Segmentation and Grouping

How and what do we see?


Fundamental Problems

- *Focus of attention, or grouping*
  - *What subsets of pixels do we consider as possible objects?*
    - *All connected subsets?*
- *Representation*
  - *How do we model the shape, color and appearance of natural objects?*
- *Matching*
  - *How do we compare these models against images?*

Polyhedral objects

- *Representation*
  - *Graphs of vertices connected by links corresponding to the corners and edges of the polyhedron, respectively.*
    - *Metric information associated with the vertices of the graph*
- *Matching*
  - *Pose estimation*
- *Segmentation*
  - *How do we find the projections of the corners of the polyhedron in the image*

Combinatorics of polyhedra recognition

- *4 - point perspective solution*
  - *unique solution for 6 pose parameters*
  - *computational complexity of* n^4m^4
- *3 - point perspective solution*
  - *generally two solutions per triangle pair, but sometimes more*
  - *reduced complexity of* n^3m^3

Reducing the combinatorics of pose estimation

- *Problem # 1: we are looking for an object in an image but the image does not contain the object*
  - *only discover this after comparing all* n^4 *quadruples of image features against all* m^4 *quadruples of object features.*
- *How can we reduce the number of matches?*
  - *consider only quadruples of object features that are*
    - *simultaneously visible - extensive preprocessing*
    - *diameter 2 subgraphs of the object graph*
      - *but in some images no such subgraphs might be visible*

Reducing the combinatorics of pose estimation

- *Reducing the number of matches*
  - *consider only quadruples of image features that*
    - *are connected by edges*
    - *are "close" to one another*
- *Problem # 2: Image contains instances of MANY objects with occlusions*
- *Generally, try to group the image junctions into sets that are probably from a single object, and then only construct quadruples from within a single group*

Image segmentation

- *Definition 1: Partition the image into connected subsets that maximize some "uniformity" criteria.*
- *Definition 2: Identify possibly overlapping but maximal connected subsets that satisfy some uniformity criterion.*

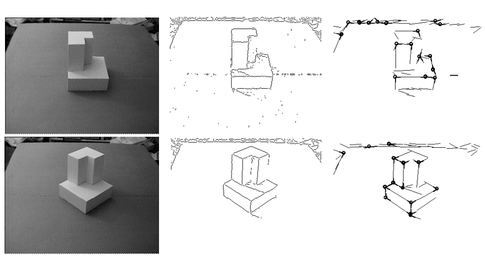
Approaches to segmentation

- *Edge detection*
  - *Biological systems are sensitive to color and texture edges*
  - *Detect and identify collections of edges that "outline" an object or are likely to be part of the outline of a single object*
- *Region detection*
  - *Identify (possibly multiple) partitions of the image into uniform regions*

Edge detection

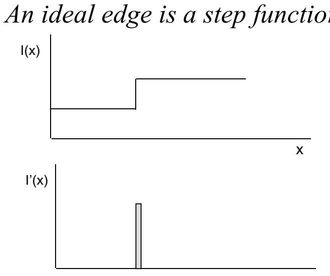
- *Gradient based edge detection*
- *Edge detection by function fitting*
- *Second derivative edge detectors*
- *Edge grouping*

Example images

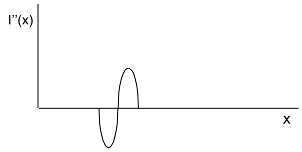


1-D edge detection

- *An ideal edge is a step function*



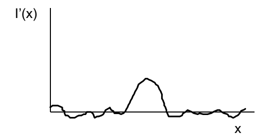
1-D edge detection



- *The first derivative of I(x) has a **peak** at the edge*
- *The second derivative of I(x) has a **zero crossing** at the edge*

1-D edge detection

- *More realistically, image edges are **blurred** and the regions that meet at those edges have **noise** or variations in intensity.*
  - *blur - high first derivatives near edges*
  - *noise - high first derivatives within regions that meet at edges*



Edge detection in 2-D

- *Let f(x,y) be the image intensity function. It has derivatives in all directions*
  - *the **gradient** is a vector whose first component is the direction in which the first derivative is highest, and whose second component is the magnitude of the first derivative in that direction.*
- *If f is continuous and differentiable, then its gradient can be determined from the directional derivatives in any two orthogonal directions*

magnitude = $[(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2}$

- *direction =* $\tan^{-1}(\frac{\partial f / \partial y}{\partial f / \partial x})$

Edge detection in 2-D

- *With a digital image, the partial derivatives are replaced by finite differences:*
  - $\Delta_x f = f(x,y) - f(x-1, y)$
  - $\Delta_y f = f(x,y) - f(x, y-1)$
- *Alternatives are:*
  - $\Delta_{2x} f = f(x+1,y) - f(x-1,y)$
  - $\Delta_{2y} f = f(x,y+1) - f(x,y-1)$
- *Robert's gradient*
  - $\Delta_+ f = f(x+1,y+1) - f(x,y)$ ↗ 0 1 / -1 0
  - $\Delta_- f = f(x,y+1) - f(x+1, y)$ ↘ 1 0 / 0 -1

Edge detection in 2-D

- *How do we combine the directional derivatives to compute the gradient magnitude?*
  - *use the root mean square (RMS) as in the continuous case*
  - *take the maximum absolute value of the directional derivatives*

Combining smoothing and differentiation - fixed scale

- *Local operators like the Roberts give high responses to any intensity variation*
  - *local surface texture*
- *If the picture is first smoothed by an averaging process, then these local variations are removed and what remains are the "prominent" edges*
  - *smoothing is blurring, and details are removed*
- *Example* $f_{2x2}(x,y) = 1/4[f(x,y) + f(x+1,y) + f(x,y+1) + f(x+1,y+1)]$

Smoothing - basic problems

- *What function should be used to smooth or average the image before differentiation?*
  - *box filters or uniform smoothing*
    - *easy to compute*
    - *for large smoothing neighborhoods assigns too much weight to points far from an edge*
  - *Gaussian, or exponential, smoothing*

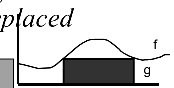
$$(1/2\pi\sigma)e^{-(x^2+y^2)/2\sigma^2}$$

Smoothing and convolution

- *The convolution of two functions, f(x) and g(x) is defined as*

$$h(x) = \int_{-\infty}^{\infty} g(x')f(x-x')dx' = g(x) * f(x)$$

- *When the functions f and g are discrete and when g is nonzero only over a finite range [-n,n] then this integral is replaced by the following summation:*



$$h(i) = \sum_{j=-n}^{n} g(j)f(i+j)$$

Smoothing and convolution

- *These integrals and summations extend simply to functions of two variables:*

$$h(i,j) = f(i,j) * g = \sum_{k=-n}^{n}\sum_{l=-n}^{n} g(k,l)f(i+k,j+l)$$

- *Convolution computes the weighted sum of the gray levels in each nxn neighborhood of the image, f, using the matrix of weights g.*
- *Convolution is a so-called linear operator because*
  - $g*(af_1 + bf_2) = a(g*f_1) + b(g*f_2)$

Gaussian smoothing

- *Advantages of Gaussian filtering*
  - *rotationally symmetric (for large filters)*
  - *filter weights decrease monotonically from central peak, giving most weight to central pixels*
  - *Simple and intuitive relationship between size of σ and size of objects whose edges will be detected in image.*
  - *The gaussian is separable:*

$$e^{-\frac{(x^2+y^2)}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} * e^{-\frac{y^2}{2\sigma^2}}$$

Advantage of seperability

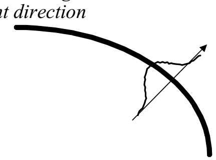
- *First convolve the image with a one dimensional horizontal filter*
- *Then convolve the result of the first convolution with a one dimensional vertical filter*
- *For a kxk Gaussian filter, 2D convolution requires k^2 operations per pixel*
- *But using the separable filters, we reduce this to 2k operations per pixel.*

Advantages of Gaussians

- *Convolution of a Gaussian with itself is another Gaussian*
  - *so we can first smooth an image with a small Gaussian*
  - *then, we convolve that smoothed image with another small Gaussian and the result is equivalent to smoother the original image with a larger Gaussian.*
  - *If we smooth an image with a Gaussian having sd σ twice, then we get the same result as smoothing the image with a Gaussian having standard deviation $(2\sigma)^{1/2}$*

Combining smoothing and differentiation - fixed scale

- *Non-maxima supression - Retain a point as an edge point if:*
  - *its gradient magnitude is higher than a threshold*
  - *its gradient magnitude is a local maxima in the gradient direction*



simple thresholding will compute thick edges

Summary of basic edge detection steps

- *Smooth the image to reduce the effects of local intensity variations*
  - *choice of smoothing operator practically important*
- *Differentiate the smoothed image using a digital gradient operator that assigns a magnitude and direction of the gradient at each pixel*
- *Threshold the gradient magnitude to eliminate low contrast edges*

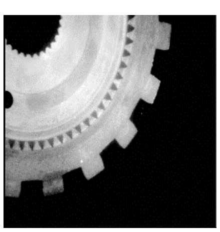
Summary of basic edge detection steps

- *Apply a nonmaxima suppression step to thin the edges to single pixel wide edges*
  - *the smoothing will produce an image in which the contrast at an edge is spread out in the neighborhood of the edge*
  - *thresholding operation will produce thick edges*

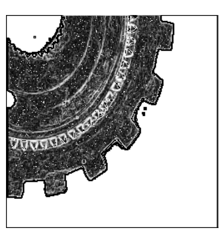
The scale-space problem

- *Usually, any single choice of* σ *does not produce a good edge map*
  - *a large* σ *will produce edges form only the largest objects, and they will not accurately delineate the object because the smoothing reduces shape detail*
  - *a small* σ *will produce many edges and very jagged boundaries of many objects.*
- *Scale-space approaches*
  - *detect edges at a range of scales* $[\sigma_1, \sigma_2]$
  - *combine the resulting edge maps*
    - *trace edges detected using large* σ *down through scale space to obtain more accurate spatial localization.*

Examples

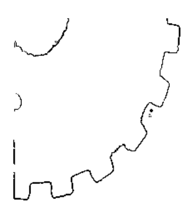


Gear image

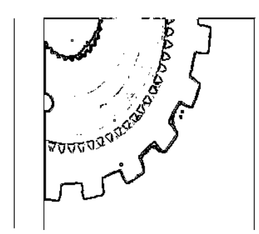


3x3 Gradient magnitude

Examples

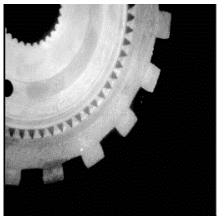


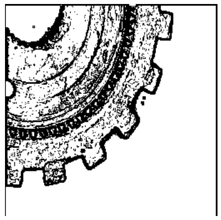
High threshold



Medium threshold

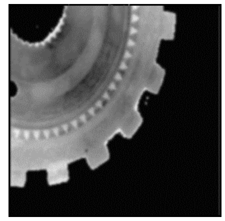
Examples



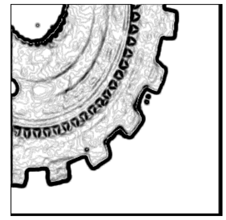


low threshold

Examples

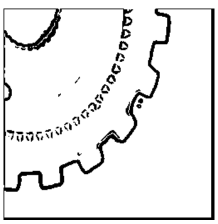


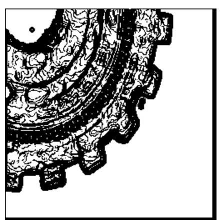
Smoothed 5x5 Gaussian



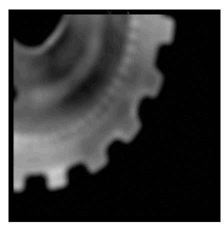
3x3 gradient magnitude

Examples

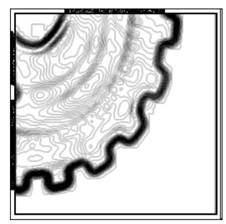




Examples

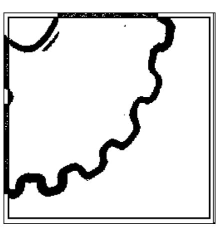


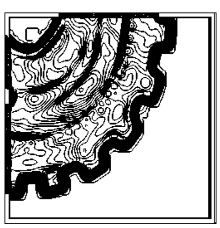
smoothed 15x15 Gaussian



3x3 gradient magnitude

Examples





Laplacian edge detectors

- *Directional second derivative in direction of gradient has a zero crossing at gradient maxima*
- *Can "approximate" directional second derivative with Laplacian*

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

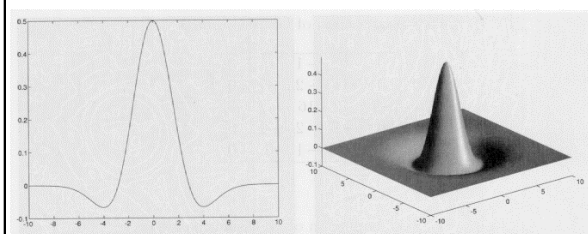
0	1	0
1	-4	1
0	1	0

- *Its digital approximation is*
  - $\nabla^2 f(x,y) = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4 f(x,y)$
    $= [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)] + [f(x,y+1)-f(x,y)] - [f(x,y) - f(x,y-1)]$

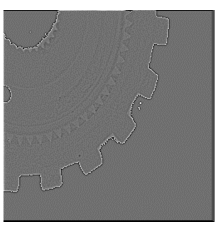
Laplacian edge detectors

- *Laplacians are also combined with smoothing for edge detectors*
  - *Take the Laplacian of a Gaussian smoothed image - called the Mexican Hat operator or DoG (Difference of Gaussians)*
  - *Locate the zero-crossing of the operator*
    - *these are pixels whose DoG is positive and which have neighbor's whose DoG is negative or zero*
  - *Usually, measure the gradient or directional first derivatives at these points to eliminate low contrast edges.*

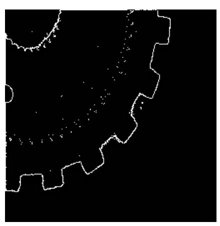
Laplacian of Gaussian or "Mexican Hat"



Laplacian of Gaussian

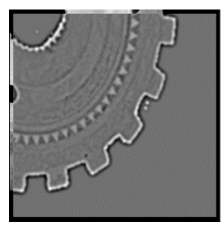


5x5 Mexican Hat - Laplacian of Gaussian

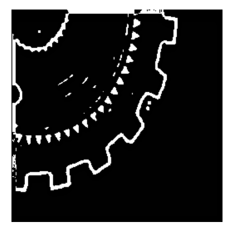


Zero crossings

Laplacian of Gaussian



13 x 13 Mexican hat



zero crossings

Edge linking and following

- *Group edge pixels into chains and chains into large pieces of object boundary.*
  - *can use the shapes of long edge chains in recognition*
    - *Curvature – high curvature points are possible corners*
    - *Junctions – where individual chains meet*

Grouping chains

- *How do we know if two chains should be combined into a single, longer chain*
  - *Edge detector leaves gaps in edges due to low contrast, complex image structure where more than two regions meet*
- *More generally, can we optimally partition the set of chains into groups that maximize some "reasonable" criteria*
  - *Good continuation across gaps*
  - *Closure*
  - *Resistance to noise (small, irrelevant chains)*

Segmentation II- Region analysis

- *Partition images into elementary regions*
  - *Pixels*
  - *Connected components of constant brightness/color*
- *Build region adjacency graph for regions*
  - *Edge weights reflect similarity of regions that meet at that edge*
- *Reduce the graph to a smaller number of regions*
  - *Merging – eliminate weak edges*
  - *Cutting – partition graph into subgraphs*

Region merging

- *Best first region merging*
  - *Eliminate the weakest edge in the graph*
  - *Compute new properties of merged region*
    - *Average color*
    - *Texture statistics*
  - *Update edge weights for adjacent regions*
- *For a graph with n regions initially, will create n-1 new regions*

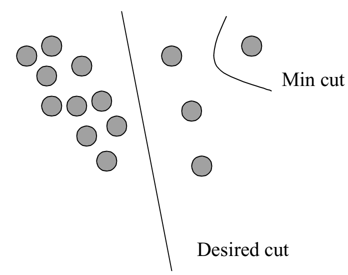
Graph splitting - cuts

- *Let A,B be a partition of the nodes in a weighted graph G.*

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

- *Optimal bi-partitioning of a graph is the one that minimizes this cut*
  - *Efficient algorithms for finding minimal cuts*
- *But minimum cuts favor small sets of isolated nodes*

Minimal cuts



Normalized cuts

- *Compute the cut cost as a fraction of the total edge connections to all the nodes in the graph:*

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

$$assoc(A,V) = \sum_{u \in A, t \in V} cut(u,t)$$

Normalized cuts

- *Finding the cut which minimizes Ncut is an NP complete problem*
- *But there is a way to obtain an approximate solution by constructing a matrix from the graph and finding the eigenvectors and eigenvalues of that matrix*
- *See Shi and Malik, IEEE T-PAMI, August 2000.*