Segmentation and low-level grouping.

Bill Freeman, MIT

6.869 April 14, 2005

Readings: Mean shift paper and background segmentation paper.

- Mean shift IEEE PAMI paper by Comanici and Meer, http://www.caip.rutgers.edu/~comanici/Papers/MsRobustApproach.pdf
- Forsyth&Ponce, Ch. 14, 15.1, 15.2.
- Wallflower: Principles and Practice of Background Maintenance, by Kentaro Toyama, John Krumm, Barry Brumitt, Brian Meyers. http://research.microsoft.com/users/jckrumm/Publications%202000/Wall%20Flower.pdf

The generic, unavoidable problem with low-level segmentation and grouping

- It makes a hard decision too soon. We want to think that simple low-level processing can identify high-level object boundaries, but any implementation reveals special cases where the low-level information is ambiguous.
- So we should learn the low-level grouping algorithms, but maintain ambiguity and pass along a selection of candidate groupings to higher processing levels.

Segmentation methods

- Segment foreground from background
- K-means clustering
- Mean-shift segmentation
- Normalized cuts

A simple segmentation technique: Background Subtraction

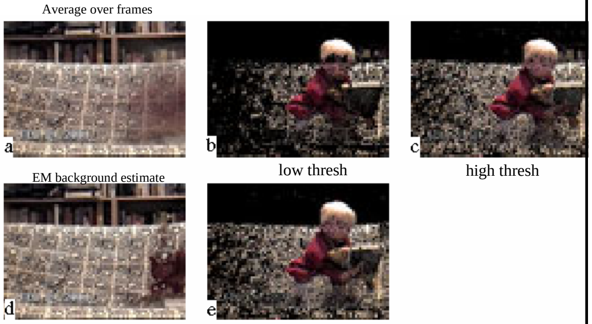
- If we know what the background looks like, it is easy to identify "interesting bits"
- Applications
  - Person in an office
  - Tracking cars on a road
  - surveillance
- Approach:
  - use a moving average to estimate background image
  - subtract from current frame
  - large absolute values are interesting pixels
    - trick: use morphological operations to clean up pixels

Movie frames from which we want to extract the foreground subject (the textbook author's child)

2 different background removal models

Background estimate | Foreground estimate | Foreground estimate

Average over frames



low thresh | high thresh

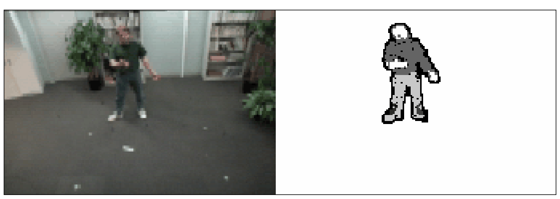
EM background estimate

EM

Static Background Modeling Examples

[MIT Media Lab Pfinder / ALIVE System]

Static Background Modeling Examples



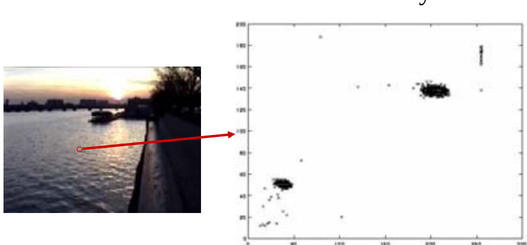
[MIT Media Lab Pfinder / ALIVE System]

Static Background Modeling Examples

[MIT Media Lab Pfinder / ALIVE System]

Dynamic Background

BG Pixel distribution is non-stationary:

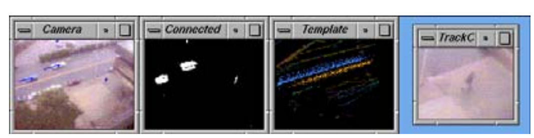


[MIT AI Lab VSAM]

Mixture of Gaussian BG model

Staufer and Grimson tracker:

Fit per-pixel mixture model to observed distrubution.



[MIT AI Lab VSAM]

Wallflower: Principles and Practice of Background Maintenance

Kentaro Toyama, John Krumm, Barry Brumitt, Brian Meyers
Microsoft Research
Redmond, WA 98052
{kentoy|jckrumm|barry|brianme}@microsoft.com

Abstract

Background maintenance *is a frequent element of video surveillance systems. We develop* Wallflower, *a three-component system for background maintenance: the pixel-level component performs Wiener filtering to make probabilistic predictions of the expected background; the region-level component fills in homogeneous regions of foreground objects; and the frame-level component detects sudden, global changes in the image and swaps in better approximations of the background.*

We compare our system with 8 other background subtraction algorithms. Wallflower is shown to outperform previous algorithms by handling a greater set of the difficult situations that can occur.

Finally, we analyze the experimental results and propose normative principles for background maintenance.

1. Introduction

Video surveillance systems seek to automatically

Bootstrapping: A training period absent of foreground objects is not available in some environments.
Foreground aperture: When a homogeneously colored object moves, change in the interior pixels cannot be detected. Thus, the entire object may not appear as foreground.
Sleeping person: A foreground object that becomes motionless cannot be distinguished from a background object that moves and then becomes motionless.
Waking person: When an object initially in the background moves, both it and the newly revealed parts of the background appear to change.
Shadows: Foreground objects often cast shadows which appear different from the modeled background.

No perfect system exists. In this paper, we hope to further understanding of background maintenance through a threefold contribution: In the next section, we describe *Wallflower*, a background maintenance algorithm that attempts to address many of the problems enumerated.

http://research.microsoft.com/users/toyama/wallflower.pd

Background removal issues

modeling process *background maintenance*. An ideal background maintenance system would be able to avoid the following problems:

Moved objects: A background object can be moved. These objects should not be considered part of the foreground forever after.
Time of day: Gradual illumination changes alter the appearance of the background.
Light switch: Sudden changes in illumination and other scene parameters alter the appearance of the background.
Waving trees: Backgrounds can vacillate, requiring models which can represent disjoint sets of pixel values.
Camouflage: A foreground object's pixel characteristics may be subsumed by the modeled background.

Bootstrapping: A training period absent of foreground objects is not available in some environments.
Foreground aperture: When a homogeneously colored object moves, change in the interior pixels cannot be detected. Thus, the entire object may not appear as foreground.
Sleeping person: A foreground object that becomes motionless cannot be distinguished from a background object that moves and then becomes motionless.
Waking person: When an object initially in the background moves, both it and the newly revealed parts of the background appear to change.
Shadows: Foreground objects often cast shadows which appear different from the modeled background.

No perfect system exists. In this paper, we hope to further understanding of background maintenance through a threefold contribution: In the next section, we describe *Wallflower*, a background maintenance algorithm that attempts to address many of the problems enumerated.

http://research.microsoft.com/users/toyama/wallflower.pd

Background Subtraction Principles

Wallflower: Principles and Practice of Background Maintenance, by Kentaro Toyama, John Krumm, Barry Brumitt, Brian Meyers.

P1: Semantic differentiation of objects should not be handled by the background maintenance module.

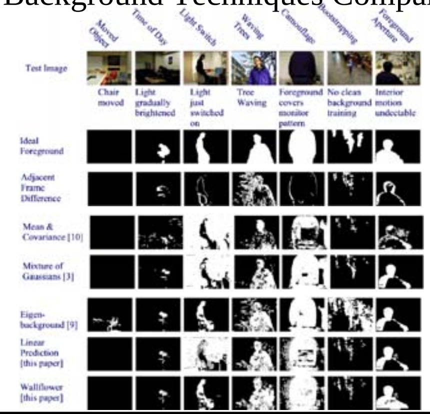
P2: Background subtraction should segment objects of interest when they first appear (or reappear) in a scene.

P3: An appropriate pixel-level stationarity criterion should be defined. Pixels that satisfy this criterion are declared background and ignored.

P4: The background model must adapt to both sudden and gradual changes in the background.

P5: Background models should take into account changes at differing spatial scales.

Background Techniques Compared



From the Wallflower Paper

Segmentation as clustering

- Cluster together (pixels, tokens, etc.) that belong together…
- Agglomerative clustering
  - attach closest to cluster it is closest to
  - repeat
- Divisive clustering
  - split cluster along best boundary
  - repeat
- Dendrograms
  - yield a picture of output as clustering process continues

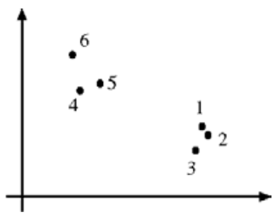
Greedy Clustering Algorithms

Algorithm 15.3: Agglomerative clustering, or clustering by merging

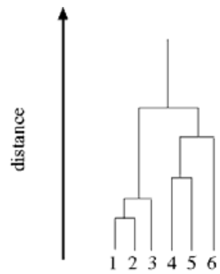
```
Make each point a separate cluster
Until the clustering is satisfactory
      Merge the two clusters with the
         smallest inter-cluster distance
end
```

Algorithm 15.4: Divisive clustering, or clustering by splitting

```
Construct a single cluster containing all points
Until the clustering is satisfactory
      Split the cluster that yields the two
         components with the largest inter-cluster distance
end
```



Data set



Dendrogram formed by agglomerative clustering using single-link clustering.

Segmentation methods

- Segment foreground from background
- K-means clustering
- Mean-shift segmentation
- Normalized cuts

K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error
- can't do this by search, because there are too many possible allocations.
- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

K-Means

Algorithm 15.5: Clustering by K-Means

```
Choose k data points to act as cluster centers
Until the cluster centers are unchanged
    Allocate each data point to cluster whose center is nearest
    Now ensure that every cluster has at least
      one data point; possible techniques for doing this include .
      supplying empty clusters with a point chosen at random from
      points far from their cluster center.
    Replace the cluster centers with the mean of the elements
      in their clusters.
end
```

Matlab k-means clustering demo

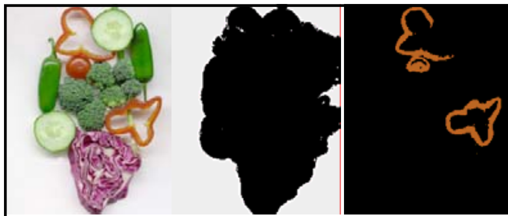
Image | Clusters on intensity (K=5) | Clusters on color (K=5)

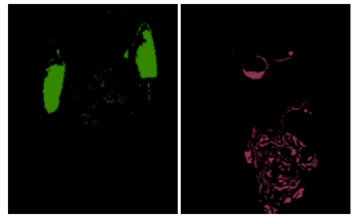
K-means clustering using intensity alone and color alone

Image

Clusters on color

K-means using color alone, 11 segments





K-means using
color alone,
11 segments.

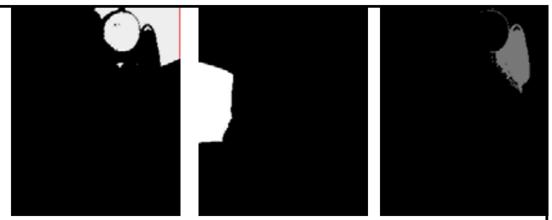
Color alone often will not yeild salient segments!

Ways to include spatial relationships

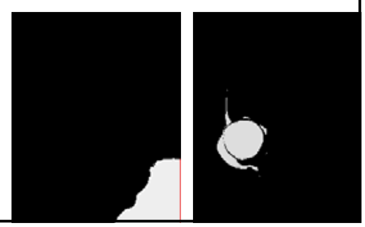
(a) Define a Markov Random Field (MRF), where the state to be estimated includes the segment index. Solve by graph cuts or BP.

(b) Augment data to be clustered with spatial coordinates.

$$z = \begin{pmatrix} Y \\ u \\ v \\ x \\ y \end{pmatrix} \begin{array}{l} \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{color coordinates} \\ \left.\begin{array}{l} \\ \end{array}\right\} \text{spatial coordinates} \end{array}$$



K-means using colour and position, 20 segments



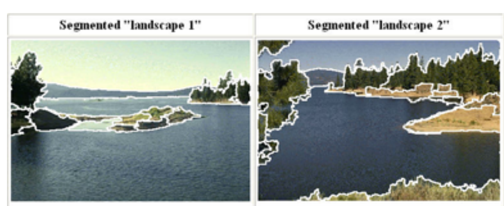
Still misses goal of perceptually pleasing segmentation!

Hard to pick K…

Segmentation methods

- Segment foreground from background
- K-means clustering
- Mean-shift segmentation
- Normalized cuts

Mean Shift Segmentation



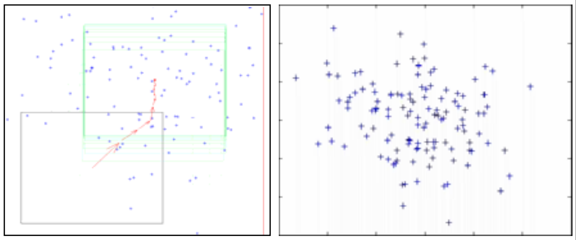
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean Shift Algorithm

Mean Shift Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the "mode" or point of highest density of a data distribution:



Mean Shift Segmentation

Mean Shift Segmentation Algorithm

1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same "peak" or mode.
5. The data these merged windows traversed are clustered together.

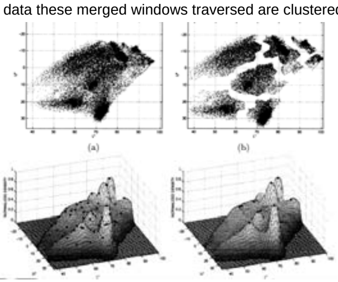


*Image From: Dorin Comaniciu and Peter Meer, Distribution Free Decomposition of Multivariate Data, Pattern Analysis & Applications (1999)2:22–30

- For your homework, you will do a mean shift algorithm just in the color domain. In the slides that follow, however, both spatial and color information are used in a mean shift segmentation.

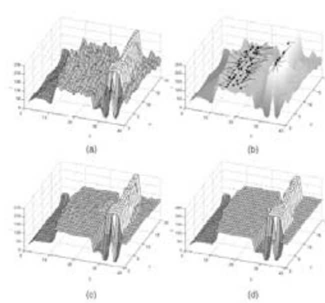
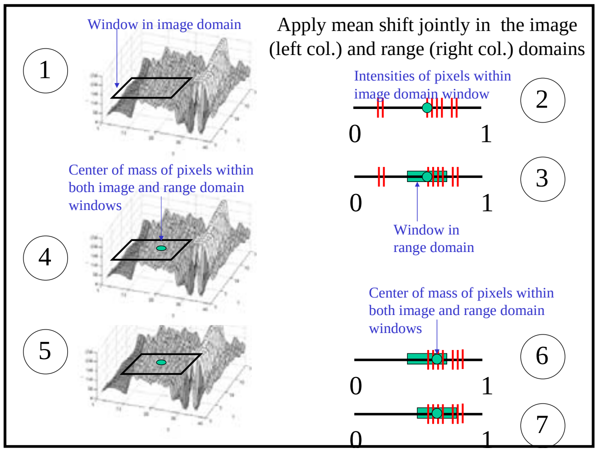


Fig. 4. Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixels on the plateau and on the line. The black dots are the points of convergence. (c) Filtering result $(h_s, h_r) = (8, 4)$. (d) Segmentation result.

Comaniciu and Meer, IEEE PAMI vol. 24, no. 5, 2002

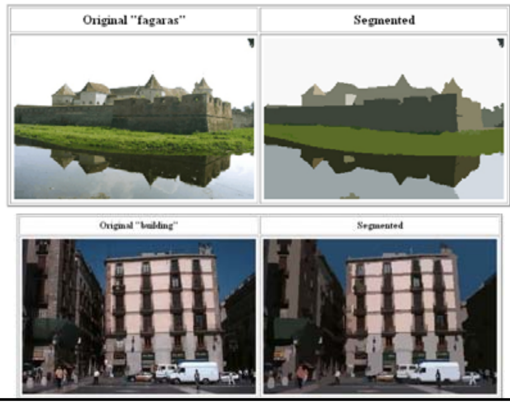
Apply mean shift jointly in the image (left col.) and range (right col.) domains



Mean Shift color&spatial Segmentation Results:

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean Shift color&spatial Segmentation Results:

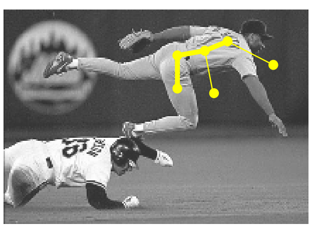


Segmentation methods

- Segment foreground from background
- K-means clustering
- Mean-shift segmentation
- Normalized cuts

Graph-Theoretic Image Segmentation

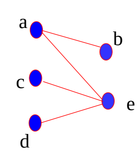
Build a weighted graph G=(V,E) from image



V: image pixels

E: connections between pairs of nearby pixels

Graphs Representations

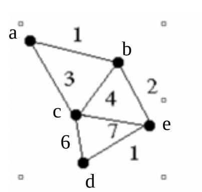


$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Weighted Graphs and Their Representations



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

Weight Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Boundaries of image regions defined by a number of attributes

- Brightness/color
- Texture
- Motion
- Stereoscopic depth
- Familiar configuration

[Malik]

Measuring Affinity

Intensity

$$aff(x,y) = \exp\left\{ -\left(\frac{1}{2\sigma_i^2}\right)\left(\left\| I(x) - I(y) \right\|^2\right) \right\}$$

Distance

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

Color

$$aff(x,y) = \exp\left\{ -\left(\frac{1}{2\sigma_t^2}\right)\left(\left\| c(x) - c(y) \right\|^2\right) \right\}$$

Eigenvectors and affinity clusters

- Simplest idea: we want a vector a giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize

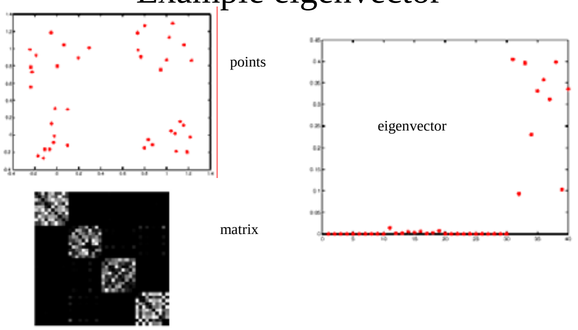
$$a^T A a$$

- But need the constraint

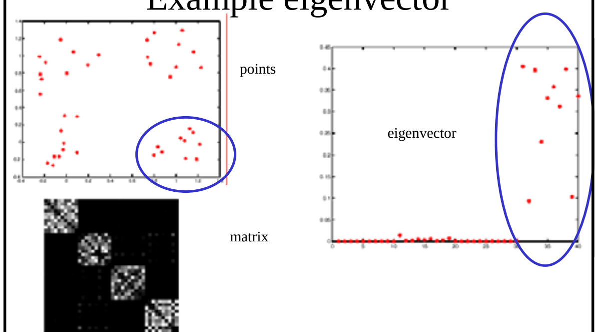
$$a^T a = 1$$

- This is an eigenvalue problem (p. 321 of Forsyth&Ponce)
- - choose the eigenvector of A with largest eigenvalue

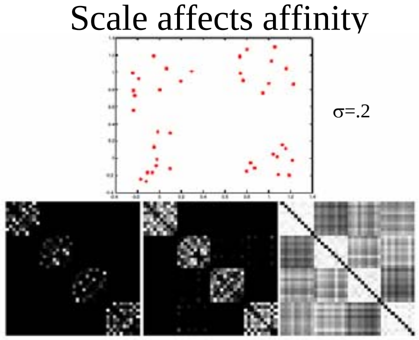
Example eigenvector



Example eigenvector



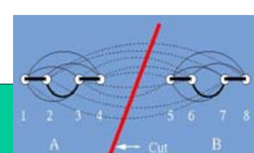
Scale affects affinity



σ=.1 σ=.2 σ=1

Some Terminology for Graph Partitioning

- How do we bipartition a graph:



$$cut(\mathrm{A},\mathrm{B}) = \sum_{u\in \mathrm{A}, v\in \mathrm{B}} \mathrm{W}(u,v),$$

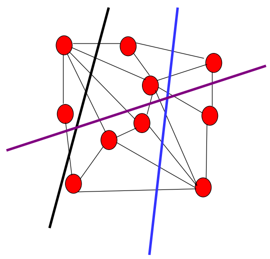
with $\mathrm{A} \cap \mathrm{B} = \emptyset$

$$assoc(\mathrm{A},\mathrm{A}') = \sum_{u\in \mathrm{A}, v\in \mathrm{A}'} \mathrm{W}(u,v)$$

A and A' not necessarily disjoint

[Malik]

Minimum Cut



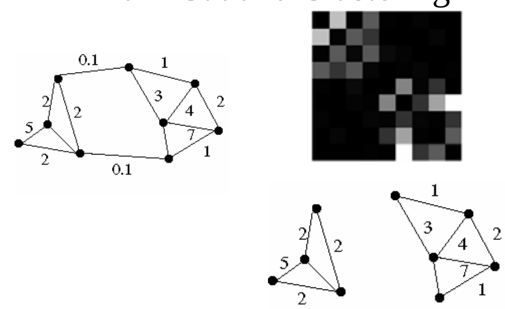
A cut of a graph *G* is the set of edges *S* such that removal of *S* from *G* disconnects *G*.

Minimum cut is the cut of minimum weight, where weight of cut <A,B> is given as

$$w(\langle A,B\rangle) = \sum_{x\in A, y\in B} w(x,y)$$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

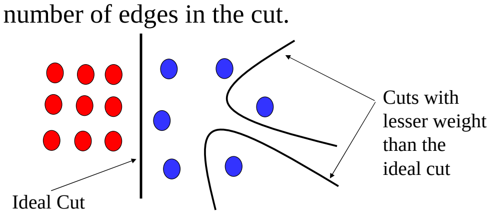
Minimum Cut and Clustering



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Drawbacks of Minimum Cut

- Weight of cut is directly proportional to the number of edges in the cut.



* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Normalized cuts

- First eigenvector of affinity matrix captures within cluster similarity, but not across cluster difference
- Min-cut can find degenerate clusters
- Instead, we'd like to maximize the within cluster similarity compared to the across cluster difference
- Write graph as V, one cluster as A and the other as B

- Minimize

$$\frac{\text{cut(A,B)}}{\text{assoc(A,V)}} + \frac{\text{cut(A,B)}}{\text{assoc(B,V)}}$$

where cut(A,B) is sum of weights with one end in A and one end in B; assoc(A,V) is sum of all edges with one end in A.

I.e. construct A, B such that their within cluster similarity is high compared to their association with the rest of the graph

Solving the Normalized Cut problem

- Exact discrete solution to Ncut is NP-complete even on regular grid,
  - [Papadimitriou'97]
- Drawing on spectral graph theory, good approximation can be obtained by solving a generalized eigenvalue problem.

[Malik]

Normalized Cut As Generalized Eigenvalue problem

$$Ncut(\text{A,B}) = \frac{cut(\text{A,B})}{assoc(\text{A,V})} + \frac{cut(\text{A,B})}{assoc(\text{B,V})} \qquad D_{ii} = \sum_j A_{ij}$$

$$= \frac{(1+x)^T(D-W)(1+x)}{k1^T D1} + \frac{(1-x)^T(D-W)(1-x)}{(1-k)1^T D1};\ k = \frac{\sum_{x_i>0} D(i,i)}{\sum D(i,i)}$$

$$= \ldots$$

after simplification, Shi and Malik derive

$$Ncut(A,B) = \frac{y^T(D-W)y}{y^T Dy}, \quad \text{with } y_i \in \{1,-b\},\ y^T D1 = 0.$$

[Malik]

Normalized cuts

- Instead, solve the generalized eigenvalue problem

$$\max_y \left(y^T(D-W)y\right) \text{ subject to } \left(y^T D y = 1\right)$$

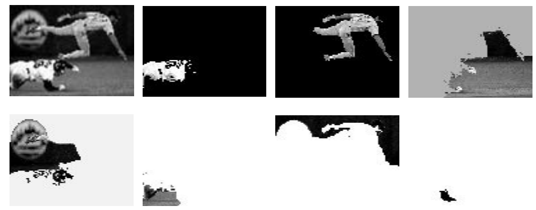
- which gives

$$(D-W)y = \lambda D y$$

- They show that the 2nd smallest eigenvector solution y is a good real-valued appox to the original normalized cuts problem. Then you look for a quantization threshold that maximizes the criterion --- i.e all components of y above that threshold go to one, all below go to -b

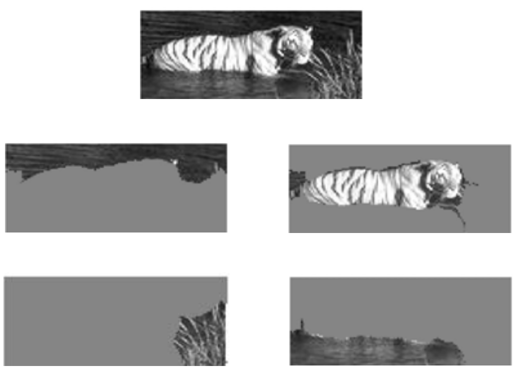
http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Brightness Image Segmentation

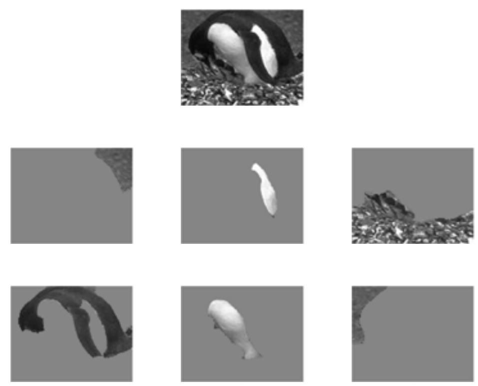


http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Brightness Image Segmentation

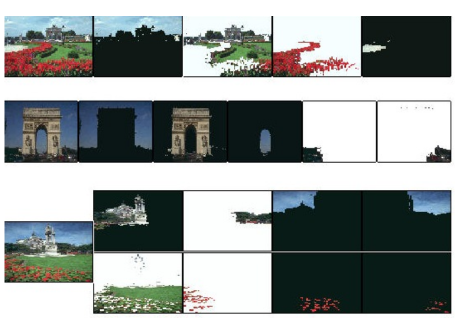


http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf



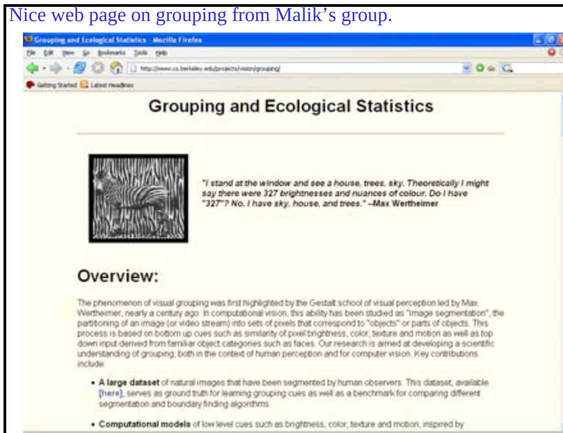
http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

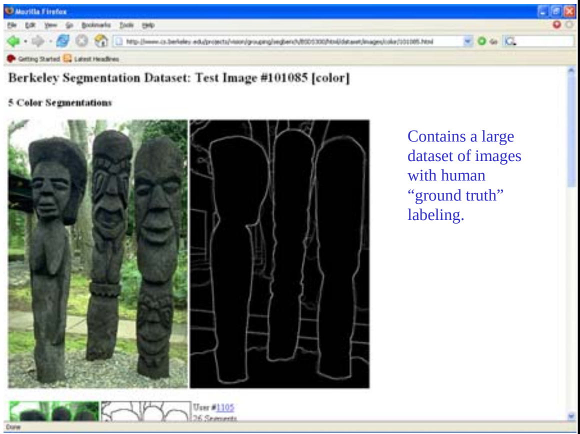
Results on color segmentation



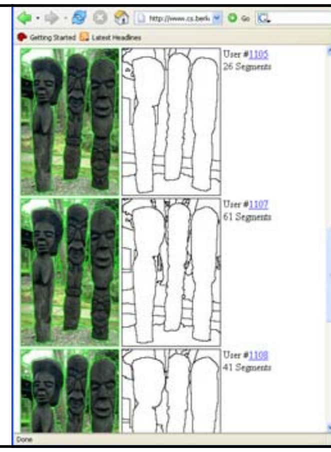
http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Nice web page on grouping from Malik's group.





Contains a large dataset of images with human "ground truth" labeling.



Of course, the human labelings differ one from another.

Line Fitting

- Hough transform
- Iterative fitting

Fitting

- Choose a parametric object/some objects to represent a set of tokens
- Most interesting case is when criterion is not local
  - can't tell whether a set of points lies on a line by looking only at each point and the next.
- Three main questions:
  - what object represents this set of tokens best?
  - which of several objects gets which token?
  - how many objects are there?

(you could read line for object here, or circle, or ellipse or...)

Fitting and the Hough Transform

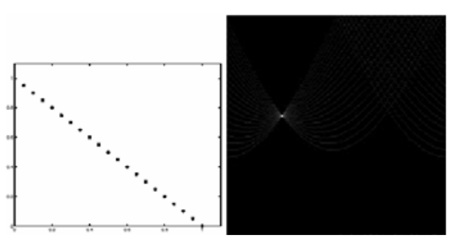
- Purports to answer all three questions
  - in practice, answer isn't usually all that much help
- We do for lines only
- A line is the set of points (x, y) such that

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

- Different choices of θ, d>0 give different lines
- For any (x, y) there is a one parameter family of lines through this point, given by

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

- Each point gets to vote for each line in the family; if there is a line that has lots of votes, that should be the line passing through the points



d

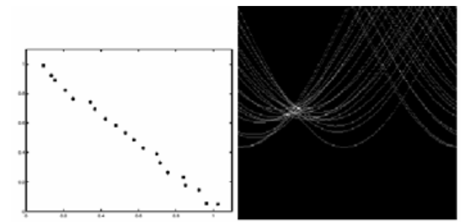
θ

tokens

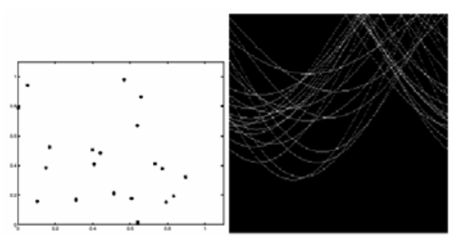
Votes for parameter values satisfying $(\sin\theta)x + (\cos\theta)y + d = 0$ at each token

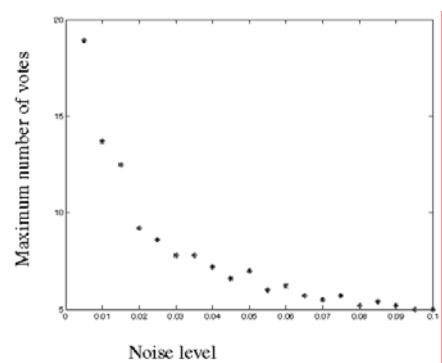
Mechanics of the Hough transform

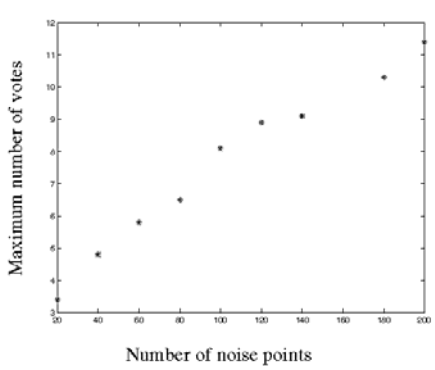
- Construct an array representing θ, d
- For each point, render the curve (θ, d) into this array, adding one at each cell
- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - count the peaks in the Hough array
- Who belongs to which line?
  - tag the votes
- Problems with noise and cell size can defeat it



tokens votes





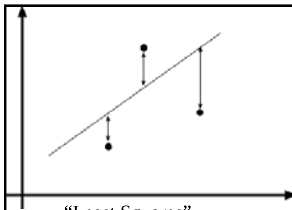


Rules of thumb for getting Hough transform to work well

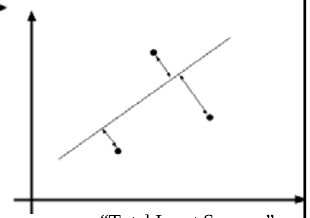
- Can work for finding lines in a set of edge points.
- Ensure minimum number of irrelevant tokens by tuning the edge detector.
- Choose the quantization grid carefully by trial and error.

Line fitting

What criteria to optimize when fitting a line to a set of points?



Line fitting can be max. likelihood - but choice of model is important



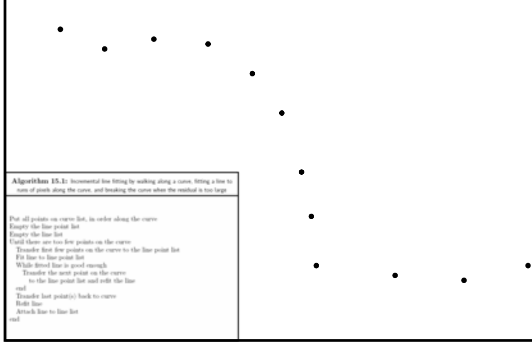
Who came from which line?

- Assume we know how many lines there are - but which lines are they?
  - easy, if we know who came from which line
- Three strategies
  - Incremental line fitting
  - K-means (described in book)
  - Probabilistic (in book, and in earlier lecture notes)

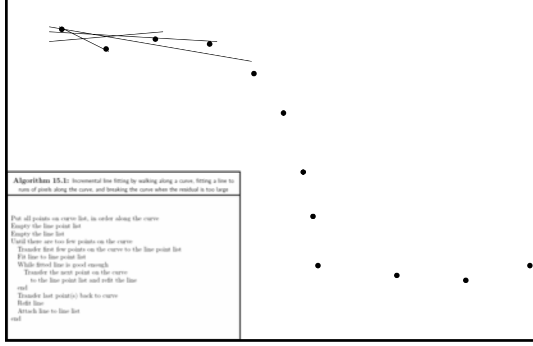
Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
  Transfer first few points on the curve to the line point list
  Fit line to line point list
  While fitted line is good enough
    Transfer the next point on the curve
      to the line point list and refit the line
  end
  Transfer last point(s) back to curve
  Refit line
  Attach line to line list
end
```

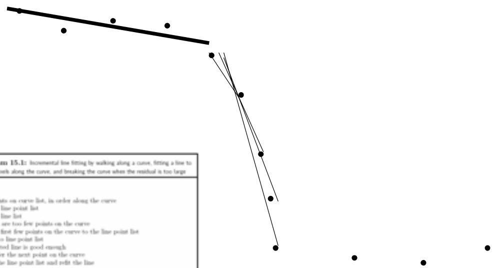
Incremental line fitting



Incremental line fitting



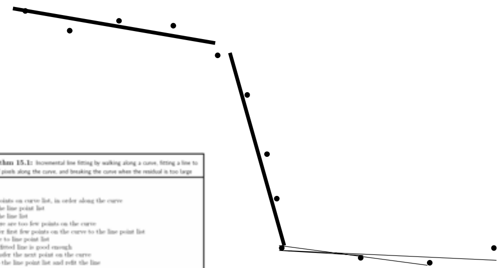
Incremental line fitting



Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
  Transfer first few points on the curve to the line point list
  Fit line to line point list
  While fitted line is good enough
    Transfer the next point on the curve
      to the line point list and refit the line
  end
  Transfer last point(s) back to curve
  Refit line
  Attach line to line list
end
```

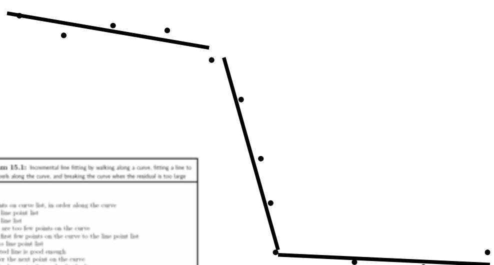
Incremental line fitting



Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
  Transfer first few points on the curve to the line point list
  Fit line to line point list
  While fitted line is good enough
    Transfer the next point on the curve
      to the line point list and refit the line
  end
  Transfer last point(s) back to curve
  Refit line
  Attach line to line list
end
```


Incremental line fitting



Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
  Transfer first few points on the curve to the line point list
  Fit line to line point list
  While fitted line is good enough
    Transfer the next point on the curve
      to the line point list and refit the line
  end
  Transfer last point(s) back to curve
  Refit line
  Attach line to line list
end
```


Fitting contours

- Two common techniques:
  - Snakes (Terzopolous, Witkin, and Kass)
  - Dynamic programming methods

Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network

Amnon Sha'ashua Shimon Ullman

Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100 Israel

Abstract

When we look at images, certain salient structures often attract our immediate attention, without requiring a systematic scan of the entire image. In subsequent stages, processing resources can be allocated preferentially to these salient structures. In many cases this saliency is a property of the structure as a whole, i.e., parts of the structure are not salient in isolation.

In this paper we present a saliency measure based on curvature and curvature variation. The structures this measure emphasizes are also salient in human perception, and they often correspond to objects of interest in the image.

We present a method for computing the saliency by a simple iterative scheme, using a uniform network of locally connected processing elements. The network uses an optimization approach to produce a "saliency map", which is a representation of the image emphasizing salient locations. The main properties of the network are: (i) the computations are simple and local, (ii) globally salient structures emerge with a small number of iterations, (iii) as a by-product of the computation contours are smoothed, and gaps are filled-in.

1. Introduction

Salient structures can often be perceived in an image at a glance. They appear to attract our attention without the need to scan the entire image in a systematic manner, and without prior expectations regarding their shape. The processes involved in the perception of salient structures appear to play a useful role in segmentation and recognition, since they allow us to immediately concentrate on objects of interest in the image.

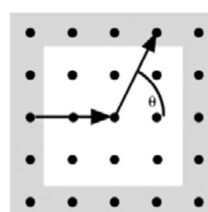
Consider the images in figures 1, 2 and 3. Certain objects in each image somehow attract our attention in a manner often described as 'preattentive'. For instance, the large blobs in Fig. 1 are prominent, although locally the blobs' contours are indistinguishable from background contours on the basis of local orientation, curvature, contrast, etc. It seems as if one must somehow capture most of the curve bounding a blob in order to perceive it as a prominent structure. The circle in Fig. 2 is immediately perceived although its contour is fragmented, implying that gaps do not hinder the immediate perception of such objects. In this case one must group together several line segments of the circle to distinguish it from the background. These examples also demonstrate that these prominent objects need not be recognized in order for them to be distinguished. The image in Fig. 3 is an edge image of a car in a cluttered background. Our attention is drawn immediately to the region of interest in the image. It seems that the car need not be recognized to attract our attention. When the image is inverted and presented for short periods, recognition becomes considerably more difficult, yet the same region remains salient.

Figure 2. A circle in a background of 200 randomly placed and oriented segments. The circle is still perceived immediately although its contour is fragmented.

The goal of this paper is to suggest what makes structures such as those in Fig. 1 – 3 salient, and to propose a mechanism for detecting salient locations in an image. A locally connected

http://www.cs.huji.ac.il/~shashua/papers/saliency.pdf

6.3.2 Saliency



http://people.csail.mit.edu/people/billf/freemanThesis.pdf

The recursive saliency calculation is as follows:

$$S_i^0 = \sigma_i \quad (6.3)$$
$$S_i^{n+1} = \sigma_i + \max_j [S_i^n f_{i,j}], \quad (6.4)$$

where S_i^k is the saliency of the ith orientation element after the kth iteration, σ_i is the local saliency of the ith element, and $f_{i,j}$ is a coupling constant between the ith and jth orientation elements. The maximization is taken over all neighboring orientation elements, j. The coupling constant penalizes sharp bends of the curve and effectively imposes a prior distribution on the expected shapes of the image contours. Shaashua and Ullman showed that after N iterations, the above algorithm will find the saliency of the most salient curve of length N originating from each contour.

http://people.csail.mit.edu/people/billf/freemanThesis.pdf

http://people.csail.mit.edu/people/billf/freemanThesis.pdf

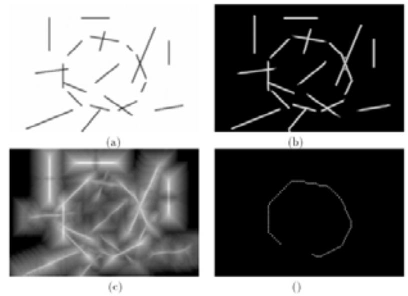


Figure 6-6: Saliency calculation. (a) Original figure, adapted from [95]. (b) Orientation evidence, based on spatial and angular local maxima of oriented filter outputs. (Shaashua and Ullman used Canny edge fragments for this step). Based on the orientation strength evidence in (b), the saliency algorithm was applied for 20 iterations. (c) shows the saliency of most salient contour of the 16 contours leaving each position. Note the "cloud" of salient values surrounding each image contour. (d) Curve traced starting from a position and orientation of high saliency. The curves traced by following the last choice of each orientation element are a reasonable approximation to the maximally

http://www.cs.huji.ac.il/~shashua/papers/saliency.pdf

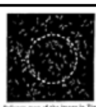


Figure 5. Saliency map of the image in Fig. 2 obtained by the network after 10 iterations. The saliency measure of each element of the circle is significantly higher than of the background elements.

Figure 6. The curve starting from the strongest element in figure 5. Virtual elements are displayed as dotted lines.

background elements. In this regard, the circle virtually 'pops-out' from the saliency map.

Figure 7. The same circle as in figure 2 but with 400 background segments.

The second point to notice is that a complete object is separated from the background although it is initially fragmented.

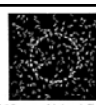


Figure 8. Saliency map of the image in Fig. 7 obtained by the network after 10 iterations.

of background elements increases considerably. To illustrate, we doubled the number of background elements as shown in Fig. 7. We applied again ten iterations to produce the saliency map in Fig. 8. Starting from the most salient element, the curve extracted by the network is identical with the one in Fig. 6.

The next example is the image in Fig. 3. Fig. 9 shows the saliency map after 30 iterations. Only the region surrounding the car is displayed. The saliency measure given to most of the elements of the car is significantly higher than that given to the background elements. Fig. 10 displays the five most salient curves obtained by tracing the most salient elements.

Note that the traced curves have been smoothed, and that the gaps have been filled in. The results suggest that the saliency computation is useful for distinguishing significant structures in

Figure 9. Saliency map of the image in Fig. 3 obtained by the network after 30 iterations. The region of interest virtually 'pops-out' from the display.