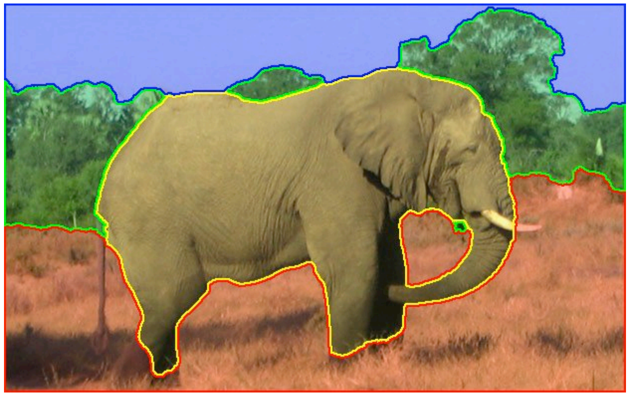
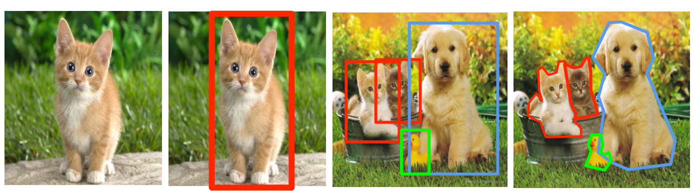
Image Segmentation

Goal: Label every pixel with its semantic category



Other Related Problems

Classification | Classification + Localization | Object Detection | Instance Segmentation



CAT | CAT | CAT, DOG, DUCK | CAT, DOG, DUCK

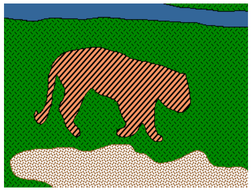
Single object | Multiple objects

F.-F. Li, A. Karpathy, J. Johnson

Applications

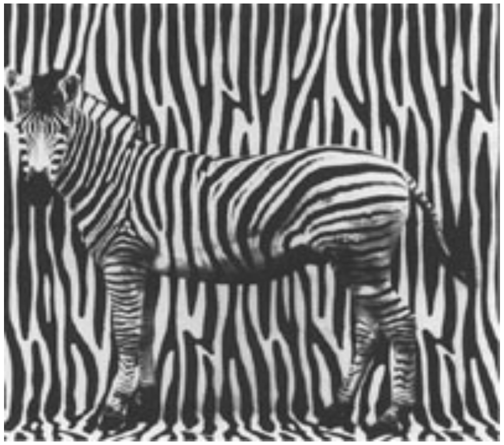
- Background Subtraction
- Cut-and-paste
- Object search
- Scene description

From Images to Objects



- What Defines an Object?
- How to Find Objects?
  - Find **boundaries** between objects
  - Find "cohesive" **regions**

Some Hard Examples



What is Segmentation?

- **Clustering image elements that "belong together"**
  - **Partitioning**
    - Divide into regions with coherent internal properties
  - **Grouping**
    - Identify sets of coherent tokens in image

- **Tokens:** Whatever we need to group
  - Pixels
  - "Superpixels" (regions with small range of color or texture)
  - Features (corners, lines, etc.)

Some Criteria for Segmentation

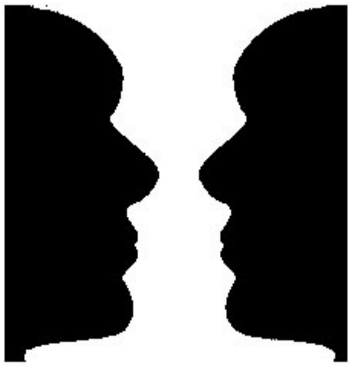
- Pixels *within a region* should have similar appearance, i.e., the statistics of their pixel intensities, colors, textures, etc. should fit some model. Region should be compact.

- The *boundaries between regions* should
  - Have discontinuities in color or texture or …
  - Smooth or piecewise smooth or …

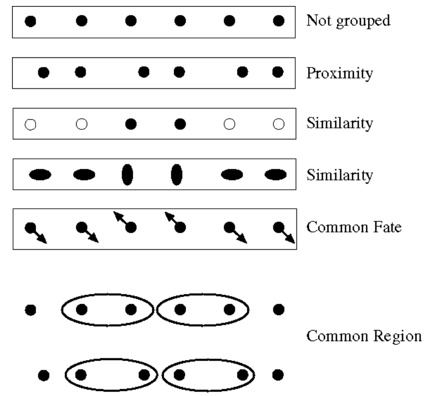
Approaches to Image Segmentation

- Segmentation by Humans
- Manual Segmentation with Games
- Automatic Segmentation Methods
- Interactive Segmentation Methods

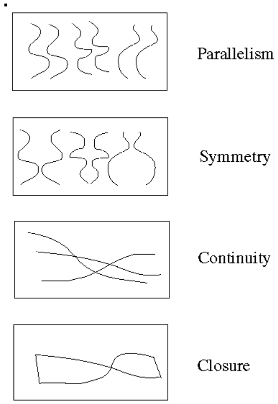
Human Vision: Figure-Ground Segmentation



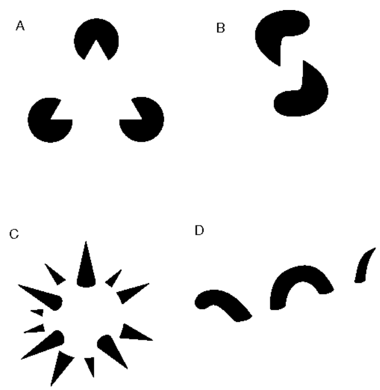
For the human visual system, Gestalt psychology identifies several properties that result in grouping/ segmentation:



For the human visual system, Gestalt psychology identifies several properties that result in grouping/ segmentation:



Groupings by Invisible Completions



* Images from Steve Lehar's Gestalt papers: http://cns-alumni.bu.edu/pub/slehar/Lehar.html

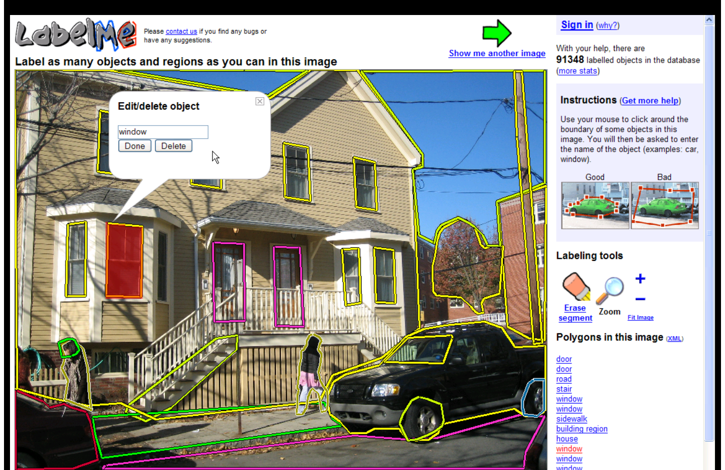
Image Segmentation by Humans

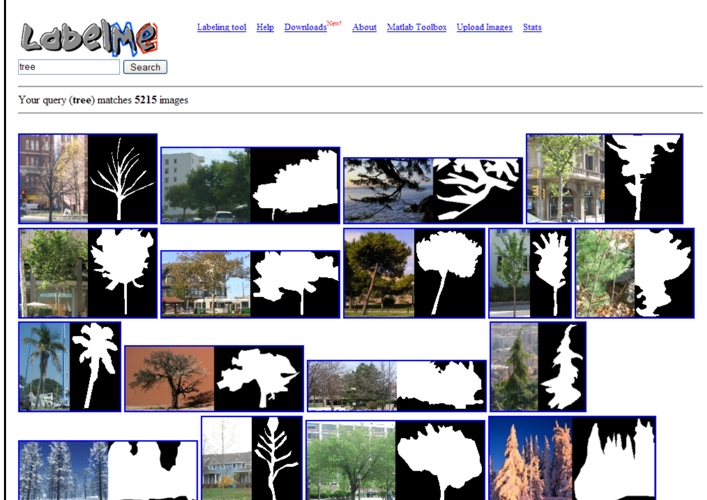
LabelMe

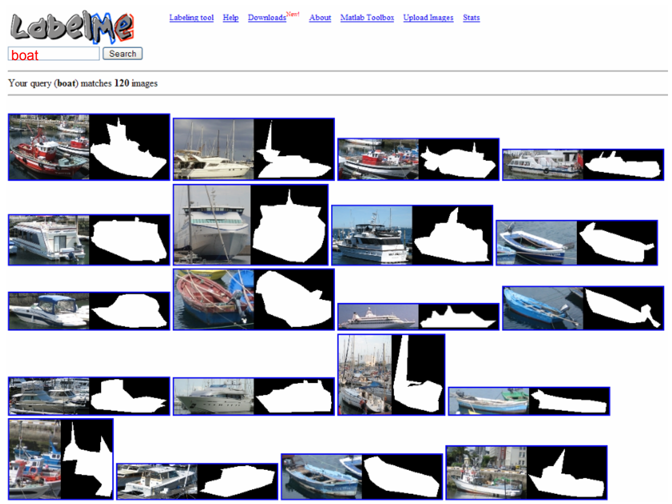
http://labelme.csail.mit.edu/

LabelMe Goals

- The goal of LabelMe is to provide an online annotation tool to build a large database of annotated (= **roughly segmented and labeled**) images by collecting contributions from many people
- Large set of scenes (indoor, outdoor) and many object classes in context
- Collect the large, high quality database of annotated images
- Images come from multiple sources, taken at many cities/countries (to help avoid overfitting)
- Allow researchers immediate access to the latest version of the database
- LabelMe Matlab toolbox





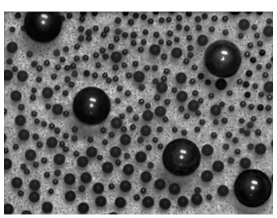
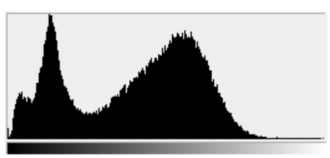


Segmentation as Clustering

- Cluster together (pixels, tokens, etc.) that belong together
- **Agglomerative clustering**
  - attach to cluster it is closest to
  - repeat
- **Divisive clustering**
  - split cluster along best boundary
  - repeat

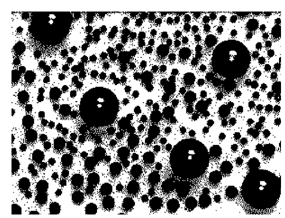
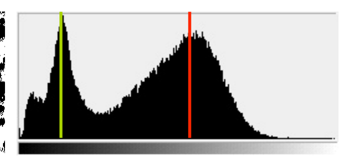
Histogram-Based Segmentation:
A Simple Agglomerative Clustering Method

- Goal
  - Segment the image into K regions
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color

Histogram-Based Segmentation:
A Simple Agglomerative Clustering Method

- Goal
  - Segment the image into K regions
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color

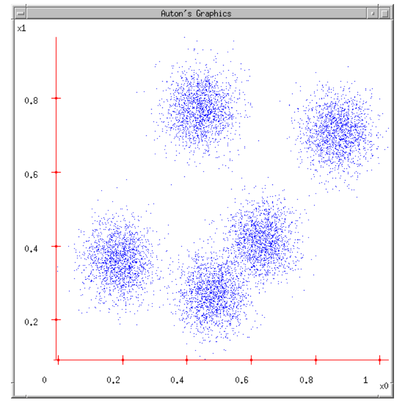
K-Means Clustering

- K-means clustering algorithm
  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 in cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, go to Step 2

- Properties
  – Will always converge to *some* solution
  – Can be a "local minimum"
    - does not always find the global minimum of objective function:
      $$\sum_{\text{clusters } i} \; \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

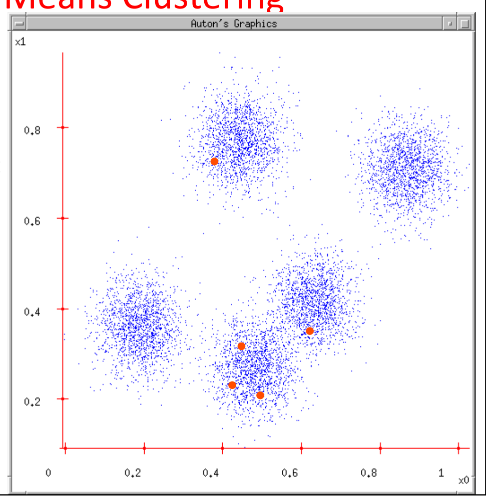
K-Means Clustering

- The dataset. Input k=5



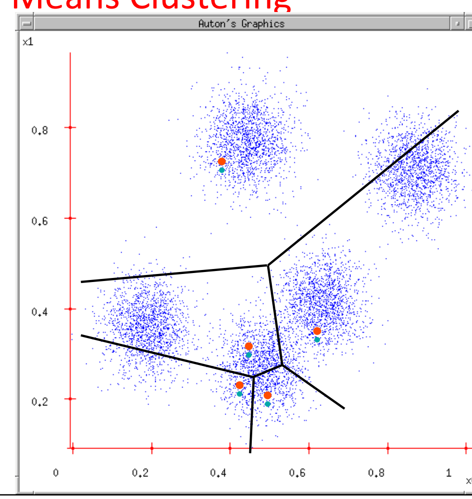
K-Means Clustering

Randomly pick 5 positions as initial cluster centers (not necessarily data points)



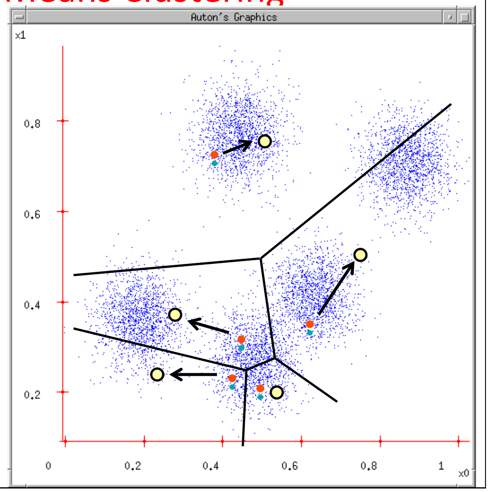
K-Means Clustering

Each point finds which cluster center it is closest to; the point belongs to that cluster



K-Means Clustering

Each cluster computes its new centroid based on which points belong to it



K-Means Clustering

Each cluster computes its new centroid, based on which points belong to it

Repeat until convergence (i.e., no cluster center moves)

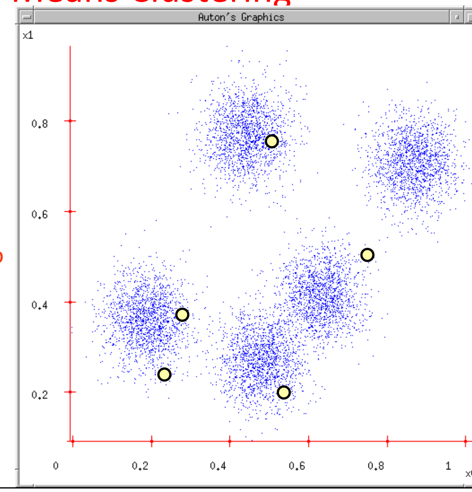


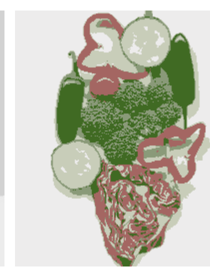
Image Segmentation using K-Means

- Select a value of K
- Select a feature vector for every pixel (color, texture, position, etc.)
- Define a similarity measure between feature vectors (e.g., Euclidean Distance)
- Apply K-Means algorithm
- Apply Connected Components algorithm
- Merge any components of size less than some threshold to an adjacent component that is most similar to it

Example

Image

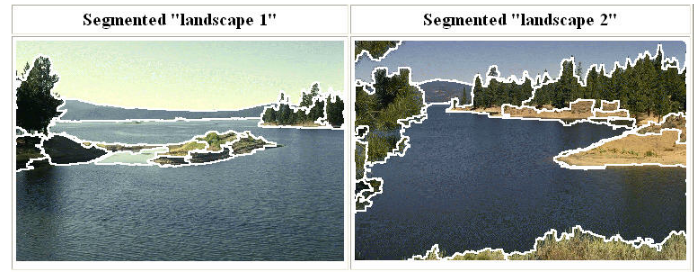
Clusters on intensity



Clusters on color

Mean Shift Algorithm

D. Comaniciu and P. Meer

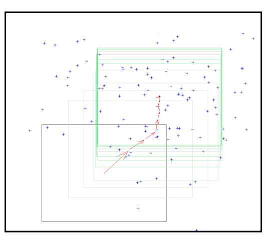


Mean Shift Algorithm

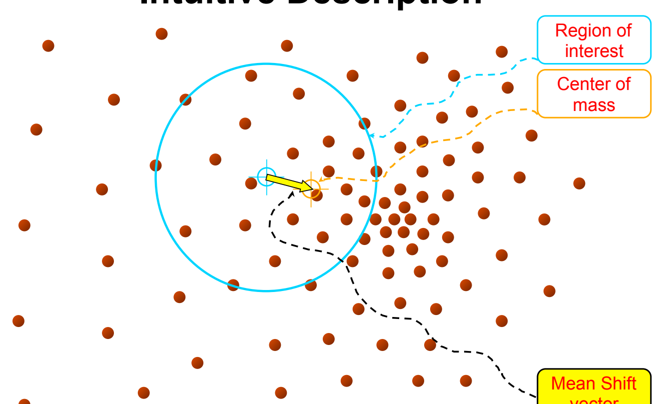
Mean Shift Algorithm

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***, i.e., point of highest density of a data distribution:



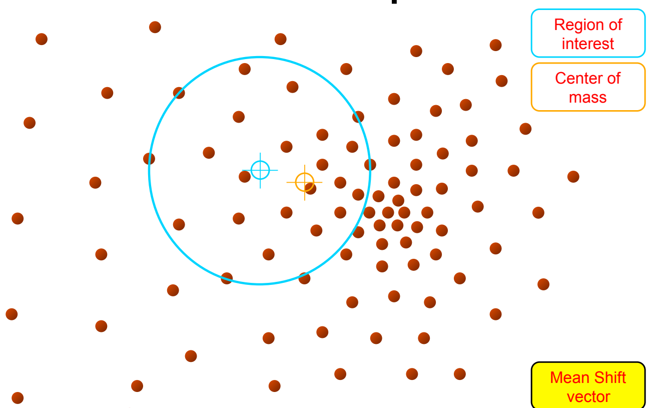
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

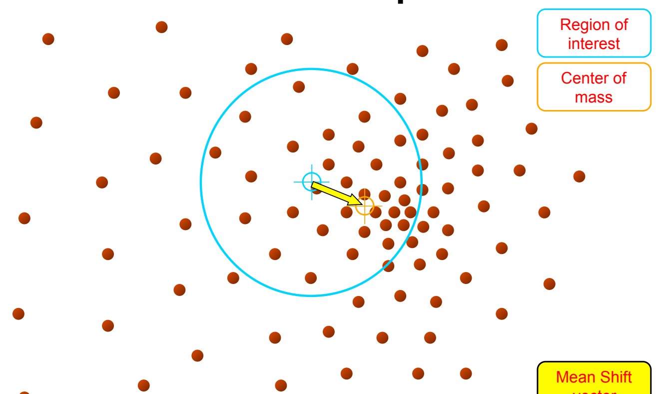
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

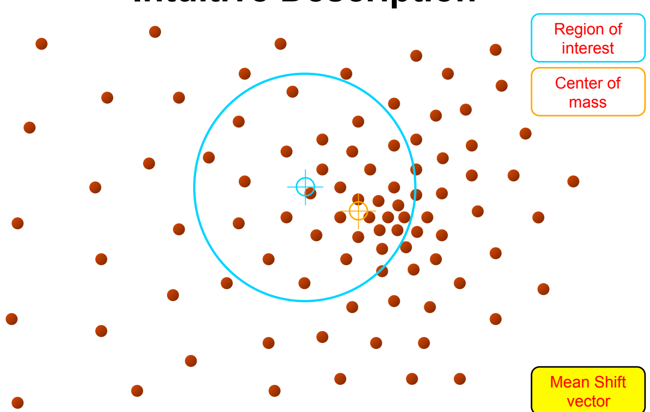
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

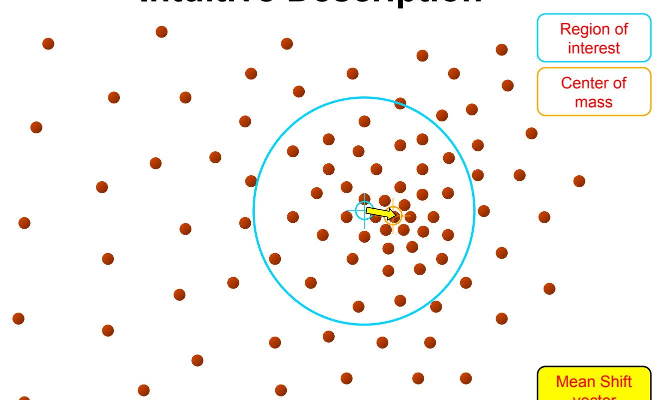
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

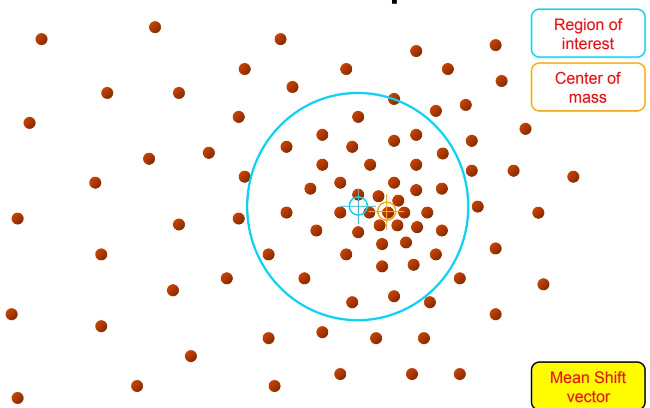
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

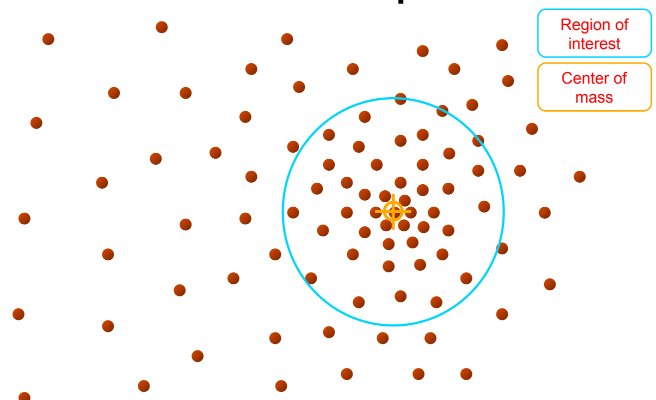
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

Intuitive Description

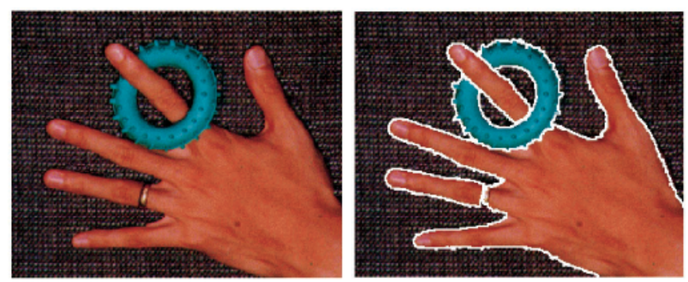


Objective : Find the densest region

Distribution of identical billiard balls

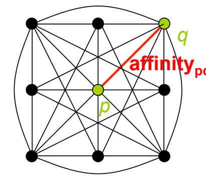
Results

Results



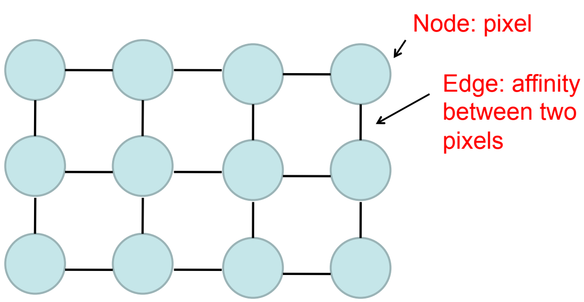
Results

Graph-based Clustering: Images as Graphs



- *Fully-connected* graph
  - node for every pixel
  - link between pairs of pixels, p,q
  - cost **affinity**$_{pq}$ for each link
    - **affinity**$_{pq}$ measures *similarity*
      - similarity is *inversely proportional* to difference in color, texture, etc.

The Image as a Graph



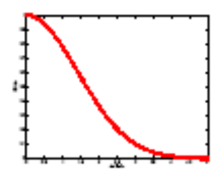
Affinity (Similarity) Measures

- Intensity

$$\mathrm{aff}(\mathbf{x},\mathbf{y}) = e^{-\|I(\mathbf{x})-I(\mathbf{y})\|^2/2\sigma_I^2}$$

- Distance

$$\mathrm{aff}(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma_d^2}$$



- Color
- Texture
- Motion

Problem Formulation

- Given an undirected graph G = (*V*, *E*), where *V* is a set of nodes, one for each data element (e.g., pixel), and *E* is a set of edges with weights representing the affinity between connected nodes
- **Find the image partition that maximizes the "similarity" within each region *and* minimizes the "dissimiliarity" between regions**
- Finding the optimal partition is NP-complete

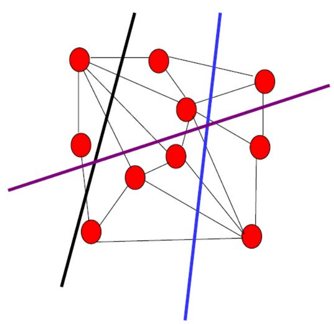
- Let A, B partition G. Therefore, A ∪ B = *V*, and A ∩ B = ∅
- The **dissimilarity** between A and B is defined as

$$cut(A,B) = \sum_{i \in A,\, j \in B} affinity_{ij}$$

= total weight of edges removed

- The **optimal bi-partition** (i.e., segment image into 2 regions) of G is the one that minimizes *cut*

Minimum Cut



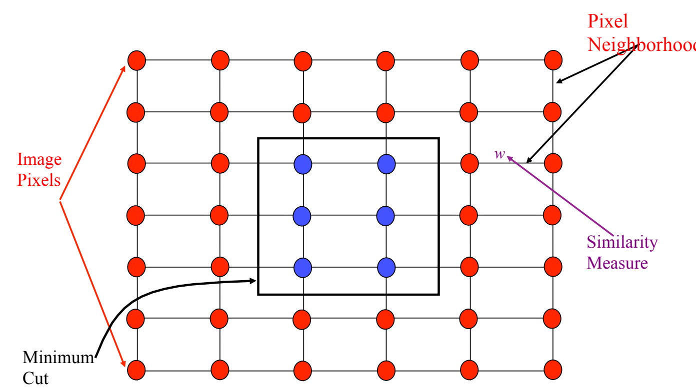
A cut of a graph *G* is the set of edges *S* such that removal of *S* from *G* disconnects *G*.

Minimum cut is the cut of minimum weight, where weight of cut <A,B> is given as

$$w(\langle A,B\rangle) = \sum_{x \in A, y \in B} w(x,y)$$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

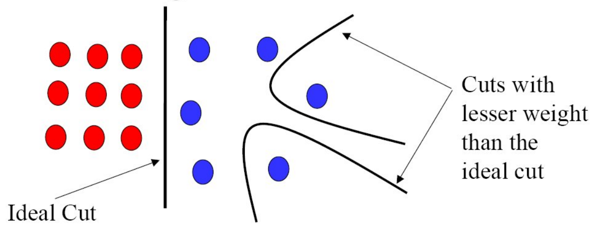
Image Segmentation & Minimum Cut



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Drawbacks of Minimum Cut

- Weight of cut is directly proportional to the number of edges in the cut.



64

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

- So, instead define the **normalized** similarity, called the ***normalized-cut***(A, B), as

$$ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)}$$

where *assoc*(*A*,*V*) = $\sum_{i \in A, k \in V} affinity_{ik}$

= total connection weight from nodes in *A* to *all nodes in G*

- *Ncut* removes the bias based on region size
- New goal: Find bi-partition that minimizes *ncut*(A, B)
- Can be found in polynomial time in number of pixels

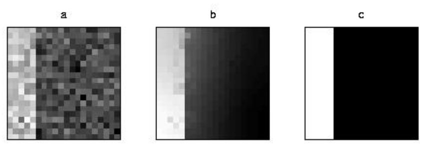


Figure 16: A synthetic image showing a noisy "step" image. Intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.2$ is added. Subplot (b) shows the eigenvector with the second smallest eigenvalue, and subplot (c) shows the resulting partition.

Synthetic Image Results

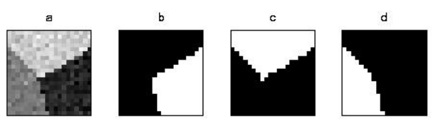
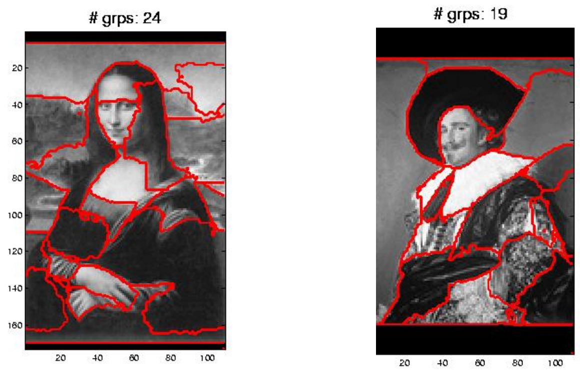


Figure 17: (a) A synthetic image showing three image patches forming a junction. Image intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.1$ is added. (b)-(d) shows the top three components of the partition.

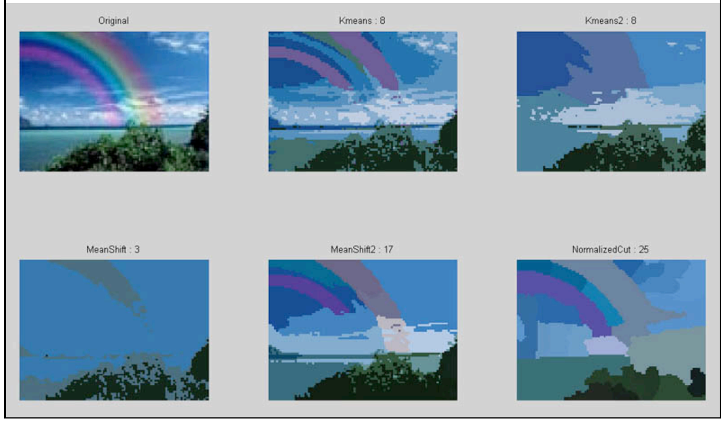
Results

(a) Input image (b) 2nd Eigenvector (c) Segmentation

Results



K-Means vs. Mean Shift vs. Normalized Cut



Some Weaknesses of Normalized Cut

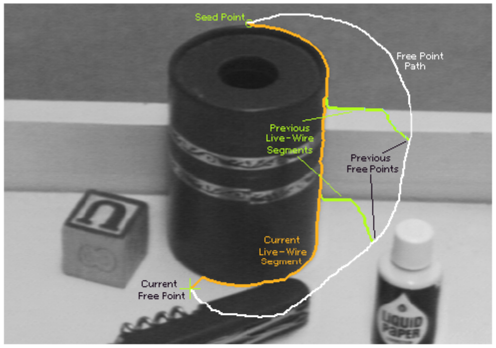
- Very large storage requirement
- Bias towards partitioning into equal segments
- Often over-segments
- Has problems with textured backgrounds

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Interactive Image Segmentation

- Boundary-based methods
  - Intelligent scissors
    - Uses local edge information
  - Snakes / Active contours
    - Uses local edge information and contour smoothness
- Graph-cut methods
  - GrabCut
    - Uses boundary and region terms

Intelligent Scissors



E. N. Mortensen and W. A. Barrett, Intelligent Scissors for Image Composition, in *Proc. SIGGRAPH*, 1995

Similar to Photoshop's "Magnetic Lasso" tool

Intelligent Scissors

- Approach answers a **basic** question
  - Q: How to find a *path from seed to mouse* that follows object **boundary** as closely as possible?
  - A: Define a path that stays as close as possible to edges

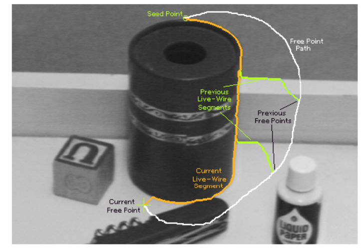
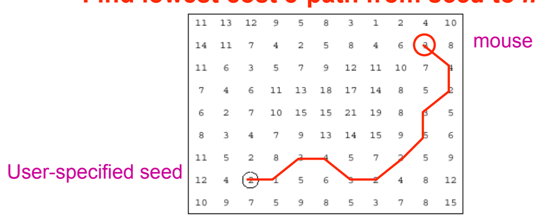


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Intelligent Scissors

- Basic Idea
  - Define edge score for each pixel
    - edge pixels have low cost
  - **Find lowest cost 8-path from *seed* to *mouse***



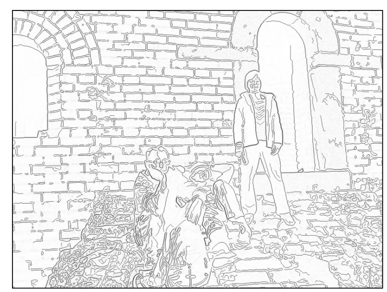
Questions

- How to define costs?
- How to find the path?

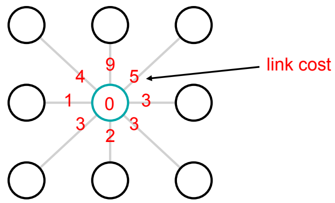
Intelligent Scissors

Define boundary cost between neighboring pixels

a) Lower if edge is present (e.g., with edge(im, 'canny'))
b) Lower if gradient is strong
c) Lower if gradient is in direction of boundary



Dijkstra’s Shortest Path Algorithm

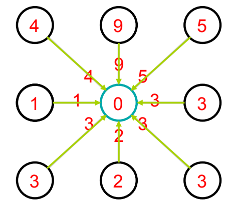


Algorithm
1. initialize node costs to ∞; set ***p*** = seed point, cost(***p***) = 0
2. expand ***p*** as follows:

   foreach of ***p***’s neighbors, ***q***, that are not already expanded

   set cost(***q***) = min(cost(***p***) + c_{pq} , cost(***q***))

Dijkstra’s Shortest Path Algorithm



Algorithm
1. initialize node costs to ∞; set ***p*** = seed point, cost(***p***) = 0
2. expand ***p*** as follows:

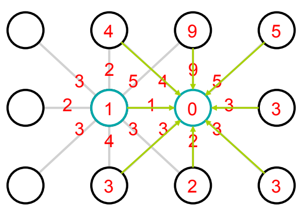
   foreach of ***p***’s neighbors, ***q***, that are not expanded

   set cost(***q***) = min(cost(***p***) + c_{pq} , cost(***q***))

   » if ***q***’s cost changed, make ***q*** point back to ***p***

   put ***q*** on the ACTIVE list (if not already there)

Dijkstra’s Shortest Path Algorithm



Algorithm
1. initialize node costs to ∞; set ***p*** = seed point, cost(***p***) = 0
2. expand ***p*** as follows:

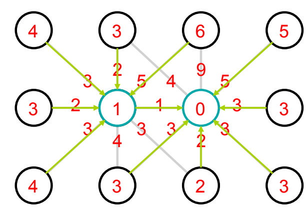
   foreach of ***p***’s neighbors, ***q***, that are not expanded

   » set cost(***q***) = min(cost(***p***) + c_{pq} , cost(***q***))

   » if ***q***’s cost changed, make ***q*** point back to ***p***

   » put ***q*** on the ACTIVE list (if not already there)
3. set ***r*** = node with minimum cost on the ACTIVE list
4. goto Step 2 with ***p*** = ***r***

Dijkstra’s Shortest Path Algorithm



Algorithm
1. initialize node costs to ∞; set ***p*** = seed point, cost(***p***) = 0
2. expand ***p*** as follows:

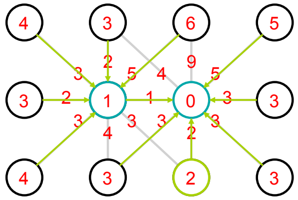
   foreach of ***p***’s neighbors, ***q***, that are not expanded

   set cost(***q***) = min(cost(***p***) + c_{pq} , cost(***q***))

   » if ***q***’s cost changed, make ***q*** point back to ***p***

   put ***q*** on the ACTIVE list (if not already there)
3. set ***r*** = node with minimum cost on the ACTIVE list
4. goto Step 2 with ***p*** = ***r***

Dijkstra's Shortest Path Algorithm



Algorithm

1. initialize node costs to ∞; set ***p*** = seed point, cost(***p***) = 0
2. expand ***p*** as follows:

   foreach of ***p***'s neighbors, ***q***, that are not expanded

   » set cost(***q***) = min(cost(***p***) + c_{pq} , cost(***q***))

   » if ***q***'s cost changed, make ***q*** point back to ***p***

   » put ***q*** on the ACTIVE list (if not already there)
3. set ***r*** = node with minimum cost on the ACTIVE list
4. goto Step 2 with ***p*** = ***r***

Dijkstra's Shortest Path Algorithm

– Computes the minimum cost path from the seed to *every* node in the graph. This set of minimum paths is represented as a *tree*

– Running time, with N pixels:

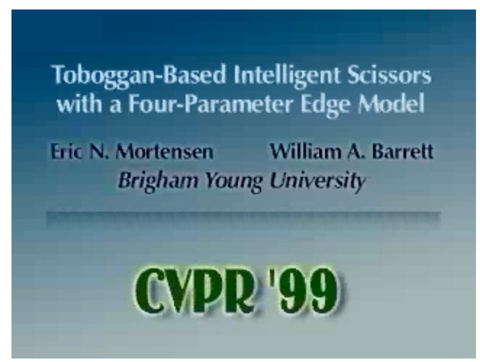
- $O(N^2)$ time if you use an active list
- $O(N \log N)$ if you use an active priority queue (heap)
- takes < second for a typical image

– Once this tree is computed, we can extract the optimal path from any point to the seed in $O(N/2)$ time

- it runs in real time as the mouse moves

Results

Results



Show video

Segmentation using Graph Cut

- Interactive image segmentation using graph cut
- Binary labeling problem: foreground vs. background
- User labels some pixels
- Exploit
  - – Statistics of known, labeled Foreground and Background pixels
  - – Smoothness of boundary
- Turn into discrete graph optimization problem
  - – Graph cut (min cut / max flow)

Graph-Cut Segmentation

Boykov and Jolly, Proc. ICCV, 2001

minimize $$E(L) = R(L) + \lambda \cdot S(L)$$

- *L* is a vector specifying the assignment of each pixel *p* as either foreground (F) or background (B)
- *R*(*L*) defines a region term specifying penalties for assigning L_p to F or B
- *S*(*L*) describes the boundary properties of the segmentation, $S_{\{p,q\}}$ is large when *p* and *q* are similar, and is close to 0 when *p* and *q* are very different

Energy Function

- Binary labeling: one value per pixel, F or B
- Energy (labeling) = region + boundary smoothness
  - – Will be minimized
- Region: for each pixel
  - – Probability that this intensity belongs to F (resp. B)
- Boundary:
  for each neighboring pixel pair
  - – Penalty for having different label
  - – Penalty is down-weighted if the two pixel intensities are very different

One labeling:

F	B	B
F	B	B
F	B	B

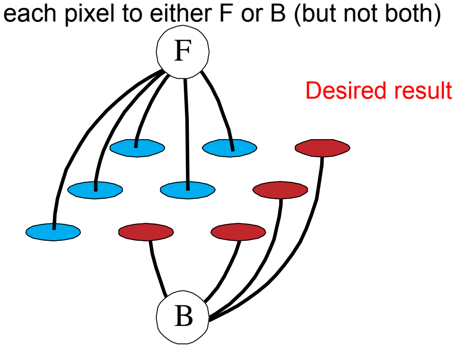
Data:

F	B	B
F	B	B
F	B	B

Labeling as a Graph Problem

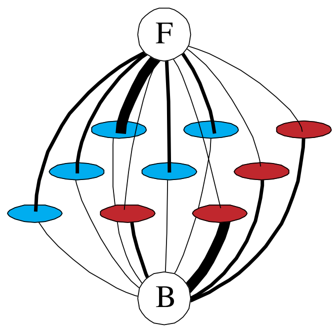
- Each pixel = node
- Add two more nodes: F and B
- Labeling: link each pixel to either F or B (but not both)

F	F	B
F	F	B
F	B	B



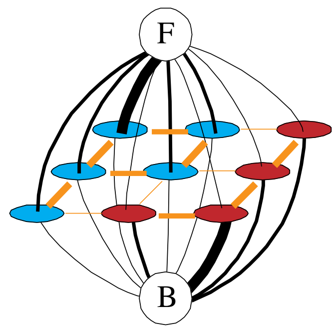
Region Term

- Put one edge between each pixel and both F and B
- Weight of edge = $-R(L_i)$



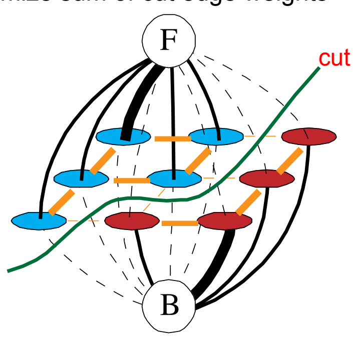
Boundary Term

- Add an edge between each neighboring pair
- Weight = S



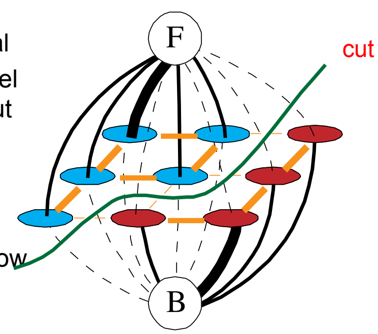
Min Cut

- Energy optimization equivalent to graph min-cut
- **Cut: remove edges to disconnect F from B**
- Minimum: minimize sum of cut edge weights



Min Cut ⇔ Labeling

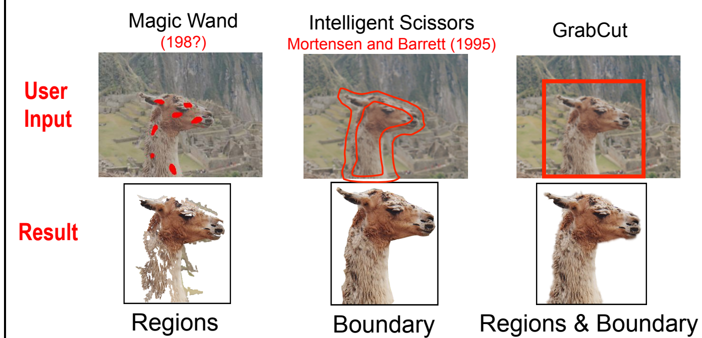
- In order to be a cut:
  - For each pixel, either the F or B edge has to be cut
- In order to be minimal
  - Only one edge label per pixel can be cut
- Can be solved in polynomial time
- Equivalent to Max-Flow problem



GrabCut
Interactive Image Segmentation

Carsten Rother
Vladimir Kolmogorov
Andrew Blake
Antonio Criminisi
Geoffrey Cross

What GrabCut Does



GrabCut Method

1. User draws bounding box; initialize border-of-box pixels as Background
2. Initialize interior pixels as Foreground (user does *not* specify foreground pixels)
3. Learn models of Foreground and Background regions
4. Apply GraphCut
5. Update Foreground and Background regions
6. Goto Step 3
7. Allow user to add cleanup strokes

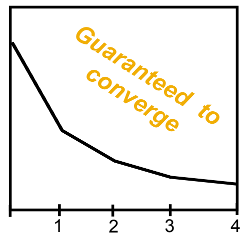
Iterated Graph Cut

User Initialization

K-means for learning K Gaussian color distributions

Graph cuts to infer the segmentation

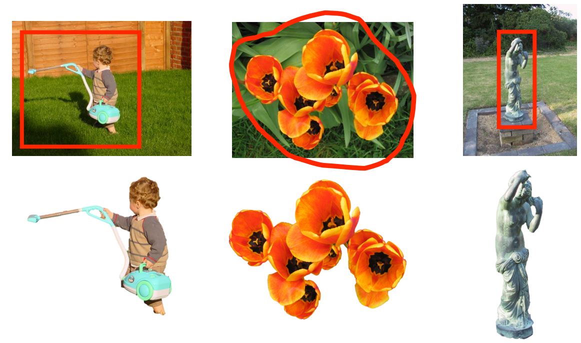
Iterated Graph Cut



Result

Energy after each Iteration

Examples



Difficult Examples

Camouflage & Low Contrast

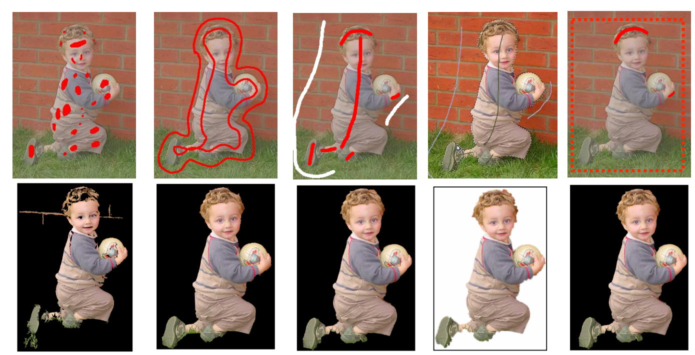
Fine structure

No telepathy

Initial Rectangle

Initial Result

Comparison



Magic Wand (198?)

Intelligent Scissors [Mortensen and Barrett, 1995]

Graph Cuts [Boykov and Jolly, 2001]

LazySnapping [Li et al., 2004]

GrabCut [Rother et al., 2004]

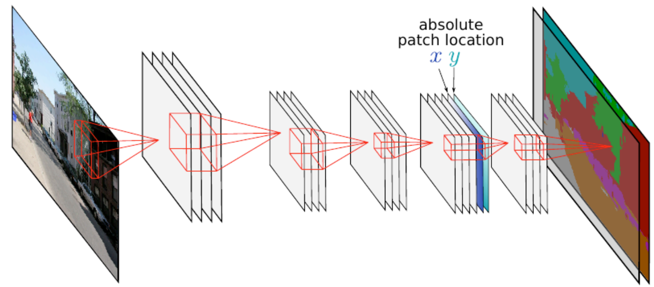
GrabCut

Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother
Vladimir Kolmogorov
Andrew Blake

Microsoft Research Cambridge

Deep Learning (Convolutional Neural Nets)



Results

FCN: Fully Convolutional Networks, Long et al., CVPR 2015

SDS: Simultaneous Detection and Segmentation Hariharan et al., ECCV 2014