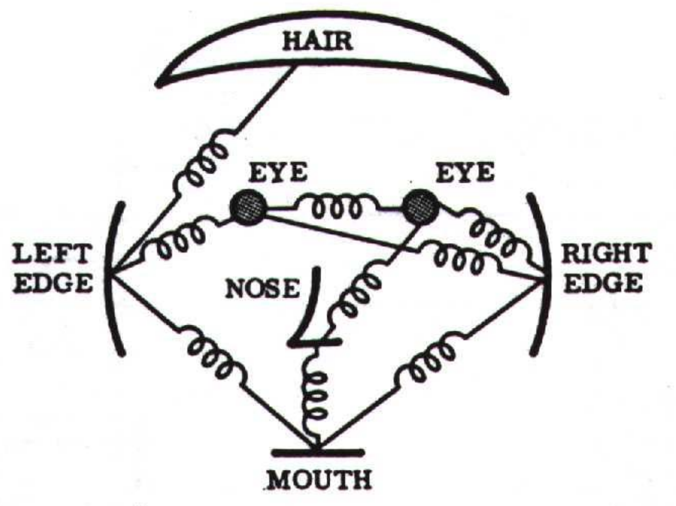
CS 664 Lecture 4
Flexible Template Matching

Prof. Dan Huttenlocher
Fall 2003

Flexible Template Matching

- Pictorial structures
  - Parts connected by springs and appearance models for each part
  - Used for human bodies, faces
  - Fischler&Elschlager, 1973 – considerable recent work

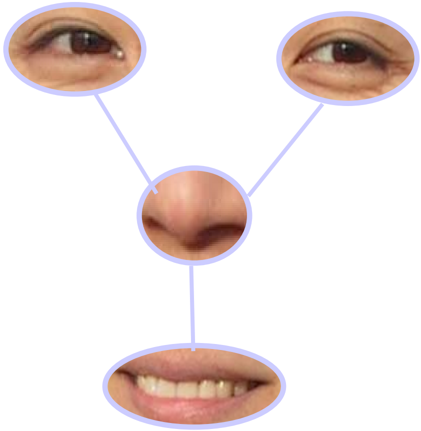


Formal Definition of Model

- Set of parts $V=\{v_1, \ldots, v_n\}$
- Configuration $L=(l_1, \ldots, l_n)$
  - Specifying locations of the parts
- Appearance parameters $A=(a_1, \ldots, a_n)$
  - Model for each part
- Edge e_{ij}, $(v_i,v_j) \in E$ for connected parts
  - Explicit dependency between part locations l_i, l_j
- Connection parameters $C=\{c_{ij} \mid e_{ij} \in E\}$
  - Spring parameters for each pair of connected parts

Flexible Template Algorithms

- Difficulty depends on structure of graph
  - Which parts are connected (E) and how (C)
- General case exponential time
  - Consider special case in which parts translate with respect to common origin
    - E.g., useful for faces



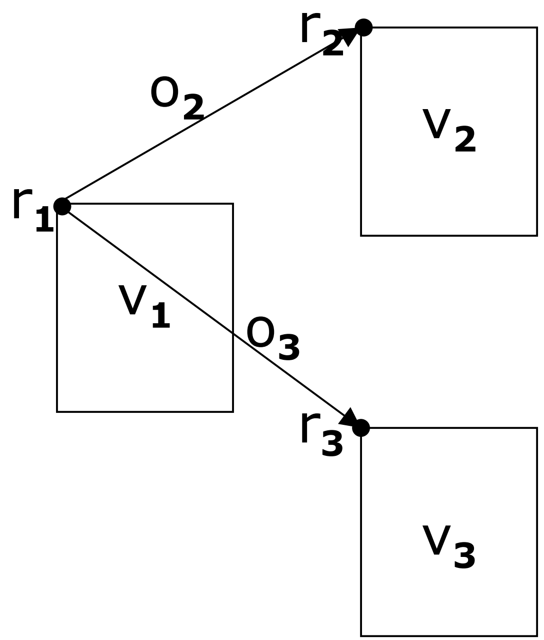
- Parts V= $\{v_1, \ldots v_n\}$
- Distinguished central part v_1
- Spring c_{i1} connecting v_i to v_1
- Quadratic cost for spring

Efficient Algorithm for Central Part

- Location $L=(l_1, \ldots, l_n)$ specifies where each part positioned in image
- Best location $\min_L (\Sigma_i m_i(l_i) + d_i(l_i,l_1))$
  - Part cost $m_i(l_i)$
    - Measures degree of mismatch of appearance a_i when part v_i placed at location l_i
  - Deformation cost $d_i(l_i,l_1)$
    - Spring cost c_{i1} of part v_i measured with respect to central part v_1
    - E.g., quadratic or truncated quadratic function
    - Note deformation cost zero for part v_1 (wrt self)

Central Part Model

- Spring cost c_{ij}: i=1, ideal location of l_j wrt l_1
  - Translation $o_j=r_j-r_1$
  - $T_j(x)=x+o_j$
- Spring cost deformation from this ideal
  - $\| l_j - T_j(l_1) \|^2$



Consider Case of 2 Parts

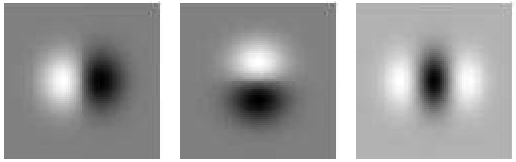
- $\min_{l1,l2} (m_1(l_1) + m_2(l_2) + \| l_2 - T_2(l_1) \|^2)$
  - Where $T_2(l_1)$ transforms l_1 to ideal location with respect to l_2 (offset)
- $\min_{l1} (m_1(l_1) + \min_{l2} (m_2(l_2) + \| l_2 - T_2(l_1) \|^2))$
  - But $\min_x (f(x) + \| x - y \|^2)$ is a distance transform
- $\min_{l1} (m_1(l_1) + D_{m2}(T_2(l_1)))$
- Sequential rather than simultaneous min
  - Don’t need to consider each pair of positions for the two parts because a distance
    - Just distance transform the match cost function, m

Several Parts wrt Reference Part

- $\min_{\mathbf{L}} (\Sigma_{\mathbf{i}} (m_{\mathbf{i}}(l_{\mathbf{i}}) + d_{\mathbf{i}}(l_{\mathbf{i}}, l_{\mathbf{1}})))$
- $\min_{\mathbf{L}} (\Sigma_{\mathbf{i}}\, m_{\mathbf{i}}(l_{\mathbf{i}}) + \| l_{\mathbf{i}} - T_{\mathbf{i}}(l_{\mathbf{1}}) \|^{\mathbf{2}})$
  - Quadratic distance between location of part $v_{\mathbf{i}}$ and ideal location given location of central part
- $\min_{\mathbf{l1}} (m_{\mathbf{1}}(l_{\mathbf{1}}) +$
  $\Sigma_{\mathbf{i>1}} \min_{\mathbf{li}} (m_{\mathbf{i}}(l_{\mathbf{i}}) + \| l_{\mathbf{i}} - T_{\mathbf{i}}(l_{\mathbf{1}}) \|^{\mathbf{2}}))$
  - i-th term of sum minimizes only over $l_{\mathbf{i}}$
- $\min_{\mathbf{l1}} (m_{\mathbf{1}}(l_{\mathbf{1}}) + \Sigma_{\mathbf{i>1}} D_{\mathbf{mi}}(T_{\mathbf{i}}(l_{\mathbf{1}})))$
  - Because $D_{\mathbf{f}}(x) = \min_{\mathbf{y}} (f(y) + \| y\text{-}x \|^{\mathbf{2}})$
  - Using same D.T. algorithms as for binary images

Application to Face Detection

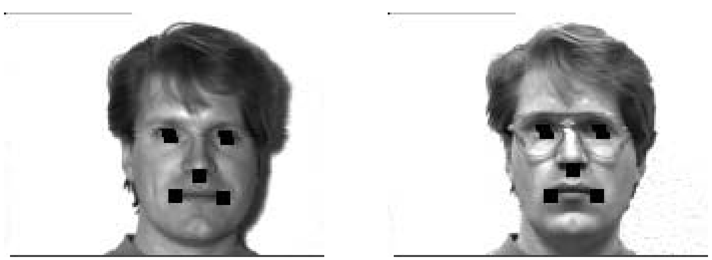
- Five parts: eyes, tip of nose, sides of mouth
- Each part a local image patch
  - Represented as response to oriented filters



  - 27 filters at 3 scales and 9 orientations
  - Learn coefficients from labeled examples
- Parts translate with respect to central part, tip of nose

Flexible Template Face Detection

- Runs at several frames per second
  - Compute oriented filters at 27 orientations and scales for part cost m_i
  - Distance transform m_i for each part other than central one (nose tip)
  - Find maximum of sum for detected location



More General Flexible Templates

- Efficient computation using distance transforms for any tree-structured model
  - Not limited to central reference part
- Two differences from reference part case
  - Relate positions of parts to one another using tree-structured recursion
    - Solve with Viterbi or forward-backward algorithm
  - Parameterization of distance transform more complex – transformation T_{ij} for each connected pair of parts

General Form of Problem

- Best location can be viewed in terms of probability or cost (negative log prob.)
  - $\max_L p(L|I,\Theta)=\text{argmax}_L p(I|L,A)p(L|E,C)$
  - $\min_L \Sigma_V m_{\mathbf{j}}(l_{\mathbf{j}}) + \Sigma_E d_{\mathbf{ij}}(l_{\mathbf{i}},l_{\mathbf{j}})$
    - $m_{\mathbf{j}}(l_{\mathbf{j}})$ – how well part $v_{\mathbf{j}}$ matches image at $l_{\mathbf{j}}$
    - $d_{\mathbf{ij}}(l_{\mathbf{i}},l_{\mathbf{j}})$ – how well locations $l_{\mathbf{i}},l_{\mathbf{j}}$ agree with model (spring connecting parts $v_{\mathbf{i}}$ and $v_{\mathbf{j}}$)
- Difficulty of maximization/minimization depends to large degree on form of graph

Minimizing Over Tree Structures

- Use dynamic programming to minimize $\Sigma_V\, m_j(l_j) + \Sigma_E\, d_{ij}(l_i,l_j)$
- Can express as function for pairs $B_j(l_i)$
  - Cost of best location of $v_{\mathbf{j}}$ given location $l_{\mathbf{i}}$ of $v_{\mathbf{i}}$
- Recursive formulas in terms of children C_j of v_j
  - $B_{\mathbf{j}}(l_{\mathbf{i}}) = \min_{\mathbf{lj}} (m_{\mathbf{j}}(l_{\mathbf{j}}) + d_{\mathbf{ij}}(l_{\mathbf{i}},l_{\mathbf{j}}) + \Sigma_{Cj}\, B_{\mathbf{c}}(l_{\mathbf{j}}))$
  - For leaf node no children, so last term empty
  - For root node no parent, so second term omitted

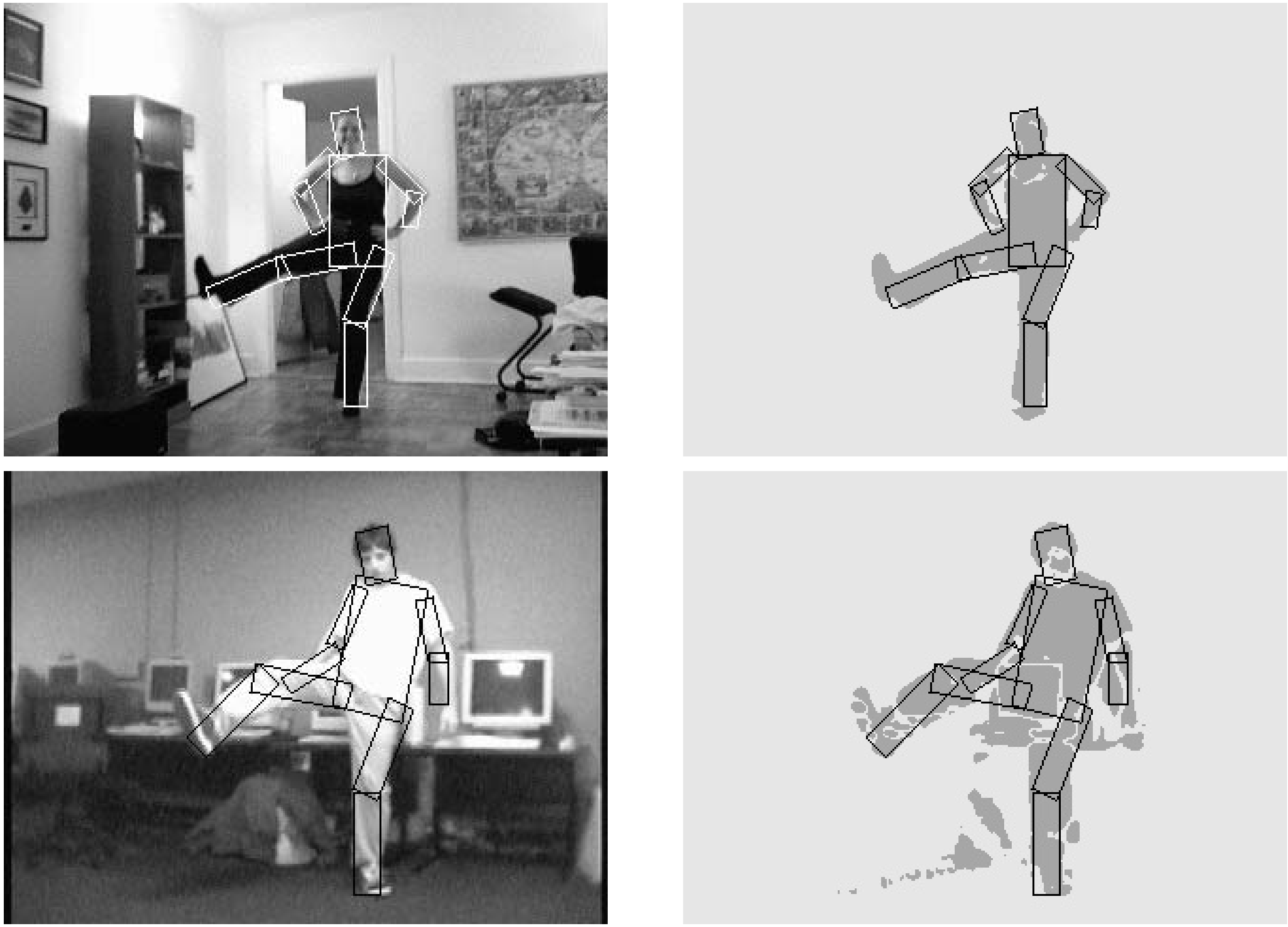
Efficient Algorithm for Trees

- MAP estimation algorithm
  - Tree structure allows use of Viterbi style dynamic programming
    - $O(ns^2)$ rather than $O(s^n)$ for s locations, n parts
    - Still slow to be useful in practice (s in millions)
  - Couple with distance transform method for finding best pair-wise locations in linear time
    - Resulting O(ns) method
- Similar techniques allow sampling from posterior distribution in O(ns) time
  - Using forward-backward algorithm

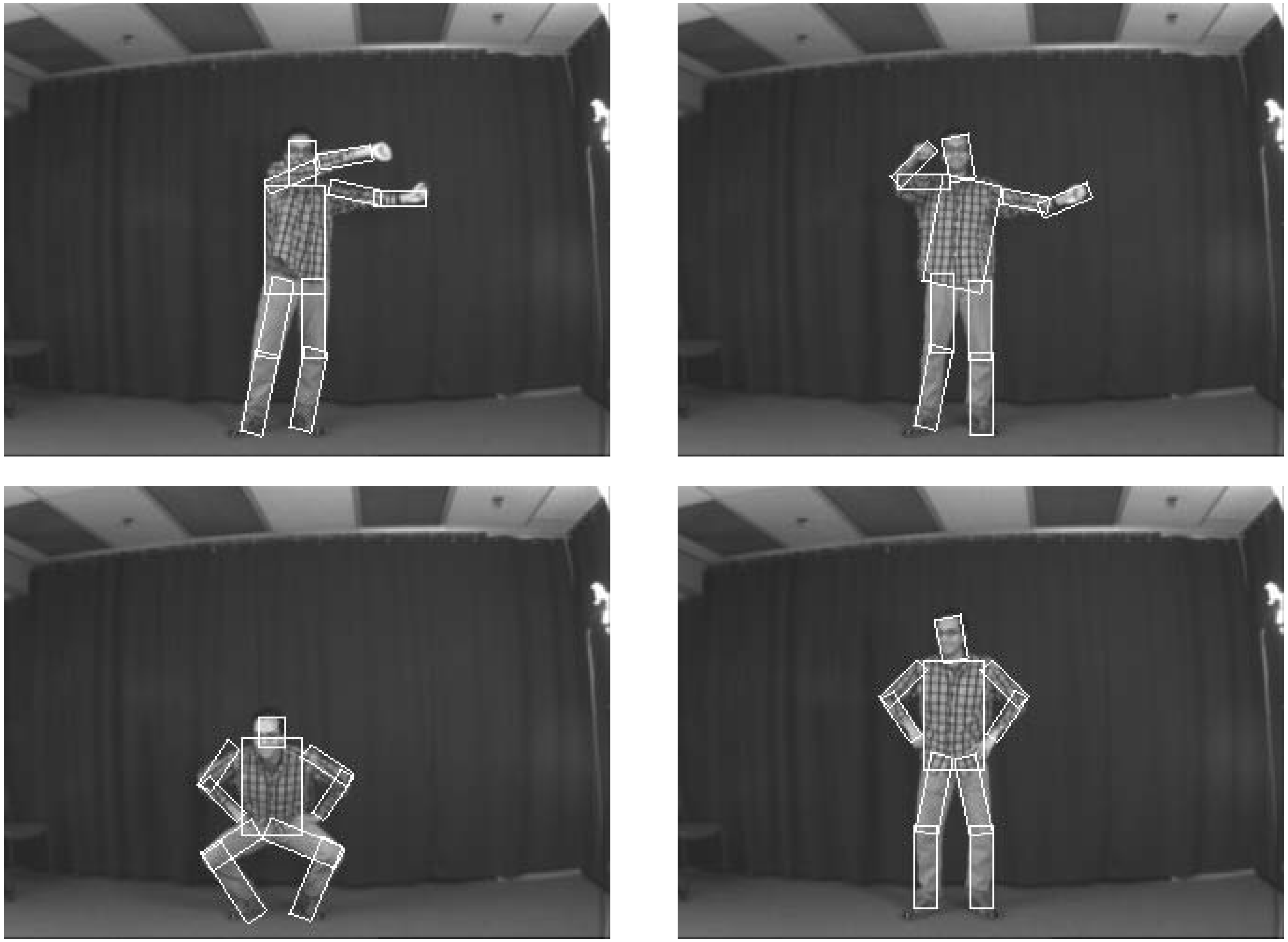
O(ns) Algorithm for MAP Estimate

- Express $B_j(l_i)$ in recursive minimization formulas as a DT $D_f(T_{ij}(l_i))$
  - Cost function
    - $f(y) = m_{\mathbf{j}}(T_{\mathbf{ji}}^{-1}(y)) + \sum_{\mathbf{Cj}} B_{\mathbf{c}}(T_{\mathbf{ji}}^{-1}(y))$
  - $T_{\mathbf{ij}}$ maps locations to space where difference between $l_{\mathbf{i}}$ and $l_{\mathbf{j}}$ is a squared distance
    - Distance zero at ideal relative locations
- Yields n recursive equations
  - Each can be computed in O(sD) time
    - D is number of dimensions to parameter space but is fixed (in our case D is 2 to 4)

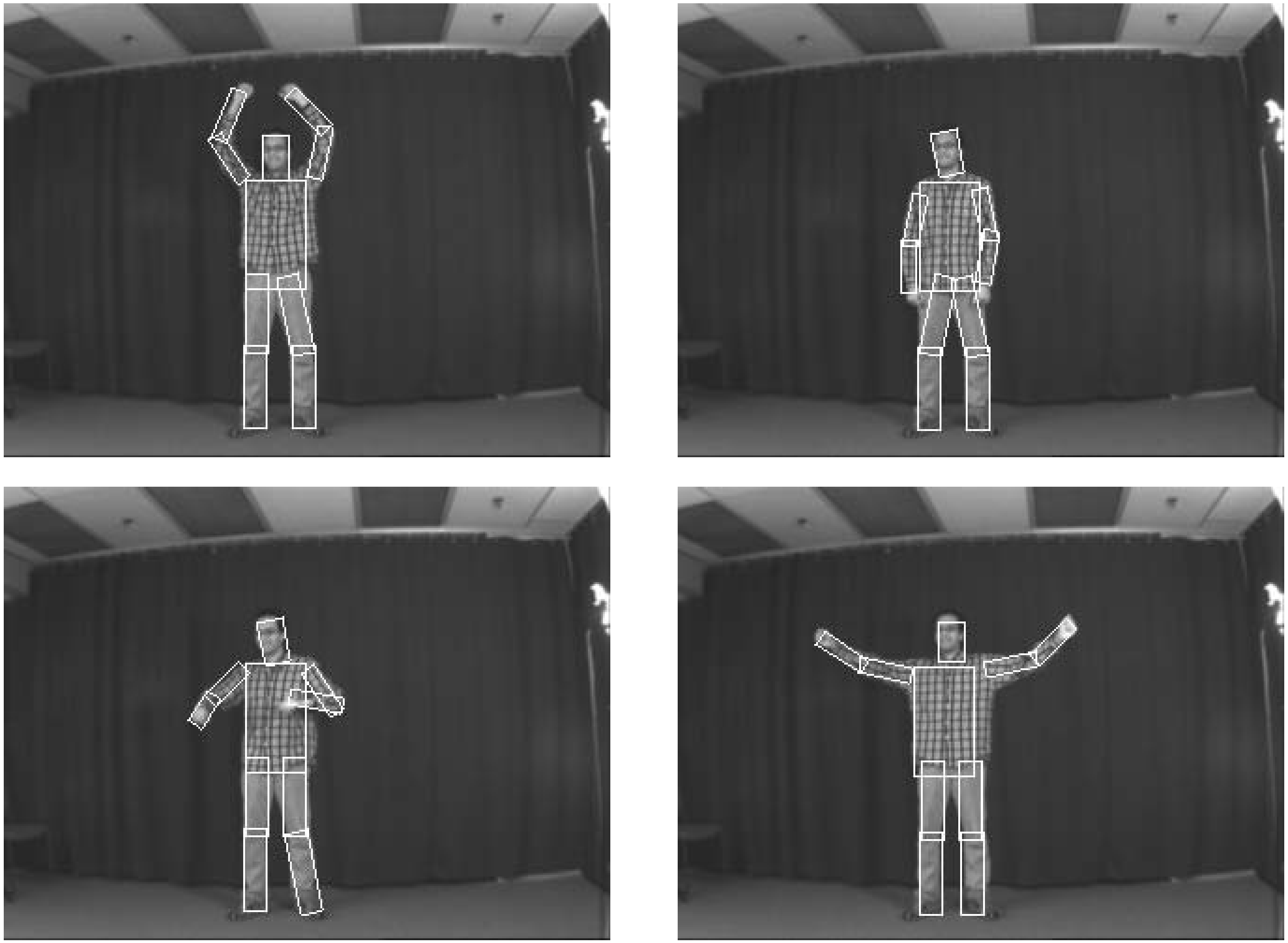
Example: Recognizing People



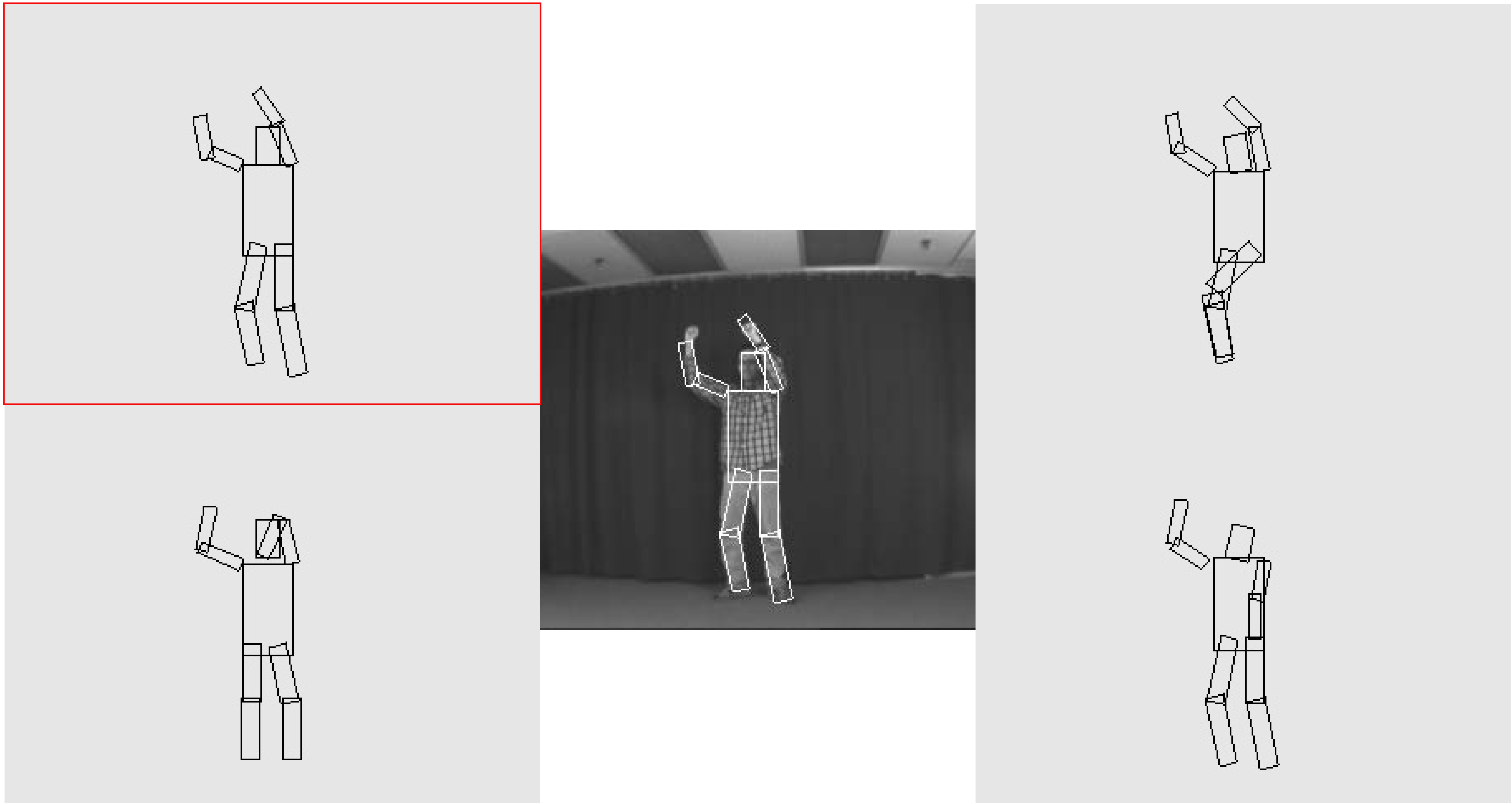
Variety of Poses



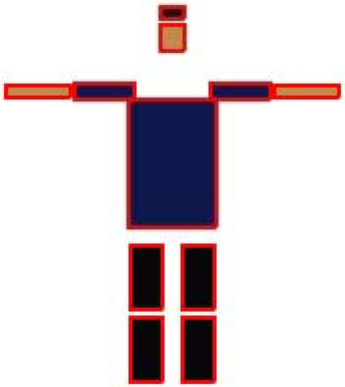
Variety of Poses

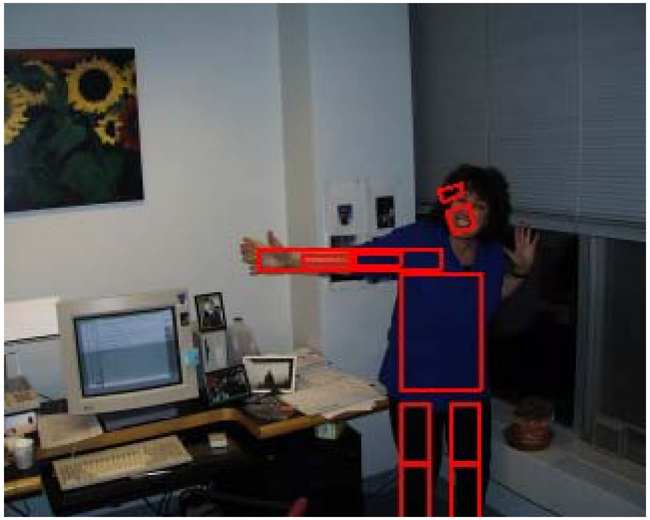


Samples From Posterior



Model of Specific Person





Bayesian Formulation of Learning

- Given example images $I^1, \ldots, I^m$ with configurations $L^1, \ldots, L^m$
  - Supervised or labeled learning problem
- Obtain estimates for model $\Theta=(A,E,C)$
- Maximum likelihood (ML) estimate is
  - $\mathrm{argmax}_\Theta\ p(I^{\mathbf{1}}, \ldots, I^{\mathbf{m}}, L^{\mathbf{1}}, \ldots, L^{\mathbf{m}} \mid \Theta)$
  - $\mathrm{argmax}_\Theta \prod_{\mathbf{k}} p(I^{\mathbf{k}},L^{\mathbf{k}}\mid\Theta)$
    - Independent examples
  - $\mathrm{argmax}_\Theta \prod_{\mathbf{k}} p(I^{\mathbf{k}}\mid L^{\mathbf{k}},A) \prod_{\mathbf{k}} p(L^{\mathbf{k}}\mid E,C)$
    - Independent appearance and dependencies

Efficiently Learning Tree Models

- Estimating appearance $p(I^k|L^k,A)$
  - ML estimation for particular type of part
    - E.g., for constant color patch use Gaussian model, computing mean color and covariance
- Estimating dependencies $p(L^k|E,C)$
  - Estimate C for pairwise locations, $p(l_i^k,l_j^k|c_{ij})$
    - E.g., for translation compute mean offset between parts and variation in offset
  - Best tree using minimum spanning tree (MST) algorithm
    - Pairs with "smallest relative spatial variation"

Example: Generic Person Model

- Each part represented as rectangle
  - Fixed width, varying length
  - Learn average and variation
    - Connections approximate revolute joints
  - Joint location, relative position, orientation, foreshortening
  - Estimate average and variation
- Learned model (used above)
  - All parameters learned
    - Including “joint locations”
  - Shown at ideal configuration