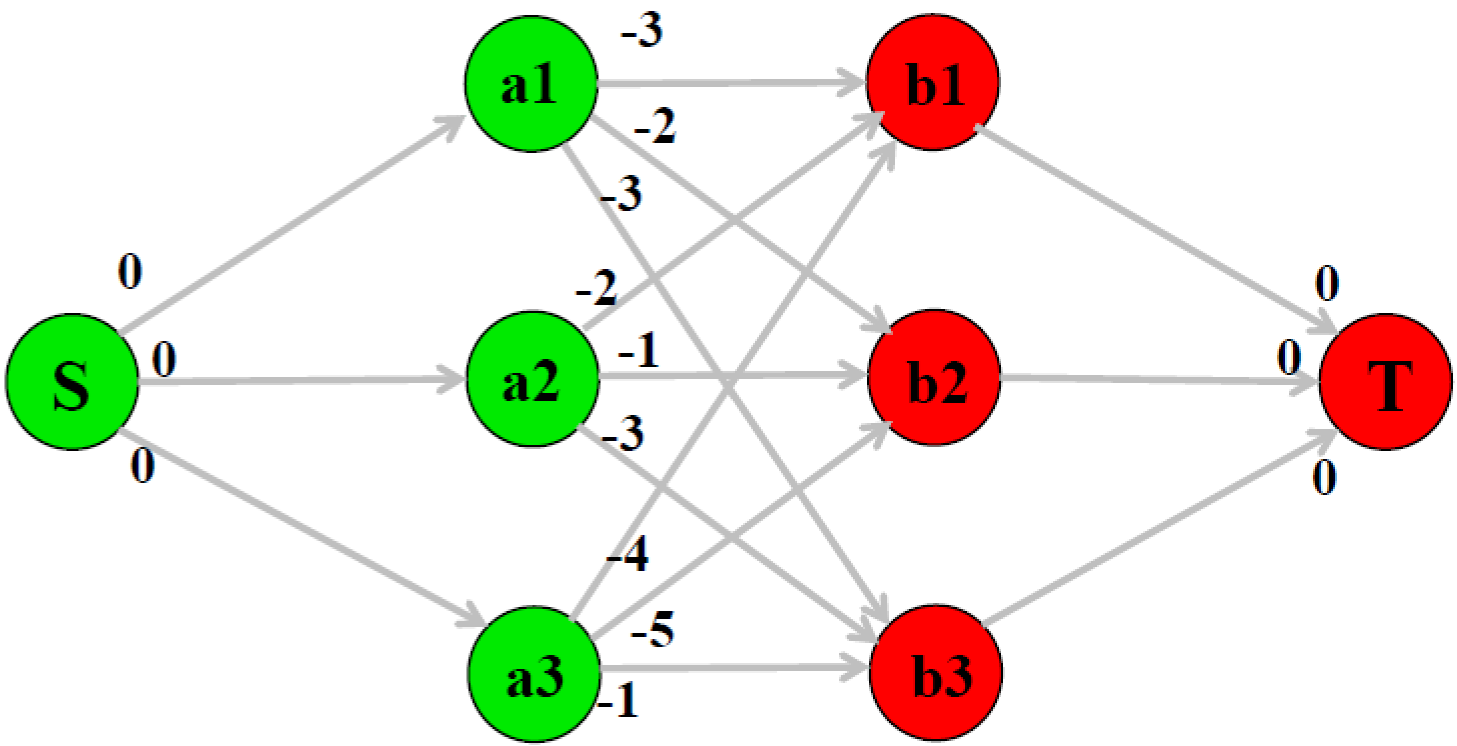
    $$\arg\min_{z_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij} z_{ij}$$

Recap: Optimal Solution

- **Greedy Algorithm**
  - Easy to program, quick to run, and yields "pretty good" solutions in practice.
  - But it often does not yield the optimal solution
- **Hungarian Algorithm**
  - There is an algorithm called Kuhn-Munkres or "Hungarian" algorithm specifically developed to efficiently solve the linear assignment problem.
  - Reduces assignment problem to bipartite graph matching.
  - When starting from an $N \times N$ matrix, it runs in $\mathcal{O}(N^3)$.

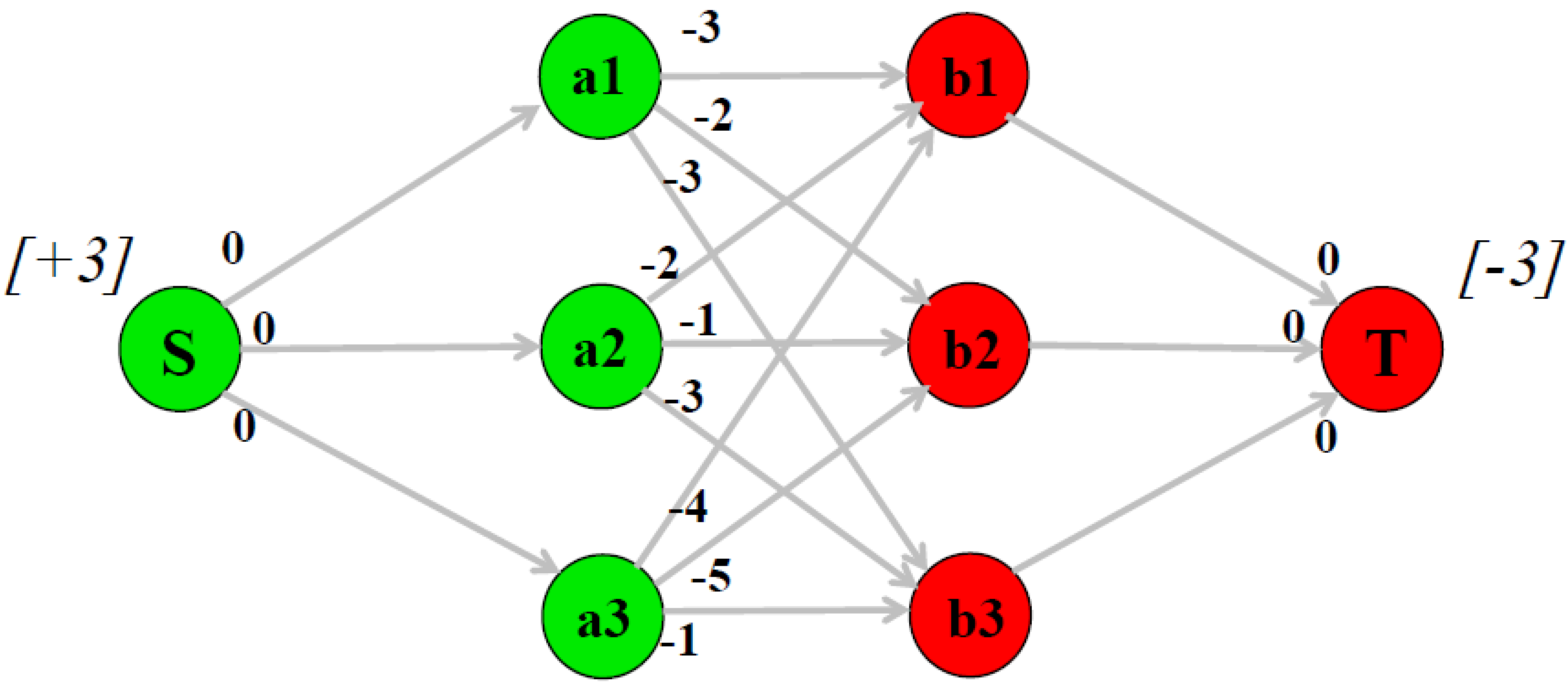
  $\Rightarrow$ If you need LAP, you should use it.

Recap: Min-Cost Flow



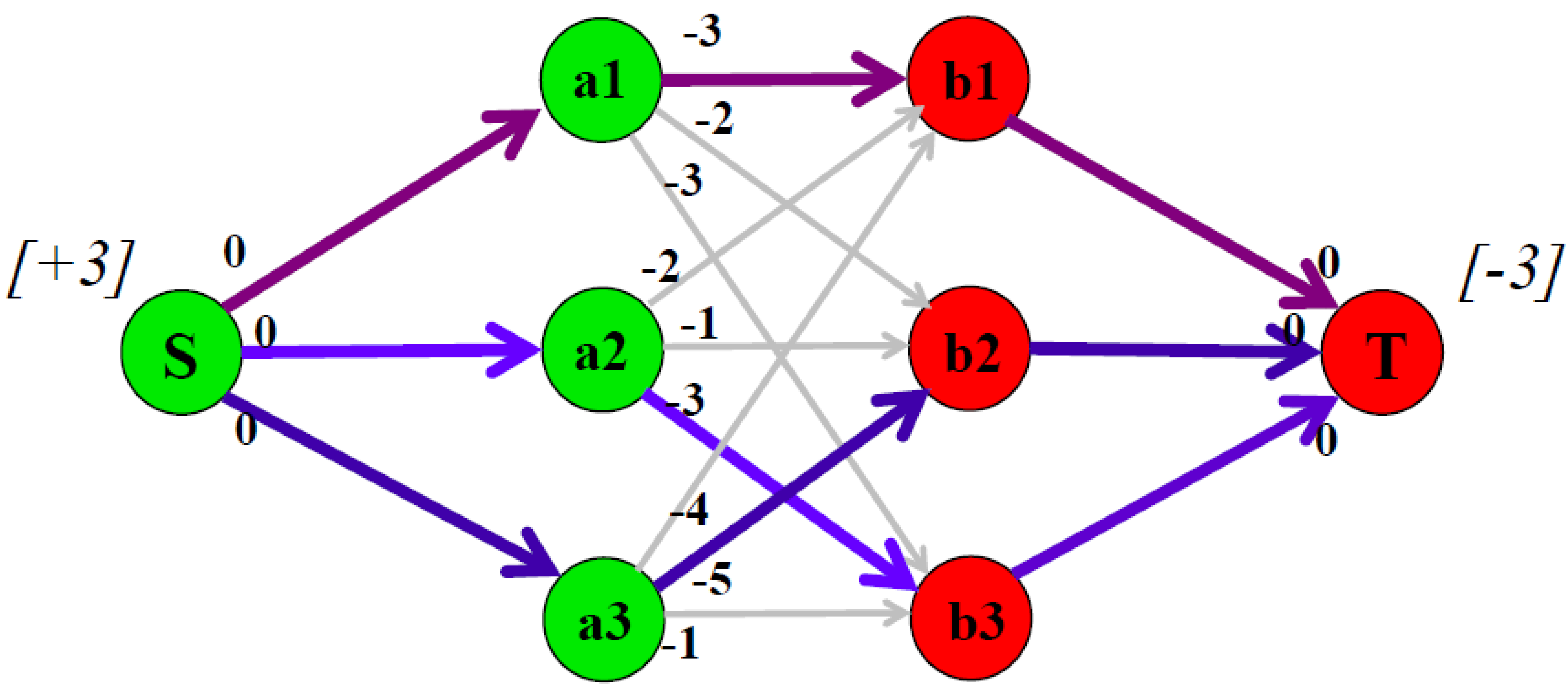
- **Conversion into flow graph**
  - Transform weights into costs $c_{ij} = \alpha - w_{ij}$
  - Add source/sink nodes with 0 cost.
  - Directed edges with a capacity of 1.

Recap: Min-Cost Flow



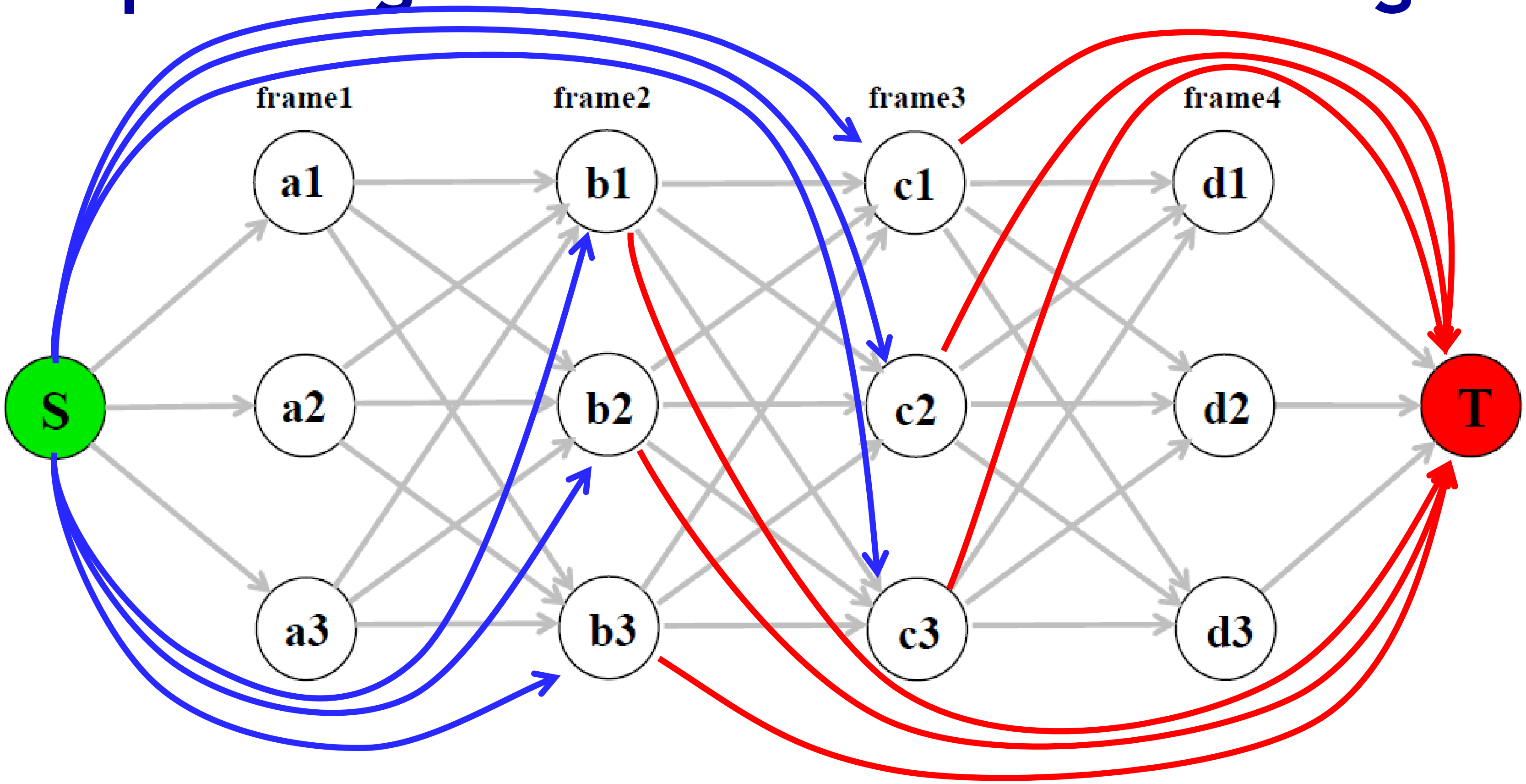
- Conversion into flow graph
  - Pump N units of flow from source to sink.
  - Internal nodes pass on flow ($\sum$ flow in $= \sum$ flow out).
  - $\Rightarrow$ Find the optimal paths along which to ship the flow.

Recap: Min-Cost Flow



- Conversion into flow graph
  - Pump N units of flow from source to sink.
  - Internal nodes pass on flow ($\sum$ flow in $= \sum$ flow out).
  - $\Rightarrow$ Find the optimal paths along which to ship the flow.

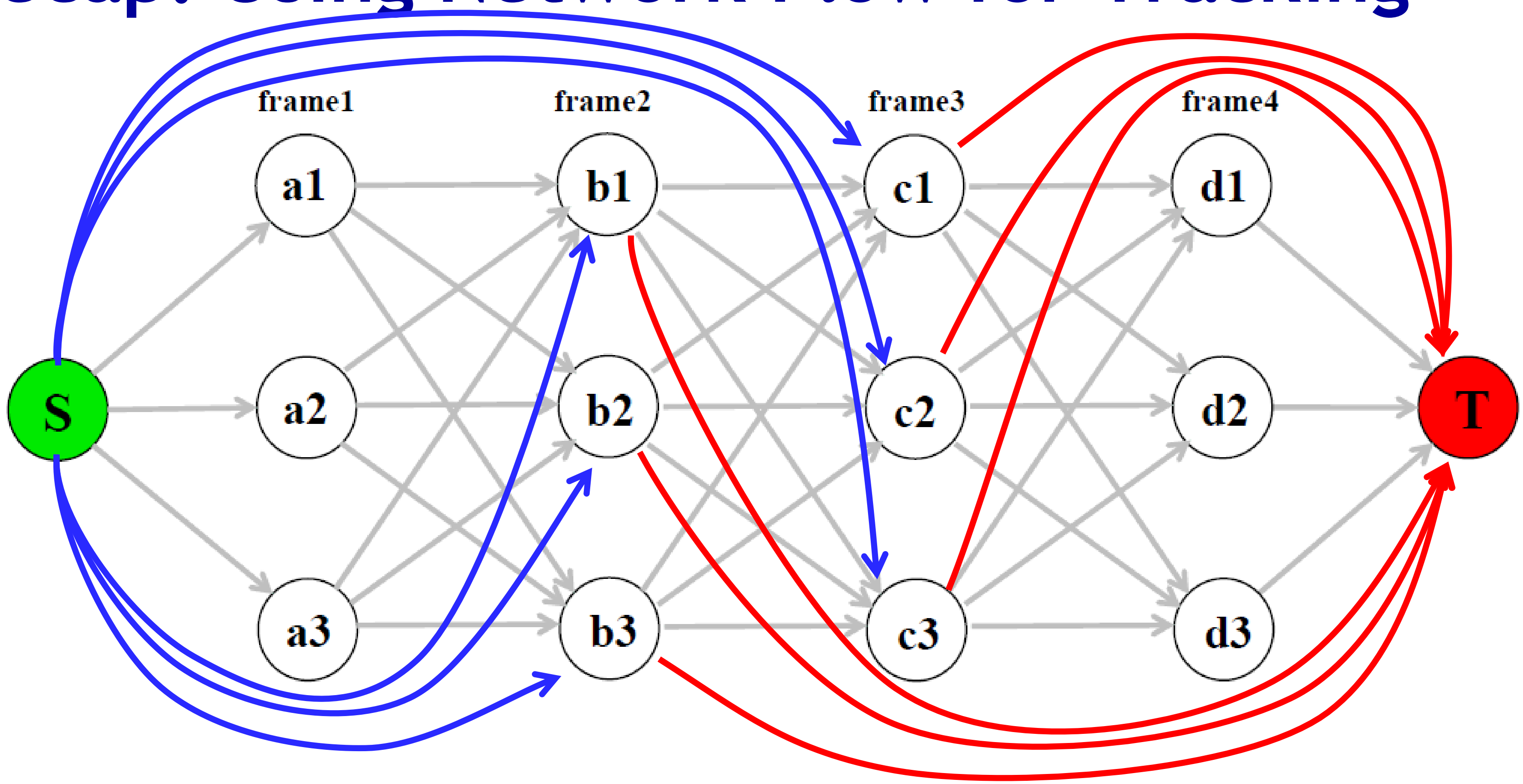
Recap: Using Network Flow for Tracking



- Complication 1
  - Tracks can start later than frame1 (and end earlier than frame4)

  ⇒ Connect the source and sink nodes to all intermediate nodes.

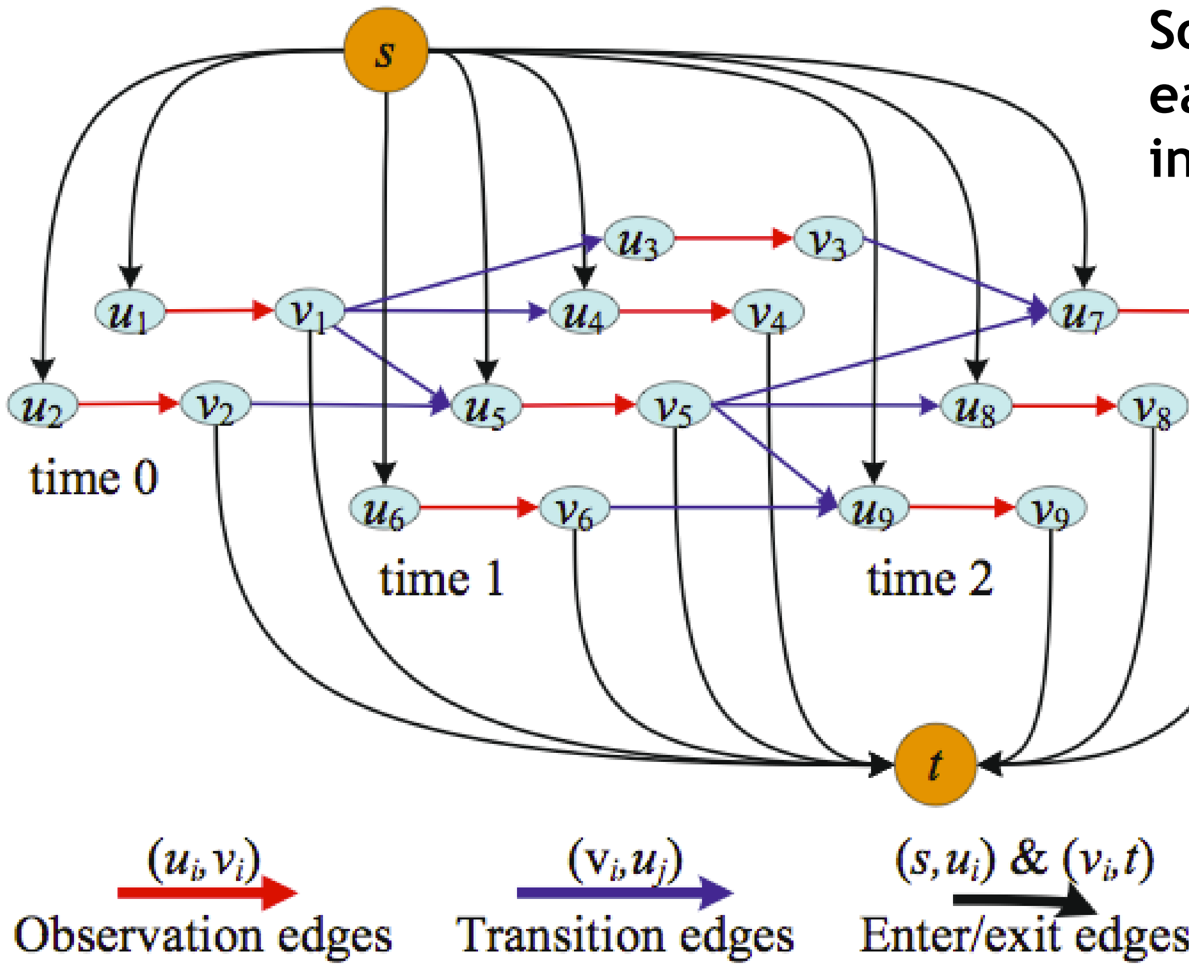
Recap: Using Network Flow for Tracking



- Complication 2
  - Trivial solution: zero cost flow!

Slide credit: Robert Collins

Recap: Network Flow Approach



Solution: Divide each detection into 2 nodes

Zhang, Li, Nevatia, Global Data Association for Multi-Object Tracking using Network Flows, CVPR'08.

image source: [Zhang, Li, Nevatia, CVPR'08]

Recap: Min-Cost Formulation

- **Objective Function**

$$\mathcal{T}* = \operatorname*{argmin}_{\mathcal{T}} \sum_i C_{in,i} f_{in,i} + \sum_i C_{i,out} f_{i,out}$$

$$+ \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_i f_i$$

- **subject to**
  - Flow conservation at all nodes

$$f_{in,i} + \sum_j f_{j,i} = f_i = f_{out,i} + \sum_j f_{i,j} \ \forall i$$

  - Edge capacities

$$f_i \leq 1$$

Outline of This Lecture

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
  - Data association
  - MHT
  - Network flow optimization
- Articulated Tracking
  - GP body pose estimation
  - Pictorial Structures

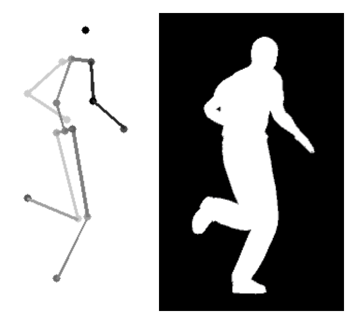
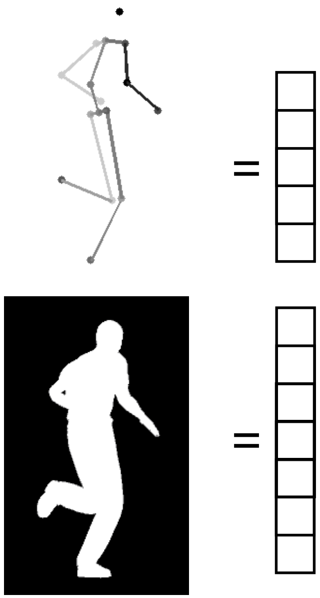


Image sources: Tomasz Svoboda, Deva Ramanan

Recap: Basic Pose Estimation Approaches

- Global methods
  - Entire body configuration is treated as a point in some high-dimensional space.
  - Observations are also global feature vectors.
  - ⇒ View of pose estimation as a high-dimensional regression problem.
  - ⇒ Often in a subspace of "typical" motions...



- Part-based methods
  - Body configuration is modeled as an assembly of movable parts with kinematic constraints.
  - Local search for part configurations that provide a good explanation for the observed appearance under the kinematic constraints.
  - ⇒ View of pose estimation as probabilistic inference in a dynamic Graphical Model.

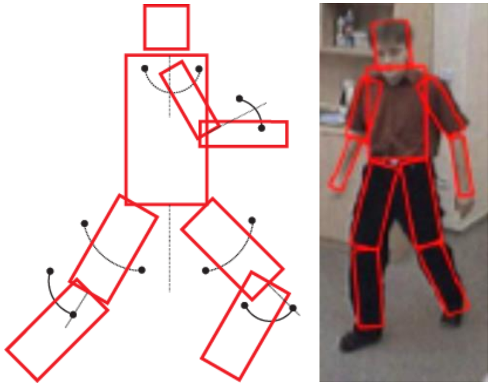
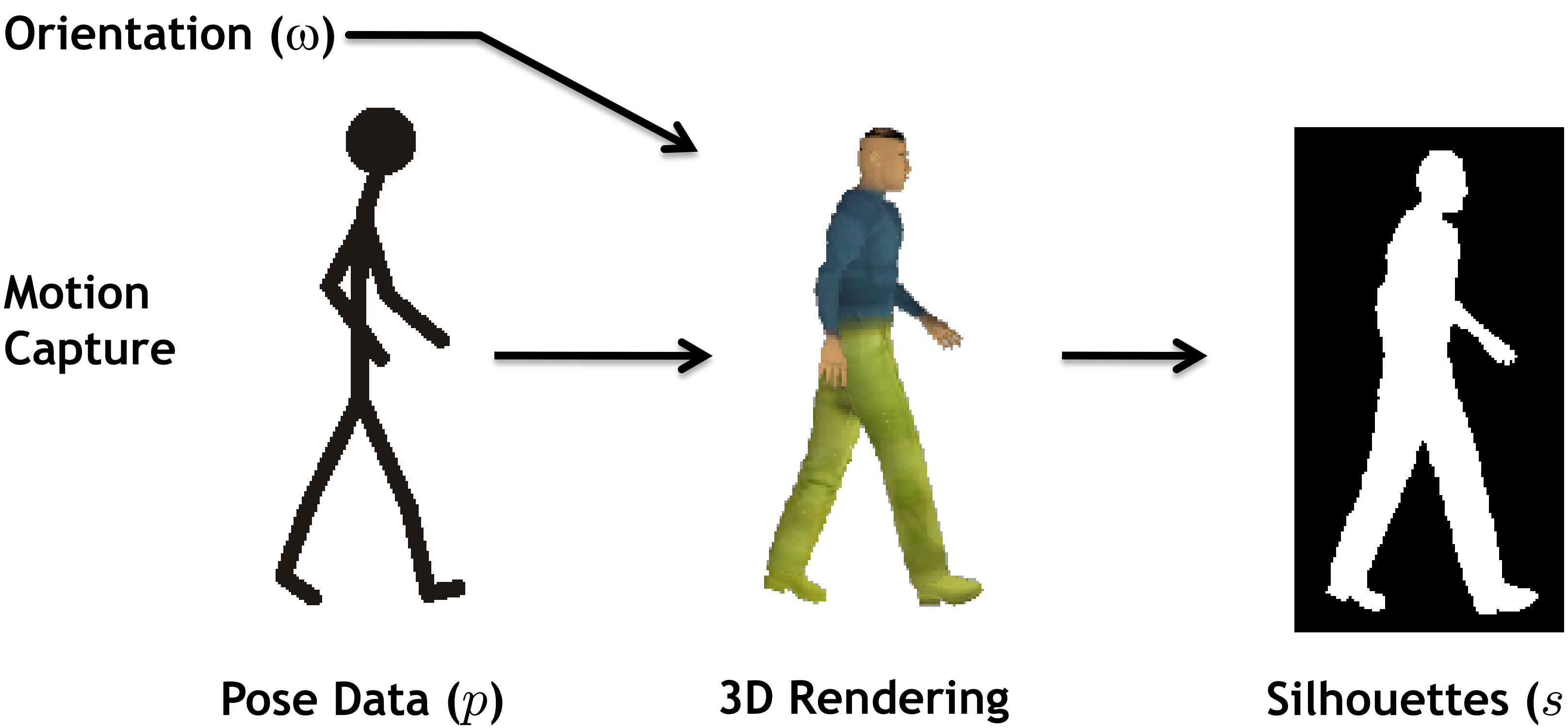


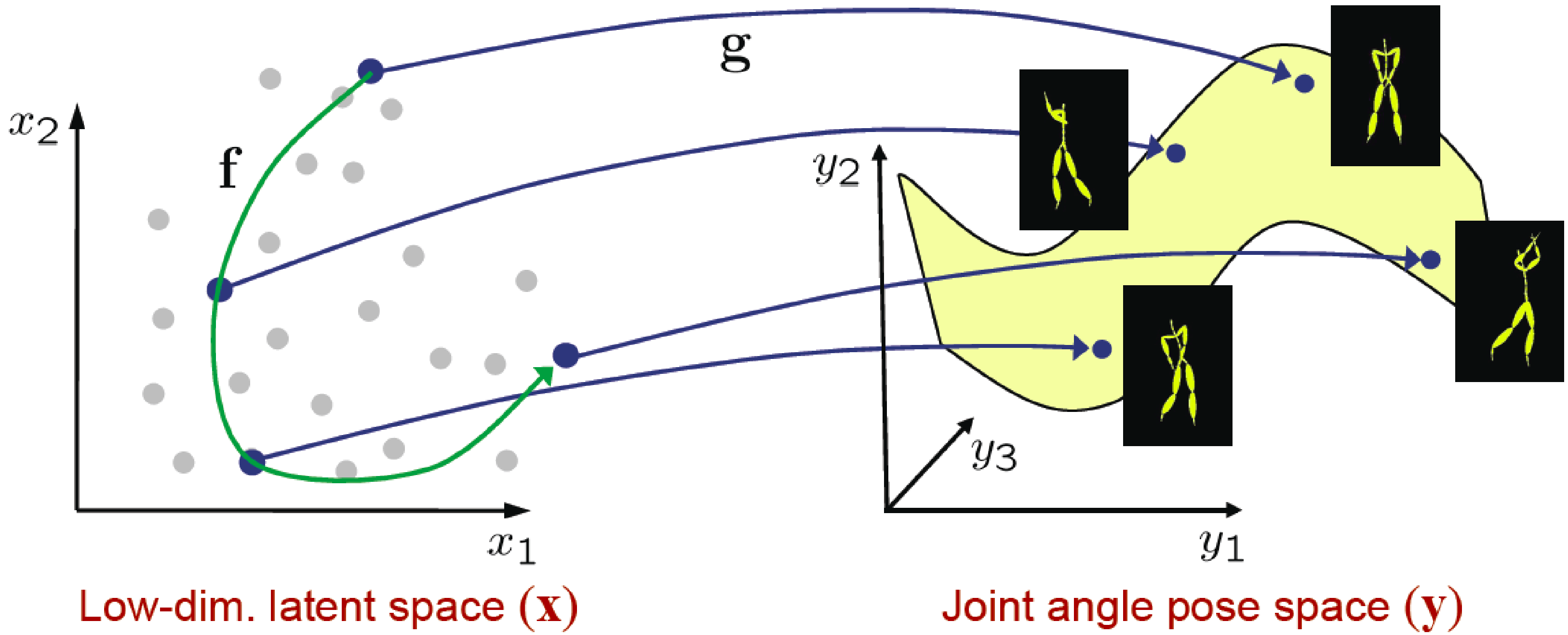
image sources: T. Jaeggli, D. Ramanan, T. Svoboda

Recap: Advantage of Silhouette Data

- Synthetic training data generation possible!
  - Create sequences of „Pose + Silhouette“ pairs
  - Poses recorded with Mocap, used to animate 3D model
  - Silhouette via 3D rendering pipeline



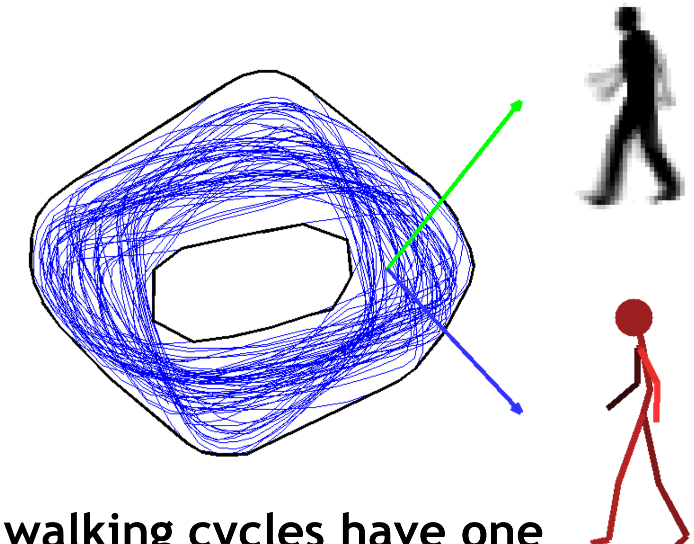
Recap: Latent Variable Models



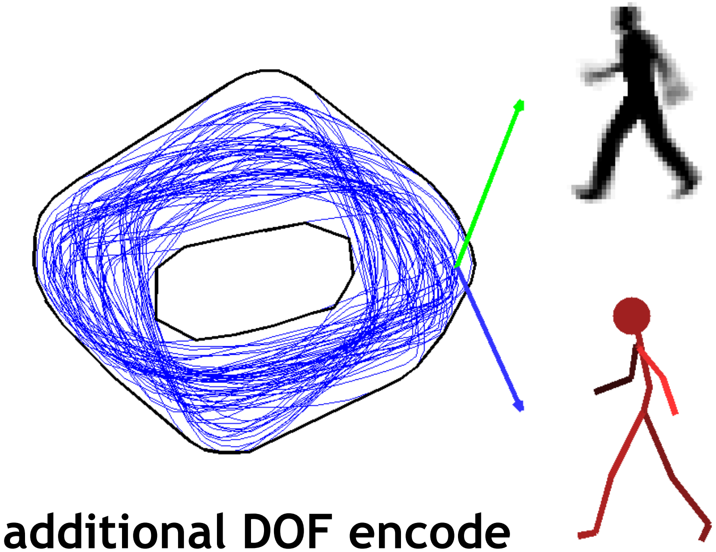
- **Joint angle pose space is huge!**
  - **Only a small portion contains valid body poses.**
  - **⇒ Restrict estimation to the subspace of valid poses for the task**
  - **Latent variable models: PCA, FA, GPLVM, etc.**

image source: R. Urtasun

Recap: Articulated Motion in Latent Space



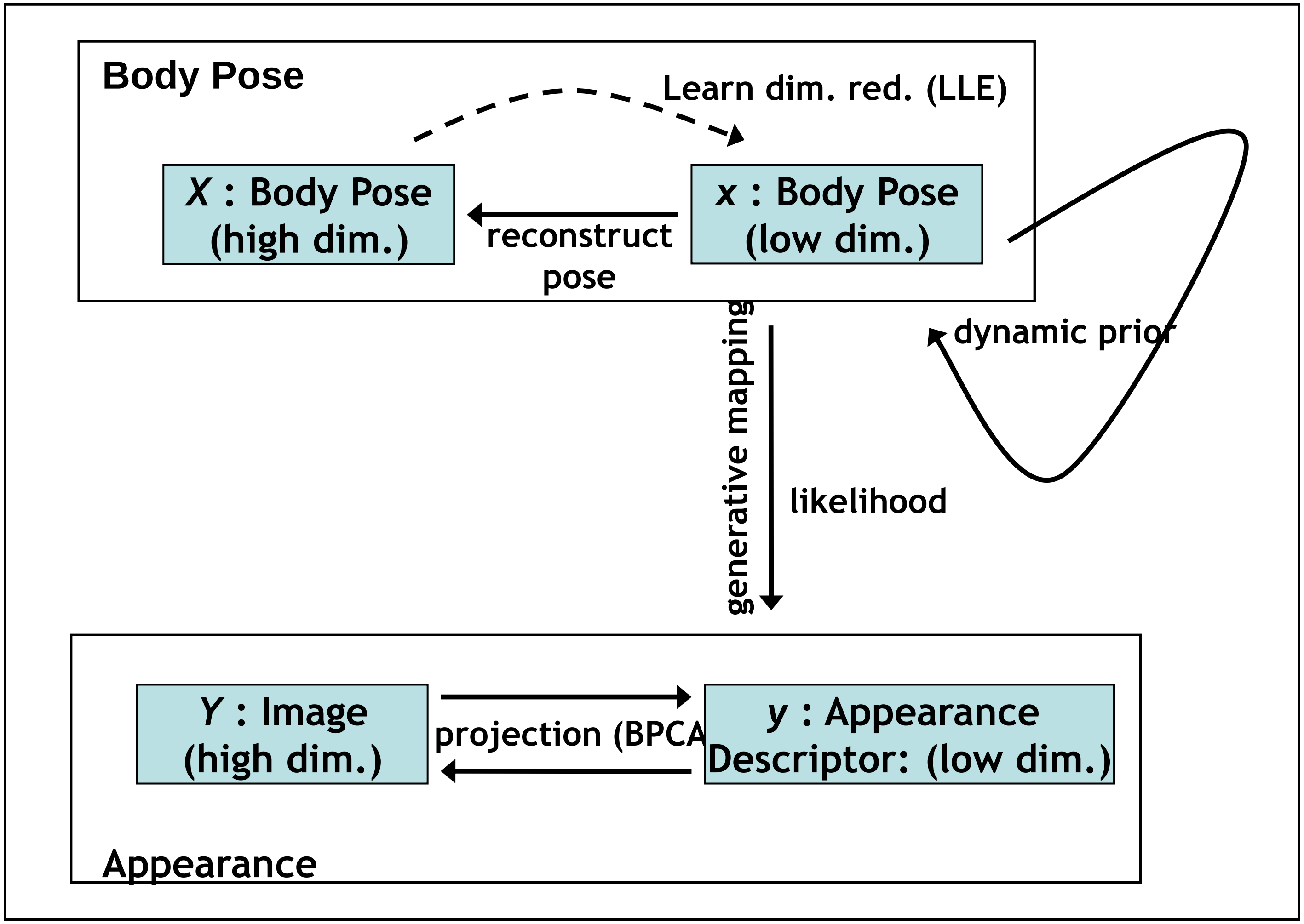
walking cycles have one main (periodic) DOF



additional DOF encode „walking style“

- Regression from latent space to
  - Pose → $p(pose \mid \mathbf{z})$
  - Silhouette → $p(silhouette \mid \mathbf{z})$
- Regressors need to be learned from training data.

Recap: Learning a Generative Mapping

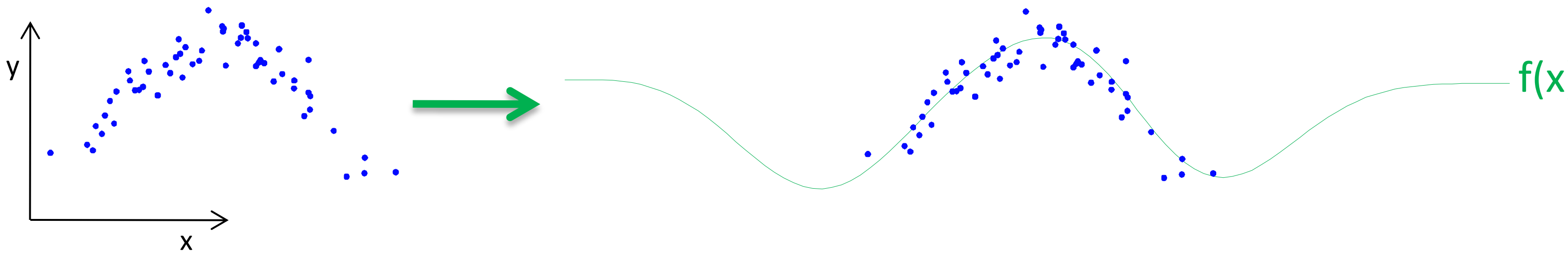


T. Jaeggli, E. Koller-Meier, L. Van Gool, "Learning Generative Models for Monocular Body Pose Estimation", ACCV 2007.

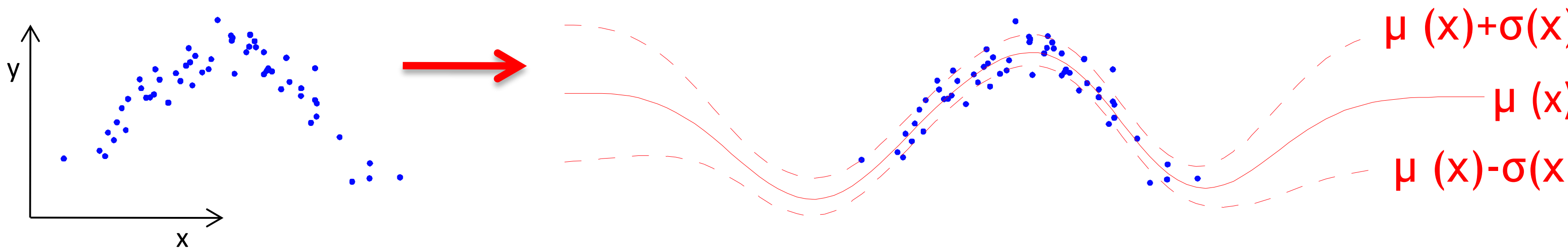
Slide credit: Tobias Jaeggli

Recap: Gaussian Process Regression

- "Regular" regression: $y = f(\mathbf{x})$



- GP regression: $p(y|\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$



Slide credit: Stefan Gammeter

Recap: GP Prediction w/ Noisy Observations

- **Calculation of posterior:**
  - Corresponds to conditioning the joint Gaussian prior distribution on the observations:

$$\mathbf{f}_\star | X_\star, X, \mathbf{t} \sim \mathcal{N}(\bar{\mathbf{f}}_\star, \text{cov}[\mathbf{f}_\star]) \qquad \bar{\mathbf{f}}_\star = \mathbb{E}[\mathbf{f}_\star | X, X_\star, \mathbf{t}]$$

  - with:

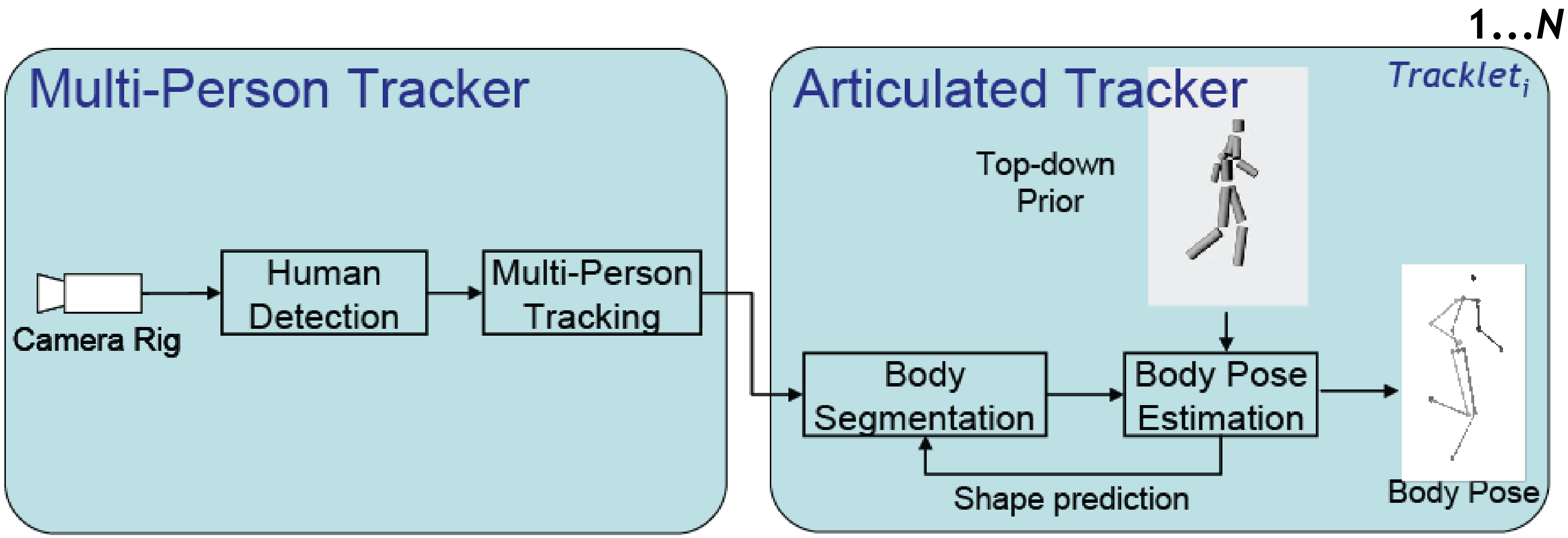
$$\bar{\mathbf{f}}_\star = K(X_\star, X)\left(K(X,X) + \sigma_n^2 I\right)^{-1} \mathbf{t}$$

$$\text{cov}[\mathbf{f}_\star] = K(X_\star, X_\star) - K(X_\star, X)\left(K(X,X) + \sigma_n^2 I\right)^{-1} K(X, X_\star)$$

⇒ This is the key result that defines Gaussian process regression!

- The predictive distribution is a Gaussian whose mean and variance depend on the test points X_* and on the kernel $k(\mathbf{x},\mathbf{x}')$, evaluated on the training data X.

Recap: Articulated Multi-Person Tracking



- **Idea: Only perform articulated tracking where it's easy!**
- **Multi-person tracking**
  - Solves hard data association problem
- **Articulated tracking**
  - Only on individual "tracklets" between occlusions
  - GP regression on full-body pose

[Gammeter, Ess, Jaeggli, Schindler, Leibe, Van Gool, ECCV'08]

Outline of This Lecture

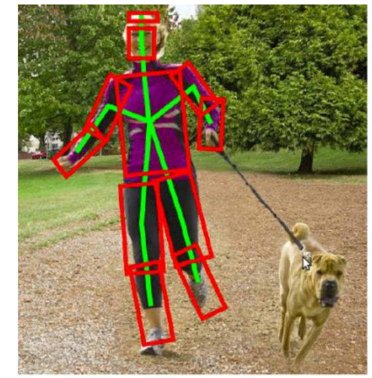
- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
  - Data association
  - MHT
  - Network flow optimization
- Articulated Tracking
  - GP body pose estimation
  - Pictorial Structures

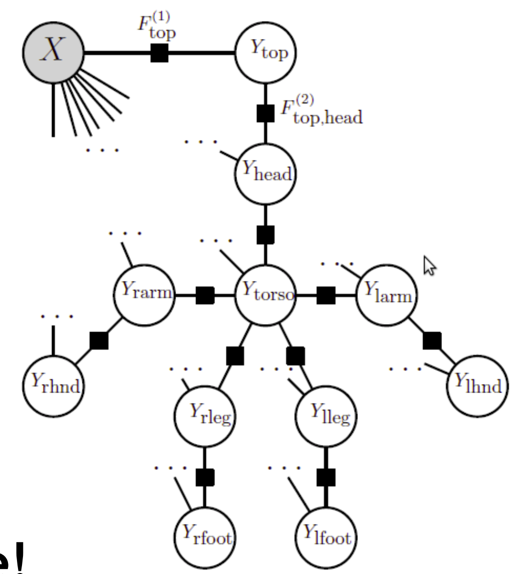
Image sources: Tomasz Svoboda, Deva Ramanan

Recap: Pictorial Structures

- Each body part one variable node
  - Torso, head, etc. (11 total)
- Each variable represented as tupel
  - E.g., $y_{torso} = (x, y, \theta, s)$ with
  - (x, y) image coordinates
  - θ rotation of the part
  - s scale
- Discretize label space y into L states
  - E.g., size of L for $y = (x, y, \theta, s)$
  - $L = 125 \times 125 \times 8 \times 4 \approx 500'000$

$\Rightarrow$ **Efficient search needed to make this feasible!**





P. Felzenszwalb, D. Huttenlocher, Pictorial Structures for Object Recognition, IJCV, Vol. 61(1), 2005.

Recap: Model Components

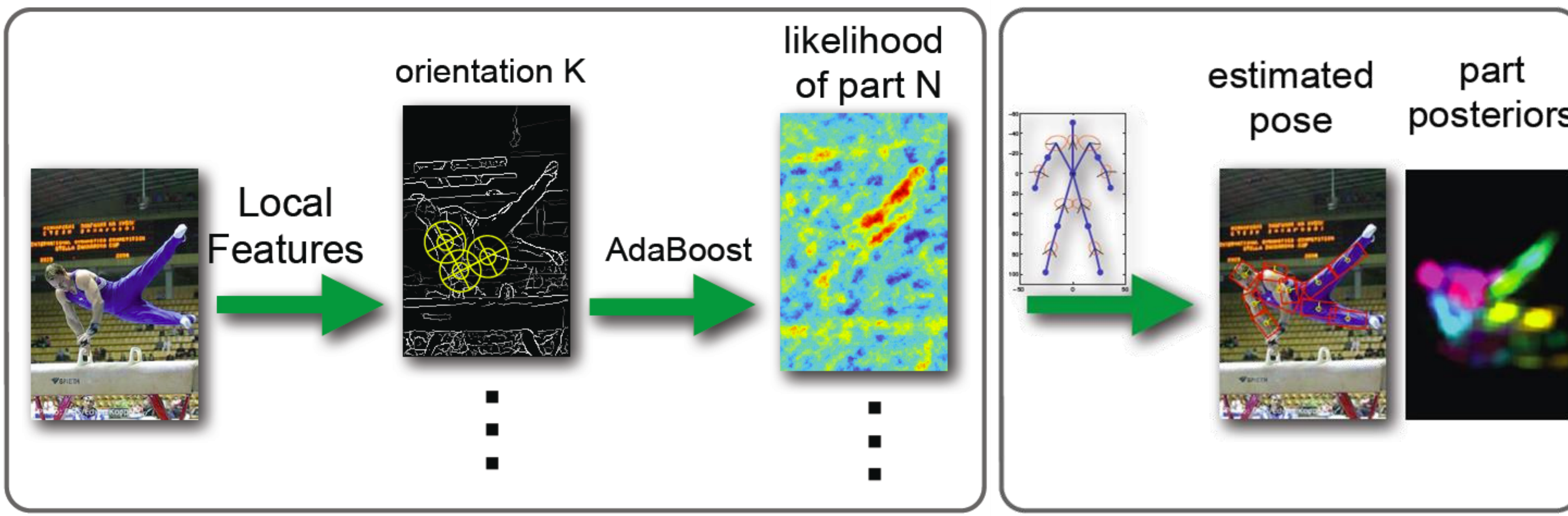
- Body is represented as flexible combination of parts

posterior over body poses

$$p(L|E) \propto p(E|L)p(L)$$

likelihood of observations

prior on body poses

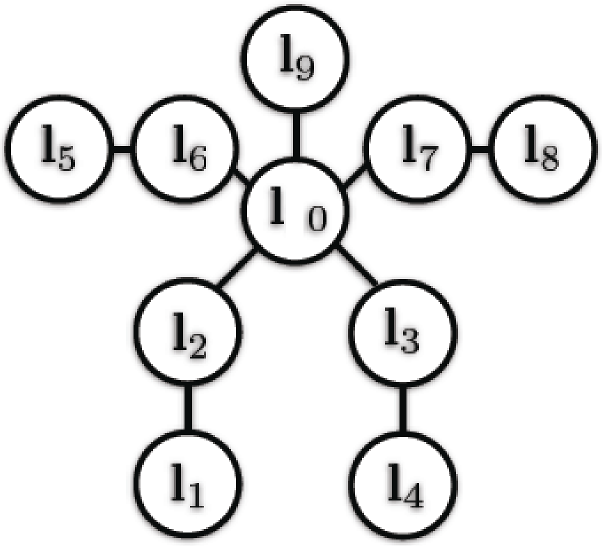


Recap: Kinematic Tree Prior

- **Notation**
  - (from [Andriluka et al., IJCV'12])
  - Body configuration

$$L = \{l_0, l_1, \ldots, l_N\}$$

  - Each body part: $l_i = (x_i, y_i, \theta_i, s_i)$

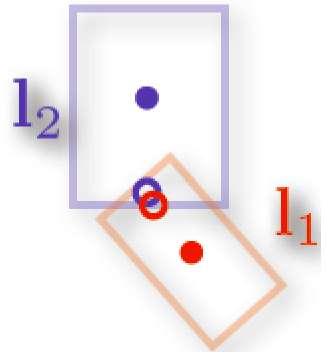


- **Prior**

$$p(L) = p(l_0) \prod_{(i,j)\in G} p(l_i|l_j)$$

  - with $p(l_0)$ assumed uniform
  - with $\mathrm{p}(l_i \mid l_j)$ modeled using a Gaussian in the transformed joint space

$$p(l_i|l_j) = \mathcal{N}\left(T_{ji}(l_i) - T_{ij}(l_j)|\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij}\right)$$

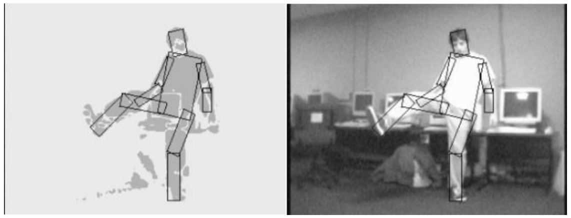


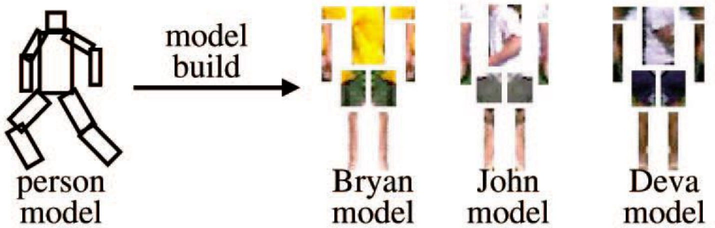
Recap: Likelihood Model

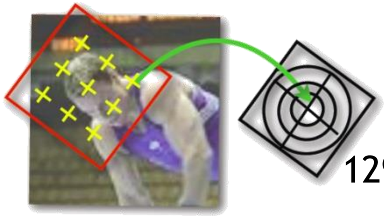
- **Assumption**
  - Evidence (image features) for each part independent of all other parts

$$p(E|L) = \prod_{i=0}^{N} p(E|l_i)$$

- **Many variants proposed in the past**
  - Based on rectangular fg regions
  - Based on color/edge models
  - Based on AdaBoost classifiers
  - ...

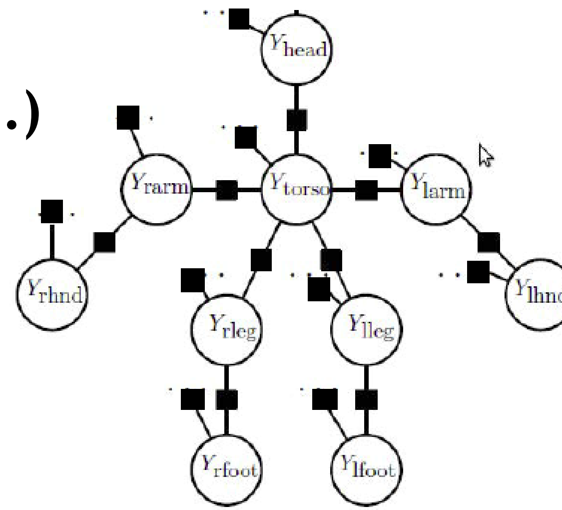






Pictorial Structures

- Potentials (= energies = factors)
  - Unaries for each body part (torso, head, ...)
  - Pairwise between connected body parts



- Body pose estimation
  - Find most likely part location
  - ⇒ Sum-product algorithm (marginals)
  - Find the best overall configuration
  - ⇒ Max-sum algorithm (MAP estimate)

- Complexity
  - Let k be the number of body parts (e.g., $k = 10$)
  - L is the size of the label space (e.g., several 100k)
  - Max-sum algorithm in general: $\mathcal{O}(k\ L^2)$
  - For specific pairwise potentials: $\mathcal{O}(k\ L)$

Recap: Efficient Inference

- **Assume d to have quadratic form**

$$d(l_1, l_0) = ||l_1 - T_1(l_0)||^2$$

- **Then** $$\min_{l_0, l_1} (m_0(l_0) + m_1(l_1) + d(l_1, l_0))$$

$$= \min_{l_0} \left(m_0(l_0) + \min_{l_1} (m_1(l_1) + d(l_1, l_0)) \right)$$

  - **with the second term a generalized distance transform (gDT).**
  - **Algorithms exist to compute gDT efficiently.**

  - **Thus** $= \min_{l_0} (m_0(l_0) + DT_{m_1}(T_1(l_0)))$

    with $DT_{m_1}(T_1(l_0)) = \min_{l_1} \{m_1(l_1) + d(l_1, l_0)\}$

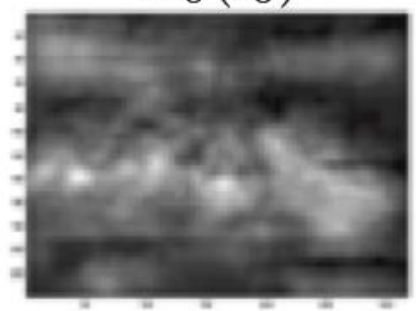
$\Rightarrow$ **Finding the best part configuration can be done sequentially, rather than simultaneously!**

Recap: Example Part Model of Motorbikes

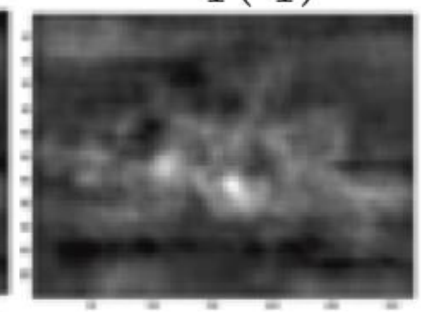
- **Model**
  - 2 parts (use both wheels), simple translation between them given by (x,y) position

1. **Part unaries (log prob)**
   – $m_0(l_0)$ and $m_1(l_1)$

$m_0(l_0)$

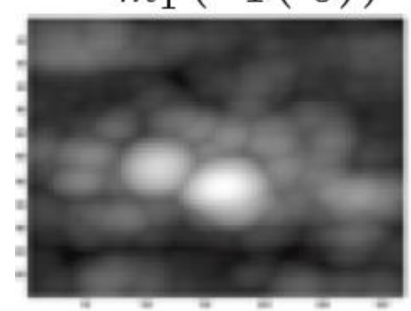


$m_1(l_1)$



2. **Distance transform of $m_1(l_1)$**

$DT_{m_1}(T_1(l_0))$



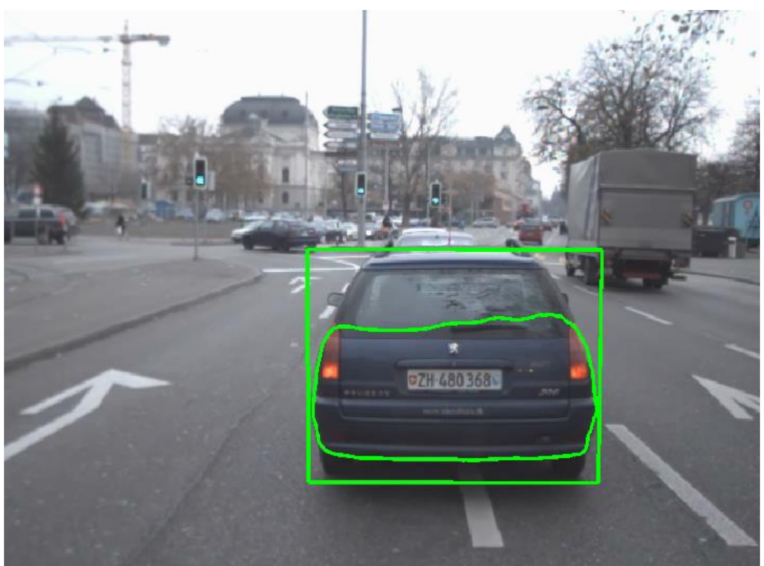
3. **Simply find minimum of sum**

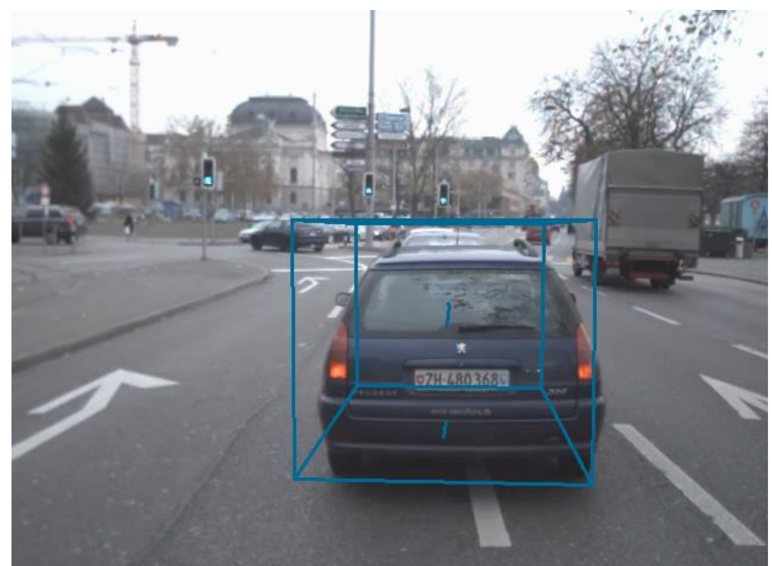
$$\min_{l_0} \left(m_0(l_0) + DT_{m_1}(T_1(l_0)) \right)$$

Any Questions?

So what can you do with all of this?

Robust Object Detection & Tracking



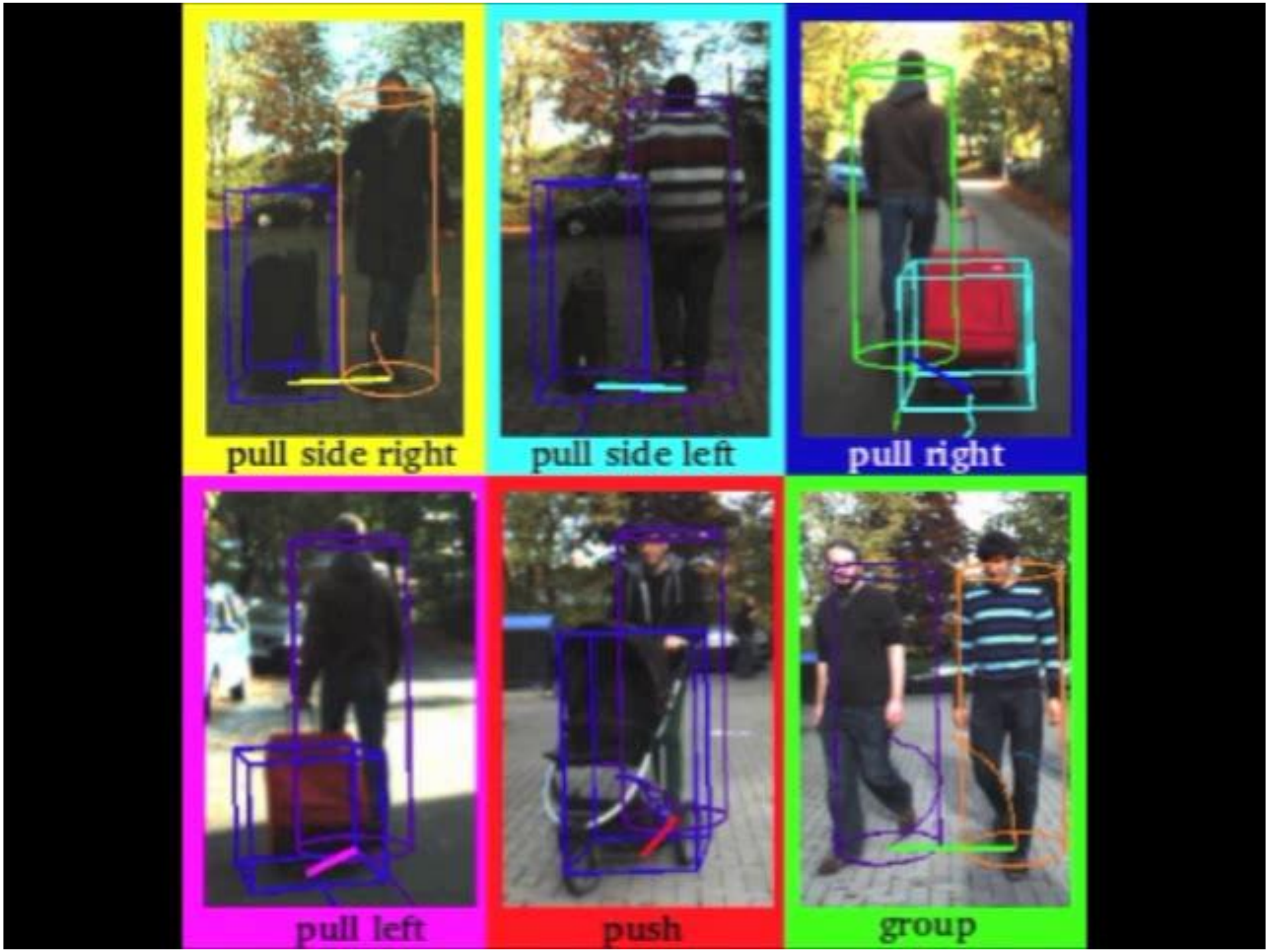


Mobile Tracking in Densely Populated Settings

(Tracking based on stereo depth only, no detector verification)

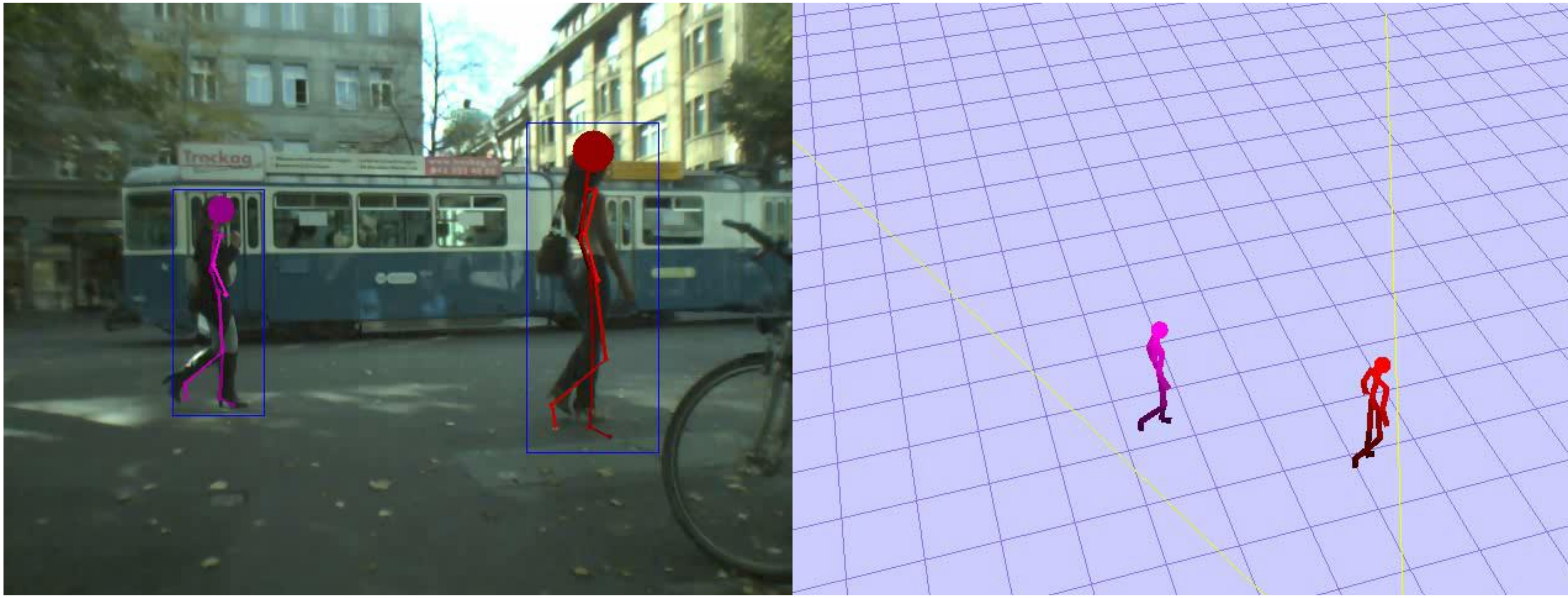
[D. Mitzel, B. Leibe, ECCV'12]

Classifying Interactions with Objects



[T. Baumgartner, D. Mitzel, B. Leibe, CVPR'13]

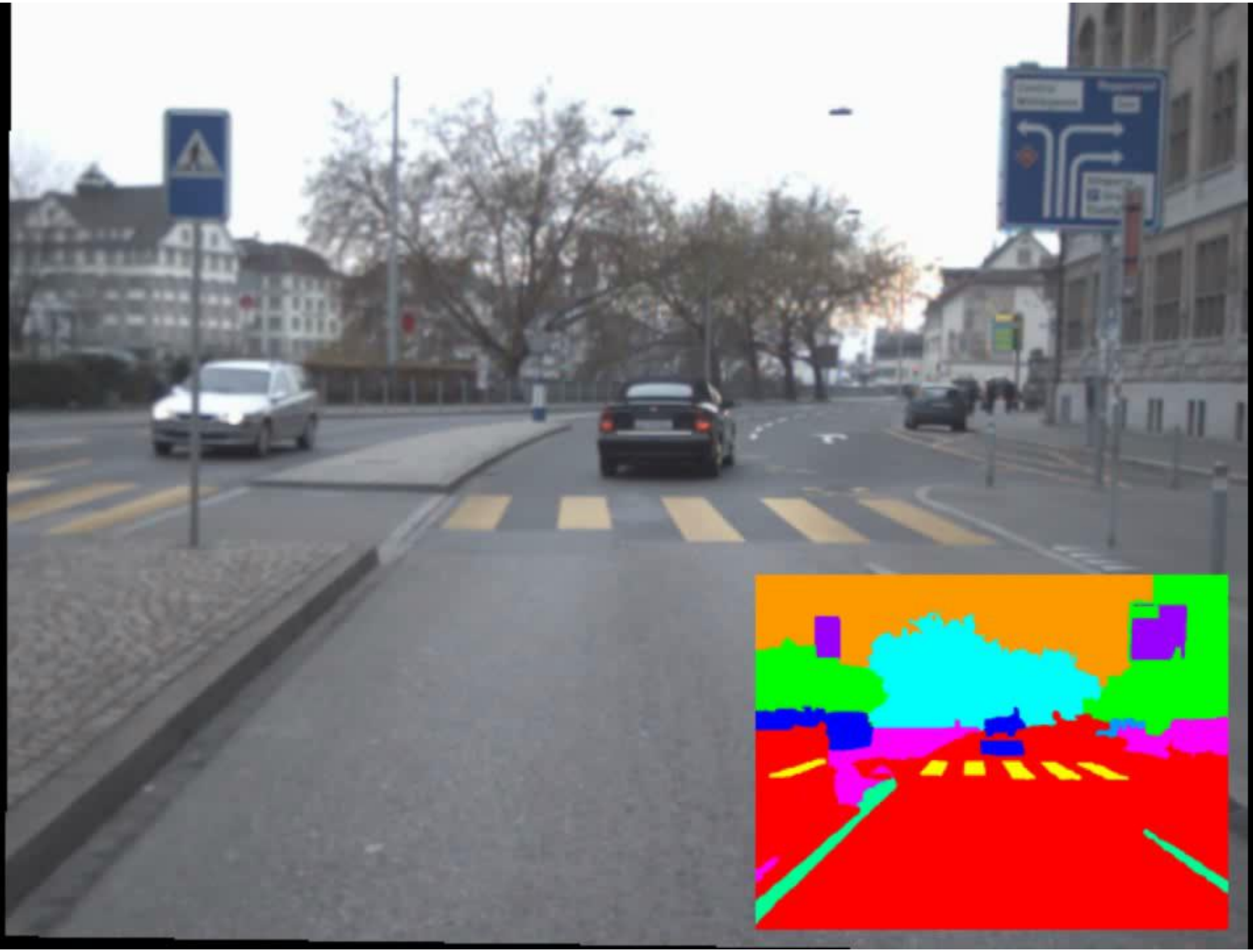
Articulated Multi-Person Tracking



- **Multi-Person tracking**
  - Recover trajectories and solve data association
- **Articulated Tracking**
  - Estimate detailed body pose for each tracked person

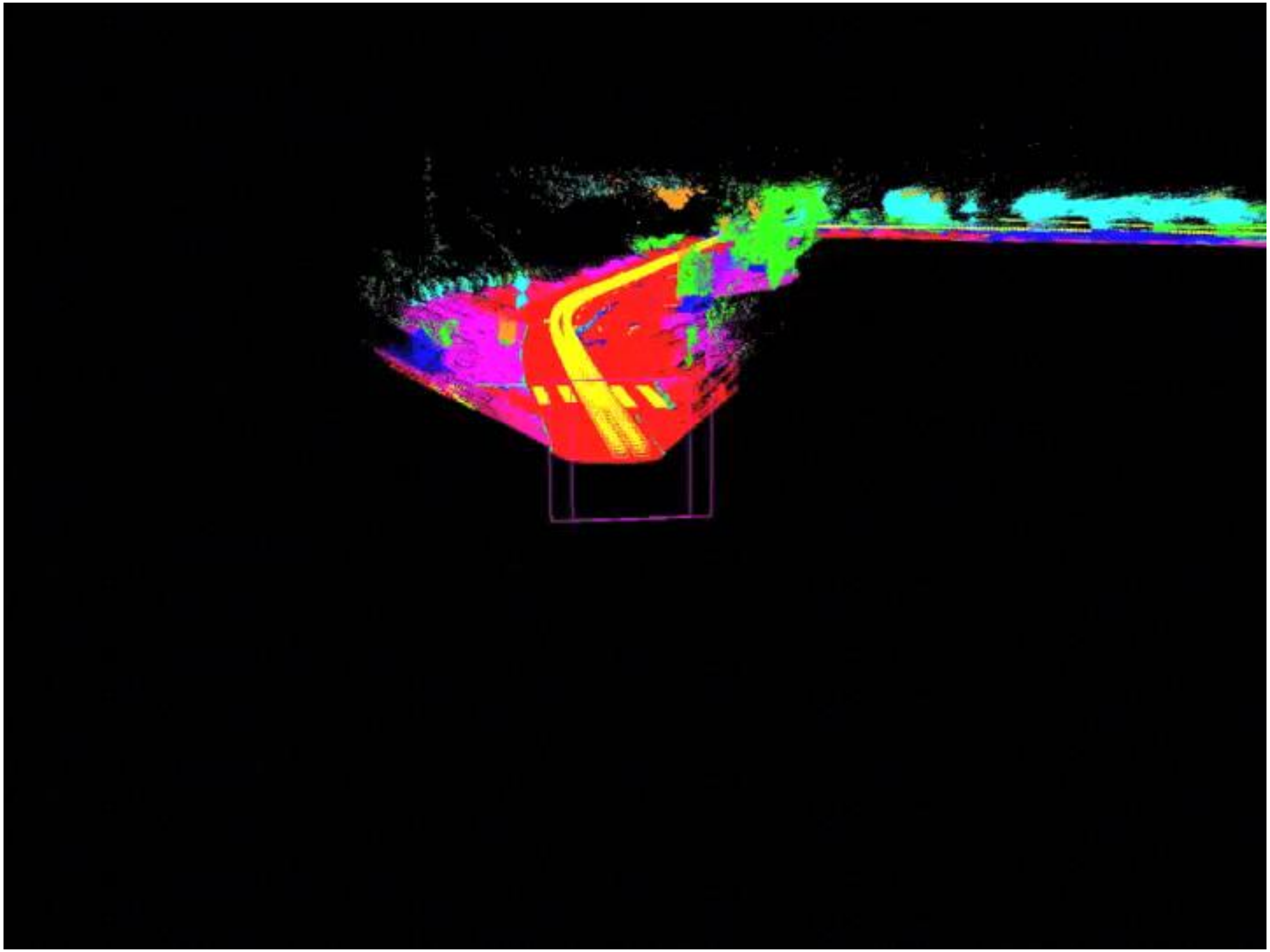
[Gammeter, Ess, Jaeggli, Schindler, Leibe, Van Gool, ECCV'08]

Semantic 2D-3D Scene Segmentation



[G. Floros, B. Leibe, CVPR'12]

Integrated 3D Point Cloud Labels



[G. Floros, B. Leibe, CVPR'12]

Any More Questions?

Good luck for the exam!