اصول پردازش تصویر

Principles of Image Processing

مصطفی کمالی تبریزی

۲۶ آبان ۱۳۹۹

جلسه هفدهم

Turbo Pixels

Alex Levinshtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, and Sven J. Dickinson
TurboPixels: Fast Superpixels Using Geometric Flows
Pattern Analysis and Machine Intelligence (PAMI), 2009

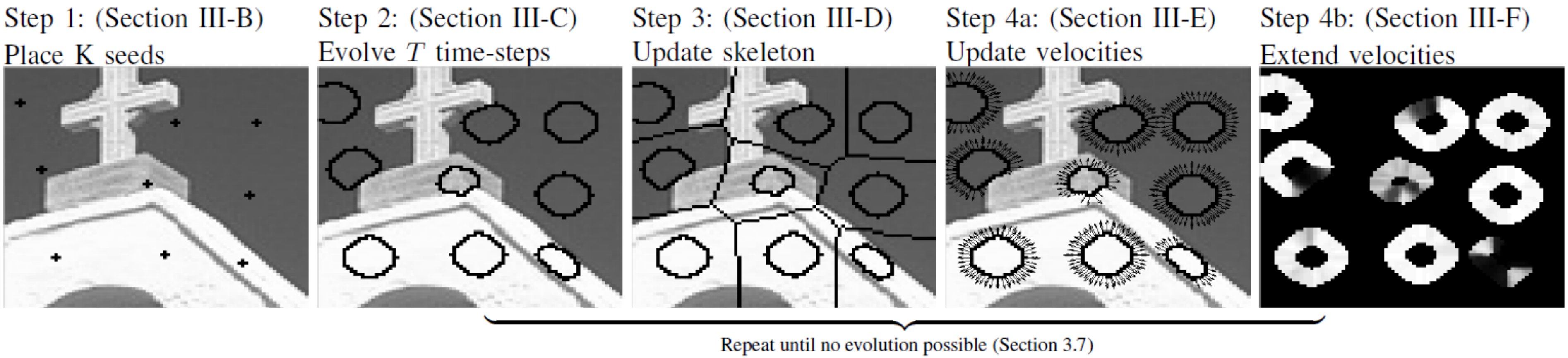


Fig. 2. Steps of the TurboPixel algorithm. In Step 4a the vectors depict the current velocities at seed boundaries. Where edges have been reached the velocities are small. In Step 4b the magnitude of velocities within the narrow band is proportional to brightness.

SUPERPIXELS FROM GEOMETRIC FLOWS

- Uniform size and coverage
- Connectivity
- Compactness
- Smooth, edge-preserving flow
- No superpixel overlap

Algorithm 1: TurboPixel Algorithm

Input: Image I, number of seeds K

Output: Superpixel boundaries B

Place K seeds on a rectangular grid in image I;

Perturb the seed positions away from high gradient regions;

Set all seed pixels to "assigned";

Set Ψ^0 to be the signed Euclidean distance from the "assigned" regions;

assigned_pixels $\leftarrow \sum_{x,y} [\Psi^0(x, y) >= 0]$;

Compute the pixel affinity $\phi(x, y)$;

$n \leftarrow 0$;

while *Change in assigned_pixels is large* **do**

  Compute the image velocity S_I;

  Compute the boundary velocity S_B;

  $S \leftarrow S_I S_B$;

  Extend the speed S in a narrow band near the zero level-set of Ψ^n;

  Compute Ψ^{n+1} by evolving Ψ^n within the narrow band;

  $n \leftarrow n + 1$;

  assigned_pixels $\leftarrow \sum_{x,y} [\Psi^n(x, y) >= 0]$;

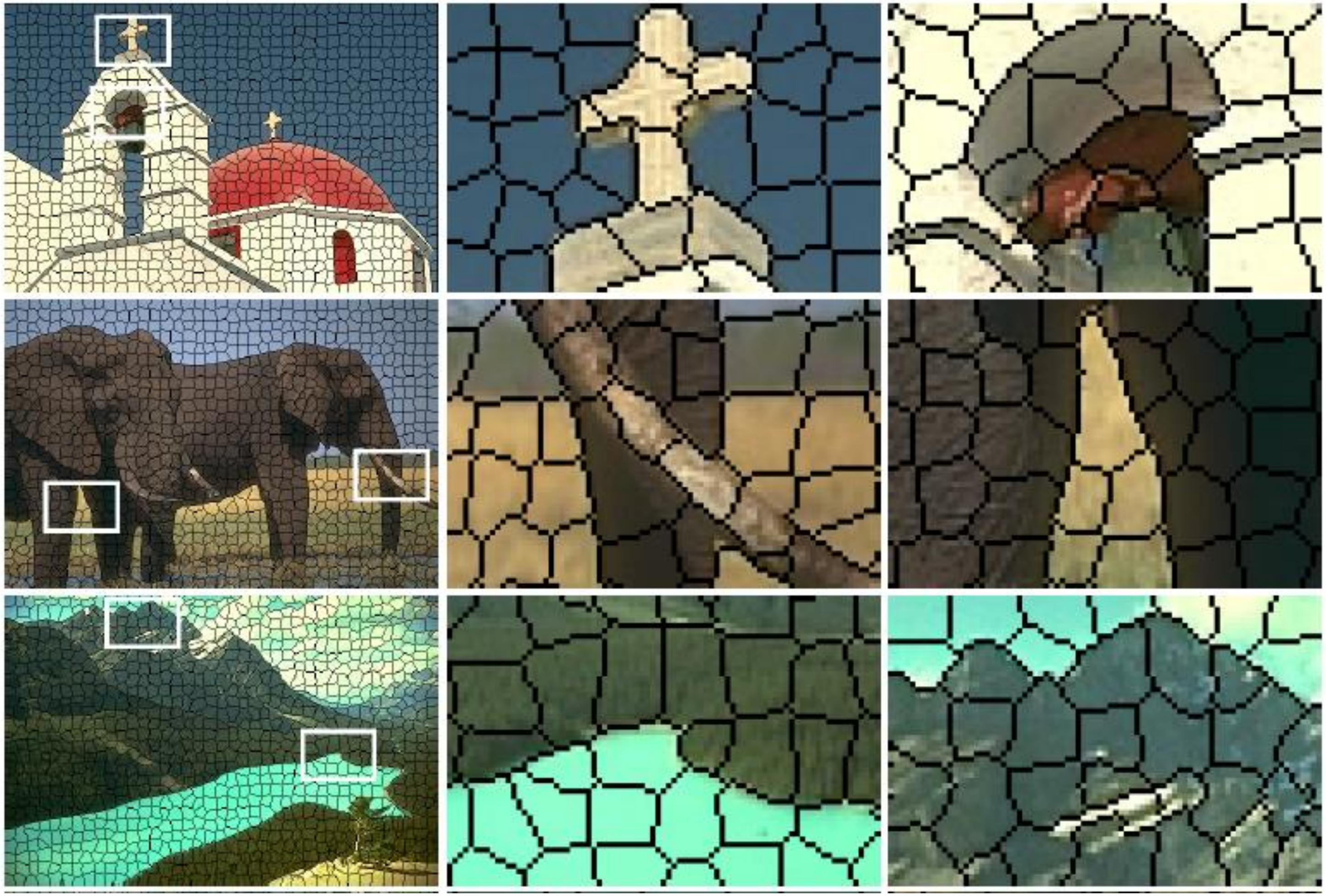
$B \leftarrow$ homotopic skeleton of Ψ^n;

return B

Turbo Pixels: Levinstein et al. 2009

http://www.cs.toronto.edu/~kyros/pubs/09.pami.turbopixels.pdf

Tries to preserve boundaries like watershed but to produce more regular regions

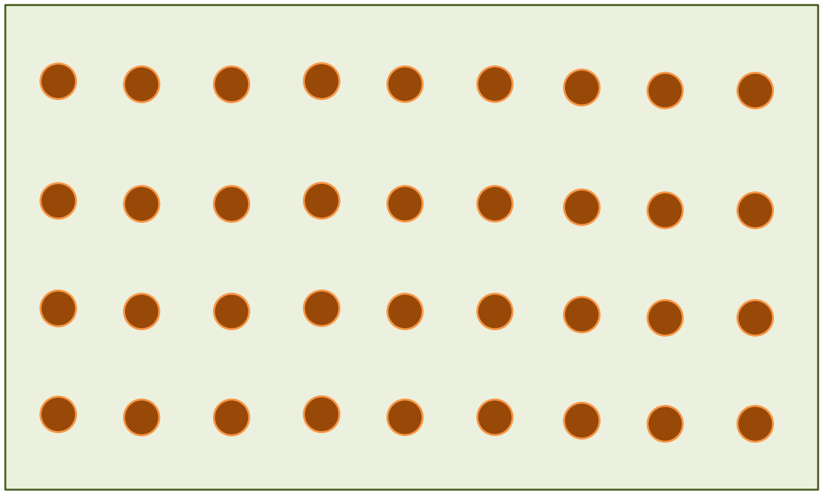


Slide: Derek Hoiem

SLIC
(Simple Linear Iterative Clustering)

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk
SLIC Superpixels
EPFL Technical Report 149300, June 2010

Clusters distributed uniformly:



$$[l, a, b, x, y]$$

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D = d_{lab} + a d_{xy} \qquad Eq.1$$

In CIELAB color space

Perceptually uniform for small changes

Simple Linear Iterative Clustering (SLIC)

Algorithm 1 Efficient superpixel segmentation

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: **repeat**
4: **for** each cluster center C_k **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
6: **end for**
7: Compute new cluster centers and residual error E $\{L1$ distance between previous centers and recomputed centers$\}$
8: **until** $E \leq$ threshold
9: Enforce connectivity.


REGAL


SPA

SLIC (Achanta et al. PAMI 2012)

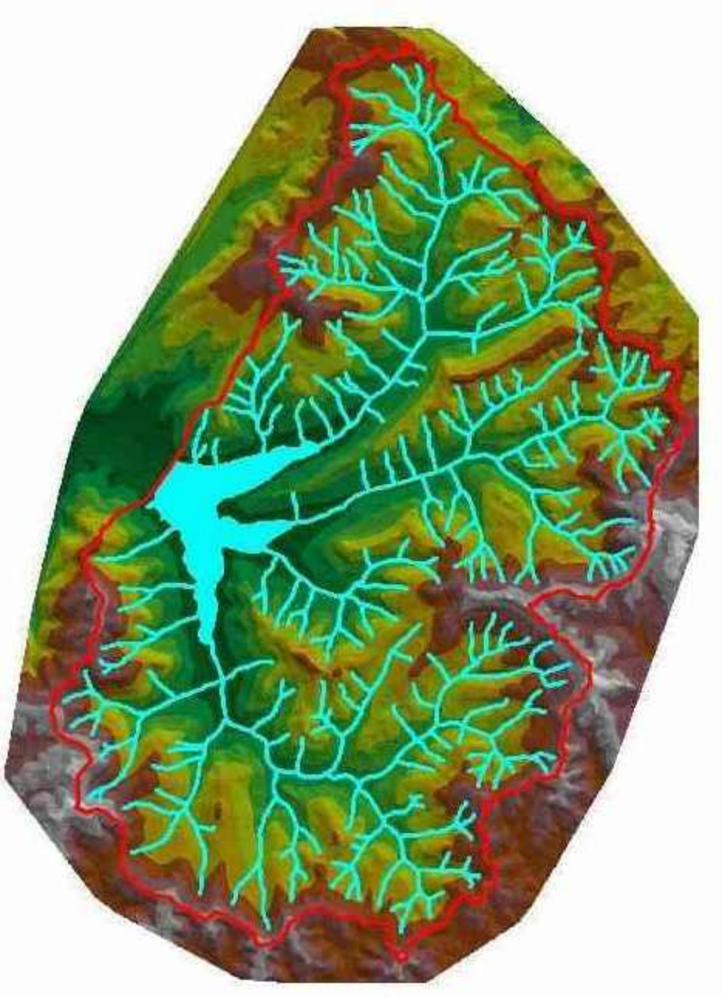
http://infoscience.epfl.ch/record/177415/files/Superpixel_PAMI2011-2.pdf

1. Initialize cluster centers on pixel grid in steps S
   - Features: Lab color, x-y position
2. Move centers to position in 3x3 window with smallest gradient
3. Compare each pixel to cluster center within 2S pixel distance and assign to nearest
4. Recompute cluster centers as mean color/position of pixels belonging to each cluster
5. Stop when residual error is small

+ Fast 0.36s for 320x240
+ Regular superpixels
+ Superpixels fit boundaries
- May miss thin objects
- Large number of superpixels

Slide: Derek Hoiem

Watershed Algorithm



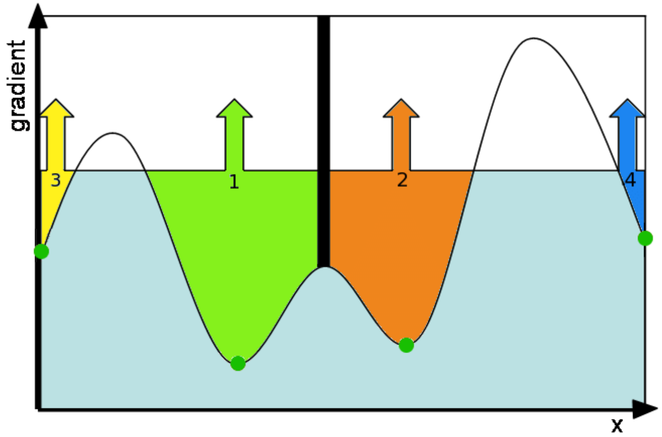


Image from: [researchgate]

Slide credit: Derek Hoiem

Meyer’s watershed segmentation

1. Choose local minima as region seeds
2. Add neighbors to priority queue, sorted by value
3. Take top priority pixel from queue
    1. If all labeled neighbors have same label, assign that label to pixel
    2. Add all non-marked neighbors to queue
4. Repeat step 3 until finished (all remaining pixels in queue are on the boundary)

Meyer 1991

Matlab: seg = watershed(bnd_im)

Slide: Derek Hoiem

Watershed usage

- Use as a starting point for hierarchical segmentation
  - Ultrametric contour map (Arbelaez 2006)

- Works with any soft boundaries
  - Pb (w/o non-max suppression)
  - Canny (w/o non-max suppression)
  - Etc.

Slide: Derek Hoiem

Watershed pros and cons

- Pros
  - Fast (< 1 sec for 512x512 image)
  - Preserves boundaries
- Cons
  - Only as good as the soft boundaries (which may be slow to compute)
  - Not easy to get variety of regions for multiple segmentations
- Usage
  - Good algorithm for superpixels, hierarchical segmentation

Slide: Derek Hoiem

Gestalt Psychology

Gestalt psychology or **gestaltism**

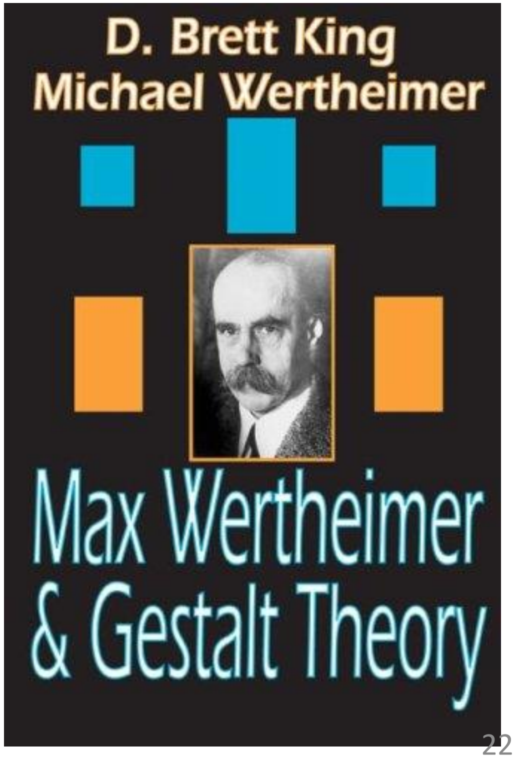
German: *Gestalt* - "form" or "whole”

Berlin School, early 20th century

Kurt Koffka, Max Wertheimer, and Wolfgang Köhler

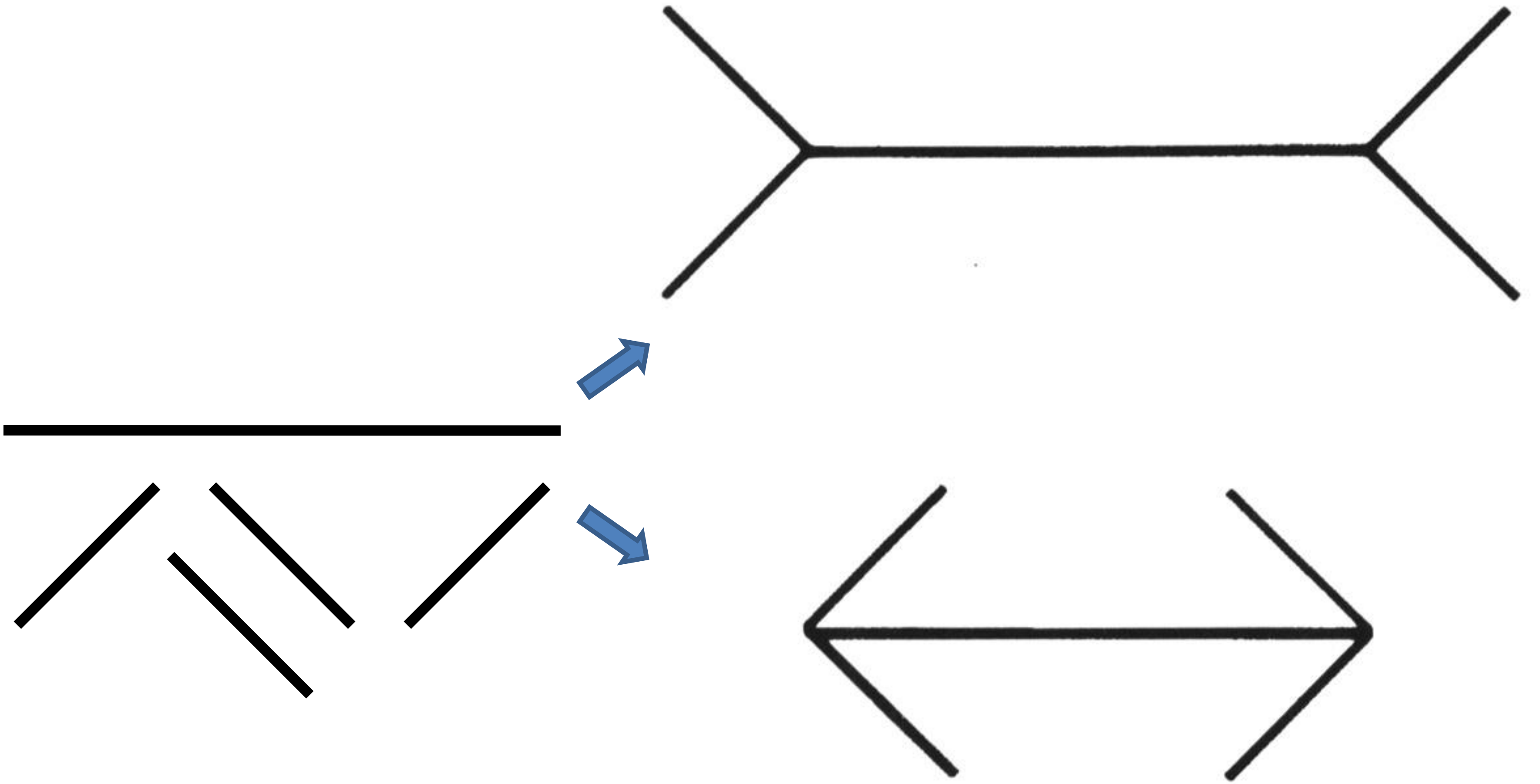
View of brain:

- whole is more than the sum of its parts
- holistic
- parallel
- analog
- self-organizing tendencies



Slide from S. Saverese

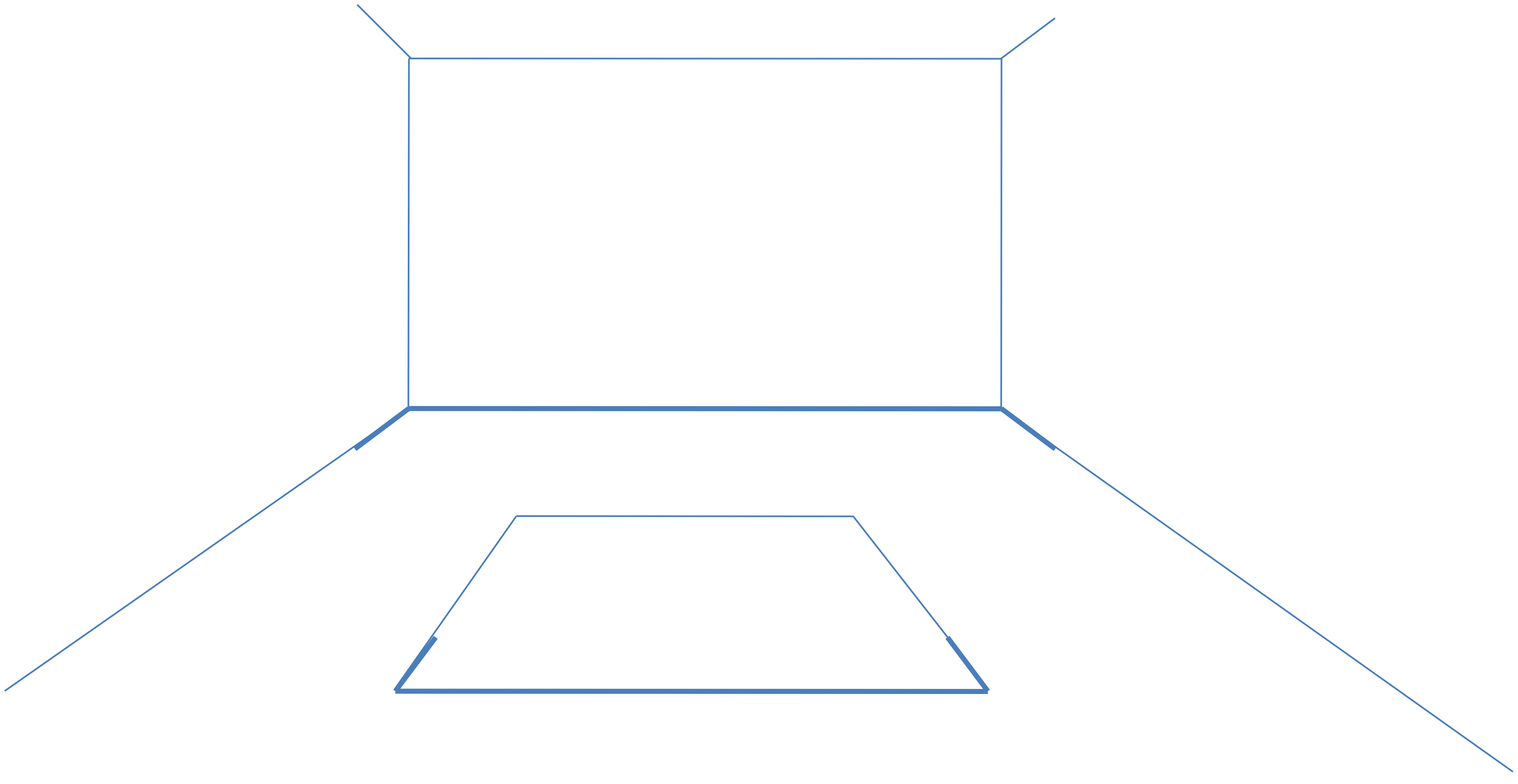
Gestaltism



The Muller-Lyer illusion

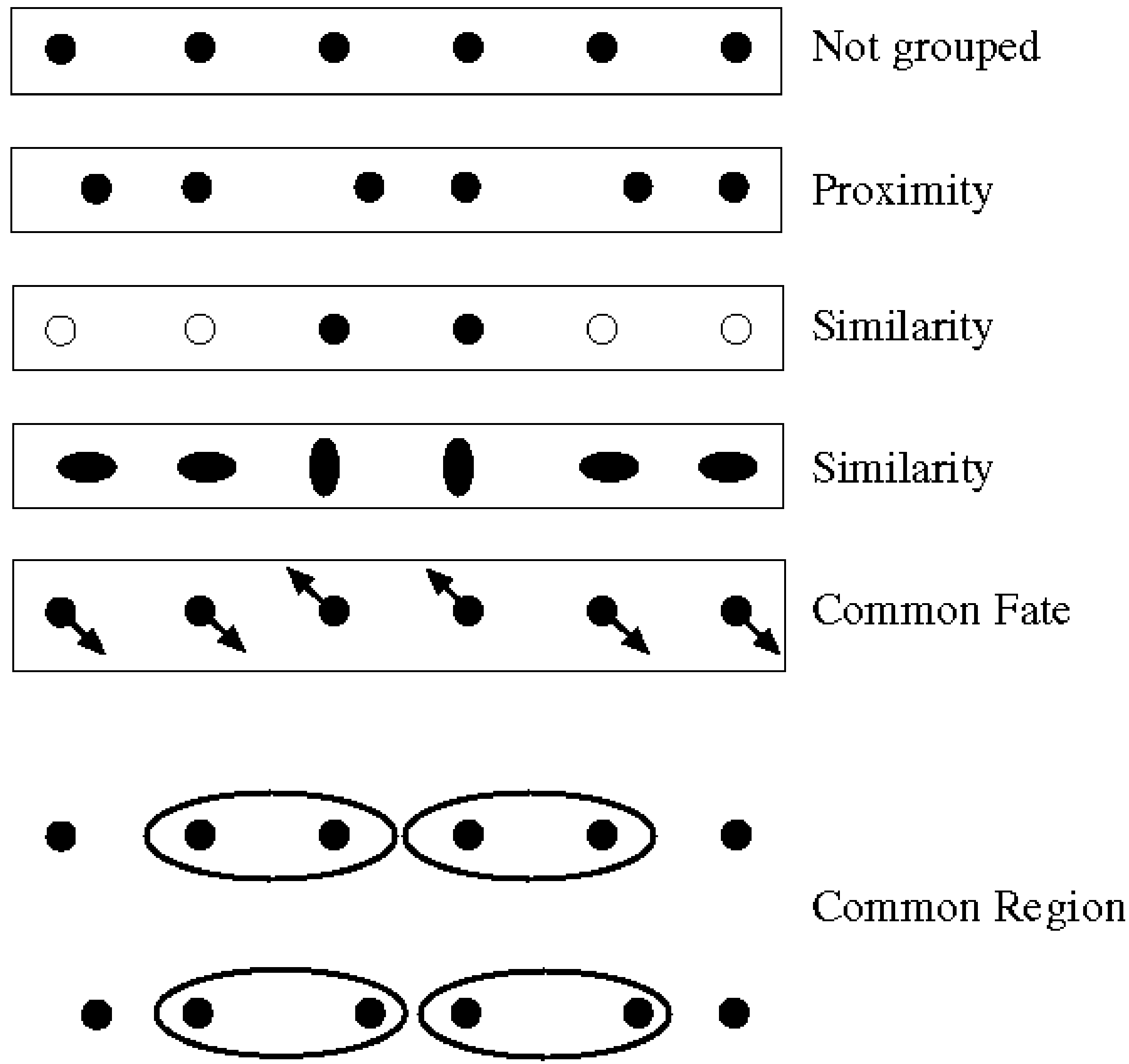
Slide: Derek Hoiem

We perceive the interpretation, not the senses



Slide: Derek Hoiem

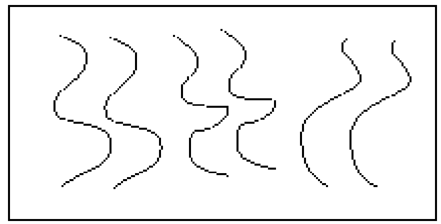
Principles of perceptual organization



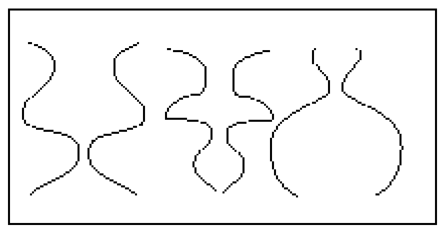
Slide: Derek Hoiem

From Steve Lehar: The Constructive Aspect of Visual Perception

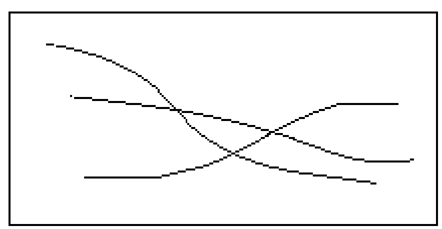
Principles of perceptual organization



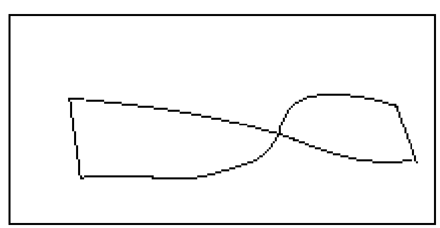
Parallelism



Symmetry



Continuity



Closure

Slide: Derek Hoiem

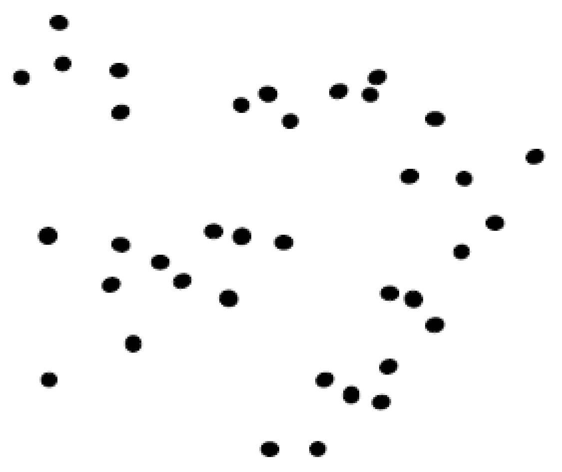
Gestalt cues

- Good intuition and basic principles for grouping
- Basis for many ideas in segmentation and occlusion reasoning
- Some (e.g., symmetry) are difficult to implement in practice

Slide: Derek Hoiem

Agglomerative Clustering

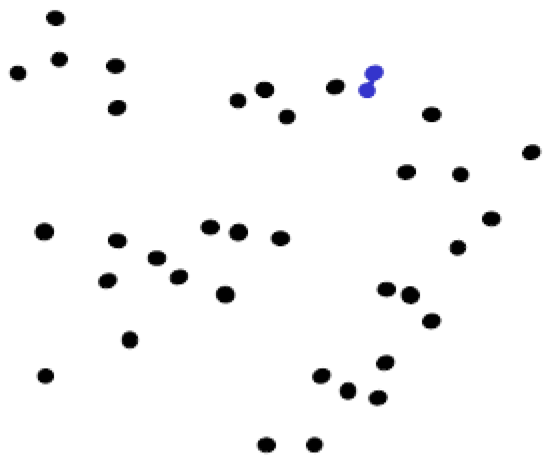
Agglomerative clustering



1. Say "Every point is its own cluster"

Slide: Derek Hoiem

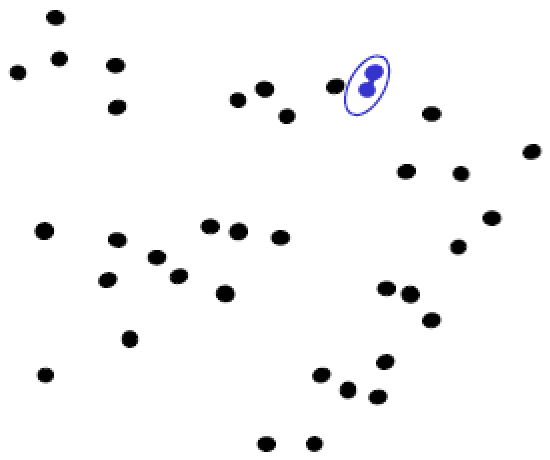
Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Slide: Derek Hoiem

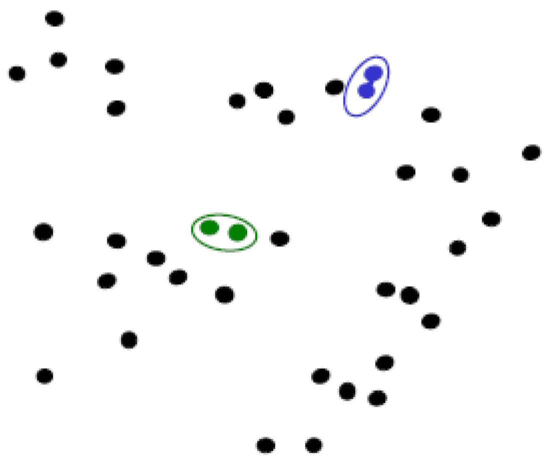
Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Slide: Derek Hoiem

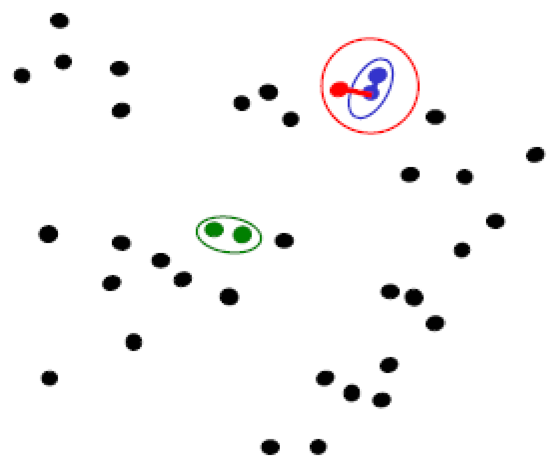
Agglomerative clustering



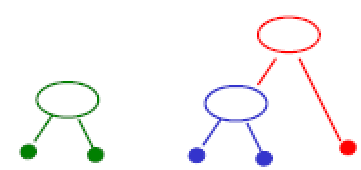
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Slide: Derek Hoiem

Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

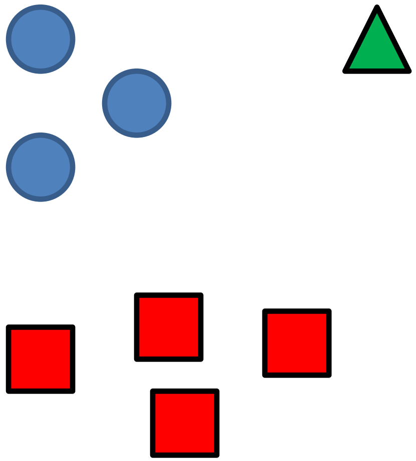


Slide: Derek Hoiem

Agglomerative clustering

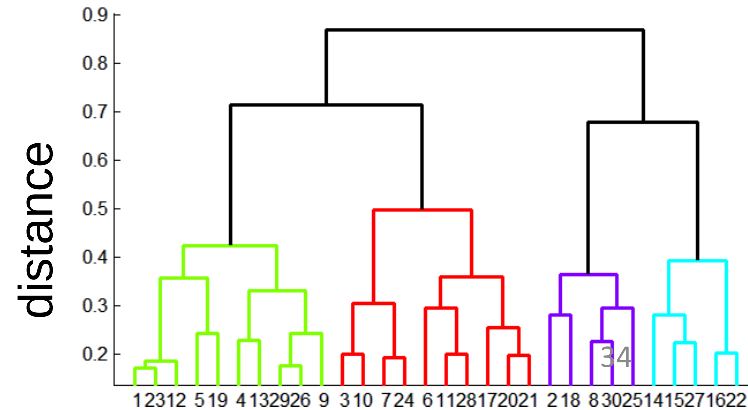
How to define cluster similarity?

- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids



How many clusters?

- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges



Slide: Derek Hoiem

Conclusions: Agglomerative Clustering

Good

- Simple to implement, widespread application
- Clusters have adaptive shapes
- Provides a hierarchy of clusters

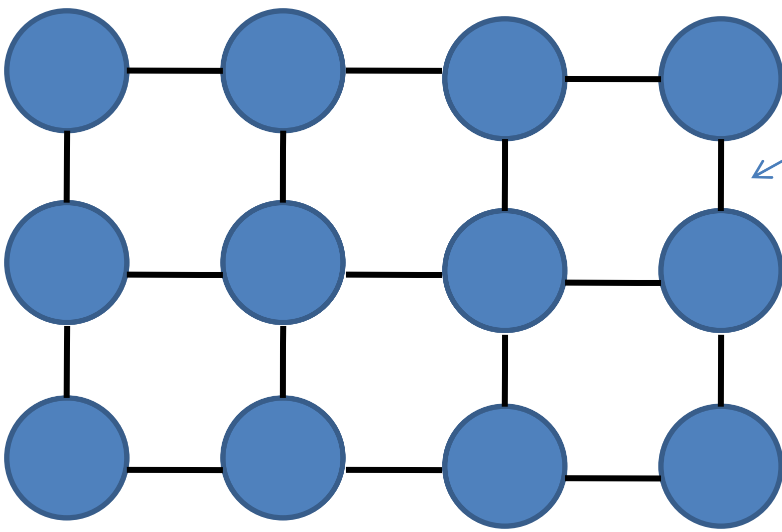
Bad

- May have imbalanced clusters
- Still have to choose number of clusters or threshold
- Need to use an “ultrametric” to get a meaningful hierarchy

Slide: Derek Hoiem

Intelligent Scissors

The Image as a Graph



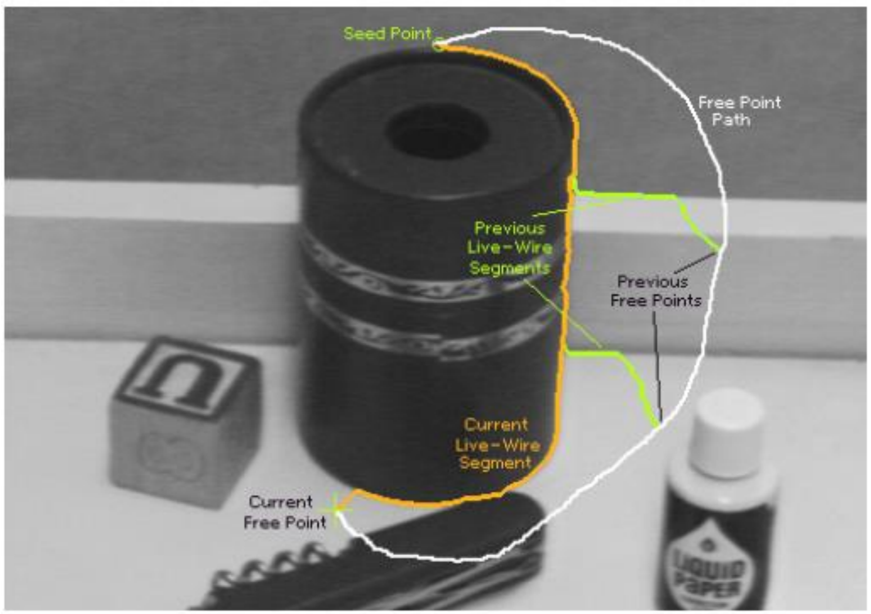
Node: pixel

Edge: cost of path or cut between two pixels

Slide: Derek Hoiem

Intelligent Scissors

Mortenson and Barrett (SIGGRAPH 1995)

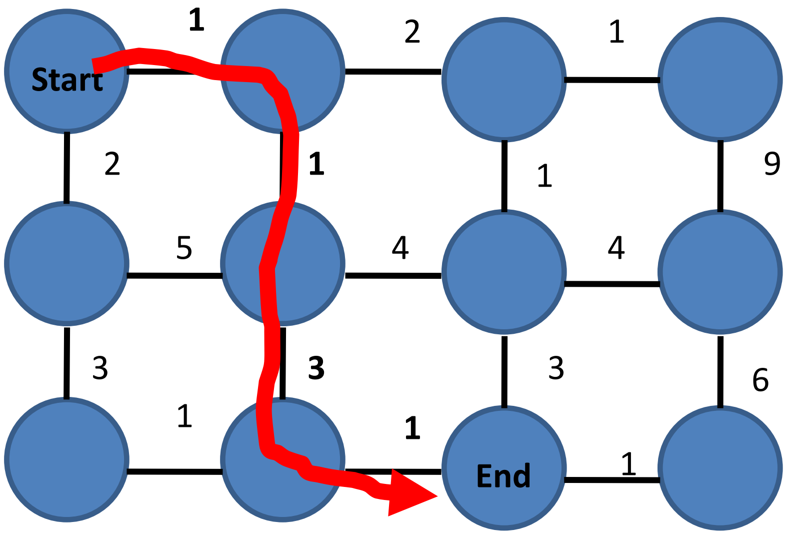


Slide: Derek Hoiem

Intelligent Scissors

Mortenson and Barrett (SIGGRAPH 1995)

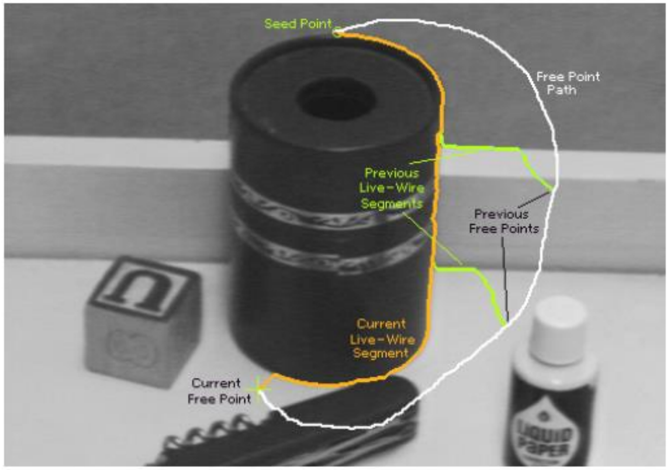
A good image boundary has a short path through the graph.



Slide: Derek Hoiem

Intelligent Scissors

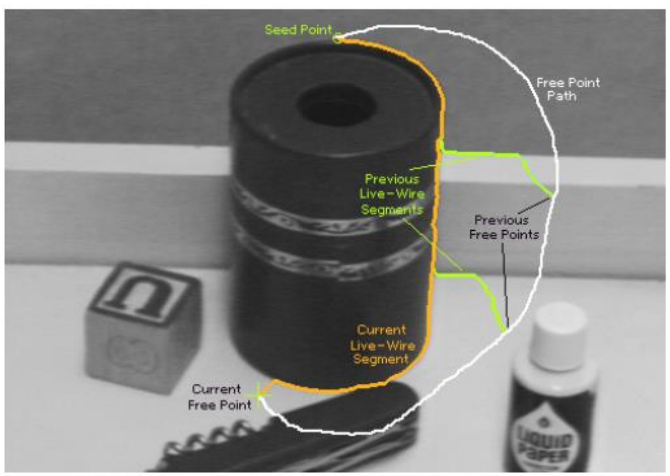
- Formulation: find good boundary between seed points
- Challenges
  - Minimize interaction time
  - Define what makes a good boundary
  - Efficiently find it



Slide: Derek Hoiem

Intelligent Scissors: method

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
4. Get path from seed to cursor, choose new seed, repeat



Slide: Derek Hoiem

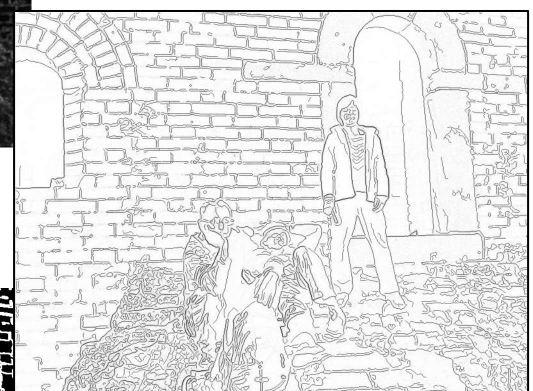
Intelligent Scissors: method

1. Define boundary cost between neighboring pixels
   a) Lower if edge is present (e.g., with $edge(im, \text{‘}canny\text{’})$)
   b) Lower if gradient is strong
   c) Lower if gradient is in direction of boundary

Slide: Derek Hoiem

Gradients, Edges, and Path Cost

Gradient Magnitude



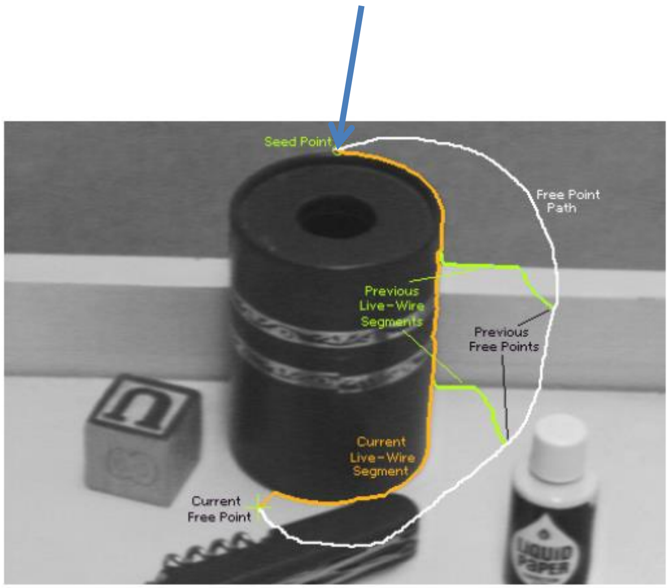
Path Cost

Edge Image

Slide: Derek Hoiem

Intelligent Scissors: method

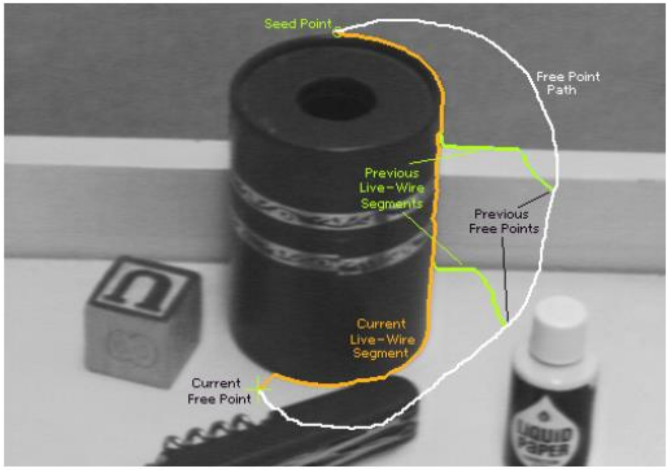
1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
   – Snapping



Slide: Derek Hoiem

Intelligent Scissors: method

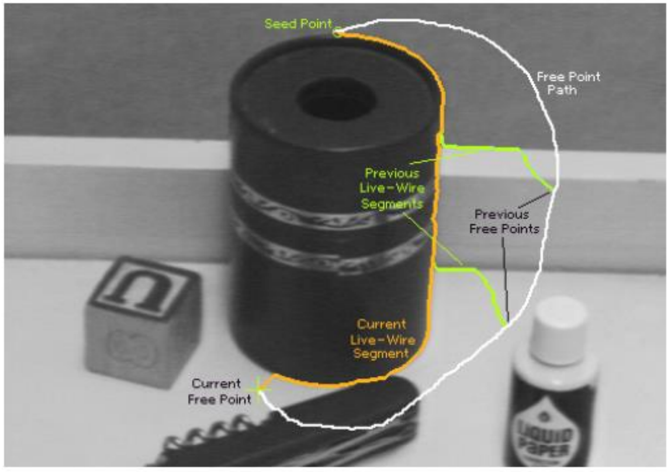
1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
   – Djikstra's shortest path algorithm



Slide: Derek Hoiem

Intelligent Scissors: method

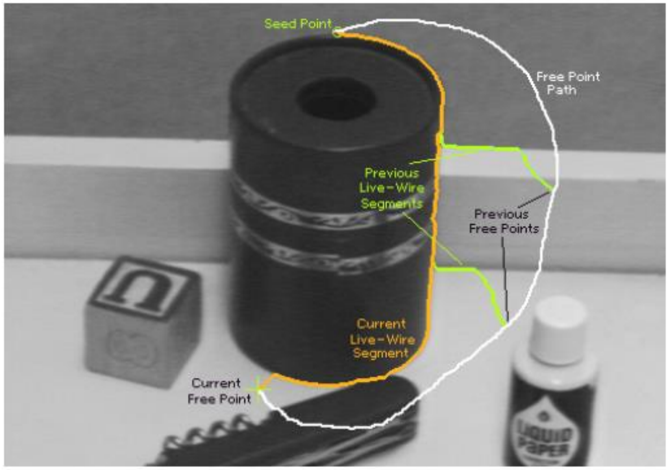
1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
4. Get new seed, get path between seeds, repeat



Slide: Derek Hoiem

Intelligent Scissors: improving interaction

1. Snap when placing first seed
2. Automatically adjust to boundary as user drags
3. Freeze stable boundary points to make new seeds



Slide: Derek Hoiem