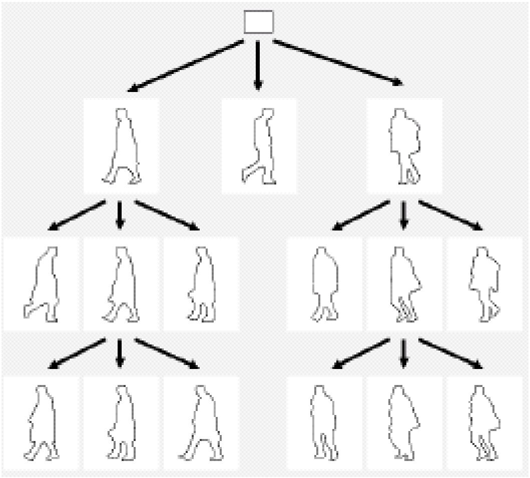
CS 664 Lecture 5
Hausdorff and Chamfer Cont.

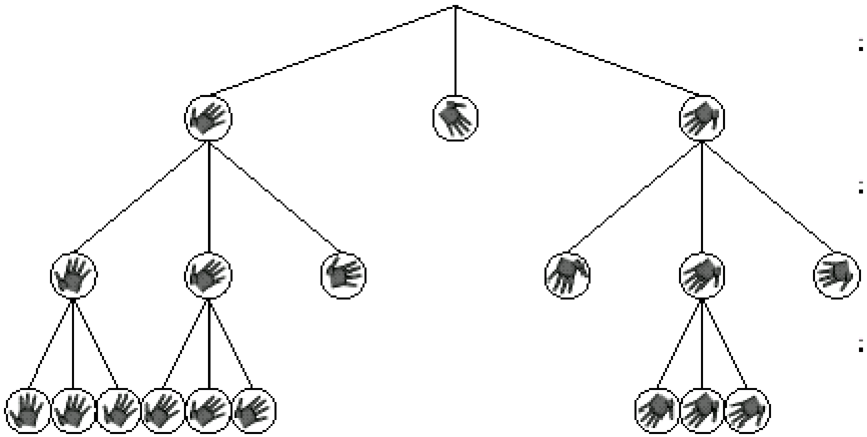
Prof. Dan Huttenlocher
Fall 2003

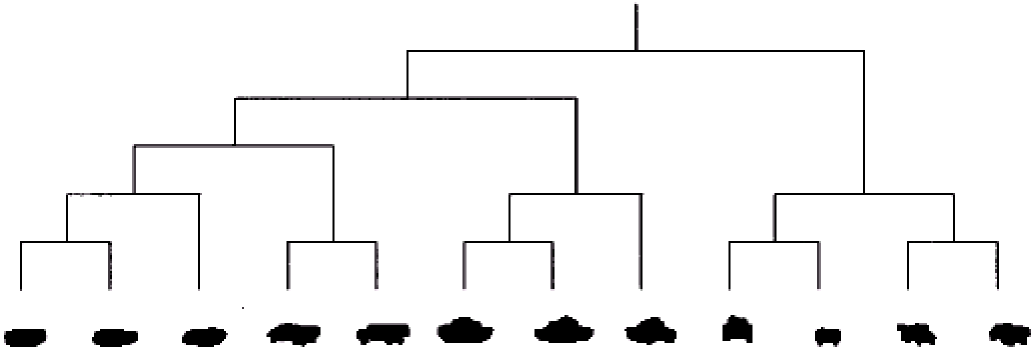
Template Clustering

- Cluster templates into tree structures to speed matching
  - Rule out multiple templates simultaneously
    - Coarse-to-fine search where coarse granularity can rule out many templates
    - Several variants: Olson, Gavrila, Stenger
- Applies to variety of DT based matching measures
  - Chamfer, Hausdorff and robust Chamfer
- Use hierarchical clustering techniques (e.g., Edelsbrunner) offline on templates

Example Hierarchical Clusters



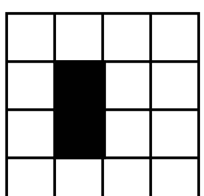




Larger pairwise differences higher in tree

DT and Morphological Dilation

- Dilation operation replaces each point of P with some fixed point set Q
  - $P \oplus Q = \cup_{\mathbf{p}} \cup_{\mathbf{q}} p+q$
- Dilation by a "disc" C^d of radius d replaces each point with a disc
  - A point is in the dilation of P by $C^{\mathbf{d}}$ exactly when the distance transform value is no more than d (for appropriate disc and distance fcn.)
  - $x \in P \oplus C^{\mathbf{d}} \iff D_{\mathbf{P}}(x) \leq d$



2	1	2	3
1	0	1	2
1	0	1	2
2	1	2	3

0	1	0	0
1	1	1	0
1	1	1	0
0	1	0	0

1	1	1	0
1	1	1	1
1	1	1	1
1	1	1	0

Dilate and Correlate Matching

- Fixed degree of "smoothing" of features
  - Dilate binary feature map with specific radius disc rather than all radii as in DT
- $h_k(A,B) \leq d \iff |A \cap B^d| \geq k$
  - At least k points of A contained in B^d
- For low dimensional transformations such as x-y-translation best way to compute
  - Dilation and binary correlation are very fast
  - For higher dimensional cases hierarchical search using DT is faster

Dot Product Formulation

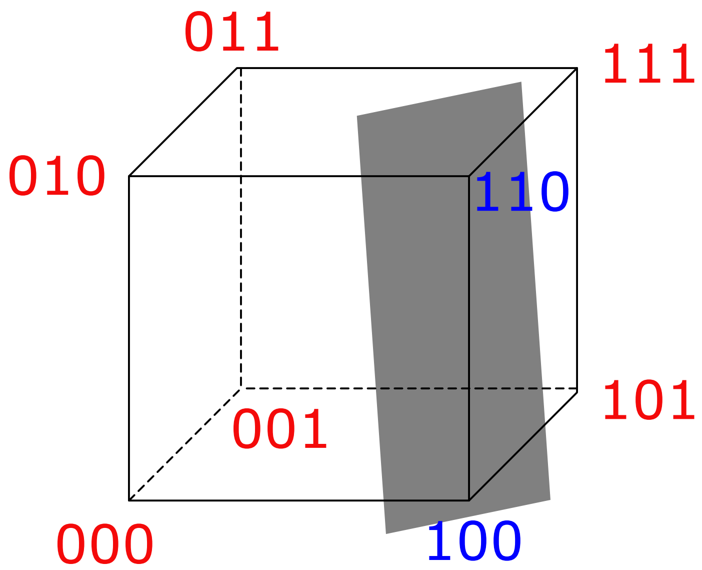
- Let $\boldsymbol{A}$ and $\boldsymbol{B}^{\mathbf{d}}$ be (binary) vector representations of A and B
  - E.g. standard scan line order
- Then fractional Hausdorff distance can be expressed as dot product
  - $h_{\mathbf{k}}(A,B) \leq d \Leftrightarrow \boldsymbol{A}\bullet\boldsymbol{B}^{\mathbf{d}} \geq k$
- Note that if B is perturbation of A by d then $\boldsymbol{A}\bullet\boldsymbol{B}$ is arbitrary whereas $\boldsymbol{A}\bullet\boldsymbol{B}^{\mathbf{d}} = \boldsymbol{A}\bullet\boldsymbol{A}$
- Hausdorff matching using linear subspaces
  - Eigenspace, PCA, etc.

Learning and Hausdorff Distance

- Learning linear half spaces
  - Dot product formulation defines linear threshold function
    - Positive if $\boldsymbol{A} \bullet \boldsymbol{B}^{\mathbf{d}} \geq k$, negative otherwise
- PAC – probably approximately correct
  - Learning concepts that with high probability have low error
  - Linear programming and perceptrons can both be used to learn half spaces in PAC sense
- Consider small number of values for d (dilation parameter) and pick best

Illustration of Linear Halfspace

- Possible images define n-dimensional binary space
- Linear function separating positive and negative examples



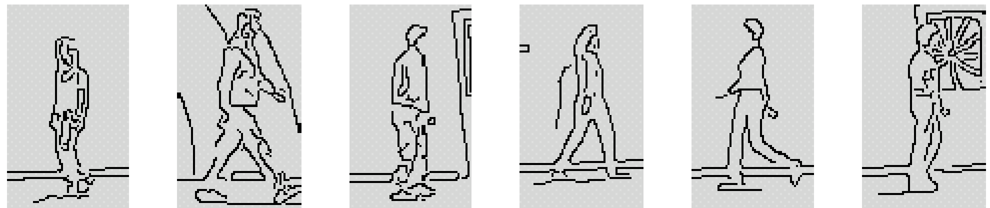
Perceptron Algorithm

- Examples x_i each with label $y_i \in \{+,-\}$
- Set initial prediction vector v to 0
- For i=1, …, m
  - If $\text{sign}(v \bullet x_i) \neq \text{sign}(y_i)$
    then $v = v + y_i x_i$
- Run repeatedly until no misclassifications on m training examples
  - Or less than some threshold number but then haven't found linear separator
- Generally need many more negative than positive examples for effective training

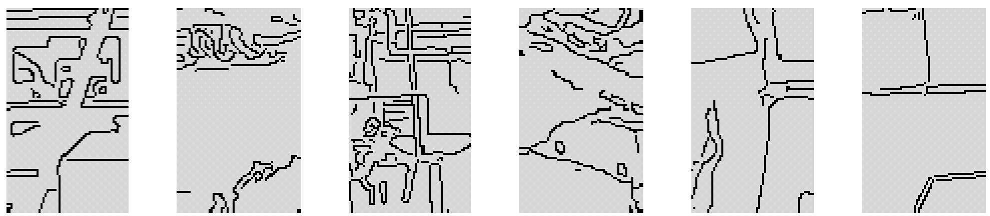
Perceptron Algorithm

- Perceptron classifier learns concepts c of form u•c $\geq$ 0
  - Our problem of form u•c $\geq$ 0
  - Map into one higher dimensional space
    - Unknown u = (-kκ …)
    - Concept c = (κ …)
    - Note in practice converges most rapidly if κ proportional to length of vector (e.g., sqrt)
- Train perceptron on dilated training data
  - Positive and negative labeled examples
- Recognize by dot product of resulting c

Learned Half-Space Templates

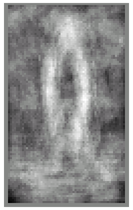


Positive examples (500)

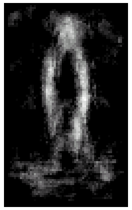


Negative examples (350,000)

All Model Coefs.

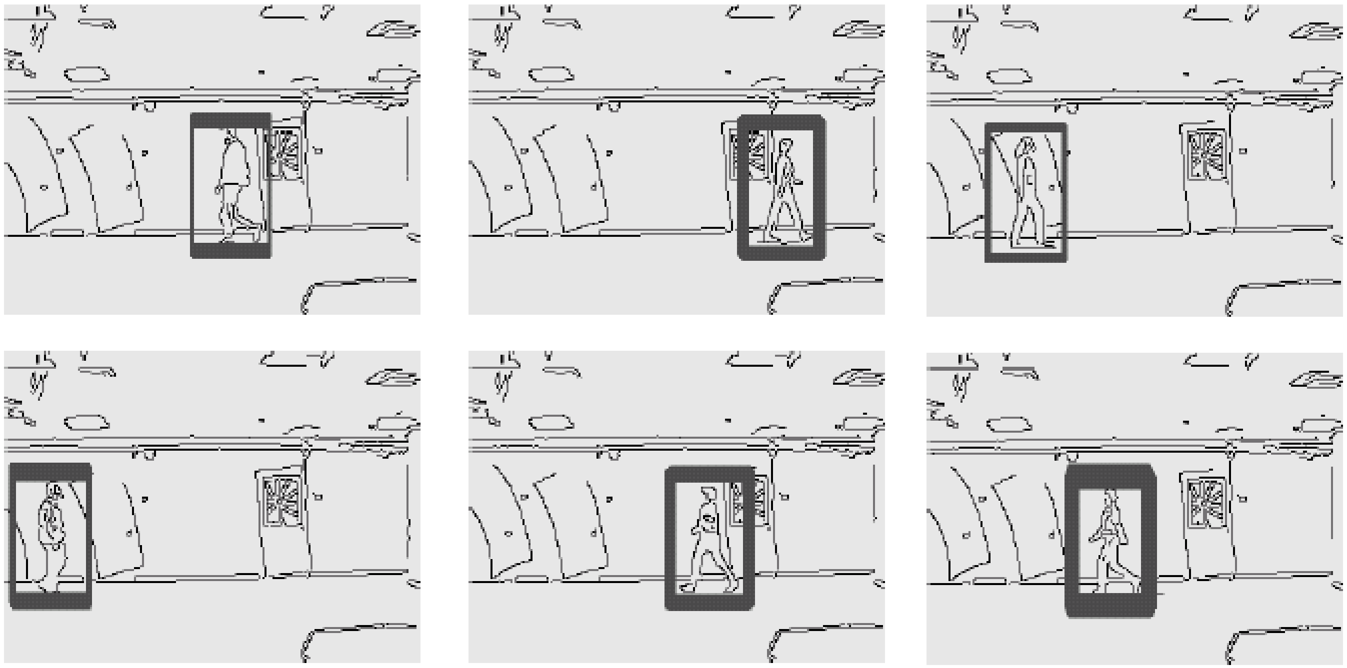


Pos. Model Coefs.



Example Model (dilation d=3, picked automatically)

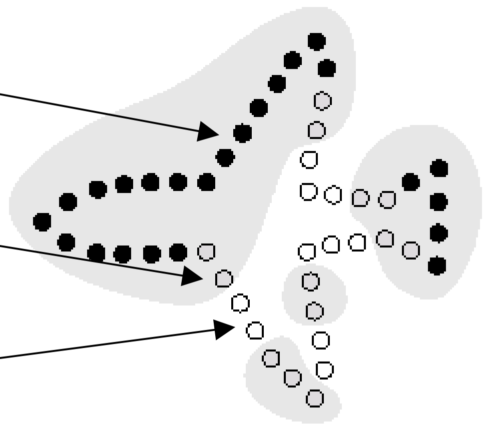
Detection Results



- Train on 80% test on 20% of data
  - No trials yielded any false positives
  - Average 3% missed detections, worst case 5%

Spatial Continuity

- Hausdorff and Chamfer matching do not measure degree of connectivity
  - E.g., edge chains versus isolated points
- Spatially coherent matching approach
  - Separate features into three subsets
    - Matchable
      - Near image features
    - Boundary
      - Matchable but near un-matchable
    - Un-matchable
      - Far from image features



Solving for Transformation

Find T which minimizes error between transformed model and data

For each datum

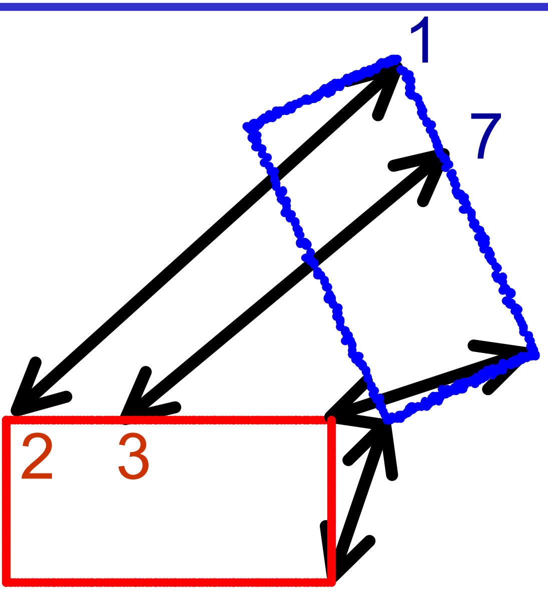
$$\epsilon(T) = -\log P(T) = \sum_j \min_i \; d(T * \mathbf{M}_i, \mathbf{D}_j)$$

Where:

- $d(\mathbf{x}, \mathbf{y})$ is a distance between points $\mathbf{x}$ and $\mathbf{y}$.
- $T * \mathbf{x}$ applies the transformation to $\mathbf{x}$
  e.g. $T = (\theta, t_x, t_y)$ for 2D

$$T * \mathbf{x} = \begin{pmatrix} x\cos\theta + y\sin\theta + t_x \\ -x\sin\theta + y\cos\theta + t_y \end{pmatrix}$$

Easy if Correspondence Known



1 – 2

7 – 3

...

Hard:

$$\epsilon(T) = \sum_j \min_i d(T * \mathbf{M}_i, \mathbf{D}_j)$$

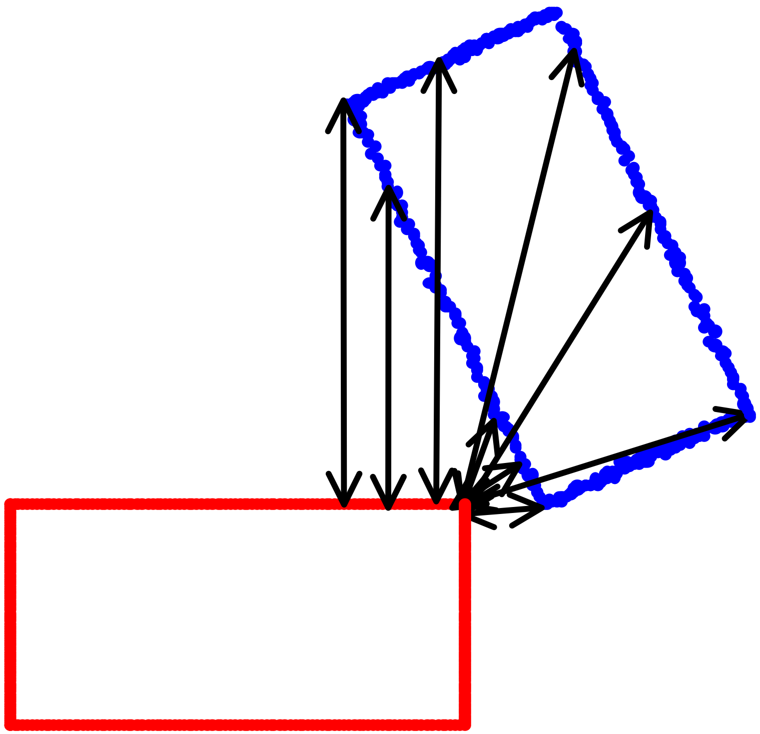
Easy:

Given *correspondences* $j \leftrightarrow \phi(j)$

Can minimize

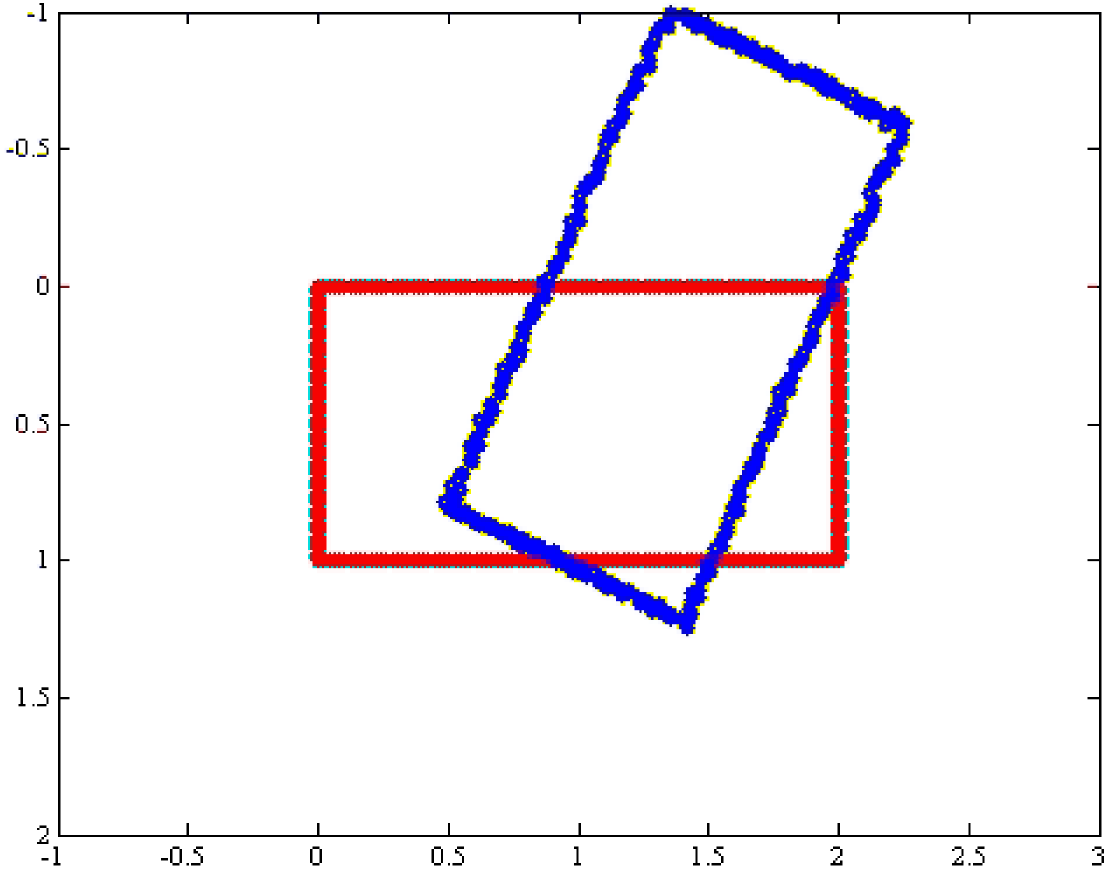
$$\sum_j d(T * \mathbf{M}_{\phi(j)}, \mathbf{D}_j)$$

Don't Know Correspondence Guess and Try to Improve



- That's OK, just choose the closest point...
- *Of course* it's wrong, but it will get us closer

ICP: Iterated Closest Point



Problems with ICP

- Slow
  - Can take many iterations
  - Each iteration slow due to search for correspondences
    - Fitzgibbons: improve this by using distance transform
- No convergence guarantees
  - Can get stuck in local minima
    - Not much to do about this
    - Can be improved by using robust distance measures (e.g., truncated quadratic)