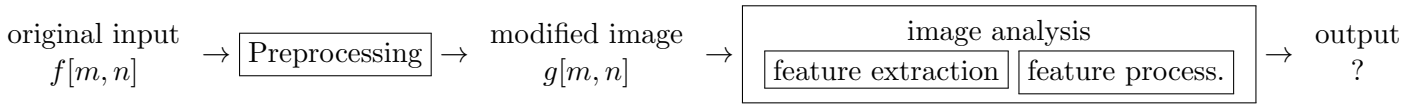
EE416: Introduction to Image Processing and Computer Vision

Il Yong Chun
Department of Electrical and Computer Engineering, the University of Hawai'i, Mānoa

October 13, 2021

3 Image segmentation and topology

This chapter focuses on image analysis methods. Most image analysis techniques extract features from images that are then used for subsequent "higher level" computer processing or vision tasks. A coarse block diagram for image analysis is the following:

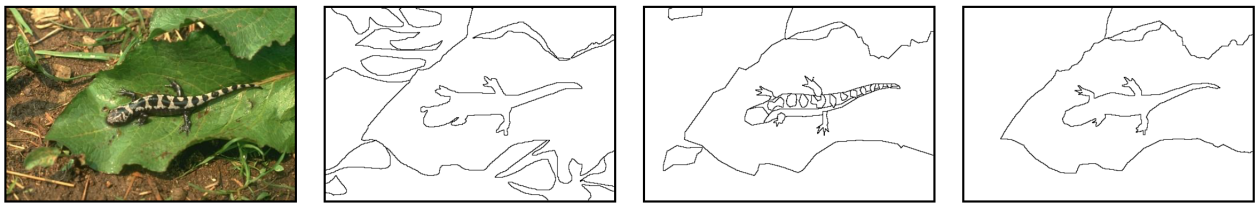


The final output is application dependent, so it can be difficult to form general theoretical frameworks here. Nevertheless, many image analysis methods are very useful in practical applications. For example, the features of interest might be the location of the edges of a tumor in a medical image such as an X-ray CT scan, and the final processed output value might be the estimated volume of that tumor. This type of quantification is important when monitoring cancer progression or treatment. In satellite-based remote sensing, the goal might be to monitor deforestation by assessing the fraction of a region that is forested. Extracted features are used extensively in image analysis and computer vision, e.g., for various object recognition tasks, as well as in medical imaging.

3.1 Edge detection: Basics

3.1.1 Introduction

- Edges
  - Edges naturally occur in images due to the discontinuities formed by occlusion
  - Edges often delineate the boundaries between distinct regions.
  - Edges often contain important visual and semantic information.
- Edge detection
  - The process of identifying pixels that fall along edges
  - For the purpose of subsequent computer processing (e.g., house detection), rather than for human visual interpretation
  - In general, trade-off exists between false alarm and missed detection rates
- Performance metric
  - Evaluation of edge detection methods can be difficult.
  - Correct labeling of edge and non-edge pixels often requires subjective interpretation.



Original Image | Subject 1 | Subject 2 | Subject 3

(Source of images: `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`)

- Best choice of edge detection scheme usually depends on task.
- Performance metrics exist and usually use synthetic data input for evaluation.

3.1.2 Gradient-based methods

- The gradient of images is defined by

$$\nabla f(x,y) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

  where $\frac{\partial f}{\partial x}$ is the "slope in the x direction".
- Compute gradient magnitude and angle:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$
$$\theta = \tan^{-1}\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

  Intuition: Notice that, $\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix}$ gives the "slope" in direction θ. By the Cauchy-Schwarz inequality, i.e., the absolute value of the dot product between two vectors is less than or equal to the product of the magnitudes of the two vectors,[1] the slope in the direction of maximum change is given by the gradient magnitude above.
- This suggests a simple gradient-based edge-detection method using a threshold:

$$e(x,y) = \begin{cases} 1, & |\nabla f(x,y)| > T, \\ 0, & \text{otherwise}, \end{cases}$$

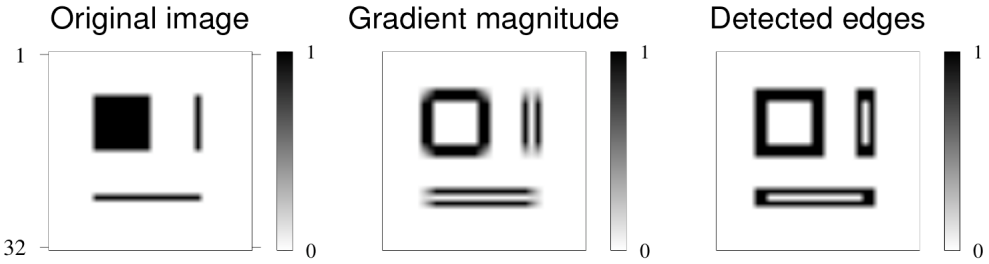
  for some threshold T.
- Choosing T
  - Too large T leads to $\boxed{?}$
  - Too small T leads to false alarms

[1] Simple proof for Euclidean space $\mathbb{R}^N$:

$$\sum_{n=1}^{N}\sum_{n'=1}^{N}(x_n y_{n'} - x_{n'} y_n)^2 = \sum_{n=1}^{N} x_n^2 \sum_{n'=1}^{N} y_{n'}^2 + \sum_{n=1}^{N} y_n^2 \sum_{n'=1}^{N} x_{n'}^2 - 2\sum_{n=1}^{N} x_n y_n \sum_{n'=1}^{N} x_{n'} y_{n'} = 2\left(\sum_{n=1}^{N} x_n^2\right)\left(\sum_{n'=1}^{N} y_{n'}^2\right) - 2\left(\sum_{n=1}^{N} x_n y_n\right)^2.$$

Because the left-hand side of the equation is greater than or equal to zero (because it is a sum of the squares of real numbers), $\left(\sum_{n=1}^{N} x_n^2\right)\left(\sum_{n=1}^{N} y_n^2\right) \geq \left(\sum_{n=1}^{N} x_n y_n\right)^2$. (Q.E.D.) There are many proofs for the Cauchy inequality and the Cauchy inequality also applies to L^2 spaces.

- Unfortunately, this approach yields edges as "strips" rather than "lines", so some method of **edge thinning** is required (cf. 1D case), as illustrated in this example figure.



  One method for edge thinning checks to see if $|\nabla f(x,y)|$ has a local maximum along at least one of the x or y directions.

3.1.3 Gradient-based methods for discrete-space images

Backgrounds in 1D. In 1D, one simple way to approximate the derivative is by **finite differences**:

$$\left.\frac{\mathrm{d}f(t)}{\mathrm{d}t}\right|_{t=n\Delta} \approx \begin{array}{ll} \frac{f[n]-f[n-1]}{\Delta} & \text{(backward difference)} \\ \frac{f[n+1]-f[n]}{\Delta} & \text{(forward difference)} \\ \frac{f[n+1]-f[n-1]}{2\Delta} & \text{(central difference)} \end{array}$$

The latter is equivalent to a filter with the impulse response $h[n] = [1/2, \boxed{0}, -1/2]$ if $\Delta = 1$.
To read: [Wiki: Numerical differentiation]

Conventional methods. Given samples $f[m,n] = f(m\Delta_x, n\Delta_y)$, we can use the central difference, for example to approximate each of the derivatives:

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{x=m\Delta_x, y=n\Delta_y} \approx \frac{f[m+1,n] - f[m-1,n]}{2\Delta_x}$$

and similarly for $\frac{\partial f(x,y)}{\partial y}$. For $\Delta_x = 1$, this is equivalent to 1D convolution with the impulse response $h_x[n] = [1/2, \boxed{0}, -1/2]$.

Directional derivatives can be computed by *applying a spatial filter.*

Other variations include

- Conventional (off center): $\begin{bmatrix} \boxed{-1} & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boxed{-1} & 0 \\ 1 & 0 \end{bmatrix}$
- Roberts (off center): $\begin{bmatrix} \boxed{0} & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \boxed{1} & 0 \\ 0 & -1 \end{bmatrix}$

Prewitt. The conventional filters above are kinds of "high-frequency-boost" filter, so noise will be amplified.

- In 2D, one way to reduce noise is to combine this filter with a low-pass filter in the perpendicular direction, forming a separable filter, such as the **Prewitt** filter (an option to Matlab's imgradientxy.m):

$$\begin{aligned} h[m,n] &= h_x[m]h_y[n], \quad \text{where } h_x[m] = [1/2, \boxed{0}, -1/2] \text{ given above and } h_y[n] = [2, \boxed{2}, 2], \\ &= \begin{bmatrix} 1 & 0 & -1 \\ 1 & \boxed{0} & -1 \\ 1 & 0 & -1 \end{bmatrix}. \end{aligned}$$

(In y-direction, Prewitt filter is given by $h[m,n] = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$.)

- The frequency response of this separable filter is

$$H(e^{\mathrm{i}\mu}, e^{\mathrm{i}\nu}) = \frac{e^{\mathrm{i}\mu} - e^{-\mathrm{i}\mu}}{2} \cdot (1 + 2\cos\nu) = \underbrace{(\mathrm{i}\sin\mu)}_{\text{“derivative”}} \underbrace{(1 + 2\cos\nu)}_{\text{low-pass}}$$

- Why is the low-pass filter part not normalized to be unity at DC?
  $\boxed{?}$

Sobel. There are many such approximations to the derivative, each with an associated name. Matlab's `edge` command has at least six different edge-finding methods.

- For example, in **Sobel's edge detection method**, one compares the gradient magnitude

$$|\nabla f(x,y)||_{x=m\Delta_x, y=n\Delta_y} \approx \sqrt{(f_x[m,n])^2 + (f_y[m,n])^2}$$

  with a threshold, where the **Sobel operator**s are

$$f_x[m,n] \triangleq f[m,n] \circledast h_x[m,n] \quad \text{and} \quad f_y[m,n] \triangleq f[m,n] \circledast h_y[m,n],$$

  in which

$$h_x[m,n] \triangleq \begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad h_y[m,n] \triangleq h_x[n,m] = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

- What is the frequency response of $h_x[m,n]$?
  $\boxed{?}$

- What might be preferable about Sobel over Prewitt?
  $\boxed{?}$

To read: [T1, §5-4.1–§5-4.3], [T2, §10.2.Gradient Operators]

Canny edge detector. In 1986, an MIT student named John Canny proposed the following formation for edge detection [1]:

- Formulate a mathematical model for the edges of interest and the noise.
  This approach is logical because presumably the edge detection method should depend on what type of “edges” are of interest: step edges, erf edges, ridge edges, roof edges...
- Formulate performance criteria that quantify desirable properties of the edge detector.
  This formulation allows objective comparisons of different methods.
- Use optimization methods to determine the best filter (over some class, such as linear methods followed by a threshold) according to the criteria.

This approach sounds promising, but some ad hoc steps crept in along the way, so we skip the derivation. Nevertheless, the method is popular.

- A long derivation led to the following filter called the **derivative of a Gaussian (DoG)**:

$$\boxed{h(x) = \frac{-x}{\chi^2} \exp\left(\frac{-x^2}{2\sigma^2}\right)},$$

  which was within 20% of the optimal filter performance.
- What is the frequency response of this filter? Derivative in space domain corresponds to multiplication by $\mathrm{i}2\pi\nu$ in the frequency domain ($\frac{\mathrm{d}g(x,y)}{\mathrm{d}x} \overset{\text{CSFT}}{\Longleftrightarrow} \mathrm{i}2\pi u G(u,v)$), and the FT of a Gaussian is a Gaussian: $H(v) = \mathrm{i}\alpha v e^{-\beta v^2}$. So for some nonnegative constants α, β, $H(v)$ can be drawn as $\boxed{?}$

- There is a localization-detection trade-off. Filter spatial scale must compromise between localization and detection criteria.

To read: [T1, §5-4.4], [T2, §10.2.The Canny Edge Detector].

3.2 Edge detection: Advances

Laplacian-based method. Locations where the second derivative has a zero crossing should correspond to edges. The generalization of the second derivative to 2D functions is the **Laplacian** given by

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \overset{\text{CSFT}}{\Longleftrightarrow} (\mathrm{i}2\pi u)^2 F(u,v) + (\mathrm{i}2\pi v)^2 F(u,v) = -(2\pi\rho)^2 F(u,v)$$

where $\rho = \sqrt{u^2+v^2}$. The laplacian or Laplace operator is sometime denoted $\nabla \cdot \nabla f$ or Δf. The notation ∇^2 is also used for the Hessian in optimization problems, so one must determine which is meant from context.

So ideas for Laplacian-based edge detection are give as follows:

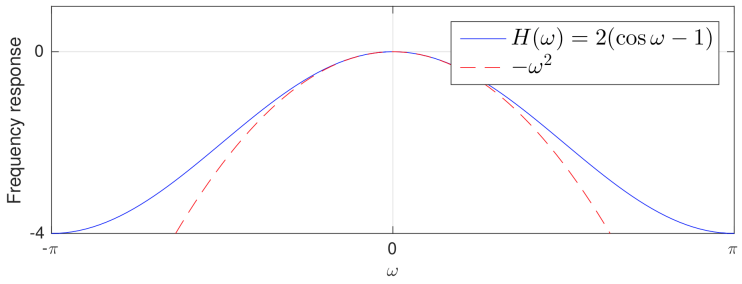
- Look for points such that $\nabla^2 f(x,y) = 0$
- **Problem:** Flat regions also have zero $\nabla^2 f(x,y)$!
- **Criteria:** $\boxed{\nabla^2 f(x,y) = 0 \text{ and } |\nabla f(x,y)| > T.}$

How can we compute (approximately) $\frac{\partial^2 f(x,y)}{\partial x^2}$ for discrete-space images?

- Because convolving with $[1,-1]$ approximates a first-order derivative, convolving again with $[1,-1]$ should approximate a second derivative:

$$h[n] = [\boxed{1}, -1] \circledast [1, \boxed{-1}] = [1, \boxed{-2}, 1]$$

  which has frequency response $H(e^{\mathrm{i}\omega}) = 2(\cos\omega - 1) \approx -\omega^2$ for $\omega \approx 0$. (Recall $\cos\omega \approx 1 - \omega^2/2$ for $\omega \approx 0$ by Taylor series.) See figure below:



- A 2D version of this filter (because the Laplacian is the sum of the derivatives with respect to x and y) corresponds to the following **discrete Laplacian operator**:

$$h_2[m,n] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

which has frequency response $H_2(e^{\mathrm{i}\mu}, e^{\mathrm{i}\nu}) = 2(\cos\mu + \cos\nu - 2) \approx -(\mu^2 + \nu^2)$ for $\mu^2 + \nu^2 \approx 0$.

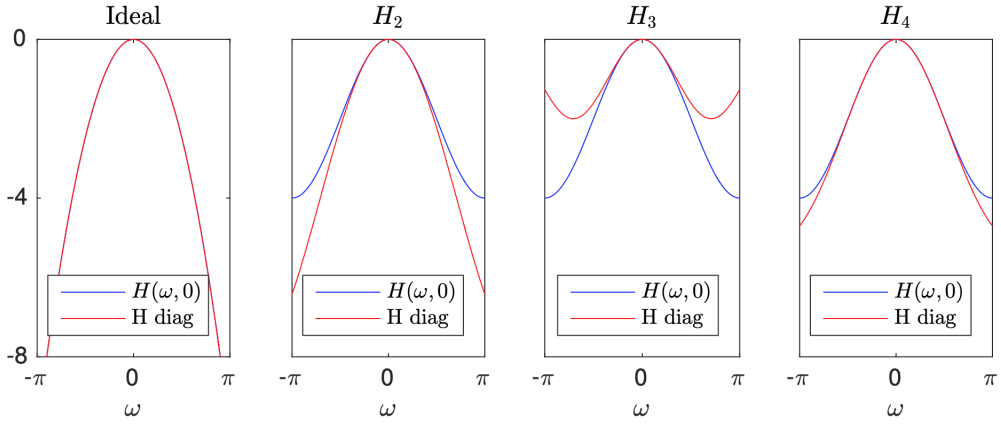
- Many other variations of discrete-space Laplacian filters are possible, e.g.,

$$h_3[m,n] = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & \boxed{-4} & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad h_4[m,n] = \frac{1}{6}(4h_2[m,n] + 2h_3[m,n]) = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & \boxed{-20} & 4 \\ 1 & 4 & 1 \end{bmatrix}.$$

Note that $h_4[m,n] = \frac{1}{6}(h[m]h[n] - 36\delta[m,n])$ where $h[n] = [1, \boxed{4}, 1]$. This form facilitates practical implementation.

Ex. Determine $H_3(e^{\mathrm{i}\mu}, e^{\mathrm{i}\nu})$ and $H_4(e^{\mathrm{i}\mu}, e^{\mathrm{i}\nu})$. Show that $H_3(e^{\mathrm{i}\mu}, e^{\mathrm{i}\nu}) \approx H_4(e^{\mathrm{i}\mu}, e^{\mathrm{i}\nu}) \approx -(\mu^2 + \nu^2)$ for $\mu^2 + \nu^2 \approx 0$.

The following figure shows horizontal $H(e^{\mathrm{i}\omega}, e^{\mathrm{i}0})$ and diagonal $H(e^{\mathrm{i}\omega/2}, e^{\mathrm{i}\omega/2})$ profiles through the frequency responses of the ideal Laplacian filter and the above practical 3×3 filters. Because of transpose symmetry of these filters, $H(e^{\mathrm{i}\omega}, e^{\mathrm{i}0}) = H(e^{\mathrm{i}0}, e^{\mathrm{i}\omega})$.



Ex. Which filter (H_2, H_3, H_4) seems the best and why?
??

Advantages and disadvantages of Laplacian-based methods:

- Advantages:
  - Zero crossing points tend to be "continuous".
  - Edge thinning is not needed.
- Disadvantages:
  - Sensitive to noise. May require preprocessing to reduce noise.
  - Numerous false edges, such as in nearly uniform regions.
    Solution: Use the additional criterion $|\nabla f(x,y)| > T$!

Marr and Hildreth methods. Marr and Hildreth, computer vision experts, argued that we should separate edges at different scales to improve subsequent image analysis. (This can help address the spurious edge problem.)

- By filtering the image with various low-pass filters prior to applying an edge detection method, we can remove "small" details and just leave the "large" scale variations for the edge detector to find, where "large" and "small" depend on the amount of filtering.
- A typical choice is a Gaussian filter:

$$h_l(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \overset{\text{CSFT}}{\Longleftrightarrow} H_l(u,v) = \exp\left(-\pi(\rho\sigma\sqrt{2\pi})^2\right)$$

  where $\rho = \sqrt{u^2+v^2}$ and σ is related to the width of the PSF.
- If the subsequent edge detection method is Laplacian based, then we can combine the filtering with the Laplacian:

$$\nabla^2\{f(x,y) \circledast h_l(x,y)\} = f(x,y) \circledast \nabla^2 h_l(x,y) = f(x,y) \circledast h(x,y)$$

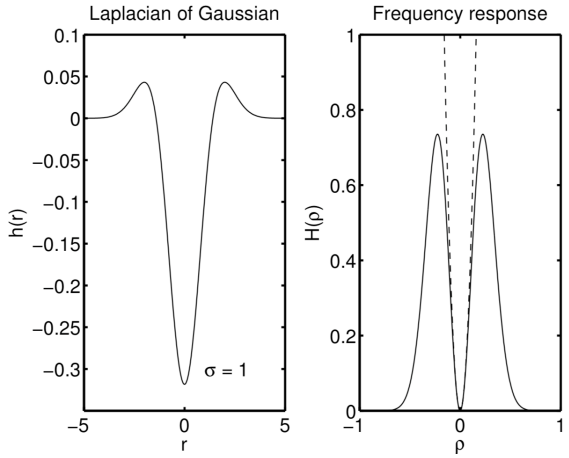
  due to the linearity of convolution, where one can show with some calculus that

$$h(x,y) = \nabla^2 h_l(x,y) = \boxed{\frac{x^2+y^2-2\sigma^2}{\sigma^4} h_l(x,y)}$$

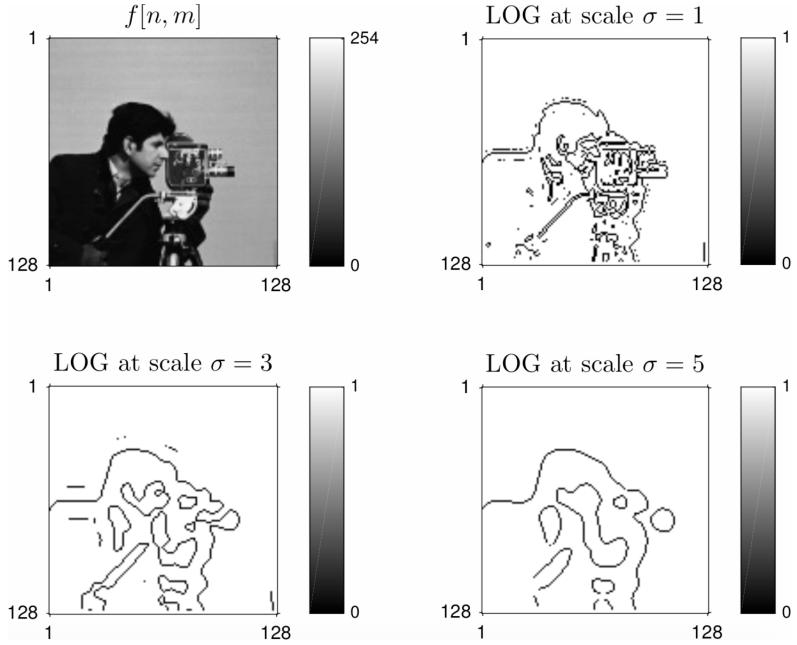
  This is called the **Laplacian of Gaussian (LoG)** filter. Note that it is not separable, but it is radially symmetric.
- By the differentiation properties of the CSFT, its frequency response is given by

$$H(u,v) = \left\{(\mathrm{i}2\pi u)^2 + (\mathrm{i}2\pi v)^2\right\} H_l(u,v) = -(2\pi\rho)^2 \exp\left(-\pi(\rho\sigma\sqrt{2\pi})^2\right),$$

  which is a bandpass filter with maximum magnitude at $\rho_{\max} = \frac{1}{2\pi\sigma\sqrt{2}}$:



  The above figure shows the impulse response $h(r)$ of the LoG filter, and its magnitude frequency response $|H(\rho)|$ which approximates the "ideal" parabola ρ^2 shown as a dashed line in the plot.
- The following figure shows edges at various scales, from Matlab's edge command with 'log' option.



To read: [T2, §10.2.The Marr-Hildreth Edge Detector]

3.3 Image topology Basics

3.3.1 2D neighborhoods

- 4-point neighborhood

$$\begin{array}{ccc} & \circ & \\ \circ & \times & \circ \\ & \circ & \end{array}$$

$\mathcal{N}(i,j) = \{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\}$

- 8-point neighborhood

$$\begin{array}{ccc} \circ & \circ & \circ \\ \circ & \times & \circ \\ \circ & \circ & \circ \end{array}$$

$\mathcal{N}(i,j) = \{(i+1,j),(i-1,j),(i,j+1),(i,j-1),(i+1,j+1),(i-1,j+1),(i+1,j-1),(i-1,j-1)\}$

- More generally, a **Neighborhood System** is any mapping with the two properties:
  1. For all $s \in \mathcal{S}$, $s \notin \mathcal{N}s$;
  2. For all $r \in \mathcal{S}$, $r \in \mathcal{N}s \rightarrow s \in \mathcal{N}r$.

3.3.2 Boundary conditions

- How do you process pixels on the boundary of an image?
- Consider the following example using a 4-point neighborhood

An example image

$$\begin{array}{cccc} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{array}$$

4 neighbors of l

$$\begin{array}{ccc} & l_1 & \\ l_4 & l & l_2 \\ & l_3 & \end{array}$$

- Free boundary condition: $\mathcal{N}l = \{h, p, k\}$, $\mathcal{N}p = \boxed{?}$
- Periodic boundary condition: $\mathcal{N}l = \{h, i, p, k\}$, $\mathcal{N}p = \{l, \boxed{?}, \boxed{?}, o\}$
- Reflective boundary condition: $\mathcal{N}l = \{h, k, p, k\}$, $\mathcal{N}p = \{l, \boxed{?}, \boxed{?}, o\}$

3.3.3 Connected sets

Connected component analysis.

- Once region boundaries are detected, it is often useful to extract regions which are not separated by a boundary.
- Any set of pixels which is not separated by a boundary is called **connected**.
- Each maximal region of connected pixels is called a **connected component**.
- The set of connected components partition an image into segments.

Connected neighbors.

- Let $\mathcal{N}s$ be a neighborhood system. (E.g., 4-point neighborhood system, 8-point neighborhood system)
- Let $c(s)$ be the set of neighbors that are connected to point s.
  For all s and r, the set $c(s)$ must have the properties that
  - $c(s) \subset \mathcal{N}s$
  - $r \in c(s) \Leftrightarrow s \in c(r)$
- Example: $c(s) = \{r \in \mathcal{N}s : X_r = X_s\}$
- Example: $c(s) = \{r \in \mathcal{N}s : |X_r - X_s| < T\}$ for some threshold value T

Connected set.

Definition 3.1. *A region $R \subset S$ is said to be connected under $c(s)$, if for all $s, r \in R$, there exists a sequence of M pixels, $s_1, \ldots, s_M$, such that*

$$s_1 \in c(s), s_2 \in c(s_1), \ldots, s_M \in c(s_{M-1}), r \in c(s_M),$$

i.e., there exist a connected path from s to r.

Ex. Consider the following image X_s:

$$\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array} \qquad \begin{array}{l} S_1 = \{s : X_s = 1\} \\ S_0 = \{s : X_s = 0\} \end{array}$$

and define $c(s) = \{r \in \mathcal{N}s : X_r = X_s\}$. Then we see that

- in 4-point neighborhood system, S_0 and S_1 are connected sets? y/n $\boxed{?}$;
- in 8-point neighborhood system, S_0 and S_1 are connected sets? y/n $\boxed{?}$.

3.4 Segmentation by region growing

3.4.1 Region growing

The main idea is to find a connected set by growing a region from a seed point s_0.

Pseudo code.

Assume that $c(s)$ is given.

```
ClassLabel = 1
Initialize Y_r for all r ∈ S
ConnectedSet(s_0, Y, ClassLabel){
    B ← {s_0}
    While(B is not empty){
        s ← any element of B
        B ← B \ {s}
        Y_s ← ClassLabel
        B ← B ∪ {r ∈ c(s) : Y_r = 0}
    }
    return(Y)
}
```

Ex. Consider the following image X with 4-point neighborhood system and $c(s) = \{r \in \mathcal{N}s : X_r = X_s\}$:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

Then, we segment the image with the following procedure:

- The list of $(n, m) \in B$: $(0, 0)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(1, 0)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(1, 1)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(2, 1)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(3, 1), (2, 2)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	0	0	0
	4	0	0	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(4, 1), (3, 2), (2, 2)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(3, 2), (2, 2)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$: $(2, 2)$, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- The list of $(n, m) \in B$ is empty, and

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

To read: [T2, §10.4.Region Growing]

3.4.2 Connected components extraction

Idea.

- Iterate through each pixel in the image.
- Extract connected set for each unlabeled pixel.

Pseudo code.

```
ClassLabel = 1
Initialize Y_r for all r ∈ S
For each s ∈ S{
    If(Y_s = 0){
        ConnectedSet(s, Y, ClassLabel) }
        ClassLabel ← ClassLabel + 1
    }
}
```

Ex. Consider the following image X with 4-point neighborhood system:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

Then, we extract connected components with the following procedure:

- For $s = (n, m) = (0, 0)$ and $\mathit{ClassLabel} = 1$, we obtain

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- For $s = (n, m) = (0, 1)$ and $\mathit{ClassLabel} = 2$, we obtain

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	2	2	2	2
	1	1	1	2	2	2
	2	0	1	1	2	2
	3	0	1	1	2	2
	4	0	1	2	2	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- For $s = (n, m) = (2, 0)$ and $\mathit{ClassLabel} = 3$, we obtain

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	2	2	2	2
	1	1	1	2	2	2
	2	3	1	1	2	2
	3	3	1	1	2	2
	4	3	1	2	2	0

, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

- For $s = (n, m) = (4, 4)$ and $\mathit{ClassLabel} = 4$, we obtain

the segmented image Y :

		m				
		0	1	2	3	4
n	0	1	2	2	2	2
	1	1	1	2	2	2
	2	3	1	1	2	2
	3	3	1	1	2	2
	4	3	1	2	2	4

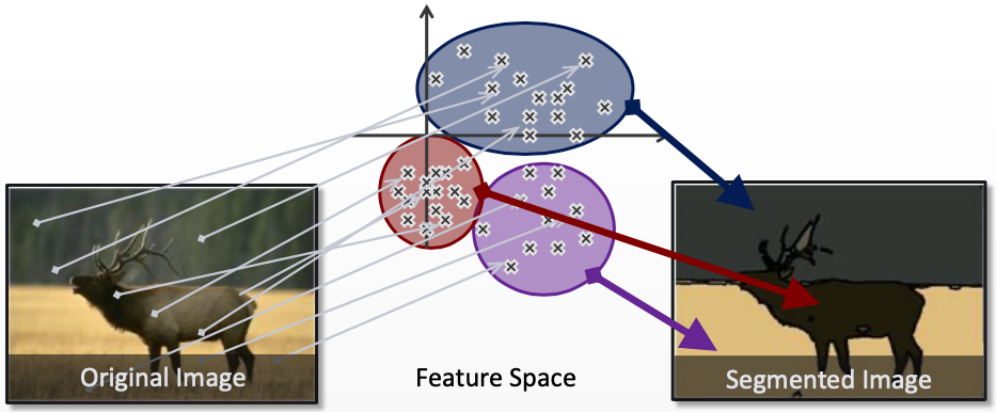
, where the image X:

		m				
		0	1	2	3	4
n	0	1	0	0	0	0
	1	1	1	0	0	0
	2	0	1	1	0	0
	3	0	1	1	0	0
	4	0	1	0	0	1

Problem: Connected component analysis can result in many small disjointed regions.
Solution: Region segmentation, e.g., region merging, region splitting, and split & merge [T2, §10.4.Region Splitting and Merging].

3.5 Segmentation by clustering

A clustering algorithm is used to find structure in the data. The pixels are represented in the feature space. Usual features include colors, pixel coordinates, and texture descriptors.



(Source of images: `http://ivrgwww.epfl.ch/`)

The basic idea behind the clustering approach is

to a partition a set of P observations (e.g., pixels) into a specified number of clusters K.

In K-means clustering, each observation is assigned to the cluster with the nearest mean, and each mean is called the *protocype* of its cluster. A **K-means algorithm** is an iterative procedure that successively refines the means until convergence is achieved.

3.5.1 Partitioning-clustering approach

Let $\{\mathbf{z}_1, \ldots, \mathbf{z}_P\}$ be the set of vector observations (samples). These vectors have the form $\mathbf{z} = [z_1, \ldots, z_Q]^T$. Each component of a vector $\mathbf{z}$ represents a numerical pixel attribute. For example, if we segment gray scale images, then $\mathbf{z} = z$. If we segment RGB color images, then $Q = 3$, each component of which is the intensity of a pixel in on of the three primary color images.

- The objective of K-means clustering is to partition the set of P observations into K ($K \leq P$) disjoint cluster set $C = \{C_1, \ldots, C_K\}$, so that the following criterion of optimality is satisfied:

$$\underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\mathbf{z} \in C_k} \|\mathbf{z} - \mathbf{m}_k\|_2^2$$

  where $\mathbf{m}_k$ is the **mean vector** (or **centroid**) of the samples in set C_k and $\|\cdot\|_2$ is the ℓ^2 vector norm defined by $\|\mathbf{x}\|_2 \triangleq \sqrt{\sum_n x_n^2}$.
- In words, we are interested in finding the set $C = \{C_1, \ldots, C_K\}$ such that the *sum of the distances* from each point in a set to the mean of that set is minimum.
- Finding the minimum is an NP-hard problem.

3.5.2 Region segmentation using K-means clustering

To find approximations to the minimum, we will consider the "standard" **K-means algorithm.**

Pseudo code.

0) **Initialization:** Specify an initial set of means $\mathbf{m}_k$, $k = 1, \ldots, K$.
1) **Assign samples to clusters:** Assign each sample to the cluster set whose mean is the closest (ties are resolved arbitrarily, but samples are assigned to only *one* cluster):

$$\mathbf{z}_p \to C_k \quad \text{if } \|\mathbf{z}_p - \mathbf{m}_k\|_2^2 < \|\mathbf{z}_p - \mathbf{m}_{k'}\|_2^2, \quad k' \neq k, \; p = 1, \ldots, P.$$

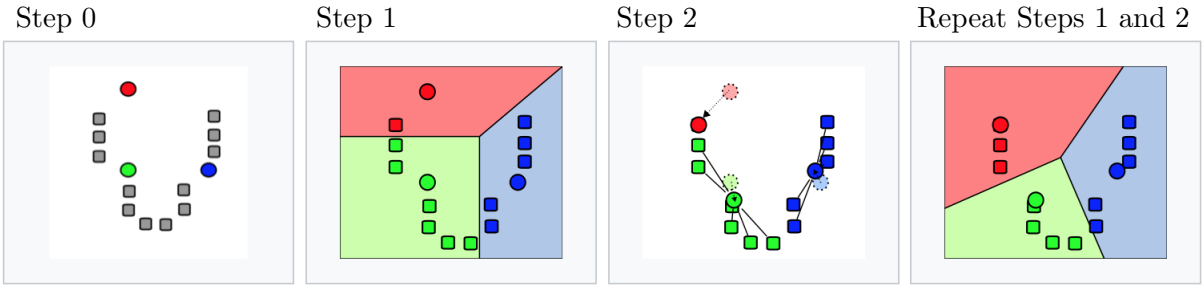
2) **Update the cluster centers (means):**

$$\mathbf{m}_k = \frac{1}{|C_k|} \sum_{\mathbf{z} \in C_k} \mathbf{z}, \quad k = 1, \ldots, K$$

where $|C_k|$ is the number of samples in cluster set C_k.
3) **Go back** to the Step 1; stop when no more new assignment (i.e., membership in each cluster no longer changes).

The K-means algorithm is the simplest partitioning method for clustering analysis and widely used in data mining applications. The computational complexity is $\mathcal{O}(K \cdot P \cdot N_{\text{Iter}})$, where N_{Iter} is the number of iterations; typically, $K, P \ll N_{\text{Iter}}$.

Illustration of K-means clustering iterations:

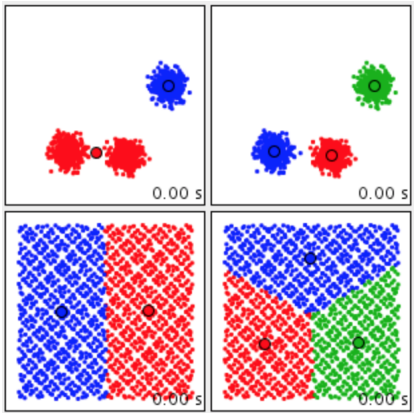


(Source of images: [Wiki])

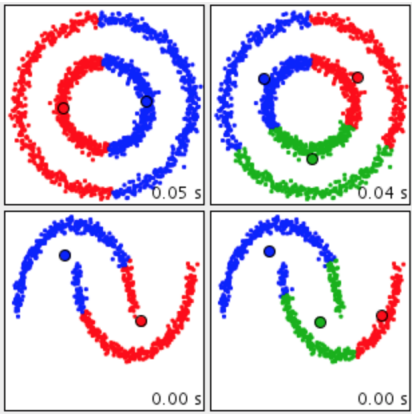
Limitations of K-means clustering algorithm:

- Number of clusters has to be known a priori priori (specify K in advance).
- Sensitive to initial seed points, could stuck in a local minimum
- Not suitable for discovering clusters with nonconvex shapes[2]

Examples of clusters with convex shapes



Examples of clusters with nonconvex shapes



[2] An object is convex if for every pair of points within the object, every point on the straight line segment that joins them is also within the object.

(Source of images: `http://commons.apache.org/proper/commons-math/userguide/ml.html`)

There are several **variants** of K-means clusterings to overcome its weakness:

- **K-Medoids:** Resistance to noise and/or outliers by choosing data points as centers (**medoids**) and using ℓ^1 norm and other distances.
- **K-Modes:** Extension to categorical data clustering analysis
- **CLARA:** Extension of the K-medoids approach to deal with large data sets
- **Mixture models (EM algorithm):** Handles uncertainty of clusters

Homework (due by 10/18, 11:55 PM; Upload your solution and Matlab codes to Laulima/Assignments)

Prob. 1. Your objective is to perform edge detection on the sampled image $s[m,n] = f(m\Delta, n\Delta)$, where $f(x,y)$ is the associated continuous space image and $\Delta = 1$. You will do this using a *combination* of gradient and Laplacian based operators.
a) Specify the condition for the detection of edges on the continuous image $f(x,y)$ using derivatives over x and y, and a single threshold T. (Hint: two conditions.)
b) Specify an approximate discretized gradient operator for the image $s[m,n]$. (Hint: use "forward" finite differences in x and y.)
c) Specify an approximate discretized Laplacian operator for the image $s[m,n]$.
d) Specify the condition for the detection of edges on the discretized image $s[m,n]$ using approximate discretized gradient and Laplacian operators. (Hint: How can we determine zero-crossings?)
e) Describe how the thresholding T should be selected. What are the tradeoffs in its selection?

Prob. 2 (Computer project). Consider the following definitions for neighborhood and connectedness:

- We will use a 8 point neighborhood and the free boundary condition. (so pixels along the boundary of the image have less than 8 neighbors each.)
- We will denote the connected neighbors of s by the following set $c(s) \subset \mathcal{N}s$:

$$c(s) = \{r \in \mathcal{N}s : |x_s - x_r| \leq T\} \tag{1}$$

where T is some thresholding value.

a) Write a function Y = ConnectedSet(s, T, img, Label) to implement the region-growing algorithm studied in class, where s is the data containing the location of the pixel s whose connected neighbors will be computed, T is the threshold in (1), img is the 2D array of pixels, Label is the integer value that will be used to label any pixel which is connected to s, and Y is a 2D array of integers which contains the class of each pixel and is passed to ConnectedSet from the main routine.
b) Use the subroutine ConnectedSet to segment the defective weld regions in the image defective-weld.tif (i.e., binary segmentation). First, threshold the image with a threshold value 254 to give multiple seed points (seed point values are 255); read instruction in [T2, Example 10.20]. Second, apply ConnectedSet with $T = 68$ and obtained seed points via thresholding above to segment the defective weld regions. Third, modify ConnectedSet to reduce the computational complexity (Hint: you don't need to re-examine points already examined in previous seed points). Finally, tune T values to better segment the defective weld regions. Note that for a small threshold, the size of most connected sets for this image will be small, resulting in a large number of connected sets in the segmentation.
c) In the report, include the original image defective-weld.tif, segmented defective weld regions from it (both with $T = 68$ and your "best" tuning), and your codes.

References

[1] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 6, pp. 679–698, Nov. 1986.