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.

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

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](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](http://research.microsoft.com/users/jckrumm/Publications%202000/Wall%20Flower.pdf)
2 different background removal models

<table>
<thead>
<tr>
<th>Background estimate</th>
<th>Foreground estimate</th>
<th>Foreground estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average over frames</td>
<td>EM background estimate</td>
<td>low thresh</td>
</tr>
<tr>
<td></td>
<td></td>
<td>high thresh</td>
</tr>
<tr>
<td></td>
<td>EM</td>
<td></td>
</tr>
</tbody>
</table>

Static Background Modeling Examples

[MIT Media Lab Pfender / ALIVE System]

Static Background Modeling Examples

[MIT Media Lab Pfender / ALIVE System]

Dynamic Background
BG Pixel distribution is non-stationary:

[MIT AI Lab VSAM]

Mixture of Gaussian BG model
Stauffer and Grimson tracker:
Fit per-pixel mixture model to observed distribution.

[MIT AI Lab VSAM]
Wallflower: Principles and Practice of Background Maintenance

Kentaro Toyama, John Krumm, Barry Brumitt, Brian Meyers
Microsoft Research
Redmond, WA 98052


Abstract
Background maintenance is a frequent element of video surveillance systems. We develop Wallflower, a three-component system for background maintenance: the wallflower, the foreground, and the background. The wallflower component identifies unique non-background objects in a scene, the foreground component learns time-based models of foreground objects, and the frame-based component detects, localizes, and tracks foreground objects. The system is designed to accommodate changes in the appearance of the background, the foreground, the lighting, and the scene. The wallflower component identifies an object as foreground only if the object has not appeared in the past. No special care has to be taken to handle changing background conditions.

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 different spatial scales.

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

Background removal issues

Background Techniques Compared

From the Wallflower Paper

Greedy Clustering Algorithms

Algorithms 15-8: Agglomerative clustering is clustering by merging

- Make each pixel a semantic cluster
- Use the clustering algorithm

Algorithms 15-9: Agglomerative clustering is clustering by splitting

- Choose a single cluster containing all points
- Use the clustering algorithm
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)

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

Matlab k-means clustering demo

K-Means clustering using intensity alone and color alone
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.

\[ \mathbf{z} = \begin{pmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{v} \end{pmatrix} \]

\[ \mathbf{z} = \begin{pmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{v} \end{pmatrix} \]

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

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

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.

---

- 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.

---

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

1. Window in image domain
2. Intensities of pixels within image domain window
3. Window in range domain
4. Center of mass of pixels within both image and range domain windows
5. Center of mass of pixels within both image and range domain windows
6. Center of mass of pixels within both image and range domain windows
7. Center of mass of pixels within both image and range domain windows

---

Mean Shift color&spatial Segmentation Results:

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

---


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

Weighted Graphs and Their Representations

Boundaries of image regions defined by a number of attributes
Measuring Affinity

Intensity
\[ \text{aff}(x,y) = \exp\left(-\frac{1}{2\sigma^2} (I(x) - I(y))^2 \right) \]

Distance
\[ \text{aff}(x,y) = \exp\left(-\frac{1}{2\sigma^2} (d(x,y))^2 \right) \]

Color
\[ \text{aff}(x,y) = \exp\left(-\frac{1}{2\sigma^2} (c(x,y))^2 \right) \]

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 \( a^T a = 1 \)

Some Terminology for Graph Partitioning

- How do we bipartition a graph:

\[ \text{cut}(A,B) = \sum_{u \in A, v \in B} W(u,v) \]
\[ \text{asso}(A,A') = \sum_{u \in A, v \in A'} W(u,v) \] with \( A \neq B \neq \emptyset \)
Minimum Cut

A cut of a graph $G$ is the set of edges $S$ such that removal of $S$ from $G$ disconnects $G$. The minimum cut is the cut of minimum weight, where weight of cut $<A, B>$ is given as

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

Minimum Cut and Clustering

Drawbacks of Minimum Cut

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

Ideal Cut

Cuts with lesser weight than the ideal cut

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 $B$.

Normalized Cut As Generalized Eigenvalue problem

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} - \frac{cut(B, A)}{assoc(B, V)}$$

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 $A$. This can be rewritten as the generalized eigenvalue problem.

$$D_y = \sum A_i$$

$$= \frac{(1+x)^T(D-W)(1+x)}{k^T Dk} - \frac{(1-x)^T(D-W)(1-x)}{(1-k)^T Dk}$$

$$= \frac{\sum_{i} A_i D_{i, i}}{\sum_{i} D_{i, i}}$$

after simplification, Shi and Malik derive

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

- Instead, solve the generalized eigenvalue problem
  \[ \max \left( y^T (D - W) y \right) \text{subject to } \left( y^T D y = 1 \right) \]
- which gives
  \[ (D - W)y = \lambda_D y \]
- They show that the 2nd smallest eigenvector solution \( y \) is a good real-valued approx 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 -1.

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 \(\theta, d \neq 0\) 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
- 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 $0, d$
- For each point, render the curve $(\theta, d)$ 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

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)

Incremental line fitting

• Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to a set of points 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
  Reset line
  Attach line to line list
end
### Incremental line fitting

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

### Fitting contours

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

---

**6.3.2 Saliency**

The recursive saliency calculation is as follows:

\[ S_i^\beta = \sigma_i \]

\[ S_i^{t+1} = \sigma_i + \max_j(S_j^\beta f_{ij}) \]

where \( S_i^\beta \) is the saliency of the \( i \)th orientation element after the \( t \)th iteration, \( \sigma_i \) is the initial saliency of the \( i \)th element, and \( f_{ij} \) is a coupling constant between the \( i \)th and \( j \)th 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. Shashua and Ullman showed that after \( N \) iterations, the above algorithm will find the saliency of the most salient curve of length \( \nu \) originating from each contour.


Figure 6-6: Saliency calculation. (a) Original figure, adapted from [96], (b) Orientation evidence, based on spatial and angular local maxima of oriented filter outputs. (c) Shashua and Ullman's contour edge fragments for this step. Based on the orientation strength evidence in (b), the saliency algorithm was applied for 50 iterations. (c) The saliency of most salient contour of the 10 contours having each position. Note the "visual" of salient values surrounding each image contour. (d) Curve traced starting from a position and orientation of high saliency. The curve traced by following the last choice of each orientation element are a reasonable approximation to the maximally