Grouping (epilogue)
Logistics

- We have contacted projects that we foresee a problem with
- HW4 (last one!) out; due next Friday [Object recognition]
- Final project: presentation deadline the same, but writeup deadline is extended from June 3rd to June 5th
The problem with mincuts
Normalized cuts

\[ Ncut(V_1, \ldots V_K) = \sum_k \frac{cut(V_k, V \setminus V_k)}{vol(V_k)} \]

\[ cut(A, B) = \sum_{i \in A, j \in B} W_{ij} \]

\[ vol(A) = \sum_{i \in A, j \in V} W_{ij} \]

“Ratio of red to blue”
Normalized cuts: integer program

\[ G = (V,E) \]

\[ Ncut(X) = \sum_k \frac{x_k^T (D - W) x_k}{x_k^T D x_k} \]

\( x_k \in \{0, 1\}^{\lvert V \rvert}, X = [x_1, \ldots, x_K], X1_K = 1_N \)

\[ D = \text{diag}(D_i), D_i = \sum_j W_{ij} \]

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

\[ D-W = \text{"Laplacian matrix of } G\text{"} \]
Normalized cuts: quadratic program

\[ G = (V,E) \]

\[ N_{cut}(X) = \sum_k \frac{x_k^T (D - W)x_k}{x_k^T Dx_k} \]

\[ x_k \in \mathbb{R}^{|V|} \]

Real-valued relaxation
Generalized eigenvalue problem

$$\min_x x^T (D - W)x \quad s.t. \quad x^T Dx = 1$$

$$(D - W)x = \lambda Dx$$

$$z = D^{\frac{1}{2}} x$$

$$D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} z = \lambda z$$

$x$ (or $z$) are given by the K-smallest eigenvalues of this linear system
How to discretize eigenvectors?

Heuristic that works well: interpret $k$ eigenvectors as coordinates for points in $\mathbb{R}^k$ and cluster again (using say, k-means)

K-means on eigenvectors “spectral clustering”

Intuition: eigenvectors capture transitive similarity properties
Variational image segmentation

\[ E(f, C) = \int_{\Omega} (I(x) - f(x))^2 dx + \int_{\Omega \setminus C} |\nabla f(x)| dx + \int_{C} ds \]

f: piecewise smooth approximation of image \( I \)
C: set of closed curves around each segment
Variational image segmentation

\[
E(f, C) = \int_{\Omega} (I(x) - f(x))^2 dx + \int_{\Omega \setminus C} |\nabla f(x)| dx + \int_{C} ds
\]

Optimization of a function is known as a “calculus of variations” problem


We can generalize the notion of “0-gradient at optimum” to functions using Euler-Lagrange equations

Allows us to write solutions as partial differential equations (PDEs)

Lots of pretty math…
Variational image segmentation

\[ E(f, C) = \int_{\Omega} (I(x) - f(x))^2 dx + \int_{\Omega\setminus C} |\nabla f(x)| dx + \int_{C} ds \]

Optimization of a function is known as a “calculus of variations” problem.


We can generalize the notion of “0-gradient at optimum” to functions using Euler-Lagrange equations.

Allows us to write solutions as partial differential equations (PDEs)

My own intuition: apply local-search / gradient descent
Mumford-Shah functional

\[ E(f, C) = \int_{\Omega} (I(x) - f(x))^2 \, dx + \int_{\Omega \setminus C} |\nabla f(x)| \, dx + \int_{C} ds \]

With out discontinuity term, we could discretize and reformulate as a giant least-squares problem.
Image snakes / active counter models

Evolution of Explicit Boundaries

http://en.wikipedia.org/wiki/Active_contour_model
How is segmentation used in practice?

Superpixels

“The pixel grid is an arbitrary spatial unit imposed by digitization”
How is segmentation used in practice?

Generate a “soup” of overlapping segments and process each

Replace scanning window with a “selective search”
How is segmentation used in practice?

Motion segmentation

Intuitively, we apply template matching on small patches from one frame to the next.

We’ll talk more about video analysis in last week of lecture.
Layered motion segmentation

Layered Representation for Motion Analysis

John Y. A. Wang
The MIT Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Edward H. Adelson
The MIT Media Laboratory
Dept. Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
Aside: segmentation for depth
“Easy” version of motion segmentation: background subtraction

Let’s go over some simple approaches...
A naive approach

B = I(0);

\[ \text{loop time } t \]
\[ I(t) = \text{next frame}; \]
\[ \text{diff} = \text{abs}[B - I(t)]; \]
\[ M(t) = \text{threshold}(\text{diff}, \lambda); \]
\[ \text{...} \]
\[ \text{end} \]

(Note: pseudocode is written for grayscale images)
How well does it work?

Background subtraction does a reasonable job of extracting the shape of an object, provided the object intensity/color is sufficiently different from the background.
Objects that enter the scene and stop continue to be detected, making it difficult to detect new objects that pass in front of them.

If part of the assumed static background starts moving, both the object and its negative ghost (the revealed background) are detected.
More problems

Background subtraction is sensitive to changing illumination and unimportant movement of the background (for example, trees blowing in the wind, reflections of sunlight off of cars or water).

Background subtraction cannot handle movement of the camera.

(Aside: this is why “motion segmentation” feels like a better perspective on this problem)
• Background model is replaced with the previous image.

\[
\begin{align*}
B(0) &= I(0); \\
&\ldots \\
&\text{loop time } t \\
&I(t) = \text{next frame}; \\
&\text{diff} = \text{abs}[B(t-1) - I(t)]; \\
&M(t) = \text{threshold}(\text{diff}, \lambda); \\
&\ldots \\
&B(t) = I(t); \\
&\text{end}
\end{align*}
\]
How well does it work?

Frame differencing is very quick to adapt to changes in lighting or camera motion.

Objects that stop are no longer detected. Objects that start up do not leave behind ghosts.

However, frame differencing only detects the leading and trailing edge of a uniformly colored object. As a result very few pixels on the object are labeled, and it is very hard to detect an object moving towards or away from the camera.
Adjusting temporal scale of differencing

Note what happens when we adjust the temporal scale (frame rate) at which we perform two-frame differencing …

Define $D(N) = \| I(t) - I(t+N) \|$

more complete object silhouette, but two copies (one where object used to be, one where it is now).
A neat “trick”: 3-frame differencing

The previous observation is the motivation behind three-frame differencing

- $D(-15)$: where object was, and where it is now
- $D(+15)$: where object is now, and where it will be

AND

where object is now!
But it's hard to find a good frame rate

Choice of good frame-rate for three-frame differencing depends on the size and speed of the object

This worked well for the person
Adaptive background subtraction

- Current image is “blended” into the background model with parameter $\alpha$
- $\alpha = 0$ yields simple background subtraction, $\alpha = 1$ yields frame differencing

$$
B(0) = I(0);
...
\text{loop time } t
I(t) = \text{next frame};
\text{diff} = \text{abs}[B(t-1) - I(t)];
M(t) = \text{threshold}(\text{diff}, \lambda);
...
B(t) = \alpha I(t) + (1-\alpha)B(t-1);
\text{end}
$$
Adaptive statistical modelling of background

Learn distribution in color space
Use some previous method to identify foreground/background pixels.

Mark each pixel with the last “time” it was declared foreground.
Efficient implementation

- Motion images are combined with a linear decay term
- also known as motion history images (Davis and Bobick)

\[
\begin{align*}
B(0) &= I(0); \\
H(0) &= 0; \\
\text{loop time } t \\
I(t) &= \text{next frame}; \\
diff &= \text{abs}[B(t-1) - I(t)]; \\
M(t) &= \text{threshold}(\text{diff}, \lambda); \\
tmp &= \text{max}[H(t-1) - \gamma, 0]; \\
H(t) &= \text{max}[255\times M(t), \text{tmp}]; \\
\ldots
\end{align*}
\]

\[B(t) = I(t);\]
end