GENERAL

- Project Proposals Deadline was Sunday
  - We'll be providing you with feedback over the next week.
- Problem Set 3 Due Thursday
- Friday Office Hours will be (always) in Lopata 103
GROUPING & SEGMENTATION

Partition the set of pixels into disjoint sets or groups

Dual of the edge detection problem!
GROUPING & SEGMENTATION

But what is the basis of this grouping?

- Physical
  - Lie on the same surface / plane
  - Made of the same material
  - Moving together rigidly
GROUPING & SEGMENTATION

But what is the basis of this grouping?

- Semantic
  - Same object
  - Foreground / background
  - Interesting / non-interesting

Semantic segmentation: often humans will disagree on what goes where.
Simplest Version: Superpixel Segmentation

- Partition Image into a large number of segments called superpixels.
- Many segments, each segment relatively small.
- Oversegmentation of the image
  - Each object / plane / surface might be broken into multiple segments
  - But (hope) each segment does not cross a boundary.

- Can be based on appearance alone

- Simplifies further processing (dealing with $K$ segments instead of $N$ pixels)
GROUPING & SEGMENTATION

Image

Superpixels (different levels)
GROUPING & SEGMENTATION

SLIC Superpixels

Achanta et al., 2010. Simple Linear Iterative Clustering.

Formally, given an image $I[n]$ with $N$ pixels, you want group the pixels into $K \ll N$ super pixels.

You want to determine a label

$$L[n] \in \{1, 2, \ldots K\}$$

for every pixel $n$, based on some metric.

Note the value of $L$ doesn't matter. What matters is similar pixels have the same label. This is clustering!

The final output we care about is $K$ sets

$$S_k = \{n : L[n] = k\}$$
SLIC Superpixels

We will want to group pixels that appear similar and are close by into the same super-pixel.

Define an "augmented" image $I'[n]$ where each $I'[n] \in \mathbb{R}^5$

- First 3 dimensions are R,G,B
- Two dimensions are $x$ and $y$ co-ordinates.

For grayscale images, $I'[n] \in \mathbb{R}^3$. 

SLIC Superpixels

Determine labeling $L[n]$ to minimize the following cost:

$$L = \arg\min_{L} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n:L[n]=k} \|I'[n] - \mu_k\|^2$$

Here, each $\mu_k \in \mathbb{R}^5$.

- This is K-means clustering.
- Easy to see that $\mu_k$ will be the mean of the $I'$ vectors of pixels assigned to label $k$.
- We're saying that all pixels assigned the label $k$ should be close to each other in the squared distance sense of their augmented vectors.
- This augmented vector encodes both appearance and location.
- So we want pixels that look the same and are close-by to have the same label.
SLIC Superpixels

\[ L = \arg \min_{L} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n:L[n]=k} ||I'[n] - \mu_k||^2 \]

- Typically, use Lab color space instead of RGB.
- You can weight the contribution of location vs appearance by normalizing \((x, y)\) in \(I'\) differently.

\[ I'[n] = [I[n]_R, I[n]_G, I[n]_B, \alpha n_x, \alpha n_y]^T \]
GROUPING & SEGMENTATION

\[ L = \arg \min_{\mathcal{L}} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n : L[n] = k} ||I'[n] - \mu_k||^2 \]

K-Means: Lloyd's algorithm

- Begin with some initial assignment \( L[n] \) (more later).
- At each iteration ...

**Step 1:** For each \( k \), assign

\[ \mu_k = \text{Mean}\{I'[n] \}_{L[n] = k} \]

**Step 2:** For each \( n \), assign

\[ L[n] = \arg \min_k ||I'[n] - \mu_k||^2 \]

- Does this converge?
- How do we initialize?
- Do we really need to do \( K \times N \) computations of \( ||I'[n] - \mu_k||^2 \)?
SLIC: Initialization

- Actually, begin with an assignment of \( \{\mu_k\} \) (and do a step 2).
- Given desired number of super-pixels \( K \), choose \( K \) points on a grid.
  - Spaced horizontally and vertically apart by \( S = \sqrt{\frac{HW}{K}} \)
- Set each \( u_k = I'[n_k] \) as the augmented vector of one of these points.
- In step 2, each seed is going to attract pixels in its neighborhood that are most like it.
- Sometimes this initialization gives you a 'seed' that lies right on an edge.
  - Bad because pixel on either side of edge will often look nothing like it.
SLIC: Initialization

- Actually, begin with an assignment of \( \{ \mu_k \} \) (and do a step 2).
- Given desired number of super-pixels \( K \), choose \( K \) points on a grid.
  - Spaced horizontally and vertically apart by \( S = \sqrt{\frac{HW}{K}} \).
- Set each \( u_k = I'[n_k] \) as the augmented vector of one of these points.
- In step 2, each seed is going to attract pixels in its neighborhood that are most like it.
- Sometimes this initialization gives you a 'seed' that lies right on an edge.
  - Bad because pixel on either side of edge will often look nothing like it.
- Solution: Look in a 3x3 neighborhood, and choose pixel with lowest gradient magnitude.
At any given iteration, for step 2:

- Don't consider all possible $K$ for every $n$.
- Instead, say that a pixel $n$ can only be assigned to a cluster $k$ if $n$ is within a $2S \times 2S$ window around the spatial co-ordinates in $u_k$.
- Note that $\mu_k$'s will no longer be on a regular grid.
SLIC: Minimization

At any given iteration, for step 2:

- Initialize $\text{min\_dist}[n]$ to Infinity for all $n$
- Loop through each $u_k$, and consider pixels in $2S \times 2S$ window around $\mu_k$
  - This will be a regular grid.
- For each pixel in this window, compute distance of $I'[n]$ to $\mu_k$, compare to $\text{min\_dist}[n]$, if lower, update $\text{min\_dist}[n]$ and update $L[n]$.

Do we need to loop over $K$? Can get some parallelism if you're clever about it.
GROUPING & SEGMENTATION

SLIC: Uses

Given a set of super-pixels $S_k = \{ n : L[n] = k \}$:

- You can "denoise" your image by smoothing independently within each $S_k$.
  - Replace all intensities by their mean.
  - Fit intensity to be a linear function of $n$.
- You can "denoise" other scene properties
  - Filter your stereo cost volume within each super-pixel.
  - Take your disparities within each super-pixel, and fit them to a plane.
  - Do the aggregation for Lucas-Kanade flow estimation within each super-pixel.
- Build super-pixels with intensity + other information
  - Get an initial estimate of disparity, add it to your augmented vector $I'[n]$.
  - Get a super-pixel segmentation. Smooth cost-volume, re-estimate disparities.
  - Repeat segmentation ... 
- Group pixels (instead of super-pixels) into objects or by semantic labels
GROUPING & SEGMENTATION

Graph-based Methods

Foreground / Background Segmentation

Image from Rother et al., GrabCuts.
Graph-based Methods

Assign a label of 1 (foreground) or 0 (background) for each pixel in the image.

Let's say user has labeled some pixels as foreground or background.
(or these are noisy / sparse predictions from some algorithm)
Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} C[n, L[n]] + \sum_{(n,n') \in E} S_{n,n'}(L[n], L[n']) \]

Kind of like our stereo setup, but binary labeling problem.
GROUPING & SEGMENTATION

Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} C[n, L[n]] + \sum_{(n,n') \in \mathbb{E}} S_{n,n'}(L[n], L[n']) \]

* Comes from user input / algorithm

* E.g., 0 for unlabeled pixels.

* Very high / infinite cost at \( n \) for \( L[n] \)

* different from user label
GROUPING & SEGMENTATION

Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} \ C[n, L[n]] + \sum_{(n,n') \in E} S_{n,n'}(L[n], L[n']) \]

Again, pairs of neighboring pixels. Horizontal / Vertical / Diagonal
GROUPING & SEGMENTATION

Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} C[n, L[n]] + \sum_{(n,n') \in E} S_{n,n'}(L[n], L[n']) \]

Now will depend on pixel location. Often based on intensity differences / whether there is an edge.
Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} C[n, L[n]] + \sum_{(n, n') \in E} S_{n, n'}(L[n], L[n']) \]
Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} C[n, L[n]] + \sum_{(n,n') \in E} S_{n,n'}(L[n], L[n']) \]

Break set of vertices into two disconnected graphs by removing edges. Cost = sum of weight on edges you removed.
Formally, let's say our smoothness cost \( S_{n,n'}(l, l') = w_{n,n'} \delta[l! = l'] \), for \( w_{n,n'} \geq 0 \).

\[
L = \arg \min_{L[n] \in \{0,1\}} \sum_n C[n, L[n]] + \sum_{(n,n') \in \mathcal{E}} w_{n,n'} \delta[L[n]! = L[n']] 
\]

- Build a graph with vertices \( V = \{n\} \cup \{0, 1\} \).
- Place an edge between every \((n, n') \in \mathcal{E}\) with weight \( w_{n,n'} \).
- Place an edge between \((n, 0)\) \( \forall n \) with weight \( C[n, 1] \) (assuming costs are positive).
- Place an edge between \((n, 1)\) \( \forall n \) with weight \( C[n, 0] \) (assuming costs are positive).
- Partition the vertices into sets \( A, B \) such that \( 0 \in A, 1 \in B \), to minimize \( \text{Cut}(A, B) \).
  - The cut is defined as the sum of the weights of the edges going between vertices in \( A \) to vertices in \( B \).
- Can be solved in polynomial time (e.g., Stoer-Wagner)
- Assign all pixels in \( A \) label 0, and all pixels in \( B \) label 1.
GROUPING & SEGMENTATION

Graph-based Methods

\[ L = \arg \min_{L[n] \in \{0,1\}} C[n, L[n]] + \sum_{(n, n') \in E} S_{n,n'}(L[n], L[n']) \]

- Initialize unary costs $C$ with user labels. Do a segmentation.
- Now look at the foreground and background pixels. Fit a probability distribution to each (mixture of Gaussians, histogram)
- Update $C[n,L]$ based on how well the intensity at location $n$ fits with the foreground/background distributions (for $L=1/0$)
GROUPING & SEGMENTATION

- Polynomial Time for Binary Segmentation
- NP-hard for multi-label cases. \( L[n] \in \{A, B, C, \ldots \} \)
  - Remember, this is the same as our stereo case.
- But approximate algorithms available
  - Typically different algorithms work well here than for stereo
GROUPING & SEGMENTATION

Multi-label Case: \( L[n] = \{A, B, C, \ldots \} \)

- Begin with some initial assignment of \( L[n] \) (perhaps the pixel-wise minimizer of \( C \))
- Then update \( L \) by making one of two kinds of moves in each iteration
- \( \alpha \)-Expansion
  - Choose one of the labels (say \( A \))
  - Build a binary segmentation problem where \( 1 = A, 0 = \) everything else
  - Set \( C[n, 0] = \infty \) for all pixels \( n \) where the current label is already \( A \)
  - Set \( C[n, 0] = \) cost of its current assigned label for every other pixel
  - Set \( C[n, 1] = \) cost of \( A \) for every other pixel
  - Do a min-cut. Replace all pixels labeled 1 with \( A \).
- \( \alpha - \beta \) Swap
  - Choose a pair of labels (say \( A \) and \( B \))
  - Now define a new graph, containing only pixels that currently have label \( A \) or \( B \).
  - Solve the binary segmentation problem
- Iterate through these different kinds of moves for different choices of labels.
GROUPING & SEGMENTATION

References

- Rother et al., GrabCut -Interactive Foreground Extraction using Iterated Graph Cuts, SIGGRAPH 2004.
Next Time

- Min-cut can often lead to isolated points

- Avoid with a method called "Normalized Cuts"