Object Recognition Using Pictorial Structures

Daniel Huttenlocher
Computer Science Department

Joint work with Pedro Felzenszwalb, MIT AI Lab

In This Talk

- Object recognition in computer vision
  - Brief definition and overview
- Part-based models of objects
  - Pictorial structures for 2D modeling
- A Bayesian framework
  - Formalize both learning and recognition problems
- Efficient algorithms for pictorial structures
  - Learning models from labeled examples
  - Recognizing objects (anywhere) in images

Object Recognition

- Given some kind of model of an object
  - Shape and geometric relations
  - Two- or three-dimensional
  - Appearance and reflectance – color, texture, …
  - Generic object class versus specific object
- Recognition involves
  - Detection: determining whether an object is visible in an image (or how likely)
  - Localization: determining where an object is in the image

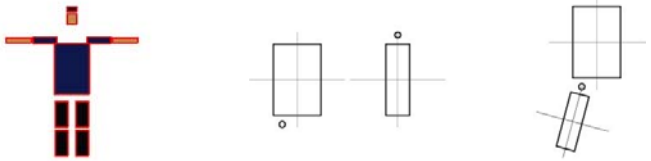
Our Recognition Goal

- Detect and localize multi-part objects that are at arbitrary locations in a scene
  - Generic object models such as person or car
  - Allow for "articulated" objects
  - Combine geometry and appearance
  - Provide efficient and practical algorithms



Pictorial Structures

- Local models of appearance with non-local geometric or spatial constraints
  - Image patches describing color, texture, etc.
  - 2D spatial relations between pairs of patches
- Simultaneous use of appearance and spatial information
  - Simple part models alone too non-distinctive



A Brief History of Recognition

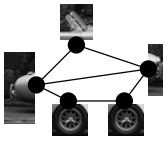
- Pictorial structures date from early 1970's
  - Practical recognition algorithms proved difficult
- Purely geometric models widely used
  - Combinatorial matching to image features
  - Dominant approach through early 1990's
  - Don't capture appearance such as color, texture
- Appearance based models for some tasks
  - Templates or patches of image, lose geometry
    - Generally learned from examples
  - Face recognition a common application

Other Part-Based Approaches

- Geometric part decompositions
  - Solid modeling (e.g., Biederman, Dickinson)
- Person models
  - First detect local features then apply geometric constraints of body structure (Forsyth & Fleck)
- Local image patches with geometric constraints
  - Gaussian model of spatial distribution of parts (Burl & Perona)
  - Pictorial structure style models (Lipson et al)

Formal Definition of Our Model

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



Quick Review of Probabilistic Models

- Random variable X characterizes events
  - E.g., sum of two dice
- Distribution p(X) maps to probabilities
  - E.g., 2 → 1/36, 5 → 1/9, …
- Joint distribution p(X,Y) for multiple events
  - E.g., rolling a 2 and a 5
  - p(X,Y)=p(X)p(Y) when events independent
- Conditional distribution p(X|Y)
  - E.g., sum given the value of one die
- Random field is set of dependent r.v.'s

Problems We Address

- Recognizing model $\Theta=(A,E,C)$ in image I
  - Find most likely location L for the parts
    - Or multiple highly likely locations
  - Measure how likely it is that model is present
- Learning a model Θ from labeled example images $I^1,\ldots, I^m$ and $L^1, \ldots,L^m$
  - Known form of model parameters A and C
    - E.g., constant color rectangle
      - Learn a_i: average color and variation
    - E.g., relative translation of parts
      - Learn c_{ij}: average position and variation

Standard Bayesian Approach

- Estimate posterior distribution $p(L|I,\Theta)$
  - Probabilities of various configurations L given image I and model Θ
    - Find maximum (MAP) or high values (sampling)
- Proportional to $p(I|L,\Theta)p(L|\Theta)$ [Bayes' rule]
  - Likelihood $p(I|L,\Theta)$: seeing image I given configuration and model
    - Fixed L, depends only on appearance, $p(I|L,A)$
  - Prior $p(L|\Theta)$: obtaining configuration L given just the model
    - No image, depends only on constraints, $p(L|E,C)$

Class of Models

- Computational difficulty depends on Θ
  - Form of posterior distribution
- Structure of graph G=(V,E) important
  - G represents a Markov Random Field (MRF)
    - Each r.v. depends explicitly on neighbors
  - Require G be a tree
    - Prior on relative location $p(L|E,C) = \prod_{E} p(l_i,l_j|c_{ij})$
    - Natural for models of animate objects – skeleton
    - Reasonable for many other objects with central reference part (star graph)
    - Prior can be computed efficiently

Class of Models

- Likelihood $p(I|L,A) = \prod_i p(I|l_i,a_i)$
  - Product of individual likelihoods for parts
    - Good approximation when parts don't overlap
- Form of connection also important – space with "deformation distance"
  - $p(l_i,l_j|c_{ij}) \propto \eta(T_{ij}(l_i)-T_{ji}(l_i),0,\Sigma_{ij})$
    - Normal distribution in transformed space
  - T_{ij}, T_{ji} capture ideal relative locations of parts and Σ_{ij} measures deformation
    - Mahalanobis distance in transformed space (weighted squared Euclidean distance)

Bayesian Formulation of Learning

- Given example images I^1, …, I^m with configurations L^1, …, L^m
  - Supervised or labeled learning problem
- Obtain estimates for model $\Theta=(A,E,C)$
- Maximum likelihood (ML) estimate is
  - $\text{argmax}_\Theta\ p(I^1, \ldots, I^m, L^1, \ldots, L^m|\Theta)$
  - $\text{argmax}_\Theta \prod_k p(I^k,L^k|\Theta)$ independent examples
- Rewrite joint probability as product – appearance and dependencies separate
  - $\text{argmax}_\Theta \prod_k p(I^k|L^k,A) \prod_k p(L^k|E,C)$

Efficiently Learning 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

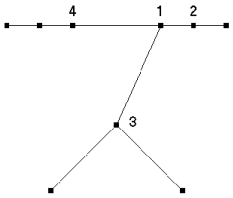
Example: Generic Face Model

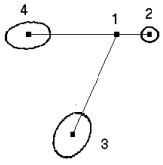
- Each part a local image patch
  - Represented as response to oriented filters

  - Vector a_i corresponding to each part
- Pairs of parts constrained in terms of their relative (x,y) position in the image
- Consider two models: 5 parts and 9 parts
  - 5 parts: eyes, tip of nose, corners of mouth
  - 9 parts: eye split into pupil, left side, right side

Learned 9 Part Face Model

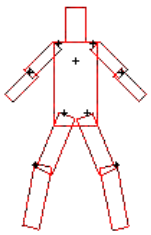
- Appearance and structure parameters learned from labeled frontal views
  - Structure captures pairs with most predictable relative location – least uncertainty
  - Gaussian (covariance) model captures direction of spatial variations – differs per part





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 10 part model
  - All parameters learned
    - Including "joint locations"
  - Shown at ideal configuration



Bayesian Formulation of Recognition

- Given model Θ and image I, seek "good" configuration L
  - Maximum a posteriori (MAP) estimate
    - Best (highest probability) configuration L
    - $L^* = \text{argmax}_L \, p(L|I,\Theta)$
  - Sampling from posterior distribution
    - Values of L where $p(L|I,\Theta)$ is high
      - With some other measure for testing hypotheses
- Brute force solutions intractable
  - With n parts and s possible discrete locations per part, $O(s^n)$

Efficiently Recognizing Objects

- 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)
  - New dynamic programming method for finding best pair-wise locations in linear time
    - Resulting O(ns) method
    - Requires a "distance" not arbitrary cost
- Similar techniques allow sampling from posterior distribution in O(ns) time

The Minimization Problem

- Recall that best location is
  - $L^* = \text{argmax}_L p(L|I,\Theta) = \text{argmax}_L p(I|L,A)p(L|E,C)$
- Given the graph structure (MRF) just pairwise dependencies
  - $L^* = \text{argmax}_L \prod_{\mathbf{V}} p(I|l_i,a_i) \prod_{\mathbf{E}} p(l_i,l_j|c_{ij})$
- Standard approach is to take negative log
  - $L^* = \text{argmin}_L \Sigma_V m_j(l_j) + \Sigma_E d_{ij}(l_i,l_j)$
    - $m_j(l_j) = -\log p(I|l_j,a_j)$ – how well part v_j matches image at l_j
    - $d_{ij}(l_i,l_j) = -\log p(l_i,l_j|c_{ij})$ – how well locations l_i,l_j agree with model

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_j given location l_i of v_i
- Recursive formulas in terms of children C_j of v_j
  - $B_j(l_i) = \min_{lj} (m_j(l_j) + d_{ij}(l_i,l_j) + \Sigma_{Cj} B_c(l_j))$
  - For leaf node no children, so last term empty
  - For root node no parent, so second term omitted

Running Time

- Compute minimum using these equations
  - Start with leaf nodes, build up sub-trees
- $O(ns^2)$ running time for n parts and s locations of each part
  - Each part pair defining one equation $B_j(l_i)$
    - $O(s^2)$ time per pair, O(n) pairs
- When d_{ij} is distance don't need to consider location pairs
  - Define $B_j(l_i)$ as a kind of distance transform
    - For each location of v_j minimum location of v_i

Classical Distance Transforms

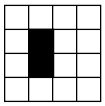
- Defined for set of points, P,

$$\Delta_P(x) = \min_{y \in P} ||x - y||$$

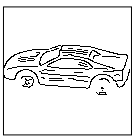
  - For each location x distance to nearest y in P
  - Think of as cones rooted at each point of P
- Commonly computed on a grid Γ using

$$\Delta_P(x) = \min_{y \in \Gamma} (||x - y|| + 1_B(y))$$

  - Where $1_B(y) = 0$ when $y \in P$, ∞ otherwise

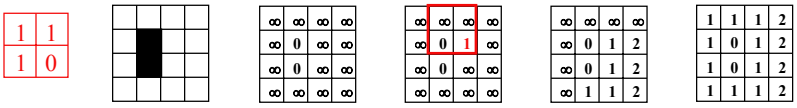


1	1	1	2
1	0	1	2
1	0	1	2
1	1	1	2



Computing Distance Transforms

- Two pass algorithm for L_1 norm
  - O(sD) time for s locations on a D-dim grid
  - On each pass, min sum of mask and distance array ("in place")
- Simple method to approximate L_p norms
- More involved exact method for L_2 that also reports which point is closest



Generalized Distance Transforms

- Replace indicator function with arbitrary f
  - $\Delta_f(x) = \min_{y \in \Gamma} (||x - y|| + f(y))$
- Intuitively, for grid location x, find y where f(y) plus distance to x is "small"
  - A distance plus a cost for each location
- Change in $\Delta_f(x)$ is bounded by change in x
  - Small value of f "dominates" nearby large values
- This generalized distance transform (GDT) computed same way as classic DT

O(ns) Algorithm for MAP Estimate

- Can express $B_j(l_i)$ in recursive minimization formulas as a GDT $\Delta_f(T_{ij}(l_i))$
  - Cost function for GDT
    - $f(y) = m_j(T_{ji}^{-1}(y)) + \Sigma_{Cj} B_c(T_{ji}^{-1}(y))$
  - T_{ij} maps locations to space where difference between l_i and l_j is a squared distance
    - Distance zero at ideal relative locations
- Have 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)

Recognizing Faces

Generic model of frontal view

- Using learned 5- and 9-part models
  - Local oriented filters for parts
  - Relatively small spatial variation in part locations
  - Similar overall size and orientation of face
- MAP estimation to find best match
  - Posterior estimate of configuration L is accurate because parts do not overlap
  - Consider all possible locations in image
  - Runs at several frames per second on a desktop workstation

Example: Recognizing Faces

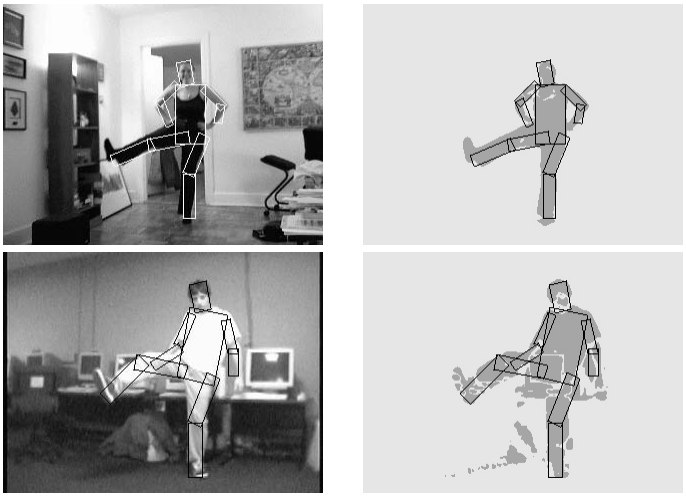
Example: Recognizing People

- Frontal view models
  - Generic model using binary rectangles for parts
    - Match to "difference image"
  - Specific model using color rectangles for parts
    - Match to original image
- Sampling posterior to find good matches
  - Posterior estimate of L can be high for several configurations due to overlap of parts
  - Use best of 200 samples
    - Measured using correlation (Chamfer matching)
  - Search over all locations runs in under minute

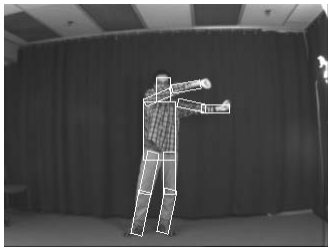
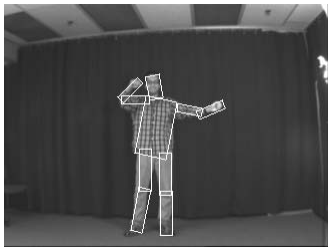
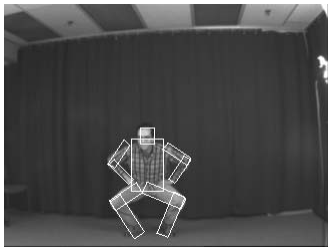
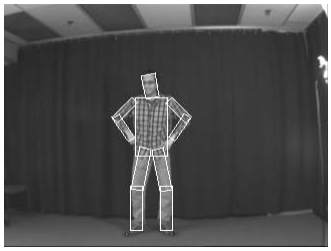
Sampling the Posterior

- Generate good possible matches as hypotheses
  - Locations where $p(L|I,\Theta)$ large
  - Validate or compare using another technique
    - Here use a correlation-like measure (Chamfer)
- Computation similar to MAP estimation
  - Recursive equations, one per part
  - Ability to solve each equation in linear time
    - Via convolution with Gaussian
    - Linear time dynamic programming approximation using box filters (due to Wells)

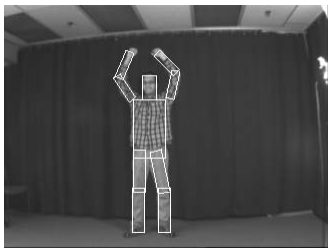
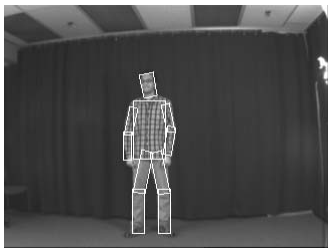
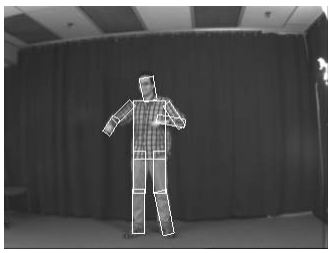
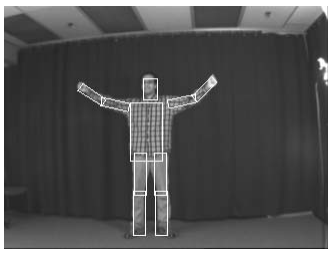
Example: Recognizing People



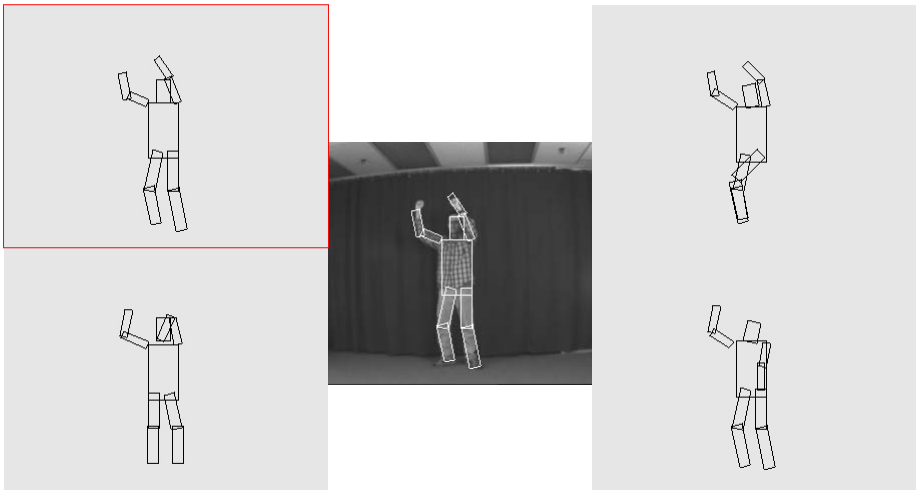
Variety of Poses

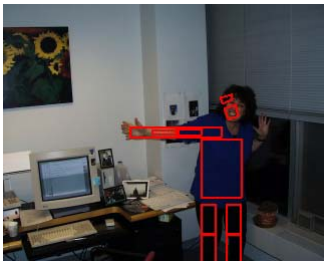
Variety of Poses

Samples From Posterior



Model of Specific Person



Summary

- Pictorial structures combine local part appearance and global spatial constraints
  - Don't try to localize parts first – exploit context
  - Suitable for generic models of object classes
- Bayesian framework provides natural learning problem – ML estimation
  - Only requires placing part models in images; structure and parameters are learned
- Practical algorithms for searching over all possible locations in image
  - Best match or good matches (high posterior)

What's Next

- Allow for occluded parts
  - Make part likelihood $p(I|l_i,a_i)$ a robust measure
- Apply to tracking people in video
  - Incorporate location at previous time frame into prior
    - Use for more efficient methods
- Start with generic models and use to learn person specific models
  - Discriminate between people
- Use person and face methods together