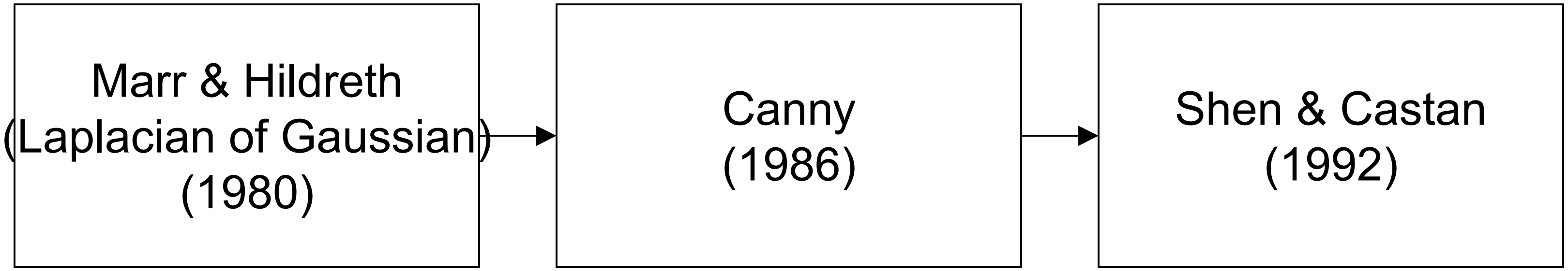
Image Segmentation: Edge-based

Michael A. Wirth, Ph.D.
University of Guelph
Computing and Information Science
Image Processing Group

Edge-based Segmentation

- The history of edge detection



Laplacian of Gaussian

- The Laplacian is seldom used on its own for edge detection because of its sensitivity to noise.
- The Laplacian-of-Gaussian (LoG) uses a Gaussian filter to blur the image and a Laplacian to enhance edges.
  - Also known as Marr & Hildreth edge detector
- Edge localisation is done by finding zero-crossings.

Laplacian of Gaussian

- Reference:
  - Marr, D., and Hildreth, E., “Theory of edge detection”, *Proceedings of the Royal Society of London*, Series B, 1980, **207**:pp.187-217

Laplacian of Gaussian

Algorithm: LoG

- Convolve the image with a two-dimensional Gaussian function
- Compute the Laplacian of the convolved image → *L*
- Identify edge pixels as those for which there is a zero-crossing in *L*.

Laplacian of Gaussian

- A radially-symmetric 2D Gaussian:

$$G_\theta(x, y) = e^{\left(\frac{-r^2}{2\sigma^2}\right)}$$

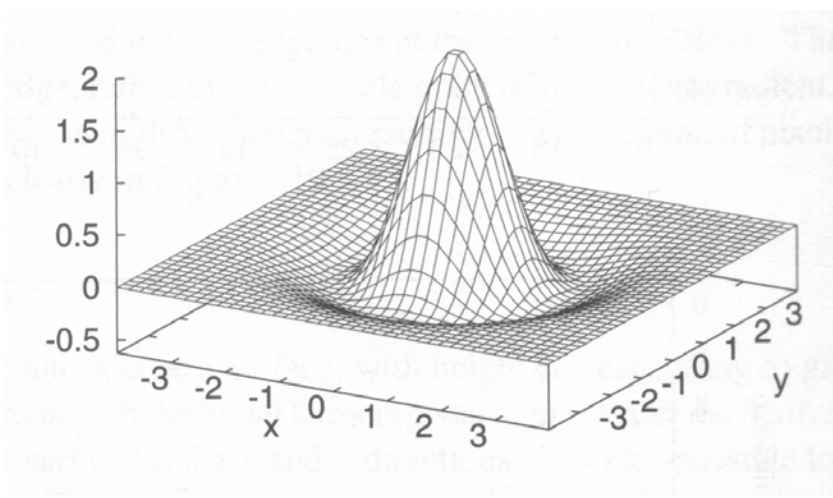
where $r^2 = x^2 + y^2$

- The Laplacian of this is:

$$G(x, y) = \left(\frac{r^2 - \sigma^2}{\sigma^4}\right) e^{\left(-\frac{r^2}{2\sigma^2}\right)}$$

Laplacian of Gaussian

- This function has a minimum at its origin, but it is usual to invert the filter, so that it has a maximum at its origin.
  - classic "Mexican hat" shape



Laplacian of Gaussian

- The value of σ determines the width of the filter and controls the amount of smoothing produced by the Gaussian component.
  - σ tunes the filter to detect edges at different scales
  - Should have a half-width of at least 3σ

Laplacian of Gaussian

- 5×5 Laplacian of Gaussian kernel

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Difference of Gaussian

- The LoG can be approximated by convolving with a kernel that is the difference of two Gaussian kernels with substantially different σ's.
  - Known as the Difference-of-Gaussian (DoG)

Zero-Crossings

- To form an edge map from the LoG/DoG output → locate and mark the zero-crossings
  - Trying to detect zeros in the LoG or DoG image will fail
  - A simple zero-crossing detector may identify a zero-crossing in a $n \times n$ window, assigning an edge label if LoG/DoG image values of both polarities occur in the window
  - No edge label would be assigned if values within the window are all of one sign (either all positive, or all negative)

Scale Space

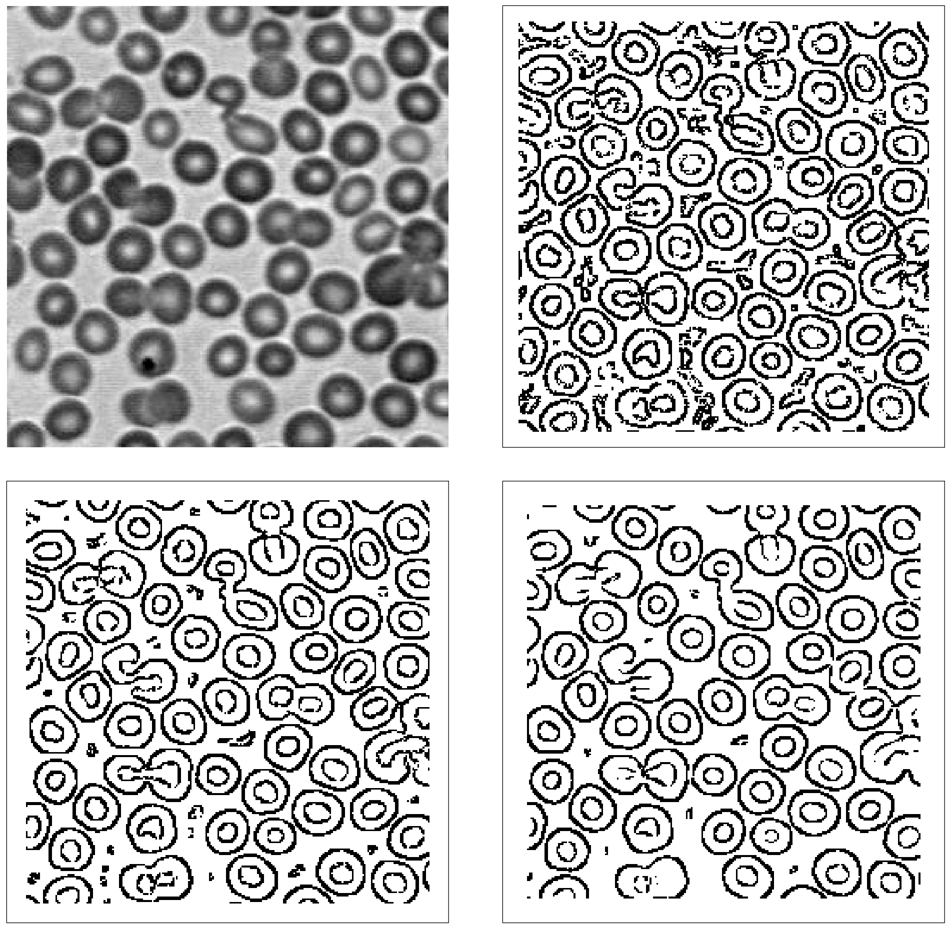
- Gaussian smoothing results in the blurring of edges and other sharp discontinuities in an image.
  - The amount of smoothing depends on the value of σ.
  - A larger σ results in better noise filtering, but at the same time loses important edge information, affecting the performance of the edge detector.
  - If a small filter is used, there is likely to be more noise due to insufficient averaging.

Scale Space

- In order to ensure a variety of scales are used:
  - Use two different Gaussians, say $\sigma \pm 0.8$

    e.g. if $\sigma=2.0$, then $\sigma_1=1.2$, $\sigma_2=2.8$
  - Select the pixels in the edge images that have zero-crossings in both scales as the edge pixels
  - More than two Gaussians can be used.

Examples of LoG

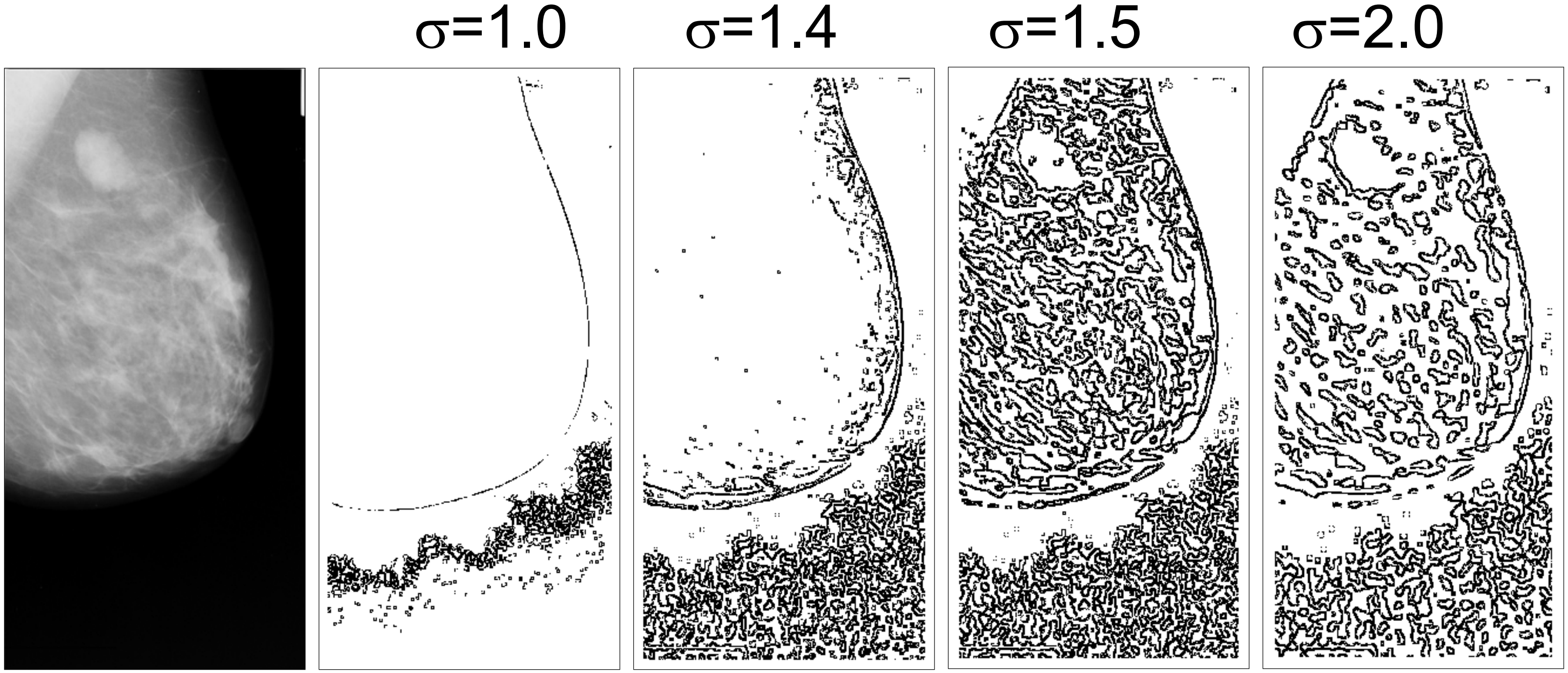


σ=1.5

σ=2.3

σ=3.0

Examples of LoG



Limitations of 2nd Derivatives

- It smoothes the shape too much
- It tends to create closed loops for edges

Extensions to Zero-Crossing

- Qian, R.J., and Huang, T.S., “Optimal edge detection in two-dimensional images”, in *ARPA Image Understanding Workshop*, Monterey, CA, ARPA, 1994, pp.1581-1588
- Mehrotra, R., and Shiming, Z., “A computational approach to zero-crossing-based two-dimensional edge detection”, *Graphical Models and Image Processing*, 1996, **58**:pp.1-17
- Hardie, R.C. and Boncelet, C.G., “Gradient-based edge detection using nonlinear edge-enhancing filters”, *IEEE Transactions on Image Processing*, 1995, **4**:1572-1577
- Alparone, L., Baronti, S., and Casini, A., “A novel approach to the suppression of false contours”, in *International Conference on Image Processing*, IEEE, 1996, pp.825-828

Canny Edge Detector

- The Canny edge detector addresses the fact that for edge detection, there is a tradeoff between noise reduction (smoothing) and edge localisation.
  - A form of optimal edge detection
- Reference:
  - Canny, J., “A computational approach to edge detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, **8**(6):pp.679-698

Canny Edge Detector

Algorithm: Canny Edge Detection

– Smooth the image with a Gaussian filter
– Compute the gradient magnitude and orientation
– Apply *non-maximal suppression* to the gradient magnitude image
– Use *hysteresis thresholding* to detect and link edges

Canny Edge Detector

- The smoothing step employs a Gaussian low-pass filter:
  - The standard deviation, σ, determines the width of the filter and hence the amount of smoothing
  - A filter with a large σ will suppress much of the noise, but also smooth away the weakest edges

Canny Edge Detector

- The edge enhancement step simply involves calculation of the gradient vector at each pixel in the smoothed image.
- Efficient implementations combine the smoothing and enhancement steps by convolving the image with a derivative of the Gaussian kernel.

Canny Edge Detector

- The localization step has two stages:
  - Non-maximal suppression
  - Hysteresis thresholding

Canny Edge Detector

- Non-maximal suppression:
  - Pixels that are not local maxima are removed
  - Thins the wide ridges around local maxima in gradient magnitude images down to edges that are only one pixel wide

Canny Edge Detector

Algorithm: Non-maximal Suppression

- Quantize edge directions eight ways according to 8-connectivity
- For each pixel with non-zero edge magnitude, inspect the two adjacent pixels indicated by the direction of its edge
- If the edge magnitude of either of these two exceeds that of the pixel under inspection, mark it for deletion
- When all pixels have been inspected, erase to zero all edge pixels marked for deletion

Canny Edge Detector

- Hysteresis thresholding:
  - Assumes the gradient magnitude G_M and direction G_θ have already been computed.
  - There are problems associated with applying a single, fixed threshold to gradient magnitude images.
    - Choosing a low threshold ensures that weak, yet meaningful edges are captured, but may also result in an excessive number of “false-positives”.
      - Gradient maxima are caused by noise rather than by features of interest.
    - Choosing a high threshold will lead to excessive fragmentation of pixes that represent significant edges in the image

Canny Edge Detector

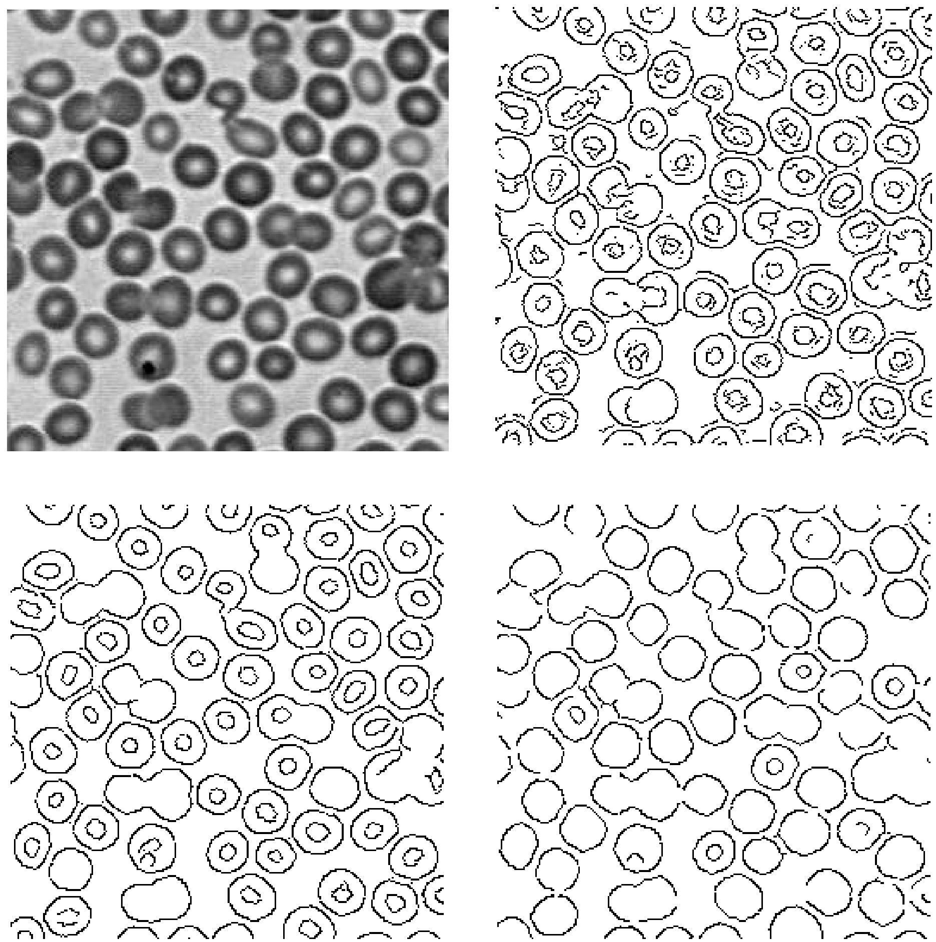
Algorithm: Hysteresis thresholding

- Uses two thresholds T_{low} and T_{high}
- If an edge response is above T_{high}, those pixels constitute a definite edge. Individual weak responses usually correspond to noise.
- If weak responses are connected to pixels with strong responses and they are above T_{low} they are edge pixels
- The T_{low} and T_{high} thresholds are set according to an estimated signal-to-noise ratio

Canny Edge Detector

- In some versions the search is conducted over all eight-neighbors of an edge pixel (edgel), in others, only the neighbors along a line normal to the gradient orientation at the edge pixel are considered.
  - This technique reduces the number of false-positives because edges are tracked only if at least one pixel has a gradient magnitude exceeding T_{high}
  - It reduces the fragmentation of contours in the edge map by allowing significant fluctuations to occur in gradient magnitude on an edge.

Examples of Canny



$\sigma=0.6$

$\sigma=1.5$

$\sigma=2.3$

Extensions to Canny

- Jalali, S., and Boyce, J.F., “Determination of optimal general edge detectors by global minimization of a cost function”, *Image and Vision Computing*, 1995, **13**:pp.683-693
- Sorrenti, D.G., “A proposal on local and adaptive determination of filter scale for edge detection”, in *Image Analysis and Processing, ICIAP’95,* Springer Verlag, 1995, pp.405-410
- Demigny, D., Lorca, F.G., Kessal, L., “Evaluation of edge detector performances with a discrete expression of Canny’s criteria”, in *International Conference on Image Processing,* IEEE, 1995, pp.169-172
- Mehrotra, R., and Shiming, Z., “A computational approach to zero-crossing-based two-dimensional edge detection”, *Graphical Models and Image Processing*, 1996, **58**:pp.1-17

Shen-Castan Detector

- The Shen-Castan edge detector uses an optima filter function:
  - Infinite Symmetric Exponential Filter (ISEF)
  - Better signal-to-noise ratios and better localisation than Canny
- Reference:
  - Shen, J., and Castan, S., “An optimal linear operator for step edge detection”, *CVGIP: Graphical Models and Understanding*, 1992, **54**(2):pp.112-133

Shen-Castan Detector

Algorithm: Shen-Castan

- Convolve the image using the ISEF
- Localise edges by finding zero-crossings of the Laplacian (similar to the Marr-Hildreth algorithm)
  - Approximate the Laplacian by subtracting the original from the smoothed image, the resulting image is a *band-limited Laplacian*
  - Compute the *Binary Laplacian Image* (BLI) by setting all the positive valued pixels to 1, and others to 0.
  - The candidate pixels are on the boundaries of the regions in the BLI

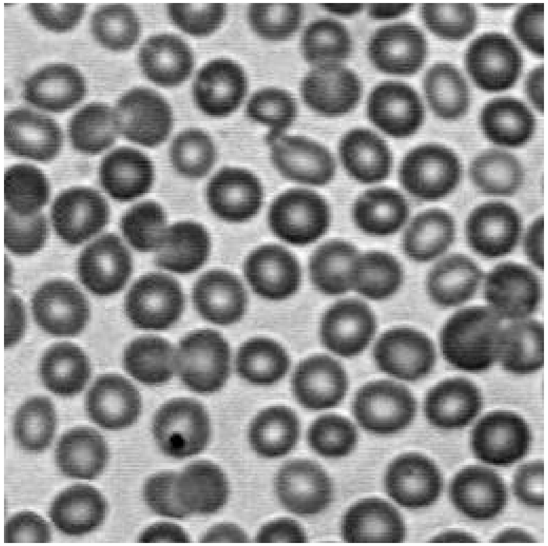
Shen-Castan Detector

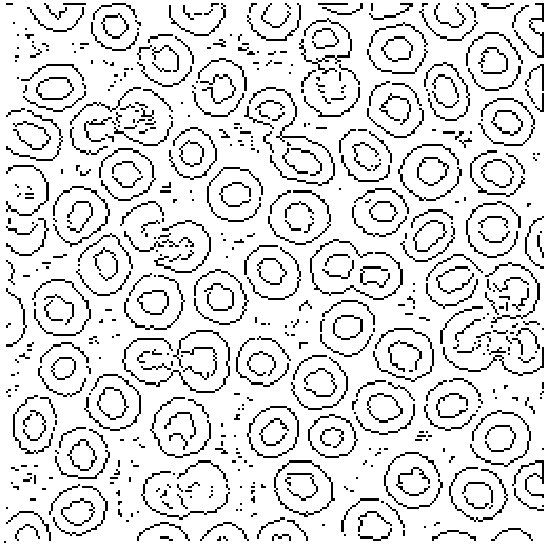
- Improve the quality of the edge pixels by *false zero-crossing suppression*
  - At the location of an edge pixel there will be a zero-crossing in the 2nd derivative of the filtered image. This means the gradient at that point is either a maximum or a minimum
  - If the change is +ve to –ve → positive zero-crossing
  - If the change is –ve to +ve → negative zero-crossing
  - Allow +ve zero-crossing to have a +ve gradient, and –ve zero-crossings to have a –ve gradient
  - Suppress all other zero-crossings

Shen-Castan Detector

- Perform adaptive gradient thresholding
  - A window with fixed with W is centred on the candidate edge pixel. If this is an edge pixel then the window will contain two regions of differing intensity separated by an edge (zero crossing contour).
  - The best estimate of the gradient at that point should be the difference in level between the two regions
- Apply hysteresis thresholding

Example of Shen-Castan





% of pixels above the hysteresis threshold = 0.7

smoothing factor = 0.7

window size = 15

thinning factor (number of pixels) = 1

Edge Image Segmentation

- An edge image is usually thresholded to decide which edges are significant, and streaking is the breaking up of an edge contour caused by the operator fluctuating above and below the threshold.
  - Can be eliminated by thresholding using some form of edge image segmentation

Edge Image Segmentation

- Edge image thresholding
  - Non-maximal suppression, and hysteresis
- Edge relaxation
- Border tracing
- Border detection as graph searching
- Border detection as dynamic programming

Other Edge Segmentation Techniques

- Boie & Cox Edge Detector
  - Boie, R.A. Cox, I., and Rehak, P., “On optimum edge recognition using matched filters”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986, pp.100-108
  - Boie, R.A. and Cox, I., “Two-dimensional optimum edge recognition using matched and wiener filters for machine vision”, in *Proceedings of the IEEE First International Conference on Computer Vision,* 1987, pp.450-456

Image Segmentation: Region-based

Michael A. Wirth, Ph.D.
University of Guelph
Computing and Information Science
Image Processing Group

Pixel Connectivity

- Pixel connectivity is a central concept in both edge- and region-based approaches to segmentation.
- Define two types of neighborhood surrounding a pixel:
  - 4-neighborhood: contains pixels, above, below, to the left and to the right of the central pixel.
  - 8-neighborhood: contains all the pixels of a 4-neighborhood, plus four diagonal neighbors.

Pixel Connectivity

- A 4-connected path from pixel p_1 to another pixel p_n is the sequences of pixels $\{p_1, p_2, \ldots, p_n\}$, where p_{i+1} is a 4-neighbor of p_i for all $i=1,\ldots,n-1$
- The path is said to be 8-connected if p_{i+1} is an 8-neighbor of p_i

Region Segmentation

- Given a set of image pixels *I*, find a partition *S* of the image *I* into a set of *n* regions R_i:

$$\bigcup_{i=1}^{n} R_i = I$$

Region Similarity

- The uniformity or otherwise of a connected region of pixels may be indicated by a uniformity predicate → a logical statement that is true only if pixels in the region are sufficiently similar in terms of intensity, colour, or some other property.
- A common uniformity predicate:

$$P(R) = \begin{cases} TRUE & if \left| f(i,j) - f(k,l) \right| \leq \Delta \\ FALSE & otherwise \end{cases}$$

Region Similarity

- Where (i,j) and (k,l) are the coordinates or neighbouring pixels in region R.
- It states that a region, R is uniform if (and only if) any two neighboring pixels differ in intensity by no more than Δ.
- Small changes in intensity from neighbour to neighbour can accumulate resulting in a large difference in intensity between opposite sides of a region.

Region Similarity

- A similar predicate:

$$P(R) = \begin{cases} TRUE & if \left| f(i,j) - \mu_R \right| \le \Delta \\ FALSE & otherwise \end{cases}$$

  – Where f(i,j) is the intensity value of a pixel from region R with coordinate (i,j), and μ_R is the mean intensity of all pixels in R except the pixel at (i,j).

Region Growing

- Region growing is bottom-up algorithm that starts with a group of pixels called *seeds* that belong to the structure of interest.
  - The aim is to grow a uniform, connected region from each seed.
  - Comparing the difference between the pixel intensity value and the mean intensity value over a region.
  - The results of region growing depend strongly on the selection of the homogeneity criterion.
  - If it is not properly chosen, the regions *leak* out into adjoining areas or merge with regions that do not belong to the structure of interest.
  - Different starting points may not grow into identical regions.

Region Growing

- A pixel is included in a region if:
  - It has not been assigned to any other region
  - It is a neighbour of that region
  - The new region created by addition of the pixel is still uniform.

Region Growing

- Segmentation of an image must satisfy a number of criteria:
  - ➊ all pixels must be assigned to regions
  - ➋ each pixel must belong to a single region only
  - ➌ each region must be a connected set of pixels
  - ➍ each region must be uniform
  - ➎ any merged pair of adjacent regions must be nonuniform

Region Growing

- Region growing satisfies criteria ❸ and ❹
- Criteria ❶ and ❷ are not satisfied:
  - In general the number of seeds defined will not be sufficient to create a region for every pixel.
- Criteria ❺:
  - Regions grown from two nearby seeds are always regarded as distinct even if those seeds are defined in a part of the image that should be segmented as a single region

Region Growing

- Various approaches to region growing:
  - Zucker, S.W., “Region growing: Childhood and adolescence”, *Computer Graphics and Image Processing*, 1976, **5**, pp.382-399
  - Fu, K.S., and Mui, J.K., “A survey on image segmentation”, *Pattern Recognition*, 1981, **13**(1):pp.3-16
  - Haralick, R.M., and Shapiro, L.G., “Survey: Image segmentation techniques”, *Computer Vision, Graphics and Image Processing*, 1985, **29**:pp.100-132

Split and Merge

- Complete segmentation is possible if a top-down approach is adopted.
  - The entire region is considered initially to be a single region
  - Divide the region into subregions
  - These subregions are then split or merged in an attempt to meet the uniformity criteria.
  - The algorithm iterates until all regions are uniform or until the desired number of regions have been established:

Algorithm

Algorithm: Split and Merge

1. Start with the entire image as a single region
2. Choose a region R. If P(R) is false, split the region into four subregions
3. Consider any two or more neighbouring subregions, $R_1, R_2, \ldots, R_n$, in the image. If $P(R_1 \cup R_2 \cup \cdots \cup R_n)$ is true, merge the *n* regions into a single region.
4. Repeat these steps until no further splits or merges take place.

Region Splitting

- If region splitting alone were used to segment the image, the final partition would likely contain many small, adjacent regions with identical properties.
- Alternate region splitting with a merging stage in which two adjacent regions R_i and R_j are combined into a new larger region if the uniformity predicate for the union of these regions is true.

$$P(R_i \cup R_j)\, is\, TRUE$$

Algorithm

Algorithm: Region Splitting

1. Form initial regions in the image
2. For each region in an image, perform the following steps:
   (a) Compute the variance in the intensity value for the region
   (b) If the variance is above a threshold, split the region
3. Repeat step 2 until no regions are split

Region Merging

- Determine the similarity between two regions.
  - Approaches are based on the intensity of regions or on the weakness of boundaries between the regions.
- There are two approaches to judging the similarity of adjacent regions:
  - Compare their mean intensities. If the mean intensities do not differ by more than some predetermined values, the regions are considered similar and should be candidates for merging.

Region Merging

- Assume that the intensity values are drawn from a probability distribution. Consider whether or not to merge adjacent regions based on the probability that they will have the same statistical distribution of intensity values.

Algorithm

Algorithm: Region Merging

1. Form initial regions in the image
2. For each region in an image, perform the following steps:
   (a) Consider its adjacent region and test to see if they are similar
   (b) If the regions are similar, merge them
3. Repeat step 2 until no regions are merged

Region Labelling

- Find a connected region of pixels that were detected by thresholding, and give all the pixels in that region their own unique value.
- The simplest and most common labeling algorithm scans the image pixel by pixel invoking a recursive labeling procedure whenever a non-zero pixel is found.

Region Labelling

- The algorithm implements the “grassfire” concept
  - Imagine a “fire” is started at a pixel, and it propagates to any of the pixel’s 4- or 8- neighbours that were also detected by thresholding.
  - Wherever a fire is started, the pixel is “burnt-away” (has its value set to zero), so that it cannot be visited again by the labelling algorithm.
  - At the end of the algorithm all pixels belonging to the region have been set to zero in the input image → making it indistinguishable from the background, and the corresponding pixels in the output image have been assigned a region number.
  - The region number is then incremented, ready for the next connected region.