Recap: Gestalt Theory

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

“i stand at the window and see a house, trees, sky. Theoretically i might say there were 327 brightnesses and nuances of colour. Do i have '327'? No. i have sky, house, and trees.”

Max Wertheimer (1880-1943)

Recap: Gestalt Factors

- These factors make intuitive sense, but are very difficult to translate into algorithms.

Recap: Image Segmentation

- Goal: identify groups of pixels that go together

Recap: K-Means Clustering

- Basic idea: randomly initialize the k cluster centers, and iterate between the two steps we just saw.
  1. Randomly initialize the cluster centers, \( c_1, \ldots, c_k \)
  2. Given cluster centers, determine points in each cluster
     - For each point \( p \) find the closest \( c_i \). Put \( p \) into cluster \( i \)
  3. Given points in each cluster, solve for \( c_i \)
  4. If \( c_i \) have changed, repeat Step 2

- Properties
  - Will always converge to some solution
  - Can be a “local minimum”
  - Does not always find the global minimum of objective function:
    \[ \sum_{i=1}^{k} \left( \sum_{p \in \text{cluster } i} ||p - c_i||^2 \right) \]
Recap: Expectation Maximization (EM)

- **Goal:** Find blob parameters $\theta$ that maximize the likelihood function:
  $$P(data|\theta) = \prod P(z|\theta)$$

- **Approach:**
  1. E-step: given current guess of blobs, compute ownership of each point
  2. M-step: given ownership probabilities, update blobs to maximize likelihood function
  3. Repeat until convergence

Recap: Mean-Shift Algorithm

- **Iterative Mode Search**
  1. Initialize random seed, and window $W$
  2. Calculate center of gravity (the “mean”) of $W$:
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence

Recap: 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

Recap: Mean-Shift Segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

Recap: Generic Clustering

- We have focused on ways to group pixels into image segments based on their appearance
  - Find groups; “quantize” feature space
- In general, we can use clustering techniques to find groups of similar “tokens”, provided we know how to compare the tokens.
  - E.g., segment an image into the types of motions present
  - E.g., segment a video into the types of scenes (shots) present

Topics of This Lecture

- **Graph theoretic segmentation**
  - Normalized Cuts
  - Using texture features
- **Hierarchical segmentation**
  - Case study: segmentation-based recognition
- **Segmentation as Energy Minimization**
  - Markov Random Fields
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
Images as Graphs

- Fully-connected graph
  - Node (vertex) for every pixel
  - Link between every pair of pixels, \((p,q)\)
  - Affinity weight \(w_{pq}\) for each link (edge)
  - \(w_{pq}\) measures similarity
  - Similarity is inversely proportional to difference (in color and position...)

Segmentation by Graph Cuts

- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low similarity (low weight)
  - Similar pixels should be in the same segments
  - Dissimilar pixels should be in different segments

Measuring Affinity

- Distance
  \[ \text{aff}(x, y) = \exp\left(-\frac{1}{2\sigma^2} \|x - y\|^2\right) \]
- Intensity
  \[ \text{aff}(x, y) = \exp\left(-\frac{1}{2\sigma^2} \|I(x) - I(y)\|^2\right) \]
- Color
  \[ \text{aff}(x, y) = \exp\left(-\frac{1}{2\sigma^2} \text{dist}(c(x), c(y))^2\right) \]
  (some suitable color space distance)
- Texture
  \[ \text{aff}(x, y) = \exp\left(-\frac{1}{2\sigma^2} \|f(x) - f(y)\|^2\right) \]
  (vectors of filter outputs)

Scale Affects Affinity

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

Graph Cut

- Set of edges whose removal makes a graph disconnected
- Cost of a cut
  \[ \text{cut}(A, B) = \sum_{p \in A, q \in B} w_{pq} \]
- A graph cut gives us a segmentation
  - What is a "good" graph cut and how do we find one?
Minimum Cut

- We can do segmentation by finding the minimum cut in a graph.
- Efficient algorithms exist for doing this.
- Drawback: Weight of cut proportional to number of edges in the cut.
- Minimum cut tends to cut off very small, isolated components.

Ideal Cut

Cuts with lesser weight than the ideal cut.

Normalized Cut (N Cut)

- A minimum cut penalizes large segments.
- This can be fixed by normalizing for size of segments.
- The normalized cut cost is:

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

\[ \text{assoc}(A,V) = \sum_{p \in A} w_{p,q} \]

- The exact solution is NP-hard but an approximation can be computed by solving a generalized eigenvalue problem.

Normalized cuts and image segmentation, PAMI 2000

NCuts as a Generalized Eigenvector Problem

- Definitions:

\[ W: \text{the affinity matrix, } W(i,j) = w_{ij}; \]
\[ D: \text{the diag. matrix, } D_{ii} = \sum_j W(i,j); \]
\[ x: \text{a vector in } \{1,-1\}^N, x(i) = 1 \Leftrightarrow i \in A. \]

- Rewriting Normalized Cut in matrix form:

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

\[ = \frac{(1+x^T)(D-W)(1-x)}{k^TD_1} + \frac{(1-x^T)(D-W)(1-x)}{(1-k)^TD_1} = \cdots \]

- After simplification, we get:

\[ \frac{1}{y^TDy} \]

This is a Rayleigh Quotient.
- Solution given by the "generalized" eigenvalue problem:

\[ (D-W)y = \lambda y \]

- Solved by converting to standard eigenvalue problem:

\[ D^{-1}(D-W)z = \lambda z, \text{ where } z = D^\dagger y \]

- Subtleties:

- Optimal solution is second smallest eigenvector.
- Gives continuous result—must convert into discrete values of y.

This is hard, as y is discrete!

Relaxation:
- Continuous.
NCuts Example

Discretization
- Problem: eigenvectors take on continuous values
  - How to choose the splitting point to binarize the image?

- Possible procedures
  a) Pick a constant value (0, or 0.5).
  b) Pick the median value as splitting point.
  c) Look for the splitting point that has the minimum NCut value:
     1. Choose a possible splitting points.
     2. Compute NCut value.
     3. Pick minimum.

NCuts: Overall Procedure
1. Construct a weighted graph \( G(V, E) \) from an image.
2. Connect each pair of pixels, and assign graph edge weights,
   \( W(i, j) = \text{Prob. that } i \text{ and } j \text{ belong to the same region.} \)
3. Solve \( (D - W)y = \lambda Dy \) for the smallest few eigenvectors. This yields a continuous solution.
4. Threshold eigenvectors to get a discrete cut
   - This is where the approximation is made (we’re not solving NP).
5. Recursively subdivide if NCut value is below a pre-specified value.

NCuts Matlab code available at http://www.cis.upenn.edu/~jshi/software/

Using Texture Features for Segmentation
- Texture descriptor is vector of filter bank outputs
- Textons are found by clustering.
Using Texture Features for Segmentation

- Texture descriptor is vector of filter bank outputs.
- Textons are found by clustering.
- Affinities are given by similarities of texton histograms over windows given by the “local scale” of the texture.

Results with Color & Texture

Summary: Normalized Cuts

- Pros:
  - Generic framework, flexible to choice of function that computes weights (“affinities”) between nodes
  - Does not require any model of the data distribution
- Cons:
  - Time and memory complexity can be high
    - Dense, highly connected graphs ⇒ many affinity computations
    - Solving eigenvalue problem for each cut
    - Preference for balanced partitions
      - If a region is uniform, NCuts will find the modes of vibration of the image dimensions

Topics of This Lecture

- Graph theoretic segmentation
  - Normalized Cuts
  - Using color and texture features
- Hierarchical segmentation
  - Case study: segmentation-based recognition
- Segmentation at Energy Minimization
  - Markov Random Fields
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

Multi-Level Segmentation

- It is often difficult to extract a single good segmentation
  - Idea: Extract a hierarchy of segmentations instead

Multiscale Segmentation Tree

- Arrange the segmentations in a tree

Sample cutsets

Example segmentations for several contrasts

Example segmentations
Multiscale Segmentation Tree

- Now an image corresponds to a segmentation tree.
- An object forms a subtree.

Case Study: Tree-Based Recognition

- Images = Trees
- Category present = Many similar subtrees
- Extracting similar subtrees = Tree matching
  - Main difficulty: structural noise
- Category model = Union of similar subtrees
- Simultaneous detection, recognition and segmentation of ALL category instances by Matching the model with an image

Example Results: Cars

10 positive out of 20 training images

Results on test images:

Example Results: Faces

6 positive out of 12 training images

Results invariant to scale and rotation

Tree-Based Recognition

- Pros/Interesting ideas
  - Instability of segmentation process circumvented by hierarchical segmentation.
  - Unsupervised learning of new category models.
  - Extraction of a semantically meaningful segment hierarchy.
  - Simultaneous recognition and segmentation.
  - Approach invariant to scale and rotation.

- Cons
  - Currently, this is still an experimental procedure.
  - Subtree matching is still very expensive: \( O(\text{nodes}^2) \).
  - Training on 20 images takes ~2 hours.
  - Algorithm doesn’t scale well to larger training sets.

Topics of This Lecture

- Graph theoretic segmentation
  - Normalized Cuts
  - Using color and texture features
- Hierarchical segmentation
  - Case study: segmentation-based recognition
- Segmentation as Energy Minimization
  - Markov Random Fields
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
**Markov Random Fields**

- Allow rich probabilistic models for images
- But built in a local, modular way
  - Learn local effects, get global effects out

**MRF Nodes as Pixels**

- Observed evidence
- Hidden “true states”
- Neighborhood relations

- Original image
- Degraded image
- Reconstruction from MRF modeling pixel neighborhood statistics

**MRF Nodes as Patches**

- Image patches
- Scene patches
- MRF Nodes as Pixels

- Image
- Scene
- Scene patches

**Network Joint Probability**

\[ P(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j) \]

- Scene
- Image
- Image-scene compatibility function
- Scene-scene compatibility function
- Local observations
- Neighboring scene nodes

**Energy Formulation**

- Joint probability
  \[ P(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j) \]

- Maximizing the joint probability is the same as minimizing the log
  \[ E(x, y) = \sum_i \log \Phi(x_i, y_i) + \sum_{i,j} \log \Psi(x_i, x_j) \]

- This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call \( E \) an energy function.

- \( \phi \) and \( \psi \) are called potentials.

**Energy Formulation**

- Energy function
  \[ E(x, y) = \sum \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \]

- Single-node potentials
- Pairwise potentials

- **Single-node potentials** \( \phi \)
  - Encode local information about the given pixel/patch
  - How likely is a pixel/patch to belong to a certain class (e.g., foreground/background)?

- **Pairwise potentials** \( \psi \)
  - Encode neighborhood information
  - How different is a pixel/patch’s label from that of its neighbor? (e.g., based on intensity/color/texture difference, edges)
Energy Minimization

- **Goal:**
  - Infer the optimal labeling of the MRF.
- Many inference algorithms are available, e.g.
  - Gibbs sampling, simulated annealing
  - Iterated conditional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts
- Recently, Graph Cuts have become a popular tool
  - Only suitable for a certain class of energy functions
  - But the solution can be obtained very fast for typical vision problems (~1MPixel/sec).

Topics of This Lecture

- **Graph theoretic segmentation**
  - Normalized Cuts
    - Using color and texture features
- **Hierarchical segmentation**
  - Case study: segmentation-based recognition
- **Segmentation as Energy Minimization**
  - Markov Random Fields
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

Graph Cuts for Optimal Boundary Detection

- Idea: convert MRF into source-sink graph

Minimum cost cut can be computed in polynomial time (max-flow/min-cut algorithms)

Simple Example of Energy

\[
E(L) = \sum_p D_p(L_p) + \sum_{p \in N} \delta(L_p \neq L_q)
\]

Boundary term

- \( D_p(t) \) and \( D_p(s) \)
- \( L_p \in \{s,t\} \)
- \( \delta(x) \) is Dirac delta function

Adding Regional Properties

- Regional bias example
  - Suppose \( I \) and \( I' \) are given “expected” intensities of object and background
  - \( D_p(t) = \exp(-\|I_p - I\|_2^2 / 2\sigma^2) \)
  - \( D_p(s) = \exp(-\|I_p - I'\|_2^2 / 2\sigma^2) \)

- NOTE: hard constrains are not required, in general.

Adding Regional Properties

- “expected” intensities of object and background \( I \) and \( I' \) can be re-estimated

- EM-style optimization
Adding Regional Properties

- More generally, regional bias can be based on any intensity models of object and background

\[ D_p(L_p) = -\log \Pr(I_p | L_p) \]

given object and background intensity histograms

\[ \text{Pr}(I_p | L_p) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(I_p - \mu)^2}{2\sigma^2}} \]

- B. Leibe
  [Boykov & Jolly, ICCV’01]
  Slide credit: Yuri Boykov

How to Set the Potentials? Some Examples

- Color potentials
  - e.g. modeled with a Mixture of Gaussians
  \[ R(x, y; \theta) = \log \sum_k \theta_k \Pr(k | x, y) N(y; \mu_k, \Sigma) \]

- Edge potentials
  - e.g. a "contrast sensitive Potts model"
  \[ \phi(x, y; g; \theta) = \frac{1}{2} e^{-\frac{1}{\beta} \sum_i g_i(y) \delta(x_i - x)} \]
  where
  \[ g_i(y) = \beta \arg \left( \frac{|y_i - y_j|}{|y_j|} \right) \]

- Parameters \( \theta \), need to be learned, too!

How Does it Work? The s-t-Mincut Problem

- Graph theoretic segmentation
  - Normalized Cuts
  - Using color and texture features

- Hierarchical segmentation
  - Case study: segmentation based recognition

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Graph cuts for image segmentation
  - s-t-mincut algorithm
  - Extension to non-binary case
  - Applications

The s-t-Mincut Problem

- What is an st-cut?
  An st-cut (S, T) divides the nodes between source and sink.

- What is the cost of an st-cut?
  Sum of cost of all edges going from 5 to 7

- \( 5 + 2 + 9 = 16 \)

- Source

- Sink

- \( v_1 \)

- \( v_2 \)

- \( v_3 \)

- \( v_4 \)

- \( v_5 \)

- \( v_6 \)

- \( v_7 \)

- \( v_8 \)

- \( v_9 \)

- \( v_{10} \)

Graph (V, E, C)

Vertices \( V = \{v_1, v_2, \ldots, v_9\} \)

Edges \( E = \{(v_1, v_2), \ldots\} \)

Costs \( C = \{c_{12}, c_{24}, \ldots\} \)

- B. Leibe
  Slide credit: Pushmeet Kohli
  [Boykov & Jolly, ICCV’01]
  [Shotton & Winn, ECCV’06]
**History of Maxflow Algorithms**

### Augmenting Path and Push-Relabel

<table>
<thead>
<tr>
<th>Year</th>
<th>Discoverer(s)</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1969</td>
<td>Ford, Fulkerson</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1970</td>
<td>Edmonds, Karp</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1971</td>
<td>Duffin, Robertson</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1977</td>
<td>Hopcroft, Tarjan</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1978</td>
<td>Even, Tarjan</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1977</td>
<td>Cherkassky</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1979</td>
<td>Cai, Hsu</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1983</td>
<td>Savage, Tarjan</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1986</td>
<td>Goldberg, Tarjan</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1987</td>
<td>Klein, Orlin</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1990</td>
<td>Abrahamson, Vempala</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1992</td>
<td>Abolhassani, et al.</td>
<td>$O(V^2E)$</td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg, Rao</td>
<td>$O(V^2E)$</td>
</tr>
</tbody>
</table>

**Notes:**
- Algorithms assume non-negative edge weights.

---

**How to Compute the s-t-Mincut?**

1. Solve the dual maximum flow problem.

2. Compute the maximum flow between Source and Sink.

3. In every network, the maximum flow equals the cost of the st-mincut.

**Flow = 0**

**Augmenting Path Based Algorithms**

1. Find path from source to sink with positive capacity.
2. Push maximum possible flow through this path.
3. Repeat until no path can be found.

**Flow = 2**

**Augmenting Path Based Algorithms**

1. Find path from source to sink with positive capacity.
2. Push maximum possible flow through this path.
3. Repeat until no path can be found.

---

**Maxflow Algorithms**

**Flow = 0**

**Augmenting Path Based Algorithms**

1. Find path from source to sink with positive capacity.
2. Push maximum possible flow through this path.
3. Repeat until no path can be found.

**Flow = 2**

**Augmenting Path Based Algorithms**

1. Find path from source to sink with positive capacity.
2. Push maximum possible flow through this path.
3. Repeat until no path can be found.

---

**Algorithm Assumptions:**
- Algorithms assume non-negative capacity.
Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Flow = 2

Source

Sink

V1
0
1
2
3

V2
4
9

Flow = 2 + 4

Source

Sink

V1
0
1
2
3

V2
4
5

Flow = 6

Source

Sink

V1
0
1
2
3

V2
4
5
0

Flow = 6 + 1

Source

Sink

V1
0
1
2
3

V2
4
5

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Flow = 2

Source

Sink

V1
0
1
2
3

V2
4
9

Flow = 6

Source

Sink

V1
0
1
2
3

V2
4
5
0

Flow = 6 + 1

Source

Sink

V1
0
1
2
3

V2
4
5
0

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Flow = 6
Maxflow Algorithms

Augmenting Path Based Algorithms
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
B. Leibe

Maxflow in Computer Vision

- Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity (m = O(n))
- Dual search tree augmenting path algorithm
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently
  - High worst-case time complexity
  - Empirically outperforms other algorithms on vision problems
  - Efficient code available on the web
  http://www.adastral.ucl.ac.uk/~vladkolm/software.html

When Can s-t Graph Cuts Be Applied?

\[ E(L) = \sum_{p} E_p(L_p) + \sum_{p \in N} E(L_p, L_q) \]

- s-t graph cuts can only globally minimize binary energies that are submodular.

- Non-submodular cases can still be addressed with some optimality guarantees.
  - Current research topic

Dealing with Non-Binary Cases

- For image segmentation, the limitation to binary energies is a nuisance.
  - Binary segmentation only
- We would like to solve also multi-label problems.
  - NP-hard problem with 3 or more labels
- There exist some approximation algorithms which extend graph cuts to the multi-label case
  - \( \alpha \)Expansion
  - \( \alpha \beta \)Swap
  - They are no longer guaranteed to return the globally optimal result.
    - But \( \alpha \)Expansion has a guaranteed approximation quality (2-approx) and converges in a few iterations.
α-Expansion Move

- **Basic idea:**
  - Break multi-way cut computation into a sequence of binary s-t cuts.

α-Expansion Algorithm

1. Start with any initial solution
2. For each label “α” in any (e.g. random) order
   1. Compute optimal α-expansion move (s-t graph cuts)
   2. Decline the move if there is no energy decrease
3. Stop when no expansion move would decrease energy

α-Expansion Moves

- In each α-expansion a given label “α” grabs space from other labels

For each move we choose the expansion that gives the largest decrease in the energy: binary optimization problem

Topics of This Lecture

- Graph theoretic segmentation
  - Normalized Cuts
  - Using color and texture features
- Hierarchical segmentation
  - Case study: segmentation-based recognition
- Segmentation as Energy Minimization
  - Markov Random Fields
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

GraphCut Applications: “GrabCut”

- Interactive Image Segmentation [Boykov & Jolly, ICCV’01]
  - Rough region cues sufficient
  - Segmentation boundary can be extracted from edges

Procedure

- User marks foreground and background regions with a brush.
- This is used to create an initial segmentation which can then be corrected by additional brush strokes.

GrabCut: Data Model

- Obtained from interactive user input
  - User marks foreground and background regions with a brush
  - Alternatively, user can specify a bounding box

Global optimum of the energy
**Perceptual and Sensory Augmented Computing**

**Computer Vision WS 09/10**

---

**GrabCut: Coherence Model**

- An object is a coherent set of pixels:
  \[
  \psi(x, y) = \gamma \sum_{(m,n) \in C} \delta[x_m \neq x] e^{-\frac{|x_m - x|}{\beta}}
  \]

- How to choose \(\gamma\)?

---

**Iterated Graph Cuts**

- Result

---

**GrabCut: Example Results**

- **This will be included in the next version of MS Office!**

---

**Applications: Interactive 3D Segmentation**

---

**Improving Efficiency of Segmentation**

- Problem: Images contain many pixels
  - Even with efficient graph cuts, an MRF formulation has too many nodes for interactive results.

- Efficiency trick: Superpixels
  - Group together similar-looking pixels for efficiency of further processing.
  - Cheap, local oversegmentation
  - Important to ensure that superpixels do not cross boundaries

- Several different approaches possible
  - Superpixel code available here

---

**Superpixels for Pre-Segmentation**

---
Summary: Graph Cuts Segmentation

- **Pros**
  - Powerful technique, based on probabilistic model (MRF).
  - Applicable for a wide range of problems.
  - Very efficient algorithms available for vision problems.
  - Becoming a de-facto standard for many segmentation tasks.

- **Cons/Issues**
  - Graph cuts can only solve a limited class of models
  - Submodular energy functions
  - Can capture only part of the expressiveness of MRFs
  - Only approximate algorithms available for multi-label case

Other Applications: Texture Synthesis

Graph-cut textures
(Kwatra, Schodl, Essa, Bobick 2003)

Similar to “image-quilting” (Efros & Freeman, 2001)

Minimal Error Boundary

Overlap error \( \Rightarrow \) min. error boundary

GraphCut Texture Synthesis Results

Original fragments

Results (with perspective correction)

GraphCut Image Synthesis Results

Source: Alyosha Efros
Application: Texture Synthesis in the Media

- Currently, still done manually...

How Do we Know?


Another Example


Segmentation: Caveats

- We’ve looked at bottom-up ways to segment an image into regions, yet finding meaningful segments is intertwined with the recognition problem.
- Often want to avoid making hard decisions too soon
- Difficult to evaluate; when is a segmentation successful?
  - Often depends on the rest of the system pipeline.

References and Further Reading

- Background information on Normalized Cuts can be found in Chapter 14 of

- Try the NCuts Matlab code at
  - http://www.cis.upenn.edu/~jshi/software/
- Try the GraphCut implementation at
  - http://www.adastral.ucl.ac.uk/~vladkolm/software.html