Lecture 14

Edges and segmentation
From Pixels to Perception: Mid-level operations of Segmentation and Grouping
Figure / Ground
Finding groups of pixels that go together
(parts, objects, textures, holes)
Figure / Ground

Emergence

http://en.wikipedia.org/wiki/Gestalt_psychology
I. Edges
What is an edge?

- Depth discontinuity
- Material change
- Texture boundary

Diagram:
- Surface normal discontinuity
- Depth discontinuity
- Surface color discontinuity
- Illumination discontinuity
Finding edges: Computing derivatives
Canny edge detector

```
edge(image,'canny')
```

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
1: Filter Image with derivatives of Gaussian 2D edge detection filters

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

\[ h_x(x, y) = \frac{\partial h(x, y)}{\partial x} = -x e^{-\frac{x^2+y^2}{2\sigma^2}} \]

\[ h_y(x, y) = \frac{\partial h(x, y)}{\partial x} = -x e^{-\frac{x^2+y^2}{2\sigma^2}} \]
Gaussian filters

\[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

Convolution with self is another Gaussian

\[ G_\sigma * G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

- Convolving two times with Gaussian kernel of width \( \sigma = \) convolving once with kernel of width \( \sigma \sqrt{2} \)
1: Filter Image with derivatives of Gaussian 2D edge detection filters

1 pixel  3 pixels  7 pixels

Smoothing filters with different scales
The Sobel Operator: A common approximation of derivative of Gaussian

- Common approximation of derivative of Gaussian

\[
\begin{array}{ccc}
1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\quad \quad
\begin{array}{ccc}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

- The standard defn. of the Sobel operator omits the $1/8$ term
  - doesn’t make a difference for edge detection
  - the $1/8$ term is needed to get the right gradient value
Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
2: Gradient: Find edge strength (magnitude) and direction (angle) of gradient

\[ h_x(x, y) = \frac{\partial h(x, y)}{\partial x} = -x \frac{-x^2 + y^2}{2\pi \sigma^4} e^{\frac{-x^2 + y^2}{2\sigma^2}} \]

\[ h_y(x, y) = \frac{\partial h(x, y)}{\partial x} = -x \frac{-x^2 + y^2}{2\pi \sigma^4} e^{\frac{-x^2 + y^2}{2\sigma^2}} \]

Magnitude: \( h_x(x, y)^2 + h_y(x, y)^2 \) \hspace{1cm} Edge strength

Angle: \( \arctan \left( \frac{h_y(x, y)}{h_x(x, y)} \right) \) \hspace{1cm} Edge normal
**Image Gradient:** gradient points in the direction of most rapid increase in intensity

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

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

Can think of it as the slope of a 3D surface

Gradient at a single point \((x, y)\) is a vector:

- Direction is the direction of maximum slope:
  \[
  \theta = \tan^{-1}\left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)
  \]

- Length is the magnitude (steepness) of the slope
  \[
  \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}
  \]
Original image

3D plot of luminance

Gradient
Issues:
1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trails; how do we identify the significant points?
3) How do we link the relevant points up into curves?
4) Noise.

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.
Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
Goal: mark points along the curve where the magnitude is biggest. How? looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues:
- at which point is the maximum
- where is the next one?

Forsyth, 2002
At \( q \), we have a maximum (1) if the value is larger than those at both \( p \) and at \( r \). Interpolate between \( p \) and \( r \) to get these values.

Predicting the next edge point: Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either \( r \) or \( s \)).

**Non maximum suppression:** check if pixel is local maximum along gradient direction
Examples:
Non-Maximum Suppression

Original image
Gradient magnitude
But some edges are broken
Non-maxima Suppressed
(remaining pixels are the local Maximum)
Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
Closing edge gaps

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.
Example: Canny Edge Detection

Original image

Strong edges only

Strong + connected weak edges

Weak edges
gap is gone

courtesy of G. Loy
Example: Canny Edge Detection
Example: Canny Edge Detection
II. Segmentation
II.1 Bottom-up segmentation

• Group together similar-looking pixels
  – “Bottom-up” process
  – Unsupervised

• Bottom-up segmentation
  – Clustering
  – Mean shift
  – Graph-based

“superpixels”
Issues

• How do we decide that two pixels are likely to belong to the same region?

• How many regions are there?
Method 1: Clustering

• Cluster similar pixels (features) together

See pdf chapter 14

Source: K. Grauman
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
A simple segmentation algorithm

• Each pixel is described by a vector
  \[ z = [r, g, b] \text{ or } [Y, u, v], \ldots \]

• Run a clustering algorithm (e.g. k-means) using some distance between pixels:
  \[ D(\text{pixel}_i, \text{pixel}_j) = \| z_i - z_j \|^2 \]
K-Means Clustering

- Given \( k \), the \( k \)-means algorithm consists of four steps:
  - Select initial centroids at random.
  - Assign each object to the cluster with the nearest centroid.
  - Compute each centroid as the mean of the objects assigned to it.
  - Repeat previous 2 steps until no change.
Data set

Dendrogram obtained by agglomerative clustering
A *Dendrogram* Shows How the Clusters are Merged Hierarchically

Decompose data objects into several levels of nested partitioning (tree of clusters), called a *dendrogram*.

A clustering of the data objects is obtained by cutting the dendrogram at the desired level. Then each connected component forms a cluster.
• K-means (k=5) clustering based on intensity (middle) or color (right) is essentially vector quantization of the image attributes
  – Clusters don’t have to be spatially coherent

See pdf chapter 14 each pixel is replaced with the mean value of its cluster
K-means using color alone (k=11 clusters)
Showing 4 of the segments, (not necessarily connected)
Some are good, some meaningless
Including spatial relationships

Augment data to be clustered with spatial coordinates.

\[
\begin{pmatrix}
  Y \\
  u \\
  v \\
  x \\
  y
\end{pmatrix}
\]

- color coordinates (or r,g,b)
- spatial coordinates

- Cluster similar pixels (features) together

\[
\begin{align*}
  R=0 & \quad G=200 & \quad B=20 \\
  X=30 & \quad Y=20 \\
  R=15 & \quad G=189 & \quad B=2 \\
  X=20 & \quad Y=400 \\
  R=3 & \quad G=12 & \quad B=2 \\
  X=100 & \quad Y=200
\end{align*}
\]
• Clustering based on (r,g,b,x,y) values enforces more spatial coherence

K-means using colour and position, 20 segments

Still misses goal of perceptually pleasing or useful segmentation
No measure of texture

Hard to pick K…
K-Means for segmentation

• Pros
  – Very simple method
  – Converges to a local minimum of the error function

• Cons
  – Memory-intensive
  – Need to pick K
  – Sensitive to initialization
  – Sensitive to outliers

Slide credit: S. Lazebnik
Method 2: Mean shift clustering

- An advanced and versatile technique for clustering-based segmentation

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

Mean shift algorithm

The mean shift algorithm seeks *modes* or local maxima of density in the feature space.

**Feature space**

$\text{(L}^*\text{u}^*\text{v}^* \text{ color values)}$
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:

Two issues:
(1) Kernel to interpolate density based on sample positions.
(2) Gradient ascent to mode.
Search window
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Search window
Center of mass
Mean Shift vector
Search window

Center of mass

Mean Shift vector
Mean shift clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

Slide by Y. Ukrainitz & B. Sarel
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.

Pixels in L*u* space

Corresponding trajectories with peaks marked as red dots

Clustering results after ~160 mean shift procedures
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
Mean Shift color & spatial Segmentation

Results:
Mean shift pros and cons

• Pros
  – Clusters are places where data points tend to be close together
  – Just a single parameter (window size)
  – Finds variable number of modes
  – Robust to outliers

• Cons
  – Output depends on window size
  – Computationally expensive
  – Does not scale well with dimension of feature space
Method 3: Graph-Theoretic Image Segmentation

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

- **V**: image pixels
- **E**: connections between pairs of nearby pixels

$W_{ij}$: probability that $i$ & $j$ belong to the same region

Segmentation = graph partition

A different way of thinking about segmentation…
Segmentation by graph cut

- Fully connected graph (node for every pixel $i,j$)
- Edge/link between every pair of pixels: $p,q$
- Each edge is weighted by the *affinity* or similarity of the two nodes:
  - cost $c_{pq}$ for each link: $c_{pq}$ measures similarity (or affinity)
  - similarity is *inversely proportional* to difference in color and position
Segmentation by graph cut

- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (similarity or affinity)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: S. Seitz
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph

2b- Build a similarity/affinity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
1- Vectors of data

We represent each pixel by a feature vector $\mathbf{x}$, and define a distance function appropriate for this feature representation (e.g. euclidean distance). Features can be brightness value, color– RGB, L*$u*$v; texton histogram, etc—and calculate distances between vectors (e.g. Euclidean distance).
Computing distance

• We represent each pixel by a feature vector $\mathbf{x}$, and define a distance function appropriate for this feature representation

• Then we can convert the distance between two feature vectors into an affinity/similarity measure with the help of a generalized Gaussian kernel:

$$\exp\left(-\frac{1}{2\sigma^2} \text{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$

Slide credit: S. Lazebnik
Affinity between pixels

Similarities among pixel descriptors

\[ W_{ij} = \exp(-\|z_i - z_j\|^2 / \sigma^2) \]

\( \sigma = \text{Scale factor…} \)

it will hunt us later
Affinity between pixels

Similarities among pixel descriptors

$$W_{ij} = \exp(-||z_i - z_j||^2 / \sigma^2)$$

Interleaving edges

$$W_{ij} = 1 - \max Pb$$

$\sigma = \text{Scale factor…}$

it will hunt us later

With $Pb = \text{probability of boundary}$
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph
2b- Build a similarity/affinity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
2a- What is a graph?
2a- What is a graph?

![Graph Diagram]

Adjacency Matrix

\[
\begin{bmatrix}
0 & - & - & - & - \\
- & 0 & - & - & - \\
- & - & 0 & - & - \\
- & - & - & 0 & - \\
- & - & - & - & 0 \\
\end{bmatrix}
\]

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

Adjacency Matrix

![Graph Diagram]
2a- What is a weighted graph?

Affinity Matrix represents the weighted links

\[
W = \begin{bmatrix}
1 & 0.1 & 0.3 & 0 & 0 \\
0 & 1 & 0.4 & 0 & 0.2 \\
0.3 & 0.4 & 1 & 0.6 & 0.7 \\
0 & 0 & 0.6 & 1 & 1 \\
0 & 0.2 & 0.7 & 1 & 1 \\
\end{bmatrix}
\]

\(W_{ij}\): probability that \(i\) & \(j\) belong to the same region

Diagonal: each point with itself is 1
Strong links/edges
Weak links/edges
No links/edges connected

See Forsyth-Ponce chapter
Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

E.g. Gaussian kernel similarity function

$$W_{ij} = e^{\frac{||x_i - x_j||^2}{2\sigma^2}}$$

Controls size of neighborhood

See Forsyth-Ponce chapter
We can do segmentation by finding the minimum cut in a graph.
Graph terminology

• Similarity matrix: \( W = \begin{bmatrix} w_{i,j} \end{bmatrix} \)

\[
\frac{-\|X(i) - X(j)\|^2}{\sigma^2_X} = w_{i,j} = e
\]

Weight matrix associated with the graph
(larger values are lighter)
Affinity matrix of a natural image

Similarity of image pixels to selected pixel
Brighter means more similar
Graph terminology

• Degree of node:

\[ d_i = \sum_j w_{i,j} \]
Graph terminology

- Volume of set:

\[ \text{vol}(A) = \sum_{i \in A} d_i, A \subseteq V \]
Graph terminology

Cuts in a graph:

\[ \text{cut}(A, \overline{A}) = \sum_{i \in A, j \in \overline{A}} w_{i,j} \]
Scale affects affinity

\[ W_{ij} = \exp\left(-\| z_i - z_j \|^2 / \sigma^2 \right) \]

- Small \( \sigma \): group only nearby points
- Large \( \sigma \): group far-away points

Dataset of 4 groups of 10 points drawn from a normal distribution with four different means

\( \sigma = 0.2 \)

See Forsyth-Ponce chapter
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph

2b- Build a similarity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
Spectral Clustering

Data

Similarities

Affinity Matrix

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University
Spectral clustering: Using Eigenvalues of the matrix

• **spectral clustering** uses the eigenvalues of the similarity/affinity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions

\[ w(i,j) \rightarrow \text{distance node } i \text{ to node } j \]
What are eigenvectors?

**Eigenvectors** represent the dimensions of data
**Eigenvalues** are the length of eigenvectors

In a case of two variables, Eigenvectors are the two lines drawn in the scatterplot

No relationship between variables

$\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}} = 1$

A linear relationship between variables

$\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}} = \infty$
Eigenvectors example

1\textsuperscript{st} Eigenvector is the all ones vector \textbf{1} (if graph is connected)

• 2\textsuperscript{nd} Eigenvector thresholded at 0 separates first two clusters from last two
• k-means clustering of the 4 eigenvectors identifies all 4 clusters
Spectral Clustering pipeline

\[ w_{i,j} = e^{-\frac{||x_i-x_j||^2}{\sigma^2}} \]

Data are projected into a lower-dimensional space (spectral/eigenvector domain) where they are easily separable.

Given number \( k \) of clusters, compute the first \( k \) eigenvectors, \( V_1, \ldots, V_k \) of the affinity matrix \( M \).

Build the matrix \( V \) with the eigenvectors as columns.

Interpret the rows of \( V \) as new data points \( Z_i \).

Cluster the points \( Z_i \) with the k-means algorithms.

Dimensionality reduction \( n \times n \rightarrow n \times k \)
Eigenvectors and blocks

- Block weight matrices have block eigenvectors:

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

\[\begin{bmatrix}
.71 \\
.71 \\
0 \\
0
\end{bmatrix} \quad \begin{bmatrix}
0 \\
0 \\
.71 \\
.71
\end{bmatrix}
\]

- Near-block matrices have near-block eigenvectors:

\[
\begin{bmatrix}
1 & 1 & .2 & 0 \\
1 & 1 & 0 & -.2 \\
.2 & 0 & 1 & 1 \\
0 & -.2 & 1 & 1
\end{bmatrix}
\]

\[\begin{bmatrix}
.71 \\
.69 \\
.14 \\
0
\end{bmatrix} \quad \begin{bmatrix}
0 \\
-.14 \\
.69 \\
.71
\end{bmatrix}
\]

\[\lambda_1 = 2 \quad \lambda_2 = 2 \quad \lambda_3 = 0 \quad \lambda_4 = 0
\]

\[\lambda_1 = 2.02 \quad \lambda_2 = 2.02 \quad \lambda_3 = -0.02 \quad \lambda_4 = -0.02
\]

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University
Spectral Space

Can put items into blocks by eigenvectors:

Clusters clear regardless of row ordering:

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University
Example eigenvector

The eigenvector corresponding to the largest eigenvalue of the affinity matrix. Most values are small, but some, corresponding to the elements of the main cluster, are large.

The 3 next eigenvectors corresponding to the next 3 largest eigenvalues of the affinity matrix. Most values are small but for (disjoint) sets of elements the values are large. This follows from the block structure of the affinity matrix.

See Forsyth Ponce, Chapter 14 given.
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph
2b- Build a similarity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
Clustering – How many groups are there?

Out of the various possible partitions, which is the correct one?
Clustering – 5 groups

Optimal?
Clustering – 5 groups

Looks optimal
What does the Affinity Matrix Look Like?
The Eigenvectors and the Clusters

*Step-Function* like behavior preferred!

Makes Clustering Easier.
The Eigenvectors and the Clusters
The Eigenvectors and the Clusters
The Affinity Matrix
The eigenvectors correspond the 2\textsuperscript{nd} smallest to the 9\textsuperscript{th} smallest eigenvalues
Issue: Number of Clusters?

$k = 3$

$k = 4$

$k = 6$

Issue: choice of kernel, for Gaussian kernels, choice of $\sigma$

$\sigma = 3$

$\sigma = 13$

$\sigma = 25$
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of **data**

2a- Build a similarity **graph**

2b- Build a similarity **matrix**

3- Calculate **eigenvectors**

4- Cut the graph: apply **threshold** to eigenvectors
Cuts in a graph:

\[
\text{cut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{i,j}
\]

- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
Partition a graph with minimum cut

\[ \text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v) \]

- **Cut**: sum of the weight of the cut edges:
- Minimum cut is the cut of minimum weight
Normalized Cut is a better measure ..

- We normalize by the total volume of connections

\[
Ncut(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}
\]

where \(\text{assoc}(A, V) = \sum_{u \in A, t \in V} w(u, t)\)
Many different methods...

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i,X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data
2- Build a similarity graph
3- Calculate eigenvectors
4- Apply threshold to largest eigenvectors

1- Get vectors of data
2- Build normalized cost matrix
3- Get eigenvectors with smallest eigenvalues
4- Apply threshold

Shi & Malik

... etc
Graph-based Image Segmentation

Image (I) \rightarrow \text{Intensity, Color, Edges, Texture} \rightarrow \text{Affinity matrix (W)} \rightarrow \text{Eigenvector} X(W) \rightarrow \text{Discretization/Thresholding}

\begin{align*}
Ncut(A, B) &= \text{cut}(A, B) \left( \frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right) \\
(D - W)X &= \lambda DX \\
X_A(i) &= \begin{cases} 
1 & \text{if } i \in A \\
0 & \text{if } i \notin A
\end{cases}
\end{align*}
The Eigenvectors

Eigenvector #7
Normalized cut
Normalized cut
II.2 Top-down Segmentation

- Separate image into coherent "things": combining object recognition with segmentation
  - Supervised or unsupervised

Berkeley segmentation database
Aim: Given an image and object category, to segment the object

Segmentation should (ideally) be
• shaped like the object e.g. cow-like
• obtained efficiently in an unsupervised manner
• able to handle self-occlusion
Examples of bottom-up segmentation

- Using Normalized Cuts, Shi & Malik, 1997