Image Segmentation

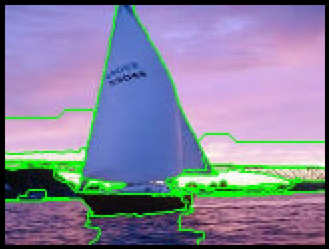
Image segmentation is the operation of partitioning an image into a collection of connected sets of pixels.

1. into regions, which usually cover the image

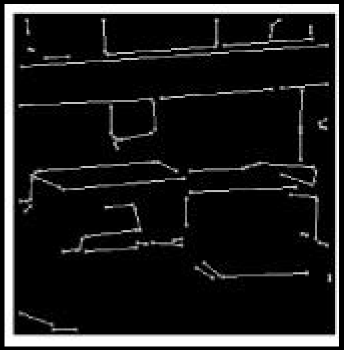
2. into linear structures, such as
   - line segments
   - curve segments

3. into 2D shapes, such as
   - circles
   - ellipses
   - ribbons (long, symmetric regions)

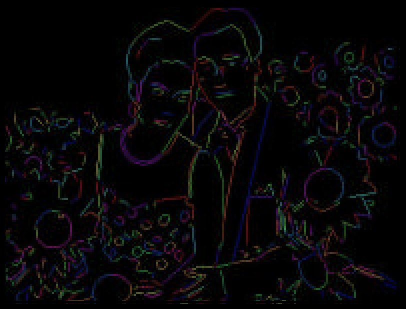
Example 1: Regions



Example 2: Straight Lines



Example 3: Lines and Circular Arcs



Region Segmentation: Segmentation Criteria

From Pavlidis

A segmentation is a partition of an image I into a set of regions S satisfying:

1. $\cup$ Si = S	Partition covers the whole image.
2. Si $\cap$ Sj = ϕ, i $\neq$ j	No regions intersect.
3. $\forall$ Si, P(Si) = true	Homogeneity predicate is satisfied by each region.
4. P(Si $\cup$ Sj) = false, i $\neq$ j, Si adjacent Sj	Union of adjacent regions does not satisfy it.

So

So all we have to do is define and implement the similarity predicate.

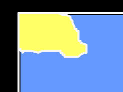
But, what do we want to be similar in each region?

Is there any property that will cause the regions to be meaningful objects?

Main Methods of Region Segmentation

1. Region Growing
2. Clustering
3. Split and Merge

Region Growing



Region growing techniques start with one pixel of a potential region and try to grow it by adding adjacent pixels till the pixels being compared are too disimilar.

- The first pixel selected can be just the first unlabeled pixel in the image or a set of seed pixels can be chosen from the image.
- Usually a statistical test is used to decide which pixels can be added to a region.

The RGGROW Algorithm

- Let R be the N pixel region so far and P be a neighboring pixel with gray tone y.
- Define the mean $\bar{X}$ and scatter S^2 (sample variance) by

$$\bar{X} = 1/N \sum_{(r,c) \in R} I(r,c)$$

$$S^2 = \sum_{(r,c) \in R} (I(r,c) - \bar{X})^2$$

The RGGROW Statistical Test

The T statistic is defined by

$$T = \left\{ \frac{(N-1) * N}{(N+1)} (y - \bar{X})^2 / S^2 \right\}^{1/2}$$

It has a T_{N-1} distribution if all the pixels in R and the test pixel y are independent and identically distributed normals (IID assumption) .

Decision and Update

- For the T distribution, statistical tables give us the probability $Pr(T \le t)$ for a given degrees of freedom and a confidence level. From this, pick suitable threshold t.
- If the computed $T \le t$ for desired confidence level, add y to region R and update $\bar{X}$ and S^2.
- If T is too high, the value y is not likely to have arisen from the population of pixels in R. Start a new region.

RGGROW Example

image

Not great!

segmentation

Clustering

- There are K clusters C1,…, CK with means m1,…, mK.
- The **least-squares error** is defined as

$$D = \sum_{k=1}^{K} \sum_{x_i \in C_k} \| x_i - m_k \|^2 .$$

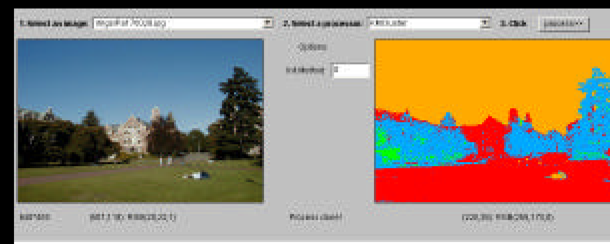
- Out of all possible partitions into K clusters, choose the one that minimizes D.

Why don't we just do this?
If we could, would we get meaningful objects?

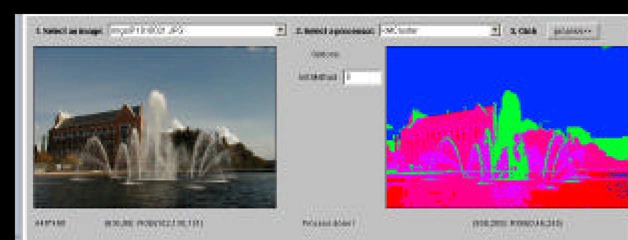
Some Clustering Methods

- K-means Clustering and Variants
- Isodata Clustering
- Histogram-Based Clustering and Recursive Variant
- Graph-Theoretic Clustering

K-Means Example 1



K-Means Example 2

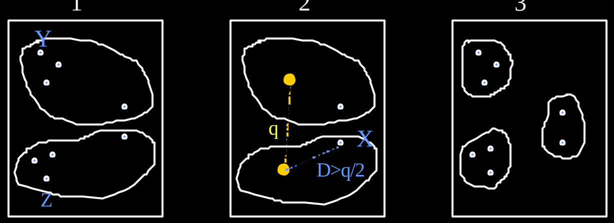


Meng-Hee Heng's K-means Variant

1. Pick 2 points Y and Z that are furthest apart in the measurement space and make them initial cluster means.
2. Assign all points to the cluster whose mean they are closest to and recompute means.
3. Let d be the max distance from each point to its cluster mean and let X be the point with this distance.
4. Let q be the average distance between each pair of means.
5. If d > q / 2, make X a new cluster mean.
6. If a new cluster was formed, repeat from step 2.

Illustration of Heng Clustering

We used this for segmentation of textured scenes.



Isodata Clustering

1. Select several cluster means and form clusters.
2. Split any cluster whose variance is too large.
3. Group together clusters that are too small.
4. Recompute means.
5. Repeat till 2 and 3 cannot be applied.

We used this to cluster normal vectors in 3D data.

Comparison

Original

K-means, K=6

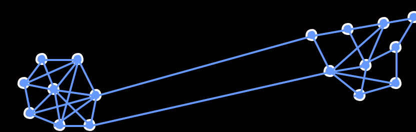
Isodata, K became 5

Ohlander's Recursive Histogram-Based Clustering

- color images of real indoor and outdoor scenes
- starts with the whole image
- selects the R, G, or B histogram with largest peak and finds clusters from that histogram
- converts to regions on the image and creates masks for each
- pushes each mask onto a stack for further clustering

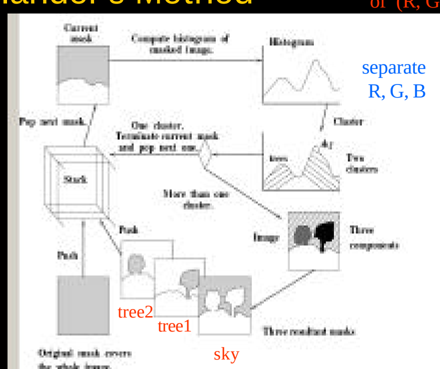
Jianbo Shi's Graph-Partitioning

- An image is represented by a graph whose nodes are pixels or small groups of pixels.
- The goal is to partition the vertices into disjoint sets so that the similarity within each set is high and across different sets is low.



Ohlander's Method

Ohta suggested using (R+G+B)/3, (R-B)/2 and (2G-R-B)/4 instead of (R, G, B).

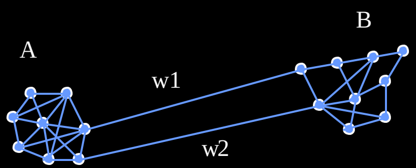


Minimal Cuts

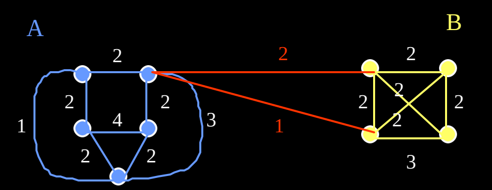
- Let G = (V,E) be a graph. Each edge (u,v) has a weight w(u,v) that represents the similarity between u and v.
- Graph G can be broken into 2 disjoint graphs with node sets A and B by removing edges that connect these sets.
- Let $cut(A,B) = \sum_{u \in A,\, v \in B} w(u,v)$.
- One way to segment G is to find the minimal cut.

Cut(A,B)

$$\text{cut}(A,B) = \sum_{u \in A,\ v \in B} w(u,v).$$



Example Normalized Cut



$$\text{Ncut}(A,B) = \frac{3}{21} + \frac{3}{16}$$

Normalized Cut

Minimal cut favors cutting off small node groups, so Shi proposed the normalized cut.

$$\text{Ncut}(A,B) = \frac{\text{cut}(A, B)}{\text{asso}(A,V)} + \frac{\text{cut}(A,b)}{\text{asso}(B,V)}$$

normalized cut

$$\text{asso}(A,V) = \sum_{u \in A,\ t \in V} w(u,t)$$

How much is A connected to the graph as a whole.

Normalize Cut in Matrix Form

$\mathbf{W}$ is the cost matrix : $\mathbf{W}(i,j) = w_{i,j}$;

$\mathbf{D}$ is the sum of costs from node i : $\mathbf{D}(i,i) = \sum_j \mathbf{W}(i,j)$;

After lots of math, we get:

$$Ncut(A,B) = \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}, \quad \text{with } \mathbf{y}_i \in \{1,-b\},\ \mathbf{y}^T\mathbf{D}\mathbf{1} = 0.$$

- Solution given by "generalized" eigenvalue problem:

$$(\mathbf{D}-\mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$$

- Solved by converting to standard eigenvalue problem:

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}z = \lambda z, \quad where\ z = \mathbf{D}^{\frac{1}{2}}\mathbf{y}$$

- optimal solution corresponds to second smallest eigenvector
- for more details, see
  - J. Shi and J. Malik, Normalized Cuts and Image Segmentation, IEEE Conf. Computer Vision and Pattern Recognition(CVPR), 1997

How Shi used the procedure

Shi defined the edge weights w(i,j) by

$$w(i,j) = e^{\|F(i)-F(j)\|_2 / \sigma I} * \begin{cases} e^{\|X(i)-X(j)\|_2 / \sigma X} & \text{if } \|X(i)-X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases}$$

where X(i) is the spatial location of node i
F(i) is the feature vector for node I
which can be intensity, color, texture, motion…

The formula is set up so that w(i,j) is 0 for nodes that are too far apart.

Examples of Shi Clustering

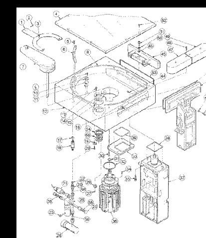
See Shi's Web Page
http://www-2.cs.cmu.edu/~jshi

Lines and Arcs Segmentation

In some image sets, lines, curves, and circular arcs are more useful than regions or helpful in addition to regions.

Lines and arcs are often used in

- object recognition
- stereo matching
- document analysis



Edge Detection

Basic idea: look for a neighborhood with strong signs of change.

Problems:

- neighborhood size
- how to detect change

81	82	26	24
82	33	25	25
81	82	26	24

Differential Operators

Differential operators

• attempt to approximate the gradient at a pixel via masks

• threshold the gradient to select the edge pixels

Example: Sobel Operator

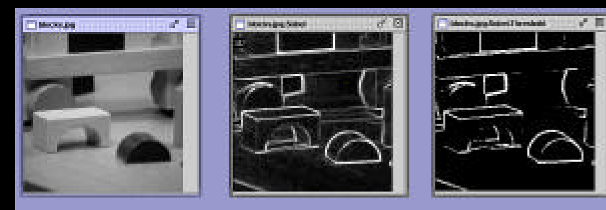
$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Sy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

On a pixel of the image
- let gx be the response to Sx
- let gy be the response to Sy

Then $g = (gx^2 + gy^2)^{1/2}$ is the gradient magnitude.

$\theta = \mathrm{atan2}(gy,gx)$ is the gradient direction.

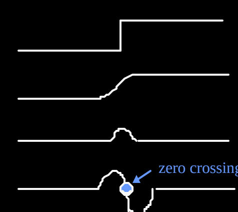
Java Toolkit's Sobel Operator



original image | gradient magnitude | thresholded gradient magnitude

Zero Crossing Operators

Motivation: The zero crossings of the second derivative of the image function are more precise than the peaks of the first derivative.



step edge

smoothed

1st derivative

2nd derivative

Marr/Hildreth Operator

- First smooth the image via a Gaussian convolution
- Apply a Laplacian filter (estimate 2nd derivative)
- Find zero crossings of the Laplacian of the Gaussian

This can be done at multiple resolutions.

Haralick Operator

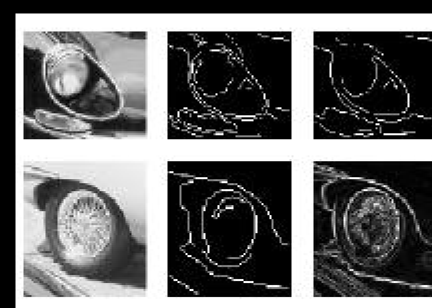
- Fit the gray-tone intensity surface to a piecewise cubic polynomail approximation.
- Use the approximation to find zero crossings of the second directional derivative in the direction that maximizes the first directional derivative.

The derivatives here are calculated from direct mathematical expressions wrt the cubic polynomial.

Canny Edge Detector

- Smooth the image with a Gaussian filter.
- Compute gradient magnitude and direction at each pixel of the smoothed image.
- Zero out any pixel response ≤ the two neighboring pixels on either side of it, along the direction of the gradient.
- Track high-magnitude contours.
- Keep only pixels along these contours, so weak little segments go away.

Canny Examples



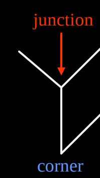
Best Canny on Kidney from Hw1

Best Canny on Blocks from Hw1

Finding Line and Curve Segments from Edge Images

Given an edge image, how do we find line and arc segments?

Method 1: Tracking



Use masks to identify the following events:

1. start of a new segment
2. interior point continuing a segment
3. end of a segment
4. junction between multiple segments
5. corner that breaks a segment into two

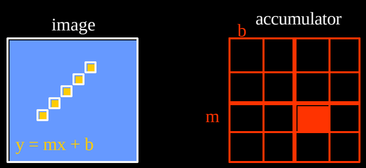
Edge Tracking Procedure

```
for each edge pixel P {
   classify its pixel type using masks
   case
      1. isolated point :          ignore it
      2.  start point :            make a new segment
      3.  interior point :         add to current segment
      4.  end point :              add to current segment and finish it
      5.  junction or corner :     add to incoming segment
                                   finish incoming segment
                                   make new outgoing segment(s)
```

The ORT package uses a fancier corner finding approach.

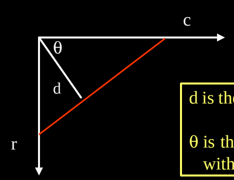
Hough Transform

- The Hough transform is a method for detecting lines or curves specified by a parametric function.
- If the parameters are p1, p2, … pn, then the Hough procedure uses an n-dimensional accumulator array in which it accumulates votes for the correct parameters of the lines or curves found on the image.



Finding Straight Line Segments

- y = mx + b is not suitable (why?)
- The equation generally used is: $d = r \sin \theta + c \cos \theta$



d is the distance from the line to origin

θ is the angle the perpendicular makes with the column axis

Procedure to Accumulate Lines

- Set accumulator array A to all zero.
  Set point list array PTLIST to all NIL.
- For each pixel (R,C) in the image {
  - compute gradient magnitude GMAG
  - if GMAG > gradient_threshold {
    - compute quantized tangent angle THETAQ
    - compute quantized distance to origin DQ
    - increment A(DQ,THETAQ)
    - update PTLIST(DQ,THETAQ) } }

Example

gray-tone image

0	0	0	100	100
0	0	0	100	100
0	0	0	100	100
100	100	100	100	100
100	100	100	100	100

DQ

-	-	3	3	-
-	-	3	3	-
3	3	3	3	-
3	3	3	3	-
-	-	-	-	-

THETAQ

-	-	0	0	-
-	-	0	0	-
90	90	40	20	-
90	90	90	40	-
-	-	-	-	-

Accumulator A

distance \ angle	0	10	20	30	40	…	90
360	-	-	-	-	-	-	-
.	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-
3	4	-	1	-	2	-	5
0	-	-	-	-	-	-	-

PTLIST

360	-	-	-	-	-	-	-
.	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-
3	*	-	*	-	*	-	*
0	-	-	-	-	-	-	-

(1,3)(1,4)(2,3)(2,4)

(3,1)
(3,2)
(4,1)
(4,2)
(4,3)

How do you extract the line segments from the accumulators?

pick the bin of A with highest value V
while V > value_threshold {

order the corresponding pointlist from PTLIST

merge in high gradient neighbors within 10 degrees

create line segment from final point list

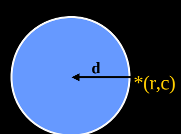
zero out that bin of A

pick the bin of A with highest value V }

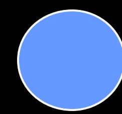
Finding Circles

Equations: $\boxed{\begin{array}{l} r = r0 + d \sin \theta \\ c = c0 + d \cos \theta \end{array}}$ r, c, d are parameters

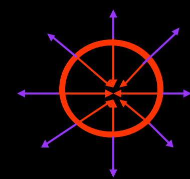
Main idea: The gradient vector at an edge pixel points to the center of the circle.



Why it works



Filled Circle:
Outer points of circle have gradient direction pointing to center.



Circular Ring:
Outer points gradient towards center.
Inner points gradient away from center.

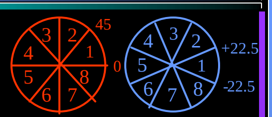
The points in the away direction don't accumulate in one bin!

Procedure to Accumulate Circles

- Set accumulator array A to all zero.
  Set point list array PTLIST to all NIL.

- For each pixel (R,C) in the image {
  For each possible value of D {
  - compute gradient magnitude GMAG
  - if GMAG > gradient_threshold {
  . Compute THETA(R,C,D)
  . R0 := R - D*cos(THETA)
  . C0 := C - D* sin(THETA)
  . increment A(R0,C0,D)
  . update PTLIST(R0,C0,D) }}

The Burns Line Finder



1. Compute gradient magnitude and direction at each pixel.
2. For high gradient magnitude points, assign direction labels to two symbolic images for two different quantizations.
3. Find connected components of each symbolic image.

- Each pixel belongs to 2 components, one for each symbolic image.
- Each pixel votes for its longer component.
- Each component receives a count of pixels who voted for it.
- The components that receive majority support are selected.

See Transparencies