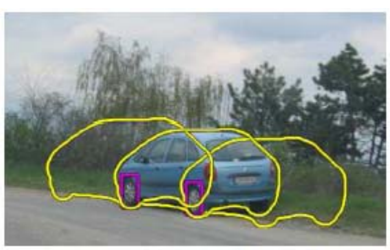
Lecture 8: Fitting

Tuesday, Sept 25



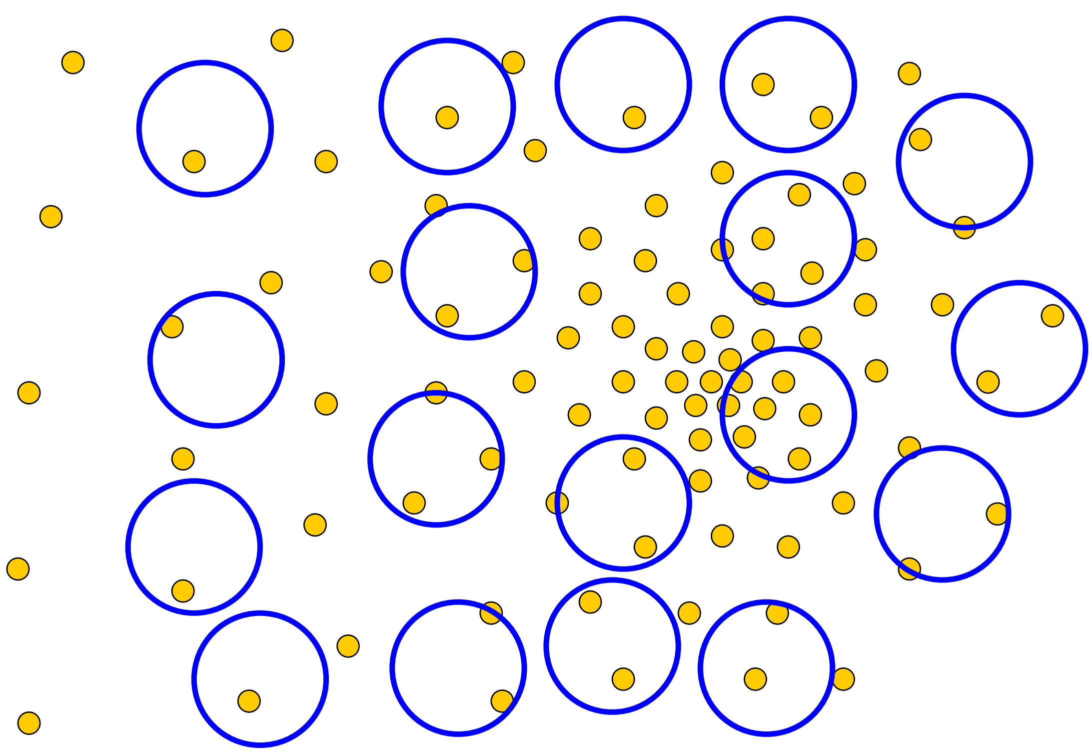

Announcements, schedule

- Grad student extensions
  - Due end of term
  - Data sets, suggestions
- Reminder: Midterm Tuesday 10/9
- Problem set 2 out Thursday, due 10/11

Outline

- Review from Thursday (affinity, cuts)
- Local scale and affinity computation
- Hough transform
- Generalized Hough transform
  - Shape matching applications
- Fitting lines
  - Least squares
  - Incremental fitting, k-means

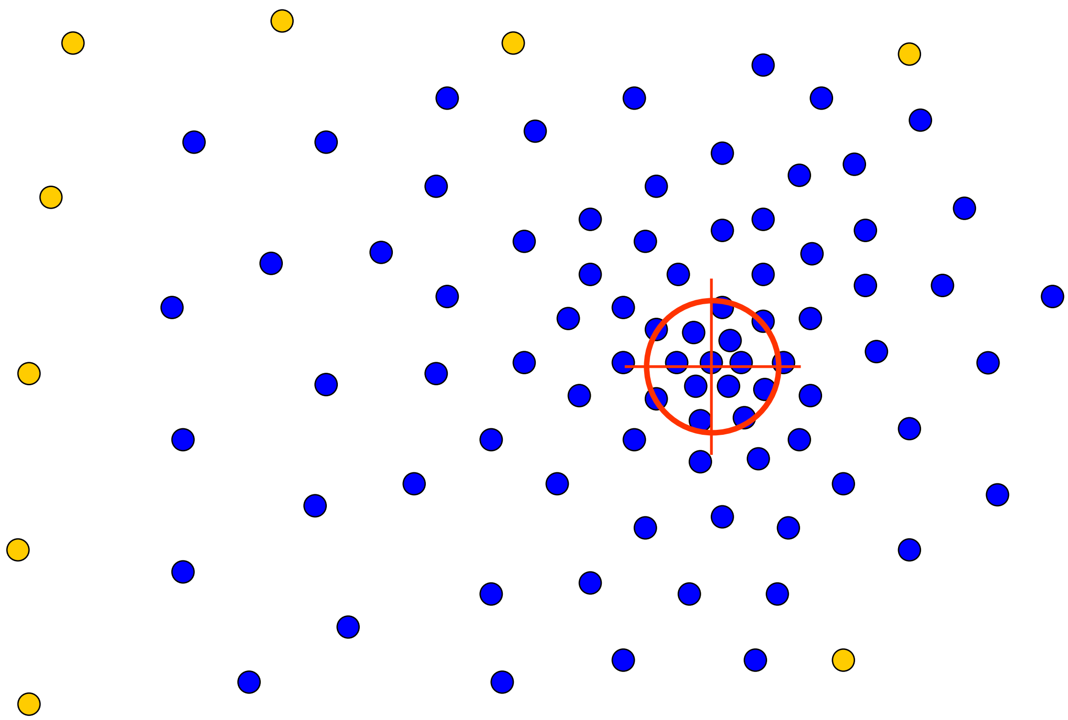
Real Modality Analysis



Tessellate the space with windows

Run the procedure in parallel

Real Modality Analysis



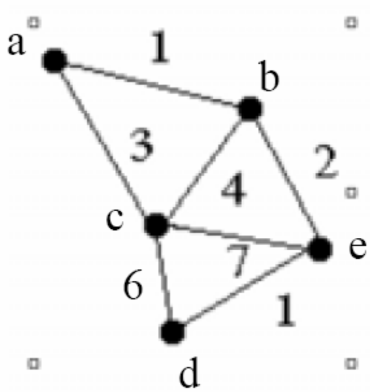
The blue data points were traversed by the windows towards the mode

Slide by Y. Ukrainitz & B. Sarel

Mean shift

- Labeling of data points: points visited by any window converging to same mode get the same label
- [Comaniciu & Meer, PAMI 2002] : If data point is visited by multiple diverging mean shift processes, “majority vote”

Weighted graph representation



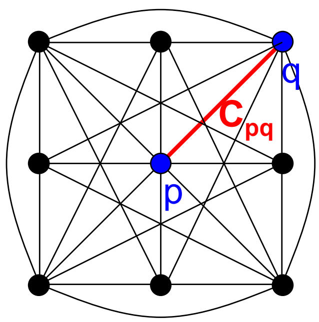
$$\begin{bmatrix} 0 & 1 & 3 & \infty & \infty \\ 1 & 0 & 4 & \infty & 2 \\ 3 & 4 & 0 & 6 & 7 \\ \infty & \infty & 6 & 0 & 1 \\ \infty & 2 & 7 & 1 & 0 \end{bmatrix}$$

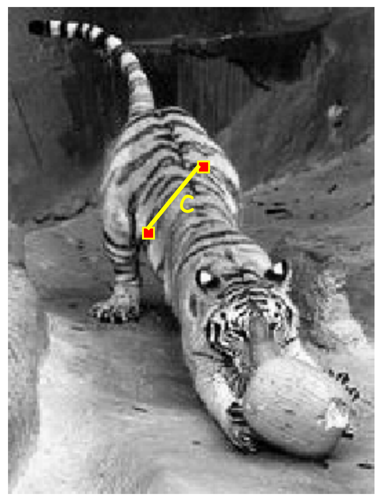
Weight Matrix

(“Affinity matrix”)

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Images as graphs



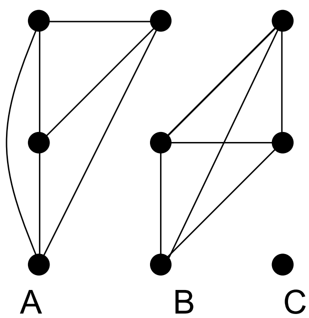


Fully-connected graph

- node for every pixel
- link between *every* pair of pixels, **p**,**q**
- similarity c_{pq} for each link
  - » similarity is *inversely proportional* to difference in color and position

Slide by Steve Seitz

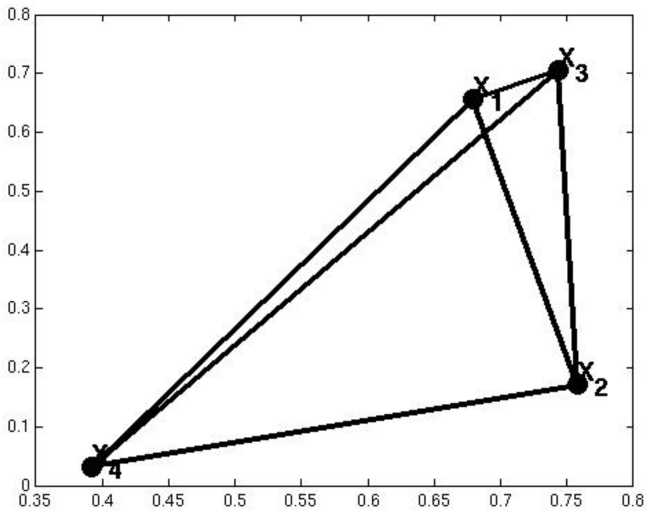
Segmentation by Graph Cuts



Break Graph into Segments

- Delete links that cross between segments
- Easiest to break links that have low similarity
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

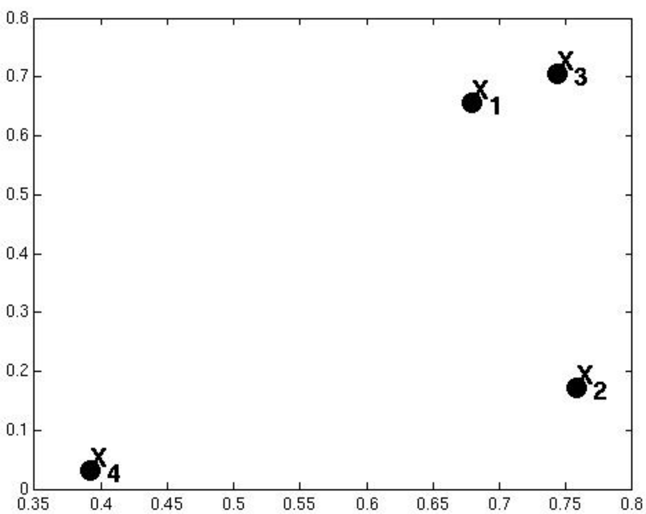
Example

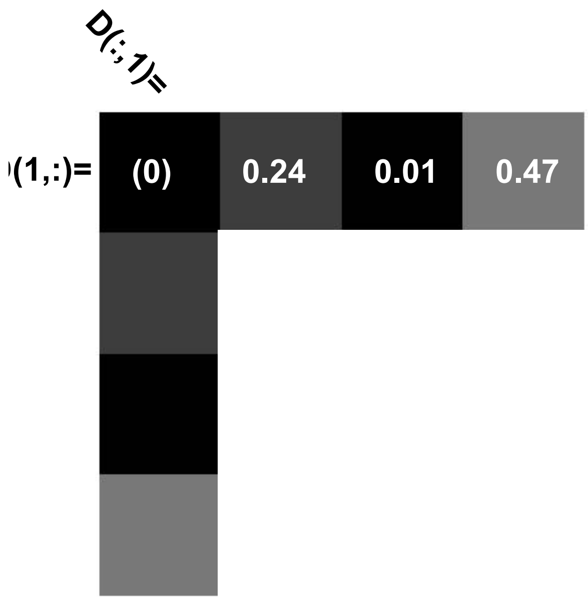


Dimension of points : d = 2

Number of points : N = 4

Distance matrix





for i=1:N

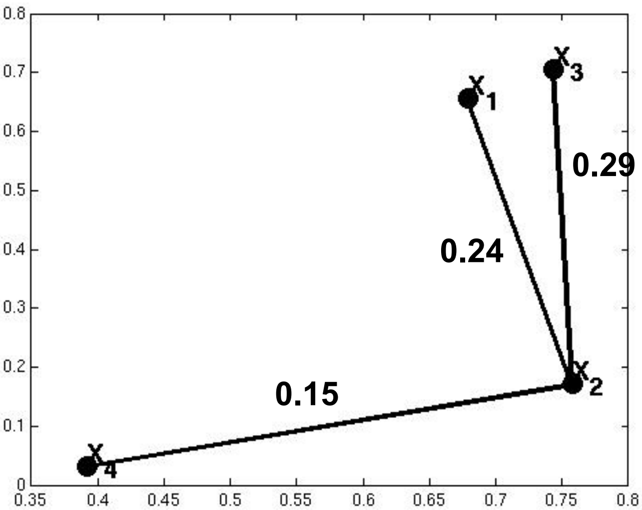
for j=1:N

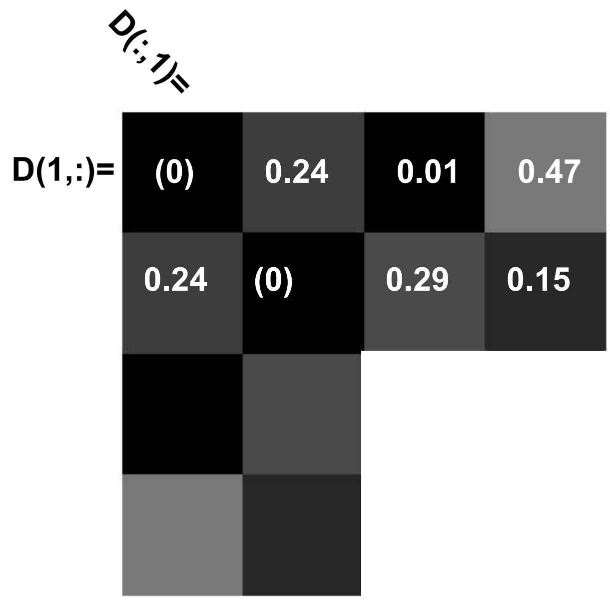
D(i,j) = $||x_i - x_j||^2$

end

end

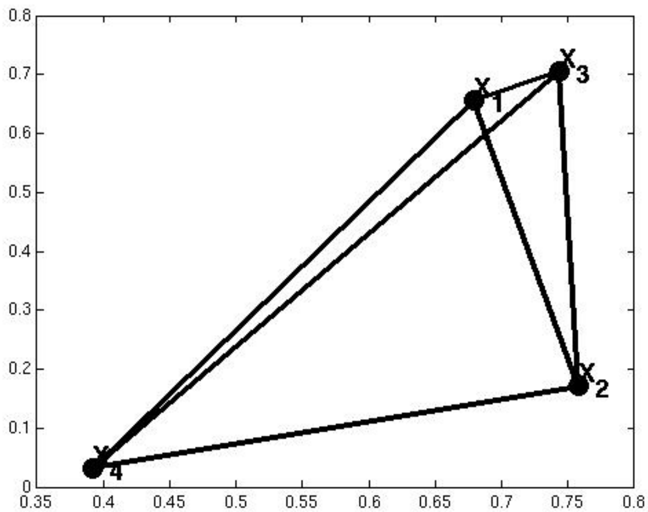
Distance matrix

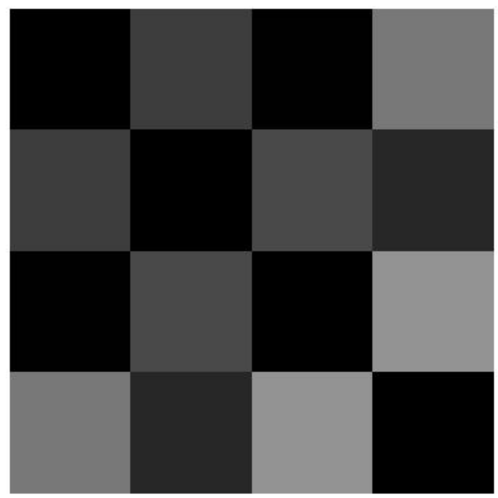




for i=1:N
  for j=1:N
    $D(i,j) = ||x_i - x_j||^2$
  end
end

Distance matrix





N x N matrix

for i=1:N

for j=1:N

D(i,j) = $||x_i - x_j||^2$

end

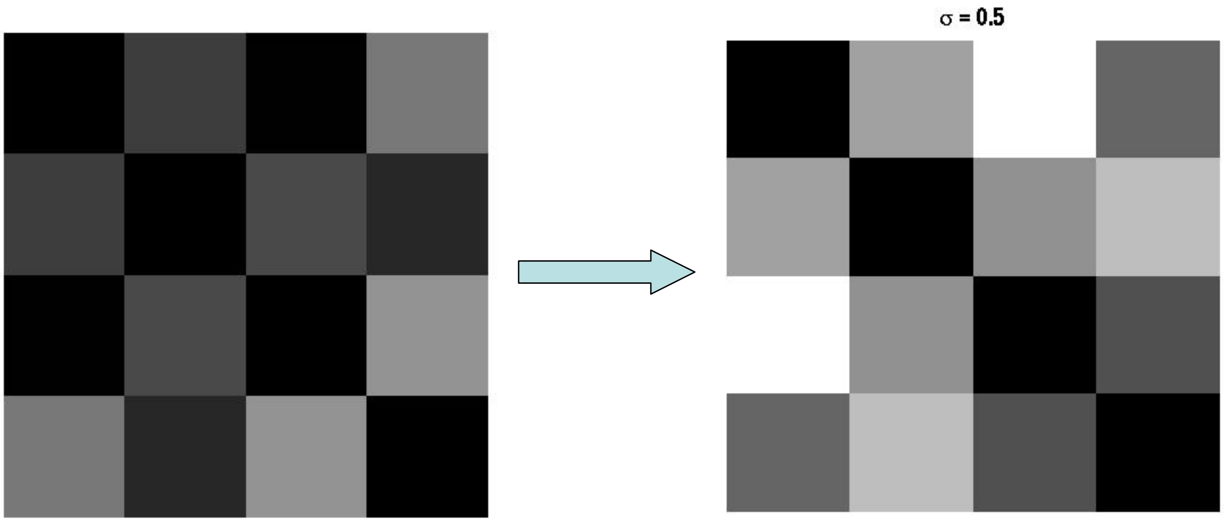
end

Measuring affinity

- One possibility: $aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)\left(\|x-y\|^2\right)\right\}$
- Map distances to similarities, use as edge weights on graph

Distances→affinities

D **A**




```
for i=1:N
  for j=1:N
    D(i,j) = ||x_i- x_j||^2
  end
end
```

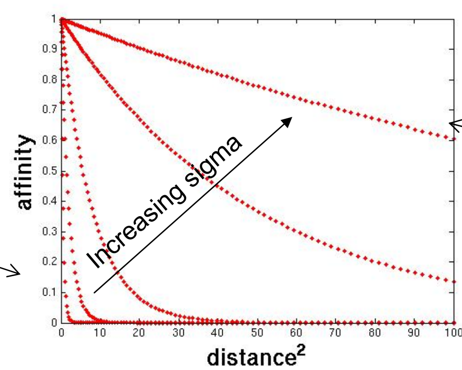
```
for i=1:N
  for j=1:N
    A(i,j) = exp(-1/(2*σ^2)*||x_i- x_j||^2)
  end
end
```

Measuring affinity

- One possibility: $aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)\left(\|x-y\|^2\right)\right\}$

- Essentially, affinity drops off after distance gets past some threshold

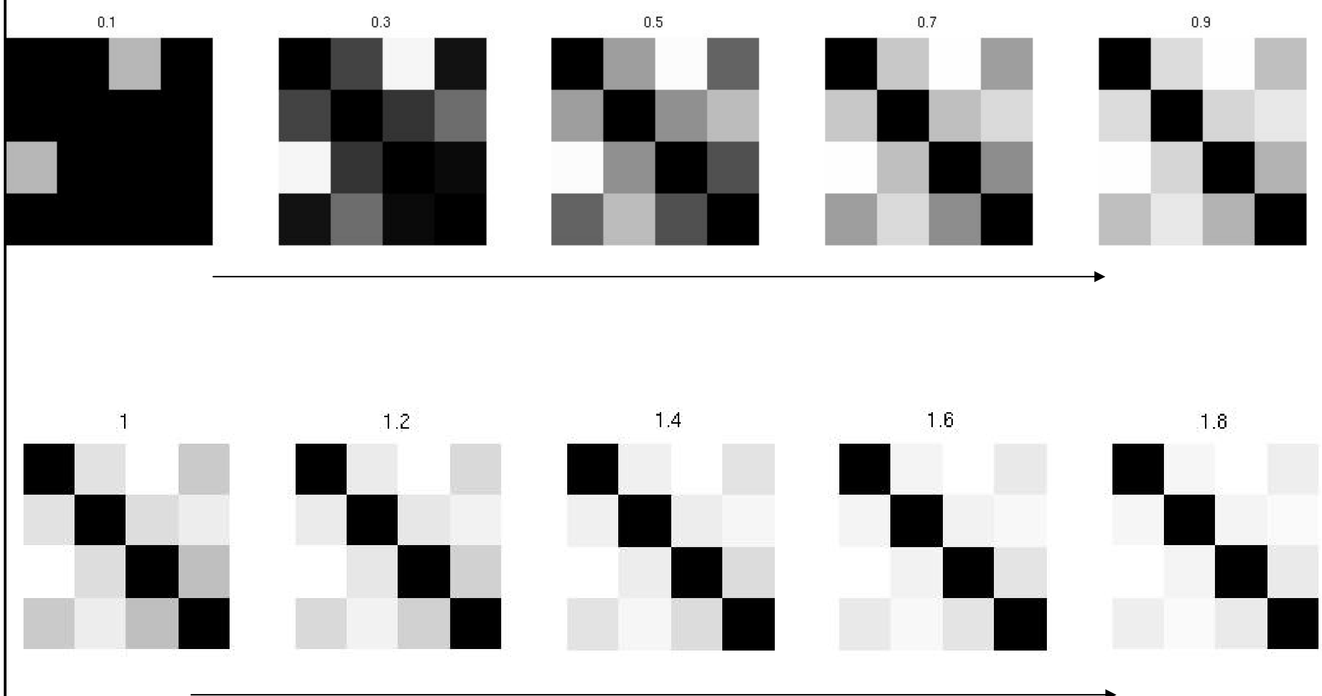
Small sigma: group only nearby points



Large sigma: group distant points

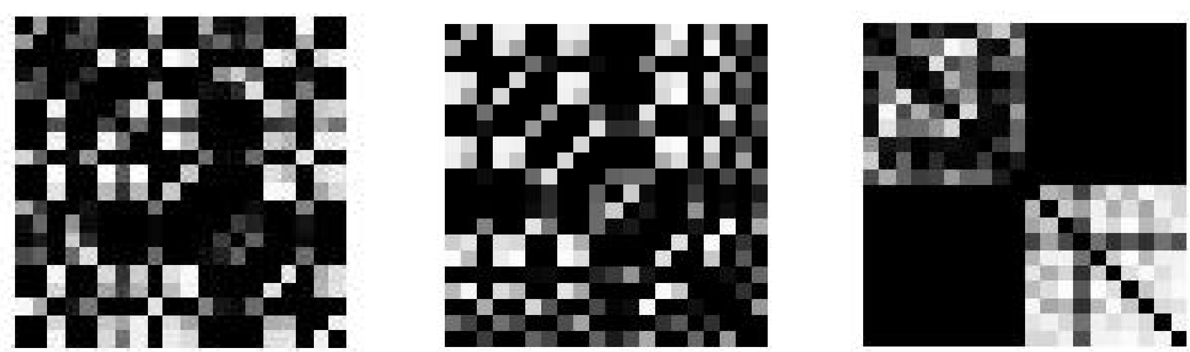
D= 

Scale affects affinity

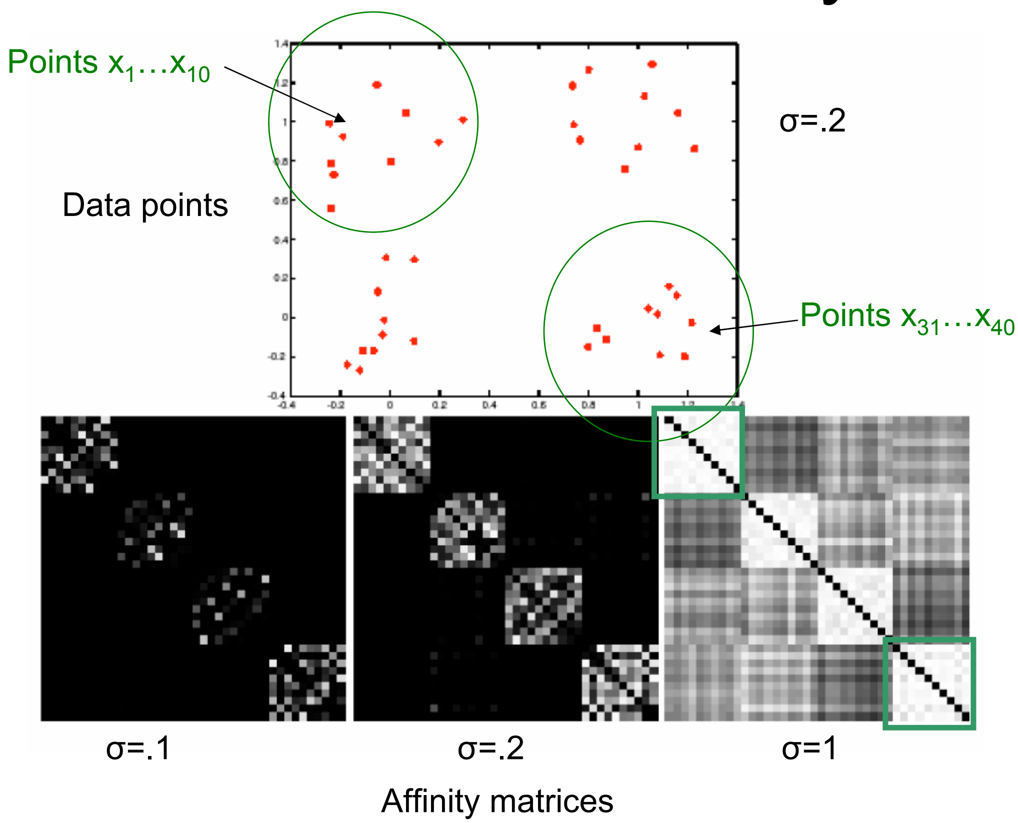


Shuffling the affinity matrix

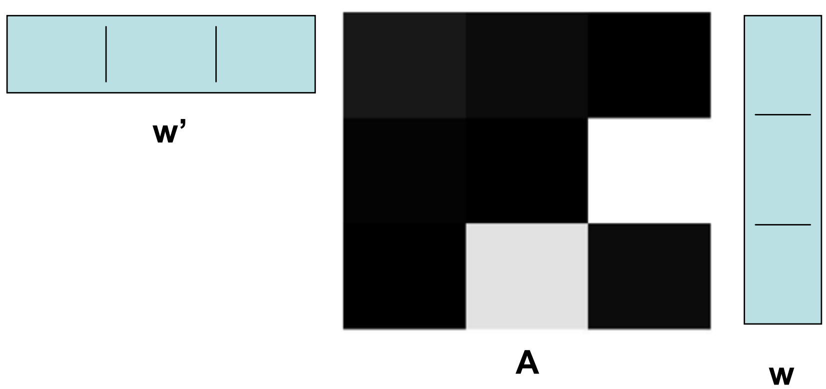
- Permute the order of the vertices, in terms of how they are associated with the matrix rows/cols



Scale affects affinity



Eigenvectors and graph cuts



$$w'Aw = \sum_i \sum_j w_i A_{ij} w_j$$

Eigenvectors and graph cuts

- Want a vector ***w*** giving the "association" between each element and a cluster
- Want elements within this cluster to have strong affinity with one another
- Maximize $\mathbf{w}^T A\, \mathbf{w}$

  subject to the constraint $\mathbf{w}^T\, \mathbf{w} = 1$

. . . Aw = λw

Eigenvalue problem : choose the eigenvector of A with largest eigenvalue

Rayleigh Quotient

- Given a symmetric matrix $\boldsymbol{A}$, find a vector $\boldsymbol{x}$ such that
- $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ is maximum AND
- $\|\boldsymbol{x}\|^2 = 1$

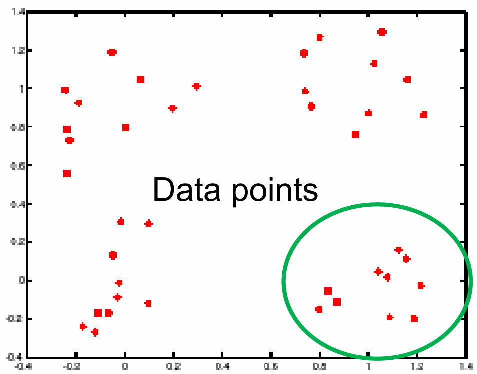
$$\updownarrow$$

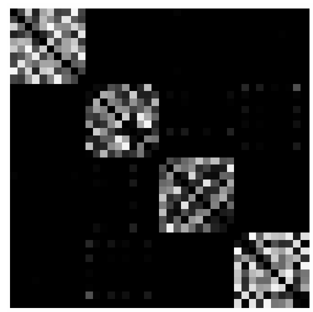
- Find $\boldsymbol{x}$ such that $\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$ is maximum.

The solution to this problem is given by the following theorem:

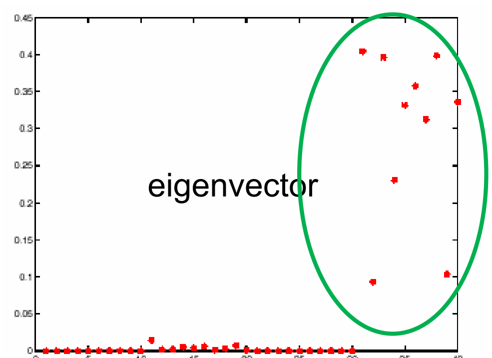
- $\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$ reaches its absolute maximum when $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$ corresponding to the *largest* eigenvalue λ_{max}.

Example





Affinity matrix



Eigenvectors and multiple cuts

- Use eigenvectors associated with k largest eigenvalues as cluster weights
- Or re-solve recursively

Scale affects affinity, number of clusters

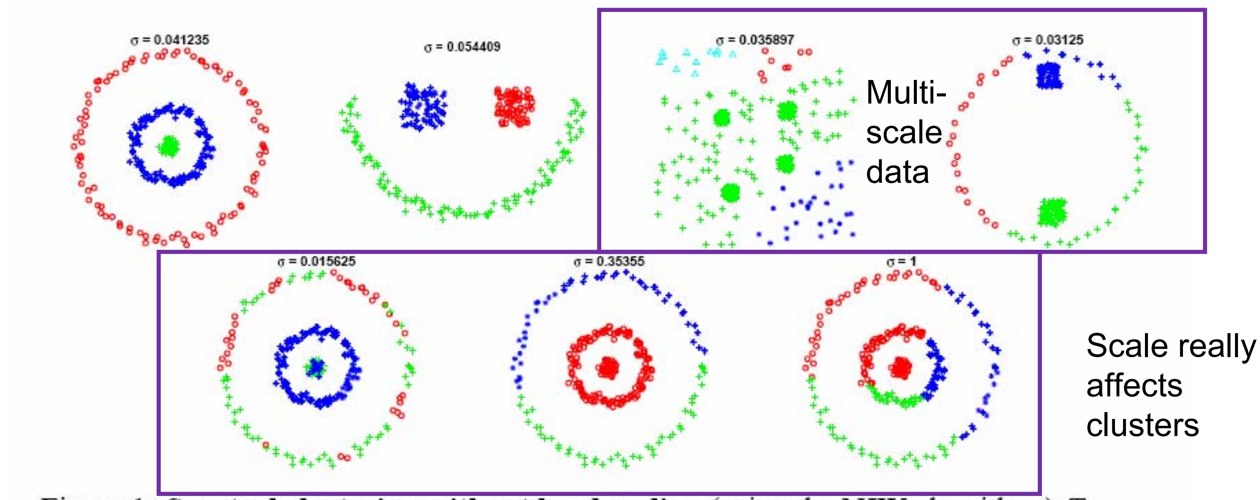


Figure 1: **Spectral clustering without local scaling** (using the NJW algorithm.) *Top row: When the data incorporates multiple scales standard spectral clustering fails. Note, that the optimal σ for each example (displayed on each figure) turned out to be different. Bottom row: Clustering results for the top-left point-set with different values of σ. This highlights the high impact σ has on the clustering quality. In all the examples, the number of groups was set manually. The data points were normalized to occupy the $[-1, 1]^2$ space.*

[Self-Tuning Spectral Clustering, L. Zelnik-Manor and P. Perona, NIPS 2004]

Local scale selection

- Possible solution: choose sigma per point

$$\hat{A}_{ij} = \exp\left(\frac{-d^2(s_i, s_j)}{\sigma_i \sigma_j}\right)$$

$$\sigma_i = d(s_i, s_K)$$

Distance to Kth neighbor for point s_i

[Self-Tuning Spectral Clustering, L. Zelnik-Manor and P. Perona, NIPS 2004]

Local scale selection

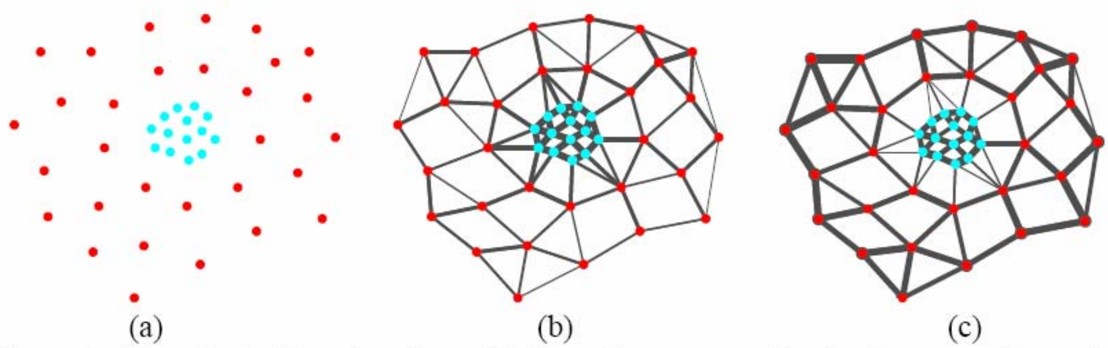
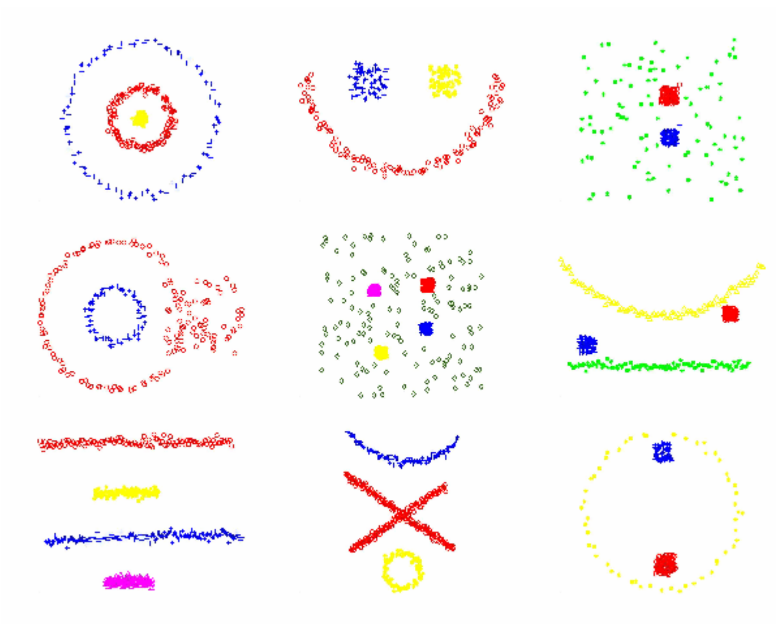


Figure 2: **The effect of local scaling.** *(a) Input data points. A tight cluster resides within a background cluster. (b) The affinity between each point and its surrounding neighbors is indicated by the thickness of the line connecting them. The affinities across clusters are larger than the affinities within the background cluster. (c) The corresponding visualization of affinities after local scaling. The affinities across clusters are now significantly lower than the affinities within any single cluster.*

[Self-Tuning Spectral Clustering, L. Zelnik-Manor and P. Perona, NIPS 2004]

Local scale selection, synthetic data



[Self-Tuning Spectral Clustering, L. Zelnik-Manor and P. Perona, NIPS 2004]

Local scale selection, image data

Image segmentation results, based on gray-scale differences alone.

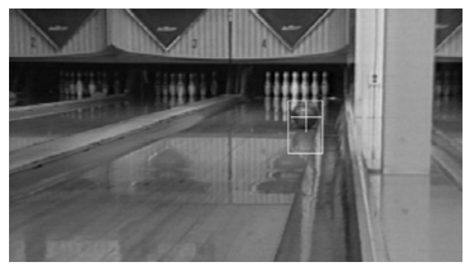
The number of clusters was set manually here to force a large number of clusters.

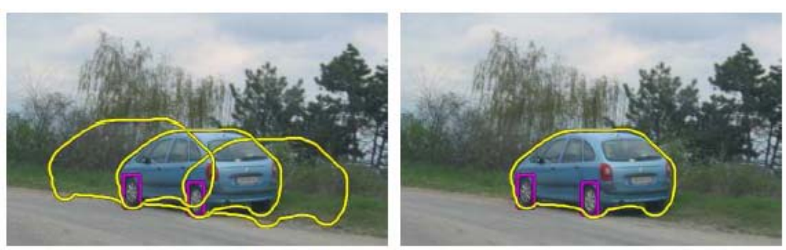
Since the scale is tuned locally for each pixel we obtained segments with both high and low contrast to the surrounding.

Zelnik-Manor & Perona, http://www.vision.caltech.edu/lihi/Demos/SelfTuningClustering.html

Fitting

- Want to associate a model with observed features





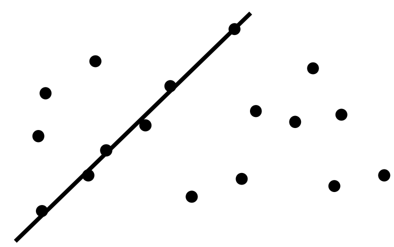
[Fig from Marszalek & Schmid, 2007]

Fitting lines

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

Difficulty of fitting lines

- Extraneous data: clutter, multiple models
- Missing data: only some parts of model are present
- Noise in the measured edge points, orientations
- **Cost:** infeasible to check all combinations of features by fitting a model to each possible subset

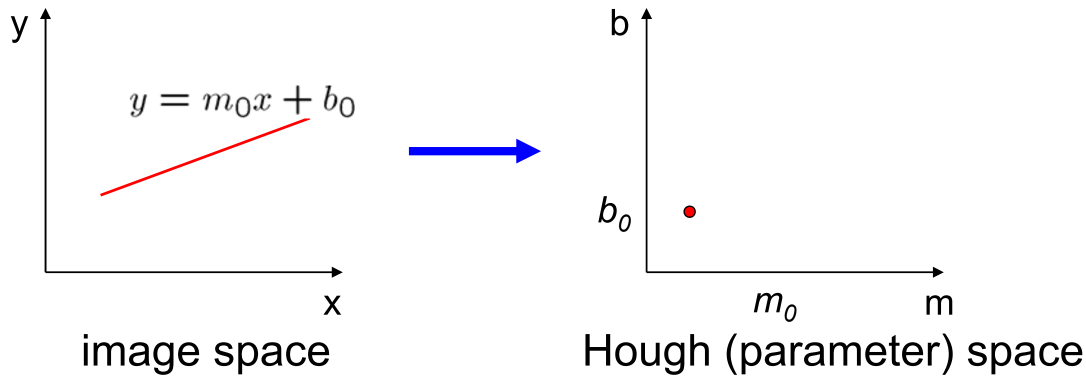


…Enter: Voting schemes

Hough transform

- Maps model (pattern) detection problem to simple peak detection problem
- Record all the structures on which each point lies, then look for structures that get many votes
- Useful for line fitting

Finding lines in an image

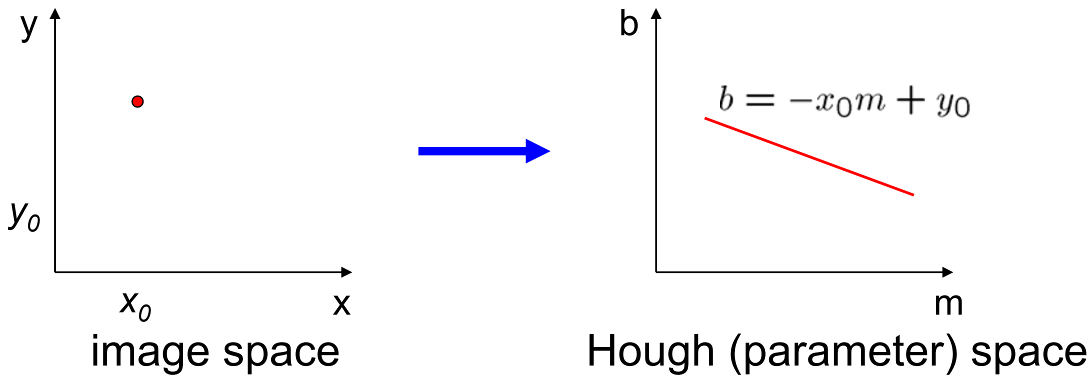


Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  – given a set of points (x,y), find all (m,b) such that y = mx + b

Slide credit: Steve Seitz

Finding lines in an image



Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - this is a line in Hough space

Finding lines in an image

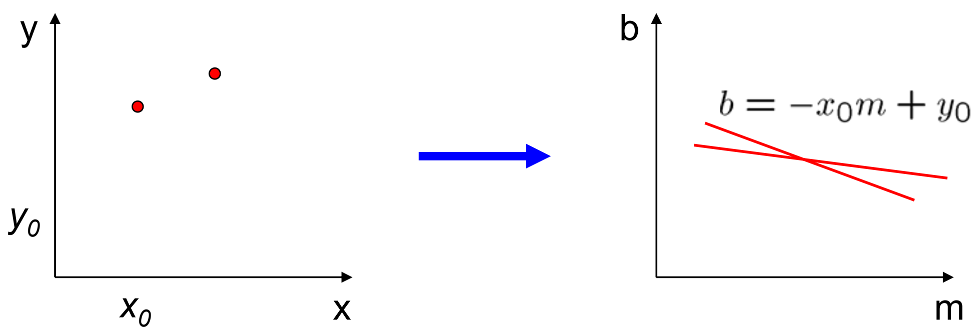


image space   Hough (parameter) space

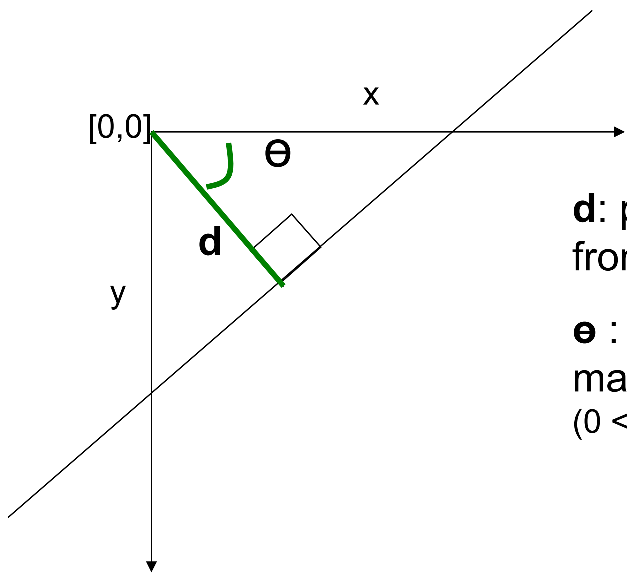
Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - Answer: the solutions of $b = -x_0m + y_0$
  - this is a line in Hough space

Polar representation for lines

- Issues with (*m*,*b*) parameter space:
  - Can take on infinite values
  - Undefined for vertical lines (x=constant)

Polar representation for lines



d: perpendicular distance from line to origin

ϴ : angle the perpendicular makes with the x-axis
(0 <= ϴ <= 2π)

$$x \cos \theta + y \sin \theta + d = 0$$

Point in image space → sinusoid segment in Hough space

Hough transform algorithm

Using the polar parameterization:

$$d = x\cos\theta + y\sin\theta$$

Basic Hough transform algorithm

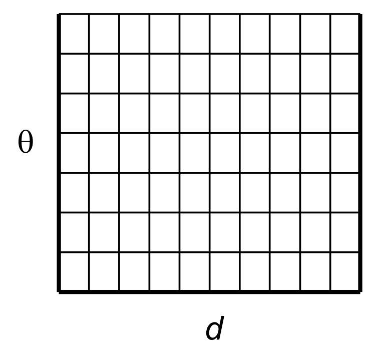
1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

   for θ = 0 to 180 // some quantization

   $d = x\cos\theta + y\sin\theta$

   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$

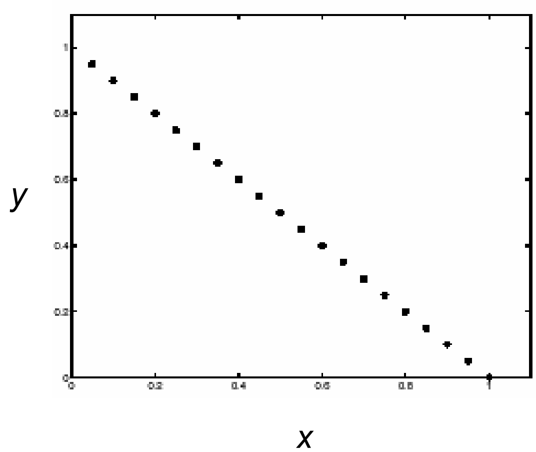
H: accumulator array (votes)



[Hough line demo]

Time complexity (in terms of number of votes)?

Example: Hough transform for straight lines



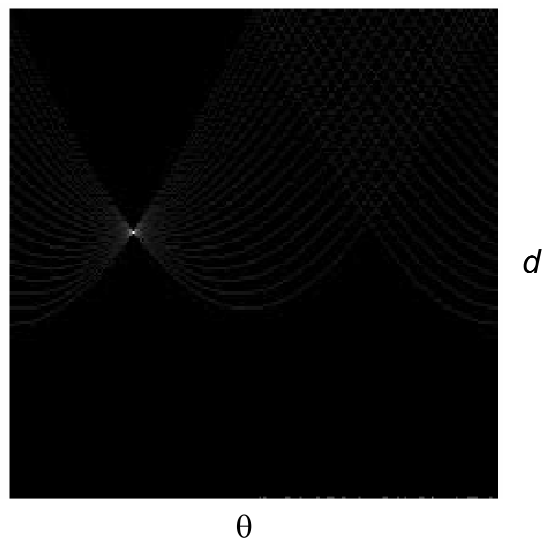
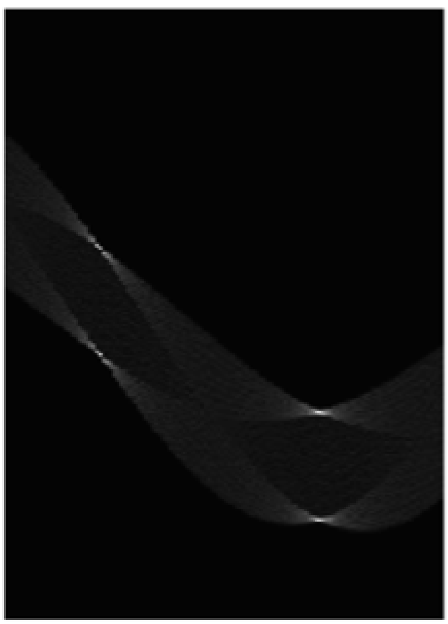


Image space edge coordinates

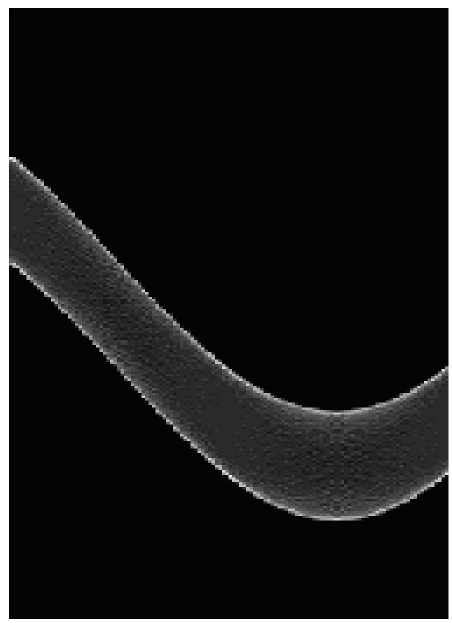
Votes

Example: Hough transform for straight lines

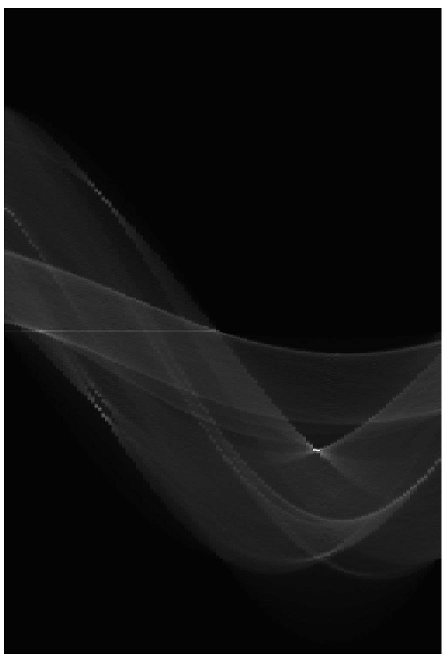
Square :

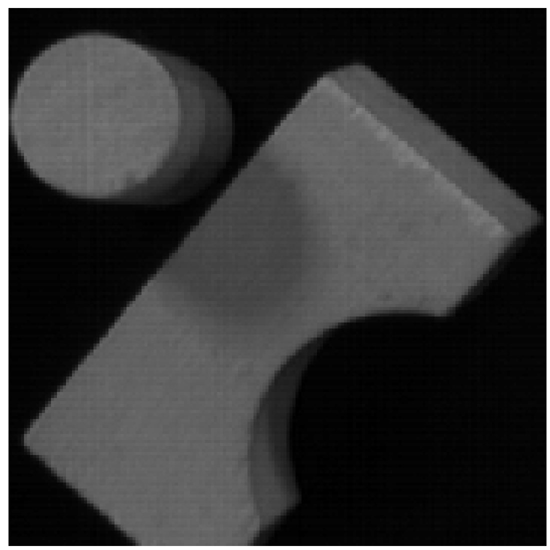


Circle :



Example: Hough transform for straight lines





Example with noise

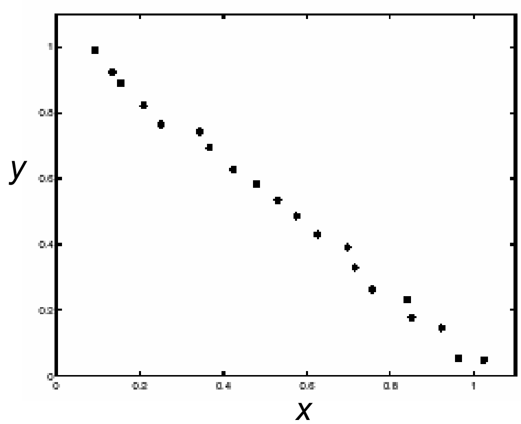
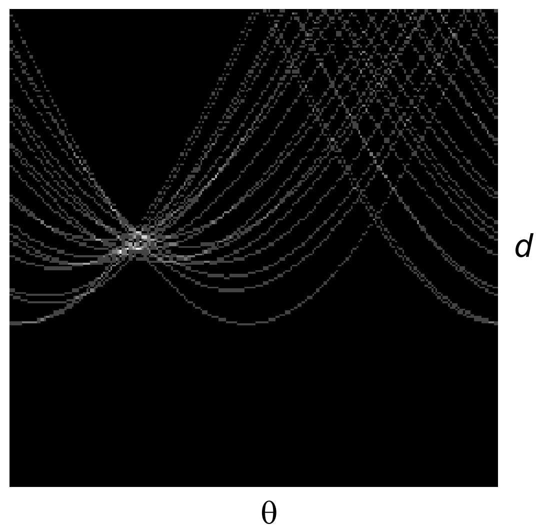
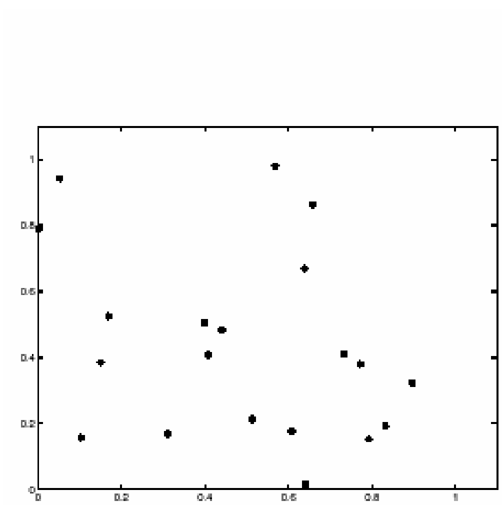


Image space
edge coordinates



Votes

Example with noise / random points



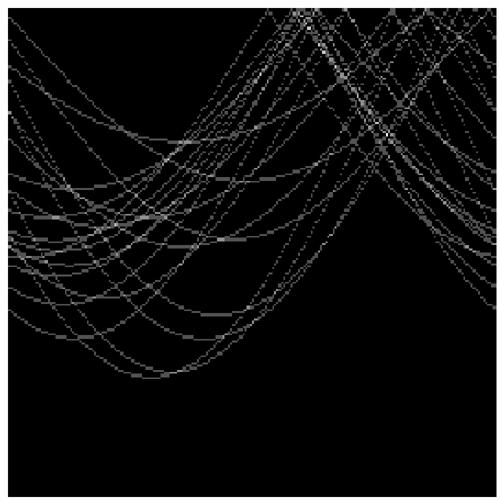


Image space edge coordinates

Votes

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image
   compute unique (d, θ) based on image gradient at (x,y)
   H[d, θ] += 1
3. same
4. same

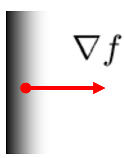
(Reduces degrees of freedom)

Recall: Image gradient

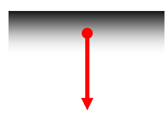
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

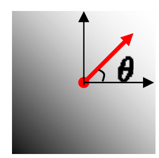
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient direction (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image
   compute unique (d, θ) based on image gradient at (x,y)
   $H[d, \theta] \mathrel{+}= 1$
3. same
4. same

(Reduces degrees of freedom)

Extension 2

- give more votes for stronger edges (use magnitude of gradient)

Extension 3

- change the sampling of (d, θ) to give more/less resolution

Extension 4

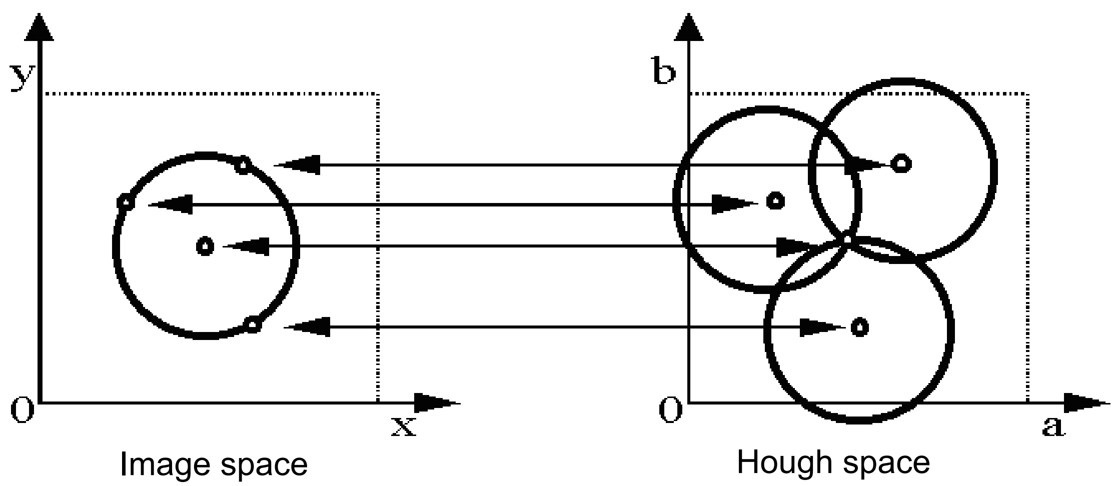
- The same procedure can be used with circles, squares, or any other shape…

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r, unknown gradient direction



Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For unknown radius r, unknown gradient direction

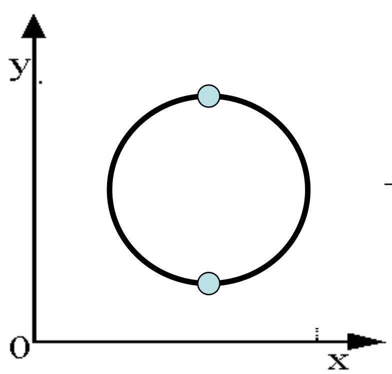
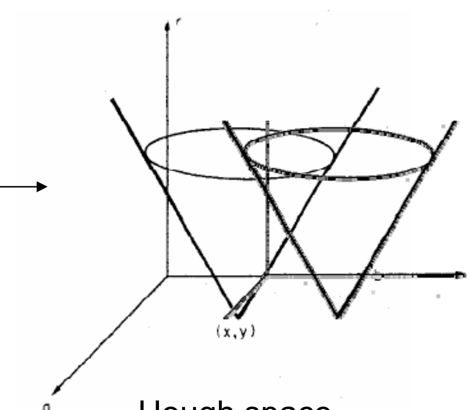


Image space



Hough space

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For unknown radius r, known gradient direction

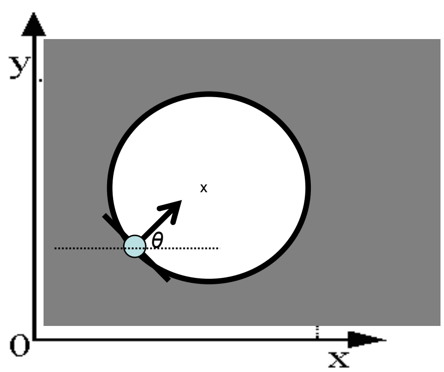
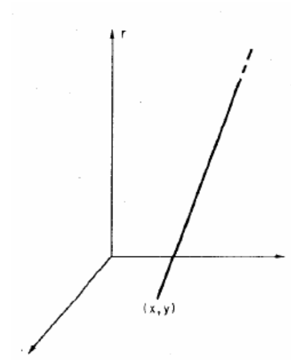


Image space



Hough space

Hough transform for circles

For every edge pixel (x,y) :

  For each possible radius value r:

    θ = gradient direction, from x,y to center

    $a = x - r\cos(\theta)$

    $b = y + r\sin(\theta)$

    $H[a,b,r]$ += 1

  end

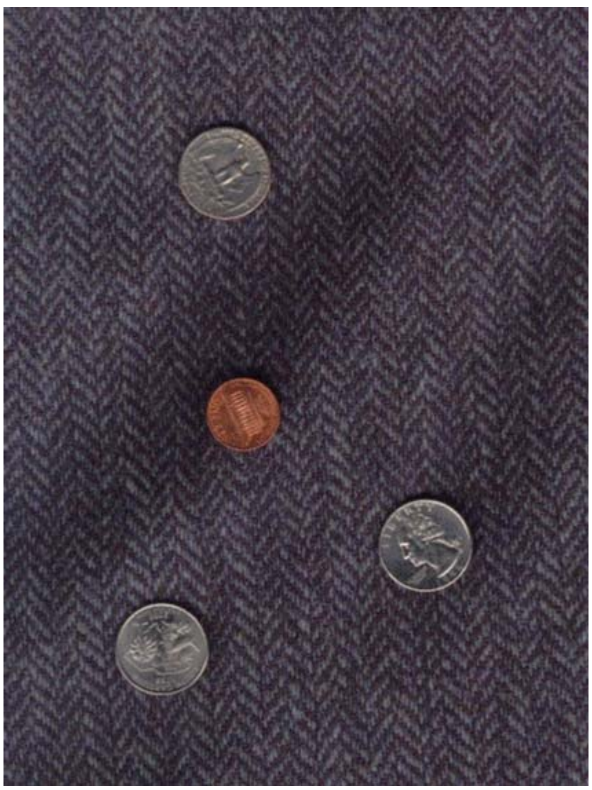
end

Real World Circle Examples

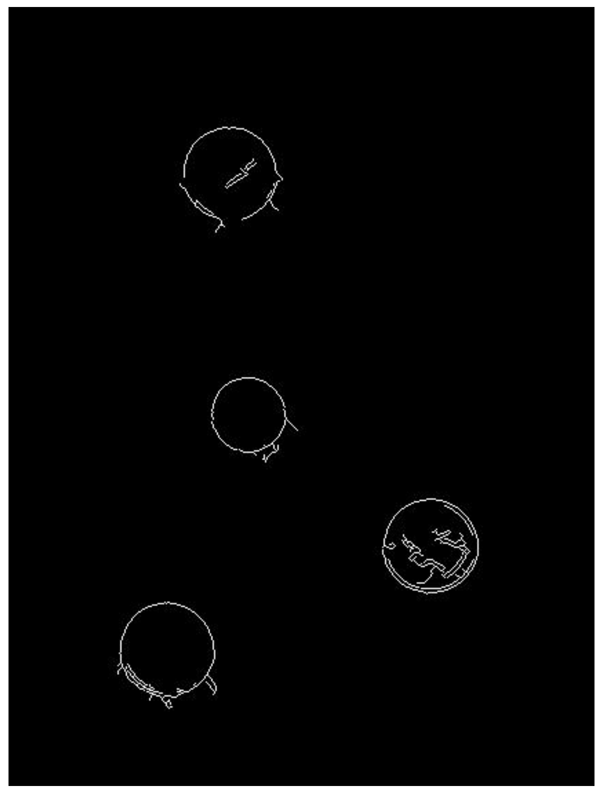
Crosshair indicates results of Hough transform, bounding box found via motion differencing.

Finding Coins

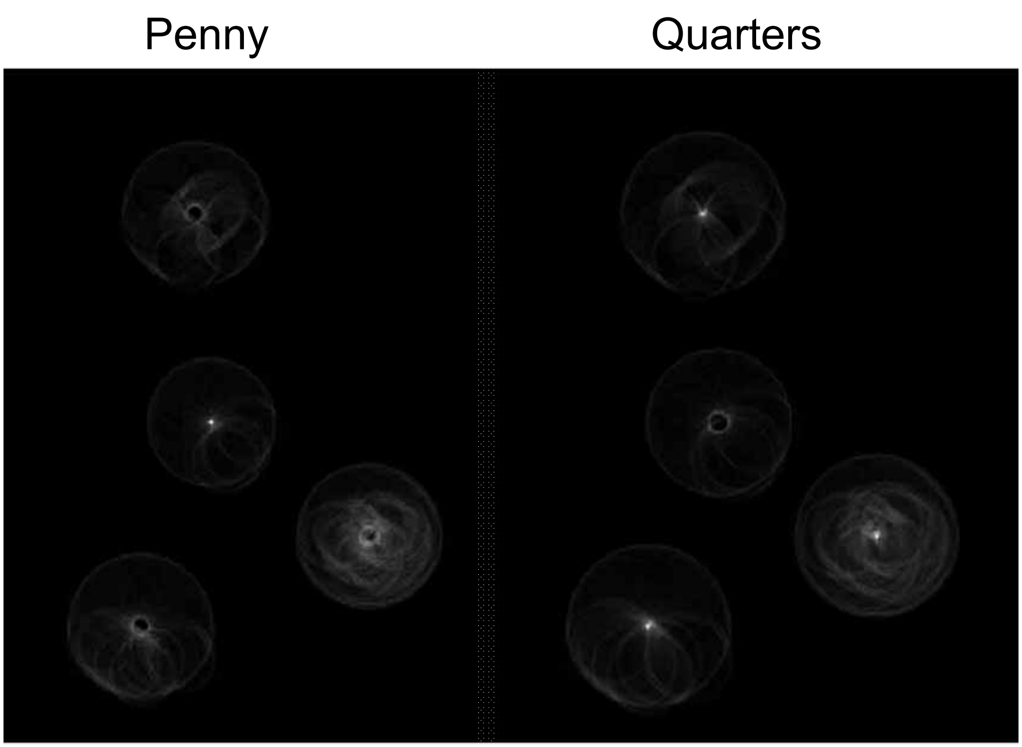
Original



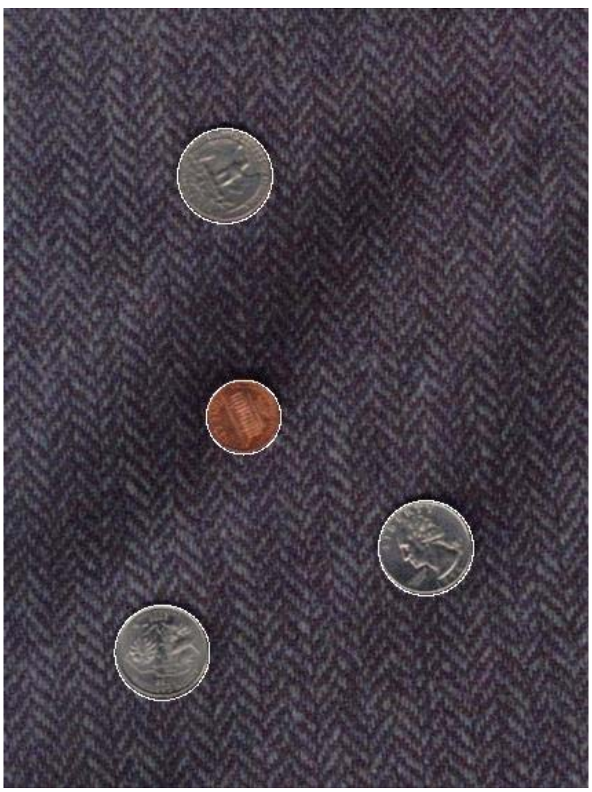
Edges (note noise)



Finding Coins



Finding Coins



Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Coin finding sample images from:
Vivek Kwatra

Hough transform for parameterized curves

For any curve with analytic form f(**x**,**a**) = 0, where

x : edge pixel in image coordinates
a : vector of parameters

1. Initialize accumulator array: H[**a**] = 0
2. For each edge pixel:
   determine each **a** such that f(**x**,**a**) = 0, and increment H[**a**]
3. Local maxima in H correspond to curves

Practical tips

- Minimize irrelevant tokens first (take edge points with significant gradient magnitude)
- Choose a good grid / discretization (trial and error)
- Vote for neighbors, also (smoothing in accumulator array)
- Utilize direction of edge to reduce free parameters by 1

Hough transform

- Pros
  - All points are processed independently, so can cope with occlusion
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
  - Can detect multiple instances of a model in a single pass

Hough transform

- Cons
  - Complexity of search time increases exponentially with the number of model parameters
  - Non-target shapes can produce spurious peaks in parameter space
  - Quantization: hard to pick a good grid size
    - Too coarse → large votes obtained when too many different lines correspond to a single bucket
    - Too fine → miss lines because some points that are not exactly collinear cast votes for different buckets

Generalized Hough transform

- What if we still know direction to some reference point (a), but allow arbitrary shapes defined by their boundary points?

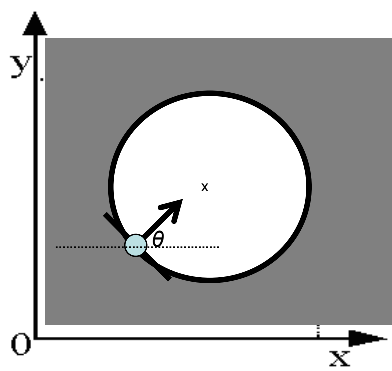


Image space

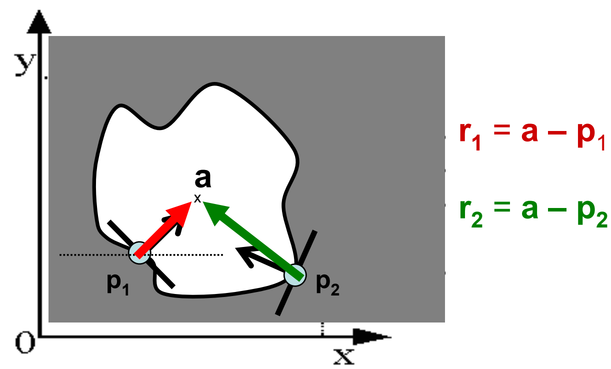


Image space

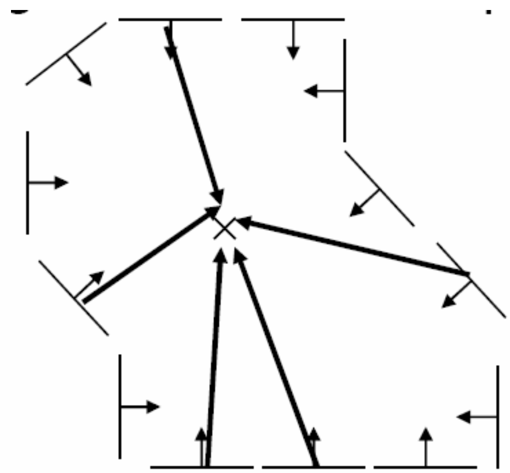
[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

Generalized Hough transform

- For model shape: construct a table storing these **r** vectors as function of gradient direction
- To detect model shape: for each edge point
  - Index into table with θ
  - Use indexed **r** vectors to vote for (x,y) position of reference point
- Peak in this Hough space is reference point with most supporting edges

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

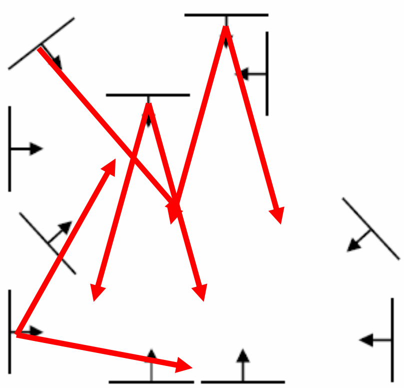
Generalized Hough transform



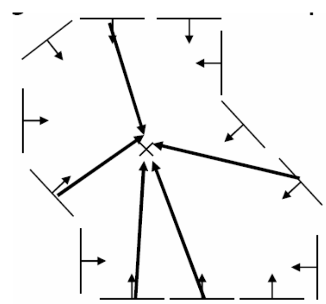
Model shape

i	ϕ_i	R_{ϕ_i}
0	0	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = 0\}$
1	$\Delta\phi$	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = \Delta\phi\}$
2	$2\,\Delta\phi$	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = 2\,\Delta\phi$
...	...	...

Generalized Hough transform



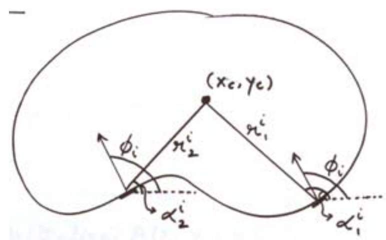
i	ϕ_i	R_{ϕ_i}
0	0	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = 0\}$
1	$\Delta\phi$	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = \Delta\phi\}$
2	$2\,\Delta\phi$	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = 2\,\Delta\phi$
...	...	...



New test shape : observed edges

Generalized Hough Transform

Find Object Center (x_c, y_c) given edges (x_i, y_i, ϕ_i)



Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$

For each edge point (x_i, y_i, ϕ_i)

For each **r** vector entry indexed in table, compute:

$$x_c = x_i + r_k^i \cos \alpha_k^i$$

$$y_c = y_i + r_k^i \sin \alpha_k^i$$

Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find peaks in $A(x_c, y_c)$

With modifications to table can generalize to add scale, orientation – increases size of accumulator array

Generalizing for scale, orientation

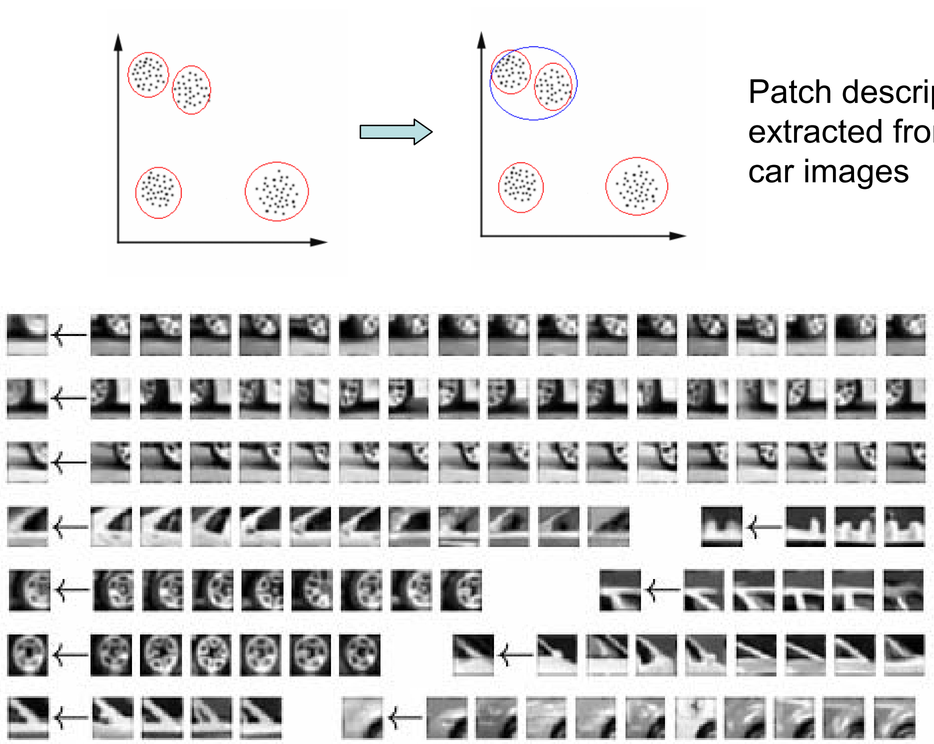
- To search for shapes at arbitrary scale and orientation
  - Add the parameters to the accumulator array (4d)
  - Update table

Example in recognition: implicit shape model

- Build 'codebook' of local appearance for each category using agglomerative clustering

[B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, 2004]

Codebook



Patch descriptors extracted from lots of car images

Example in recognition: implicit shape model

- Build ‘codebook’ of local appearance for each category using agglomerative clustering
- In all training images, match codebook entries to images with cross correlation (activate all entries with similarity > t)
- For each codebook entry, store all positions it was found, relative to object center

[B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, 2004]

Implicit shape model

- Given test image, extract patches, match to codebook entry (or entries)
- Cast votes for possible positions of object center
- Search for maxima in voting space using Mean-Shift
- (Extract weighted segmentation mask based on stored masks for the codebook occurrences)

[B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, 2004]

Implicit shape model

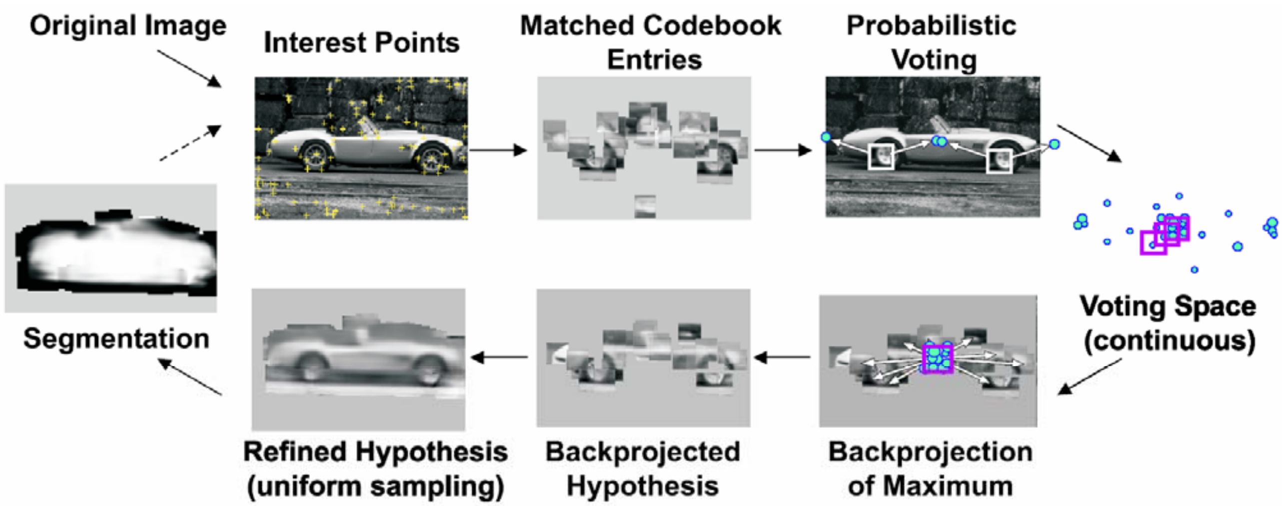


Fig. 1. *The recognition procedure. Image patches are extracted around interest points and compared to the codebook. Matching patches then cast probabilistic votes, which lead to object hypotheses that can later be refined. Based on the refined hypotheses, we compute a category-specific segmentation.*

[B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, 2004]

Grouping and fitting

- Grouping, segmentation: make a compact representation that merges similar features
  - Relevant algorithms: K-means, hierarchical clustering, Mean Shift, Graph cuts
- Fitting: fit a model to your observed features
  - Relevant algorithms: Hough transform for lines, circles (parameterized curves), generalized Hough transform for arbitrary boundaries; least squares; assigning points to lines incrementally or with k-means