IE404

Digital Image Processing

Instructor: Manish Khare

Lecture 12-15

Principle Objective of Enhancement

- Process an image so that the result will be more suitable than the original image for a specific application.

- The suitableness is up to each application.

- A method which is quite useful for enhancing an image may not necessarily be the best approach for enhancing another images

2 Domains

- Spatial Domain : (image plane)
  - Techniques are based on direct manipulation of pixels in an image
- Frequency Domain :
  - Techniques are based on modifying the Fourier transform of an image
- There are some enhancement techniques based on various combinations of methods from these two categories.

Good Images

- For human visual perception
  - The visual evaluation of image quality is a highly subjective process.
  - It is hard to standardize the definition of a good image.

- For machine perception
  - The evaluation task is easier.
  - A good image is one which gives the best machine recognition results.

- A certain amount of trial and error usually is required before a particular image enhancement approach is selected.

Spatial Domain

- Procedures that operate directly on pixels.

$$g(x,y) = T[f(x,y)]$$

where

- $f(x,y)$ is the input image
- $g(x,y)$ is the processed image
- T is an operator on f defined over some neighborhood of (x,y)

Mask/Filter

- Neighborhood of a point (x,y) can be defined by using a square/rectangular (common used) or circular subimage area centered at (x,y)
- The center of the subimage is moved from pixel to pixel starting at the top of the corner

Point Processing

- Neighborhood = 1x1 pixel
- g depends on only the value of f at (x,y)
- T = gray level (or intensity or mapping) transformation function

$$s = T(r)$$

- Where
  - r = gray level of $f(x,y)$
  - s = gray level of $g(x,y)$

Mask Processing or Filter

- Neighborhood is bigger than 1x1 pixel
- Use a function of the values of f in a predefined neighborhood of (x,y) to determine the value of g at (x,y)
- The value of the mask coefficients determine the nature of the process
- Used in techniques
  - Image Sharpening
  - Image Smoothing

Contrast Stretching

Intensity Transformation Functions: Contrast Stretching

- Produce higher contrast than the original by
  - darkening the levels below m in the original image
  - Brightening the levels above m in the original image

Thresholding

- Produce a two-level (binary) image

Intensity Transformation Functions: Thresholding Function

3 basic gray-level transformation functions

- Linear function
  - Negative and identity transformations
- Logarithm function
  - Log and inverse-log transformation
- Power-law function
  - n^{th} power and n^{th} root transformations

Identity function

- Output intensities are identical to input intensities.
- Is included in the graph only for completeness.

Image Negatives

- An image with gray level in the range ***[0, L-1]***
  where $L = 2^n$; n = 1, 2…
- Negative transformation :

$$s = L - 1 - r$$

- Reversing the intensity levels of an image.
- Suitable for enhancing white or gray detail embedded in dark regions of an image, especially when the black area dominant in size.

Example of Negative Image

Original mammogram showing a small lesion of a breast

Negative Image : gives a better vision to analyze the image

Log Transformations

$$s = c \log (1+r)$$

- c is a constant and $r \geq 0$
- Log curve maps a narrow range of low gray-level values in the input image into a wider range of output levels.
- Used to expand the values of dark pixels in an image while compressing the higher-level values.

Log Transformations

- It compresses the dynamic range of images with large variations in pixel values
- Example of image with dynamic range: Fourier spectrum image
- It can have intensity range from 0 to 10^6 or higher.
- We can't see the significant degree of detail as it will be lost in the display.

Example of Logarithm Image

Fourier Spectrum with range = 0 to 1.5×10^6

Result after apply the log transformation with c = 1, range = 0 to 6.2

Inverse Logarithm Transformations

- Do opposite to the Log Transformations
- Used to expand the values of high pixels in an image while compressing the darker-level values.

Power-Law (Gamma) Transformations

$$s = cr^{\gamma}$$

- c and γ are positive constants
- Power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels.
- c = γ = 1 ⇨ Identity function

Gamma correction

- Cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with γ varying from 1.8 to 2.5
- The picture will become darker.
- Gamma correction is done by preprocessing the image before inputting it to the monitor with $s = cr^{1/\gamma}$

Another example : MRI

a	b
c	d

(a) a magnetic resonance image of an upper thoracic human spine with a fracture dislocation and spinal cord impingement

- The picture is predominately dark
- An expansion of gray levels are desirable ⇨ needs $\gamma < 1$

(b) result after power-law transformation with $\gamma = 0.6$, c=1

(c) transformation with $\gamma = 0.4$ (best result)

(d) transformation with $\gamma = 0.3$ (under acceptable level)

Effect of decreasing gamma

- When the γ is reduced too much, the image begins to reduce contrast to the point where the image started to have very slight "wash-out" look, especially in the background

Another example

a	b
c	d

(a) image has a washed-out appearance, it needs a compression of gray levels ⇨ needs $\gamma > 1$

(b) result after power-law transformation with $\gamma = 3.0$ (suitable)

(c) transformation with $\gamma = 4.0$ (suitable)

(d) transformation with $\gamma = 5.0$ (high contrast, the image has areas that are too dark, some detail is lost)

Piecewise-Linear Transformation Functions

- Advantage:
  - The form of piecewise functions can be arbitrarily complex
- Disadvantage:
  - Their specification requires considerably more user input

Contrast Stretching

- Low contrast images can result from poor illuminations.

- Lack of dynamic range in the imaging sensor, or even the wrong setting of a lens aperture during image acquisition.

- It expands the range of intensity levels in an image so that it spans the full intensity range of display devices.

- Contrast stretching is obtained by setting

$$(r1,s1) = (rmin , 0) \text{ and } (r2,s2) = (rmax , L-1)$$

Contrast Stretching

- increase the dynamic range of the gray levels in the image
- (b) a low-contrast image : result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture of image acquisition
- (c) result of contrast stretching: $(r_1,s_1) = (r_{min},0)$ and $(r_2,s_2) = (r_{max},L-1)$
- (d) result of thresholding

Gray-level slicing

- Highlighting a specific range of gray levels in an image
  - Display a high value of all gray levels in the range of interest and a low value for all other gray levels
- (a) transformation highlights range [A,B] of gray level and reduces all others to a constant level
- (b) transformation highlights range [A,B] but preserves all other levels

Bit-plane slicing

- Highlighting the contribution made to total image appearance by specific bits
- Suppose each pixel is represented by 8 bits
- Higher-order bits contain the majority of the visually significant data
- Useful for analyzing the relative importance played by each bit of the image

Example

An 8-bit fractal image

- The (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation.
  - Map all levels between 0 and 127 to 0
  - Map all levels between 129 and 255 to 255

8 bit planes

Bit-plane 7	Bit-plane 6	
Bit-plane 5	Bit-plane 4	Bit-plane 3
Bit-plane 2	Bit-plane 1	Bit-plane 0

a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

a b c

FIGURE 3.15 Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

Histogram Processing

- Histogram of a digital image with gray levels in the range [0,L-1] is a discrete function

$$h(r_k) = n_k$$

- Where
  - r_k : the kth gray level
  - n_k : the number of pixels in the image having gray level r_k
  - $h(r_k)$: histogram of a digital image with gray levels r_k

Normalized Histogram

- dividing each of histogram at gray level r_k by the total number of pixels in the image, n

$$p(r_k) = n_k / n$$

- For k = 0,1,…,L-1
- $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k
- The sum of all components of a normalized histogram is equal to 1

Histogram Processing

- Basic for numerous spatial domain processing techniques
- Used effectively for image enhancement
- Information inherent in histograms also is useful in image compression and segmentation

Example

Dark image

Components of histogram are concentrated on the low side of the gray scale.

Bright image

Components of histogram are concentrated on the high side of the gray scale.

Example

Low-contrast image

histogram is narrow and centered toward the middle of the gray scale

High-contrast image

histogram covers broad range of the gray scale and the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than the others

Histogram Equalization

- As the low-contrast image's histogram is narrow and centered toward the middle of the gray scale, if we distribute the histogram to a wider range the quality of the image will be improved.
- We can do it by adjusting the probability density function of the original histogram of the image so that the probability spread equally

Histogram Equalization

$$s = T(r)$$

- Where $0 \leq r \leq 1$
- T(r) satisfies
  - (a). T(r) is single-valued and monotonically increasingly in the interval $0 \leq r \leq 1$
  - (b). $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$

Histogram Equalization

- Thus, an output image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k in the output image
- In discrete space, it cannot be proved in general that this discrete transformation will produce the discrete equivalent of a uniform probability density function, which would be a uniform histogram

Histogram Processing

a b

FIGURE 3.17 (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

Histogram Equalization

Intensity levels in an image may be viewed as random variables in the interval [0,L-1]

Fundamental descriptor of a random variable is its probability density function (PDF)

Let $p_r(r)$ and $p_s(s)$ denote the PDFs of r and s respectively

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = (L-1)p_r(r)$$

$$p_s(s) = \frac{1}{L-1}$$

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = (L-1)\sum_{j=0}^{k} \frac{n_j}{MN}, \quad k = 0,1,2,\ldots,L-1$$

a b

FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

Example

	before	after	Histogram equalization
Dark Image			
Bright Image			

Example

	before	after	Histogram equalization
Low Contrast Image			
High Contrast Image			

The quality is not improved much because the original image already has a broaden gray-level scale

Example

2	3	3	2
4	2	4	3
3	2	3	5
2	4	2	4

4x4 image

Gray scale = [0,9]

histogram

Gray Level(j)	0	1	2	3	4	5	6	7	8	9
No. of pixels	0	0	6	5	4	1	0	0	0	0
$\sum_{j=0}^{k} n_j$	0	0	6	11	15	16	16	16	16	16
$s = \sum_{j=0}^{k} \frac{n_j}{n}$	0	0	6 / 16	11 / 16	15 / 16	16 / 16	16 / 16	16 / 16	16 / 16	16 / 16
s x 9	0	0	3.3 ≈3	6.1 ≈6	8.4 ≈8	9	9	9	9	9

Example

3	6	6	3
8	3	8	6
6	3	6	9
3	8	3	8

Output image

Gray scale = [0,9]

Histogram equalization

Note

- It is clearly seen that
  - Histogram equalization distributes the gray level to reach the maximum gray level (white) because the cumulative distribution function equals 1 when $0 \leq r \leq L-1$
  - If the cumulative numbers of gray levels are slightly different, they will be mapped to little different or same gray levels as we may have to approximate the processed gray level of the output image to integer number
  - Thus the discrete transformation function can't guarantee the one to one mapping relationship

Histogram Matching (Specification)

- Histogram equalization has a disadvantage which is that it can generate only one type of output image.
- Histogram equalization automatically determines a transformation function produce uniform histogram
- When automatic enhancement is desired, equalization is a good approach
- There are some applications in which attempting to base enhancement on a uniform histogram is not the best approach
- In particular, it is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have.
- The method used to generate a processed image that has a specified histogram is called *histogram matching* or *specification*

Consider the continuous domain

Let $p_r(r)$ denote continuous probability density function of gray-level of input image, r

Let $p_z(z)$ denote desired (specified) continuous probability density function of gray-level of output image, z

Let s be a random variable with the property

$$s = T(r) = \int_0^r p_r(w)dw$$

Histogram equalization

Where w is a dummy variable of integration

Next, we define a random variable z with the property

$$g(z) = \int_0^z p_z(t)dt = s$$ ➡ Histogram equalization

Where t is a dummy variable of integration

thus

$$s = T(r) = G(z)$$

Therefore, z must satisfy the condition

$$z = G^{-1}(s) = G^{-1}[T(r)]$$

Assume G^{-1} exists and satisfies the condition (a) and (b)

We can map an input gray level r to output gray level z

ALGORITHM

- INPUT: Input image, Target image
- OUTPUT: Output image that has the same characteristic as the target image Steps:
- Read the Input image and the Target image.
- Obtain the histogram of the input image and the target image.
- Equalize the input and the target images using the equation (1).
- Calculate the transformation function G of the target Image.
- Map the original image gray level r_k to the final gray level z_k

Procedure

1. Obtain the transformation function T(r) by calculating the histogram equalization of the input image

$$s = T(r) = \int_0^r p_r(w)dw$$

2. Obtain the transformation function G(z) by calculating histogram equalization of the desired density function

$$G(z) = \int_0^z p_z(t)dt = s$$

Procedure

3. Obtain the inversed transformation function G^{-1}

4. Obtain the output image by applying the processed gray-level from the inversed transformation function to all the pixels in the input image

$$z = G^{-1}(s) = G^{-1}[T(r)]$$

Example

Assume an image has a gray level probability density function $p_r(r)$ as shown.

$$p_r(r) = \begin{cases} -2r + 2 & ;0 \le r \le 1 \\ 0 & ;\text{elsewhere} \end{cases}$$

$$\int_0^r p_r(w)dw = 1$$

Example

We would like to apply the histogram specification with the desired probability density function $p_z(z)$ as shown.

$$p_z(z) = \begin{cases} 2z & ;0 \le z \le 1 \\ 0 & ;\text{elsewhere} \end{cases}$$

$$\int_0^z p_z(w)dw = 1$$

Step 1:

Obtain the transformation function T(r)

$$s = T(r) = \int_0^r p_r(w)dw$$

$$= \int_0^r (-2w + 2)dw$$

$$= -w^2 + 2w\Big|_0^r$$

$$= -r^2 + 2r$$

Step 2:

Obtain the transformation function G(z)

$$G(z) = \int_0^z (2w)dw \qquad = z^2\Big|_0^z \qquad = z^2$$

Step 3:

Obtain the inversed transformation function G^{-1}

$$G(z) = T(r)$$

$$z^2 = -r^2 + 2r$$

$$z = \sqrt{2r - r^2}$$

We can guarantee that $0 \le z \le 1$ when $0 \le r \le 1$

Discrete formulation

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0,1,2,\ldots,L-1$$

$$G(z_k) = \sum_{i=0}^{k} p_z(z_i) = s_k \qquad k = 0,1,2,\ldots,L-1$$

$$z_k = G^{-1}\left[T(r_k)\right]$$

$$= G^{-1}\left[s_k\right] \qquad k = 0,1,2,\ldots,L-1$$

Example

Image of Mars moon

Image is dominated by large, dark areas, resulting in a histogram characterized by a large concentration of pixels in pixels in the dark end of the gray scale

Image Equalization

Transformation function for histogram equalization

Histogram of the result image

Result image after histogram equalization

The histogram equalization doesn't make the result image look better than the original image. Consider the histogram of the result image, the net effect of this method is to map a very narrow interval of dark pixels into the upper end of the gray scale of the output image. As a consequence, the output image is light and has a washed-out appearance.

Solve the problem

Since the problem with the transformation function of the histogram equalization was caused by a large concentration of pixels in the original image with levels near 0

a reasonable approach is to modify the histogram of that image so that it does not have this property

Histogram Specification

- (1) the transformation function G(z) obtained from

$$G(z_k) = \sum_{i=0}^{k} p_z(z_i) = s_k$$

$$k = 0,1,2,\ldots,L-1$$

- (2) the inverse transformation $G^{-1}(s)$

Result image and its histogram

Original image

After applied the histogram equalization

The output image's histogram

Notice that the output histogram's low end has shifted right toward the lighter region of the gray scale as desired.

Note

- Histogram specification is a trial-and-error process
- There are no rules for specifying histograms, and one must resort to analysis on a case-by-case basis for any given enhancement task.
- Histogram processing methods are global processing, in the sense that pixels are modified by a transformation function based on the gray-level content of an entire image.
- Sometimes, we may need to enhance details over small areas in an image, which is called a local enhancement.

Local Enhancement

(a) (b) (c)

a) Original image (slightly blurred to reduce noise)
b) global histogram equalization (enhance noise & slightly increase contrast but the construction is not changed)
c) local histogram equalization using 7x7 neighborhood (reveals the small squares inside larger ones of the original image.

- define a square or rectangular neighborhood and move the center of this area from pixel to pixel.
- at each location, the histogram of the points in the neighborhood is computed and either histogram equalization or histogram specification transformation function is obtained.
- another approach used to reduce computation is to utilize nonoverlapping regions, but it usually produces an undesirable checkerboard effect.

Explain the result in c)

- Basically, the original image consists of many small squares inside the larger dark ones.
- However, the small squares were too close in gray level to the larger ones, and their sizes were too small to influence global histogram equalization significantly.
- So, when we use the local enhancement technique, it reveals the small areas.
- Note also the finer noise texture is resulted by the local processing using relatively small neighborhoods.

Enhancement using Arithmetic/Logic Operations

- Arithmetic/Logic operations perform on pixel by pixel basis between two or more images
- except NOT operation which perform only on a single image

Logic Operations

- Logic operation performs on gray-level images, the pixel values are processed as binary numbers
- light represents a binary 1, and dark represents a binary 0
- NOT operation = negative transformation

Example of AND Operation

original image

AND image mask

result of AND operation

Example of OR Operation

original image

OR image mask

result of OR operation

Image Subtraction

$$g(x,y) = f(x,y) - h(x,y)$$

- enhancement of the differences between images

Image Subtraction

a	b
c	d

- a). original fractal image
- b). result of setting the four lower-order bit planes to zero
  - refer to the bit-plane slicing
  - the higher planes contribute significant detail
  - the lower planes contribute more to fine detail
  - image b). is nearly identical visually to image a), with a very slightly drop in overall contrast due to less variability of the gray-level values in the image.
- c). difference between a). and b). (nearly black)
- d). histogram equalization of c). (perform contrast stretching transformation)

Note

- We may have to adjust the gray-scale of the subtracted image to be [0, 255] (if 8-bit is used)
  - first, find the minimum gray value of the subtracted image
  - second, find the maximum gray value of the subtracted image
  - set the minimum value to be zero and the maximum to be 255
  - while the rest are adjusted according to the interval [0, 255], by timing each value with 255/max
- Subtraction is also used in segmentation of moving pictures to track the changes
  - after subtract the sequenced images, what is left should be the moving elements in the image, plus noise

Image Averaging

- consider a noisy image g(x,y) formed by the addition of noise η(x,y) to an original image f(x,y)

$$g(x,y) = f(x,y) + \eta(x,y)$$

Image Averaging

- if noise has zero mean and be uncorrelated then it can be shown that if

$\bar{g}(x, y)$ = image formed by averaging K different noisy images

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^{K} g_i(x, y)$$

Image Averaging

- then

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)}$$

$\sigma^2_{\bar{g}(x,y)}, \sigma^2_{\eta(x,y)}$ = variances of $\bar{g}$ and η

if K increase, it indicates that the variability (noise) of the pixel at each location (x,y) decreases.

Image Averaging

- thus

$$E\{\bar{g}(x,y)\} = f(x,y)$$

$E\{\bar{g}(x,y)\}$ = expected value of $\bar{g}$ (output after averaging)

= original image f(x,y)

Image Averaging

- Note: the images $g_i(x,y)$ (noisy images) must be registered (aligned) in order to avoid the introduction of blurring and other artifacts in the output image.

Example

a	b
c	d
e	f

- a) original image
- b) image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels.
- c). -f). results of averaging K = 8, 16, 64 and 128 noisy images

Spatial Filtering

Background

- Filter term in “Digital image processing” is referred to the subimage
- There are others term to call subimage such as mask, kernel, template, or window
- The value in a filter subimage are referred as coefficients, rather than pixels.

Basics of Spatial Filtering

- The concept of filtering has its roots in the use of the Fourier transform for signal processing in the so-called frequency domain.
- Spatial filtering term is the filtering operations that are performed directly on the pixels of an image

Mechanics of spatial filtering

- The process consists simply of moving the filter mask from point to point in an image.
- At each point (x,y) the response of the filter at that point is calculated using a predefined relationship

Linear spatial filtering

Pixels of image

The result is the sum of products of the mask coefficients with the corresponding pixels directly under the mask

Mask coefficients

$$
\begin{aligned}
f(x,y) = \; & w(-1,-1)\, f(x-1, y-1) + w(-1,0)\, f(x-1, y) + w(-1,1)\, f(x-1, y+1) + \\
& w(0,-1)\, f(x, y-1) + w(0,0)\, f(x, y) + w(0,1)\, f(x, y+1) + \\
& w(1,-1)\, f(x+1, y-1) + w(1,0)\, f(x+1, y) + w(1,1)\, f(x+1, y+1)
\end{aligned}
$$

Note: Linear filtering

- The coefficient w(0,0) coincides with image value f(x,y), indicating that the mask is centered at (x,y) when the computation of sum of products takes place.
- For a mask of size mxn, we assume that m-2a+1 and n=2b+1, where a and b are nonnegative integer. Then m and n are odd.

Linear filtering

- In general, linear filtering of an image f of size MxN with a filter mask of size mxn is given by the expression:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Discussion

- The process of linear filtering similar to a frequency domain concept called "*convolution*"

<u>Simplify expression</u>

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_{mn} z_{mn} = \sum_{i=1}^{mn} w_i z_i$$

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Where the w's are mask coefficients, the z's are the value of the image gray levels corresponding to those coefficients

Nonlinear spatial filtering

- Nonlinear spatial filters also operate on neighborhoods, and the mechanics of sliding a mask past an image are the same as was just outlined.
- The filtering operation is based conditionally on the values of the pixels in the neighborhood under consideration

Smoothing Spatial Filters

- Smoothing filters are used for blurring and for noise reduction.
  - Blurring is used in preprocessing steps, such as removal of small details from an image prior to object extraction, and bridging of small gaps in lines or curves
  - Noise reduction can be accomplished by blurring

Type of smoothing filtering

- There are 2 way of smoothing spatial filters
  - Smoothing Linear Filters
  - Order-Statistics Filters

Smoothing Linear Filters

- Linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask.
- Sometimes called “averaging filters”.
- The idea is replacing the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask.

Two 3x3 Smoothing Linear Filters

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Standard average

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Weighted average

5x5 Smoothing Linear Filters

$$\frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Smoothing Linear Filters

- The general implementation for filtering an MxN image with a weighted averaging filter of size mxn is given by the expression

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

Result of Smoothing Linear Filters

Original Image

[3x3]

[5x5]

[7x7]

Order-Statistics Filters

- Order-statistics filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result.
- Best-known “median filter”

Process of Median filter

10, 15, 20, 20, 20, 20, 20, 25, 100

5th

- Corp region of neighborhood
- Sort the values of the pixel in our region
- In the MxN mask the median is MxN div 2 +1

Result of median filter

Noise from Glass effect

Remove noise by median filter

Sharpening Spatial Filters

- The principal objective of sharpening is to highlight fine detail in an image or to enhance detail that has been blurred, either in error or as an natural effect of a particular method of image acquisition.

Introduction

- The image blurring is accomplished in the spatial domain by pixel averaging in a neighborhood.
- Since averaging is analogous to integration.
- Sharpening could be accomplished by spatial differentiation.

Foundation

- We are interested in the behavior of these derivatives in areas of constant gray level(flat segments), at the onset and end of discontinuities(step and ramp discontinuities), and along gray-level ramps.
- These types of discontinuities can be noise points, lines, and edges.

Definition for a first derivative

- Must be zero in flat segments
- Must be nonzero at the onset of a gray-level step or ramp; and
- Must be nonzero along ramps.

Definition for a second derivative

- Must be zero in flat areas;
- Must be nonzero at the onset and end of a gray-level step or ramp;
- Must be zero along ramps of constant slope

Definition of the 1st-order derivative

- A basic definition of the first-order derivative of a one-dimensional function f(x) is

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Definition of the 2nd-order derivative

- We define a second-order derivative as the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

Gray-level profile

0	0	0	1	2	3	2	0	0	2	2	6	3	3	2	2	3	3	0	0	0	0	0	0	7	7	6	5	5	3

Derivative of image profile

Analyze

- The 1st-order derivative is nonzero along the entire ramp, while the 2nd-order derivative is nonzero only at the onset and end of the ramp.

- The response at and around the point is much stronger for the 2nd- than for the 1st-order derivative

1st make thick edge and 2nd make thin edge

The Laplacian (2nd order derivative)

- Shown by Rosenfeld and Kak[1982] that the simplest isotropic derivative operator is the Laplacian is defined as

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

Discrete form of derivative

f(x-1,y)	*f(x,y)*	*f(x+1,y)*

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

f(x,y-1)
f(x,y)
f(x,y+1)

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

2-Dimentional Laplacian

- The digital implementation of the 2-Dimensional Laplacian is obtained by summing 2 components

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x^2}$$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

	1	
1	-4	1
	1	

Laplacian

0	1	0
1	-4	1
0	1	0

1	0	1
0	-4	0
1	0	1

1	1	1
1	-8	1
1	1	1

Laplacian

0	-1	0
-1	4	-1
0	-1	0

-1	0	-1
0	4	0
-1	0	-1

-1	-1	-1
-1	8	-1
-1	-1	-1

Implementation

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \textit{If the center coefficient is negative} \\ f(x,y) + \nabla^2 f(x,y) & \textit{If the center coefficient is positive} \end{cases}$$

Where f(x,y) is the original image

$\nabla^2 f(x,y)$ is Laplacian filtered image

g(x,y) is the sharpen image

Implementation

-1	-1	-1
-1	8	-1
-1	-1	-1

Implementation

Filtered = Conv(image,mask)

Implementation

filtered = filtered - Min(filtered)
filtered = filtered * (255.0/Max(filtered))

Implementation

sharpened = image + filtered
sharpened = sharpened - Min(sharpened)
sharpened = sharpened * (255.0/Max(sharpened))

Algorithm

- Using Laplacian filter to original image

- And then add the image result from step 1 and the original image

Simplification

- We will apply two step to be one mask

$$g(x,y) = f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) + 4f(x,y)$$

$$g(x,y) = 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1

Unsharp masking

- A process to sharpen images consists of subtracting a blurred version of an image from the image itself. This process, called *unsharp masking*, is expressed as

$$f_s(x, y) = f(x, y) - \bar{f}(x, y)$$

Where $f_s(x, y)$ denotes the sharpened image obtained by unsharp masking, and $\bar{f}(x, y)$ is a blurred version of $f(x, y)$

High-boost filtering

- A high-boost filtered image, f_{hb} is defined at any point (x,y) as

$$f_{hb}(x,y) = Af(x,y) - \bar{f}(x,y) \quad where\ A \geq 1$$

$$f_{hb}(x,y) = (A-1)f(x,y) + f(x,y) - \bar{f}(x,y)$$

$$f_{hb}(x,y) = (A-1)f(x,y) + f_s(x,y)$$

This equation is applicable general and does not state explicity how the sharp image is obtained

High-boost filtering and Laplacian

- If we choose to use the Laplacian, then we know $f_s(x,y)$

$$f_{hb} = \begin{cases} Af(x,y) - \nabla^2 f(x,y) & \textit{If the center coefficient is negative} \\ Af(x,y) + \nabla^2 f(x,y) & \textit{If the center coefficient is positive} \end{cases}$$

0	-1	0
-1	A+4	-1
0	-1	0

-1	-1	-1
-1	A+8	-1
-1	-1	-1

The Gradient (1st order derivative)

- First Derivatives in image processing are implemented using the magnitude of the gradient.
- The gradient of function f(x,y) is

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Gradient

- The magnitude of this vector is given by

$$mag(\nabla f) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

G_x

-1	1

G_y

1
-1

This mask is simple, and no isotropic. Its result only horizontal and vertical.

Robert's Method

- The simplest approximations to a first-order derivative that satisfy the conditions stated in that section are

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$G_x = (z_9 - z_5)$ and $G_y = (z_8 - z_6)$

$$\nabla f = \sqrt{(z_9 - z_5)^2 + (z_8 - z_6)^2}$$

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

Robert's Method

- These mask are referred to as the Roberts cross-gradient operators.

-1	0
0	1

0	-1
1	0

Sobel’s Method

- Mask of even size are awkward to apply.
- The smallest filter mask should be 3x3.
- The difference between the third and first rows of the 3x3 mage region approximate derivative in x-direction, and the difference between the third and first column approximate derivative in y-direction.

Sobel's Method

- Using this equation

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Spatial Filtering Methods
(or Mask Processing Methods)

Area or Mask Processing Methods

$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

Spatial Filtering

- The word “filtering” has been borrowed from the frequency domain.
- Filters are classified as:
  - Low-pass (i.e., preserve low frequencies)
  - High-pass (i.e., preserve high frequencies)
  - Band-pass (i.e., preserve frequencies within a band)
  - Band-reject (i.e., reject frequencies within a band)

Spatial Filtering (cont'd)

- Spatial filtering is defined by:
  (1) A neighborhood
  (2) An operation that is performed on the pixels inside the neighborhood

$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

Spatial Filtering - Neighborhood

- Typically, the neighborhood is rectangular and its size is much smaller than that of *f(x,y)*
  - e.g., 3x3 or 5x5

Spatial filtering - Operation

Assume the origin of the mask is the center of the mask.

for a 3 x 3 mask:

$$g(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) f(x+s, y+t)$$

for a K x K mask:

$$g(x,y) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(x+s, y+t)$$

Spatial Filtering - Example

w1	w2	w3
w4	w5	w6
w7	w8	w9

$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$

Area or Mask Processing Methods

$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

- A filtered image is generated as the center of the mask moves to every pixel in the input image.

Handling Pixels Close to Boundaries

pad with zeroes

or

Linear vs Non-Linear Spatial Filtering Methods

- A filtering method is linear when the output is a weighted sum of the input pixels.

w1	w2	w3
w4	w5	w6
w7	w8	w9

```
z5' = R = w1z1 + w2z2 + ... + z9w9
```

- Methods that do not satisfy the above property are called non-linear.
  - e.g.,

$$z_5' = max(z_k, k = 1, 2, \ldots, 9)$$

Linear Spatial Filtering Methods

- Two main linear spatial filtering methods:
  - Correlation
  - Convolution

Correlation

$$g(x,y) = w(x,y) \bullet f(x,y) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(x+s, y+t)$$

Correlation (cont'd)

Often used in applications where we need to measure the similarity between images or parts of images
(e.g., pattern matching).

Convolution

- Similar to correlation except that the mask is first **<u>flipped</u>** both horizontally and vertically.

$$g(x,y) = w(x,y) * f(x,y) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(x-s, y-t)$$

<u>Note</u>: if w(x,y) is symmetric, that is w(x,y)=w(-x,-y), then convolution is equivalent to correlation!

Example

Correlation:

Convolution:

How do we choose the elements of a mask?

- Typically, by sampling certain functions.

w1	w2	w3
w4	w5	w6
w7	w8	w9

Filters

- Smoothing (i.e., low-pass filters)
  - Reduce noise and eliminate small details.
  - The elements of the mask must be **positive**.
  - Sum of mask elements is 1 (after normalization)

Filters (cont'd)

- Sharpening (i.e., high-pass filters)
  - Highlight fine detail or enhance detail that has been blurred.
  - The elements of the mask contain both **positive** and **negative** weights.
  - Sum of the mask weights is 0 (after normalization)

Smoothing Filters: Averaging (Low-pass filtering)

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(a)

$$\frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{49} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(c)

Smoothing Filters: Averaging (cont'd)

- Mask size determines the degree of smoothing and loss of detail.

original 3x3 5x5 7x7

15x15 25x25

Smoothing Filters: Averaging (cont'd)

Example: extract, largest, brightest objects

Smoothing filters: Gaussian

- The weights are samples of the Gaussian function

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

7×7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

σ = 1.4

mask size is
a function of **σ** :

$$height = width = 5\sigma \text{ (subtends 98.76\% of the area)}$$

Smoothing filters: Gaussian (cont'd)

- σ controls the amount of smoothing
- As σ increases, more samples must be obtained to represent the Gaussian function accurately.

$\sigma = 3$

15 × 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

Smoothing filters: Gaussian (cont'd)

Averaging vs Gaussian Smoothing

Averaging

Gaussian

Smoothing Filters: Median Filtering (non-linear)

- Very effective for removing “salt and pepper” noise (i.e., random occurrences of black and white pixels).

averaging

median filtering

Smoothing Filters: Median Filtering (cont'd)

- Replace each pixel by the median in a neighborhood around the pixel.

Sharpening Filters (High Pass filtering)

- Useful for emphasizing transitions in image intensity (e.g., edges).

Sharpening Filters (cont'd)

- Note that the response of high-pass filtering might be negative.
- Values must be re-mapped to [0, 255]

original image

sharpened images

Sharpening Filters: Unsharp Masking

- Obtain a sharp image by subtracting a lowpass filtered (i.e., smoothed) image from the original image:

$$Highpass = Original - Lowpass$$

Sharpening Filters: High Boost

- Image sharpening emphasizes edges but details (i.e., low frequency components) might be lost.
- **High boost filter**: amplify input image, then subtract a lowpass image.

$$Highboost = A\ Original - Lowpass$$

$$= (A-1)\ Original + Original - Lowpass$$

$$= (A-1)\ Original + Highpass$$

(A-1)

Sharpening Filters: Unsharp Masking (cont'd)

- If **A=1**, we get a high pass filter

- If **A>1**, part of the original image is added back to the high pass filtered image.

A>=1
w = 9A-1

-1	-1	-1
-1	w	-1
-1	-1	-1

A=2
w = 17

-1	-1	-1
-1	17	-1
-1	-1	-1

Sharpening Filters: Unsharp Masking (cont'd)

A=1.4

A=1.9

Sharpening Filters: Derivatives

- Taking the derivative of an image results in sharpening the image.
- The derivative of an image can be computed using the gradient.

$$grad(f) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Sharpening Filters: Derivatives (cont'd)

- The gradient is a **vector** which has magnitude and direction:

$$magnitude(grad(f)) = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$

or $$\left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

(approximation)

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

Sharpening Filters: Derivatives (cont'd)

- **Magnitude:** provides information about edge strength.

- **Direction:** perpendicular to the direction of the edge.

Sharpening Filters: Gradient Computation

- Approximate gradient using finite differences:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

sensitive to vertical edges

$$\frac{\partial f}{\partial x} = \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x} = f(x+1, y) - f(x,y), \quad (\Delta x=1)$$

sensitive to horizontal edges

$$\frac{\partial f}{\partial y} = \frac{f(x, y+\Delta y) - f(x,y)}{-\Delta y} = f(x,y) - f(x, y+1), \quad (\Delta y=1)$$

Sharpening Filters: Gradient Computation (cont'd)

$$\frac{f(x3,y2) - f(x3,y3)}{y2 - y3}$$

or $$\frac{f(x,y+Dy) - f(x,y)}{-Dy} = \boxed{f(x,y) - f(x,y+1)}$$ **(grad in the y-direction)**

(y3=y2+Dy, y2=y, x3=x, Dy=1)

edge in the y-direction

$$\frac{f(x2,y1) - f(x1,y1)}{x2 - x1}$$

or $$\frac{f(x+Dx,y) - f(x,y)}{Dx} =$$

$$\boxed{f(x+1,y) - f(x,y)}$$

(grad in the x-direction)

(x2=x+Dx, x1=x, y1=y2=y, Dx=1)

Example

[redacted]

[redacted]

Sharpening Filters: Gradient Computation (cont'd)

- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using masks:

-1	1

1
-1

$\frac{\partial f}{\partial x}$ good approximation at (x+1/2,y)

$\frac{\partial f}{\partial y}$ good approximation at (x,y+1/2)

- <u>Example:</u> approximate gradient at z_5

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\frac{\partial f}{\partial x} = z_6 - z_5$$

$$\frac{\partial f}{\partial y} = z_5 - z_8$$

$$mag(grad(f)) = \sqrt{(z_6 - z_5)^2 + (z_5 - z_8)^2}$$

Sharpening Filters: Gradient Computation (cont'd)

- A different approximation of the gradient:

$$\frac{\partial f}{\partial x}(x,y) = f(x,y) - f(x+1,y+1)$$

$$\frac{\partial f}{\partial y}(x,y) = f(x+1,y) - f(x,y+1),$$

good approximation

(x+1/2,y+1/2)

•We can implement and $\frac{\partial f}{\partial x}$ ing the $\frac{\partial f}{\partial y}$ ving masks:

1	0
0	−1

0	1
−1	0

(b) Roberts

Sharpening Filters: Gradient Computation (cont'd)

- <u>Example:</u> approximate gradient at z_5

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\frac{\partial f}{\partial x} = z_5 - z_9$$

$$\frac{\partial f}{\partial y} = z_6 - z_8$$

$$mag(grad(f)) = \sqrt{(z_5 - z_9)^2 + (z_6 - z_8)^2}$$

- Other approximations

–1	–1	–1
0	0	0
1	1	1

–1	0	1
–1	0	1
–1	0	1

(c) Prewitt

–1	–2	–1
0	0	0
1	2	1

–1	0	1
–2	0	2
–1	0	1

Sobel

Example

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

Sharpening Filters: Laplacian

The Laplacian (2nd derivative) is defined as:

$$\nabla^2 = \nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

(dot product)

Approximate derivatives:

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

$$\nabla^2 f = -4f(i, j) + f(i, j+1) + f(i, j-1) + f(i+1, j) + f(i-1, j)$$

Sharpening Filters: Laplacian (cont'd)

Laplacian Mask

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

detect zero-crossings

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

Sharpening Filters: Laplacian (cont'd)

Laplacian

Sobel

2D filtering

- Cross-correlation in which the filter is flipped horizontally and vertically is called convolution

$$g = h * f$$

$$g[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[-u,-v] \cdot f[i+u, j+v]$$

$$\boxed{= \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] \cdot f[i-u, j-v]}$$

Convolution vs. Cross-Correlation

- If the kernel is symmetric

$$h(u,v) = h(-u,-v)$$

convolution = cross-correlation

Noise

- **Types of noise:**
  - Salt and pepper noise
  - Impulse noise
  - Gaussian noise
- **Due to**
  - transmission errors
  - dead CCD pixels
  - specks on lens
  - can be specific to a sensor

Original

Salt and pepper noise

Impulse noise

Gaussian noise

Practical Noise Reduction

- How can we remove noise?
- Replace each pixel with the average of a $k_x k$ window around it

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	104	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Mean filtering

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[x, y]$

$g[x, y]$

Mean filtering

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$g[x, y]$

Neighborhood Processing (filtering)

Effect of mean filters

Side effect - blur

Mean kernel

- What's the kernel for a 3x3 mean filter?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[x, y]$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$h[u, v]$

Mean kernel

- What's the kernel for a 3x3 mean filter?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[x, y]$

1/9

1	1	1
1	1	1
1	1	1

$h[u, v]$

Equal weight to all pixels within the neighborhood

Gaussian Filtering

- A **Gaussian kernel** gives less weight to pixels further from the center

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[x, y]$

$$\frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$h[u, v]$

is a discrete approximation of a Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

Mean vs. Gaussian filtering

Input image

Mean Filtering

Mean filter, 20x20

Gaussian filtering

Gaussian filter, 20x20, $\sigma = 5$

Mean vs. Gaussian filtering

Neighborhood Processing (filtering)

Gaussian Filtering

The effects of a small and a large Gaussian blur

- Low-pass filter
- Smooth color variation (low frequency) is preserved
- Sharp edges (high frequency) are removed

Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.

Image credit: Wikipedia – page on Median Filter

Median filters

- Is a median filter a kind of convolution?
  No, median filter is an example of non-linear filtering

$$Median(f_1(x)+f_2(x)) \neq Median(f_1(x))+Median(f_2(x))$$

$$1 = Median\left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix}\right) = Median\left(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right) \neq$$

$$Median\left(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}\right) + Median\left(\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right) = 1+1 = 2$$

Salt&Pepper Noise

Gaussian noise

Median filters

- What advantage does a **median filter** have over a **mean filter**?
  Better at removing Salt & Pepper noise
- Disadvantage:
  Slow

Neighborhood Processing (filtering)

Derivatives and Convolution

- Image Derivatives and Gradients
  - Used for Edge/Corner Detection
  - Computed with Finite Differences Filters
- Laplacian of Gaussians (LoG) Filter
  - Used for Edge/Blob Detection and Image Enhancement
  - Approximated using Difference of Gaussians

Reading: Forsyth & Ponce, 8.1-8.2

First Derivative

- Recall Sharp changes in gray level of the input image correspond to “peaks or valleys” of the first-derivative of the input signal.

(1D example)

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6pp.

Reading: Forsyth & Ponce, 8.1-8.2

First Derivative and convolution

$$\frac{\partial}{\partial x} f = \lim_{h \to 0} \left(\frac{f(x+h, y) - f(x, y)}{h} \right)$$

- How can we approximate it for a discrete function?

- Is this operation shift-invariant?

- Is it linear?

?	?	?
?	?	?
?	?	?

$$\nabla_x [u, v]$$

Reading: Forsyth & Ponce, 8.1-8.2

First Derivative and convolution

- Finite Difference

$$f(x+a) - f(x+b)$$

Forward, backward, and central differences

Only three forms are commonly considered: forward, backward, and central differences.

A **forward difference** is an expression of the form

$$\Delta_h[f](x) = f(x+h) - f(x).$$

$$\frac{\Delta_h[f]}{h} = \frac{f(x+h) - f(x)}{h} \xrightarrow[h \to 0]{} f'(x)$$

Slide credit: Wikipedia

Reading: Forsyth & Ponce, 8.1-8.2

First Derivative and convolution

- Finite Difference – The order of an error can be derived using Taylor Theorem

$$\frac{\Delta_h[f](x)}{h} - f'(x) = O(h) \quad (h \to 0).$$

The same formula holds for the backward difference:

$$\frac{\nabla_h[f](x)}{h} - f'(x) = O(h).$$

However, the central difference yields a more accurate approximation.

$$\frac{\delta_h[f](x)}{h} - f'(x) = O(h^2).$$

Slide credit: Wikipedia

Reading: Forsyth & Ponce, 8.1-8.2

First Derivative and convolution

- Pixel Size $\Delta x = h$
- Using Finite Central Difference

$$\frac{\partial}{\partial x} f \approx \frac{f(x+\Delta x, y) - f(x-\Delta x, y)}{2 \cdot \Delta x}$$

- We need a kernel ∇_x such that

$$\frac{\partial}{\partial x} f = \nabla_x * f$$

$$\frac{1}{2\Delta x} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\nabla_x[u,v]$$

Neighborhood Processing (filtering)

Finite differences – responds to edges

Dark = negative

White = positive

Gray = 0

$$\nabla_x * f$$

Finite differences - responding to noise

Increasing noise ->
(this is zero mean additive gaussian noise)

Finite differences and noise

- Finite difference filters respond strongly to noise
  - Noisy pixels look very different from their neighbours
  - The larger the noise ➔ the stronger is the response

- How can we eliminate the response to noise?
  - Most pixels in images look similar to their neighbours (even at an edge)
  - Smooth the image (mean/gaussian filtering)

Smoothing and Differentiation

- Smoothing before differentiation = two convolutions

$$\nabla_x * (H * f)$$

- Convolution is associative

$$\nabla_x * (H * f) = (\nabla_x * H) * f$$

$(\nabla_x * H)$

$(\nabla_y * H)$

Smoothing and Differentiation

$$(\nabla_x * H) * f$$

1 pixel 3 pixels 7 pixels

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Sobel kernels

- Yet another approximation frequently used

$$\frac{\partial}{\partial x} f \qquad\qquad \frac{\partial}{\partial y} f$$

$$\frac{1}{8\Delta x} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad\qquad \frac{1}{8\Delta y} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\nabla_x[u,v] \qquad\qquad \nabla_y[u,v]$$

Image Gradients

- Recall for a function of two variables $f(x,y)$
- The **gradient** at a point (x,y)

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \approx \begin{bmatrix} \nabla_x * f \\ \nabla_y * f \end{bmatrix}$$

- **Gradient Magnitude**

$$\| \nabla f \| = \sqrt{(\tfrac{\partial f}{\partial x})^2 + (\tfrac{\partial f}{\partial y})^2} \approx$$

- **Gradient Orientation**
  - direction of the "steepest ascend"
  - orthogonal to object boundaries in the image

Image Gradient for Edge Detection

- Typical application where image gradients are used is *image edge* detection
  - find points with large image gradients

Canny edge detector suppresses non-extrema Gradient points

Reading: Forsyth & Ponce, 8.1-8.2

Second derivatives and convolution

- Peaks or valleys of the first-derivative of the input signal, correspond to “zero-crossings” of the second-derivative of the input signal.

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6pp.

Neighborhood Processing (filtering)

Second derivatives and convolution

Taylor Series expansion

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + O(h^4)$$

add

$$+\left[f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + O(h^4) \right]$$

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4)$$

$$\frac{f(x-h) - 2f(x) + f(x+h)}{h^2} = f''(x) + O(h^2)$$

1	-2	1

Central difference approx to second derivative

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6pp.

Neighborhood Processing (filtering)

Second derivatives and convolution

- Better localized edges
- But more sensitive to noise

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Neighborhood Processing (filtering)

Second derivatives and convolution

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Second Image Derivatives

- Laplace operator

$$\Delta f = \nabla \cdot \nabla f = \nabla^2 f$$

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

rotationally invariant
second derivative for 2D functions

0	0	0
-1	2	-1
0	0	0

$+$

0	-1	0
0	2	0
0	-1	0

$=$

0	-1	0
-1	4	-1
0	-1	0

Finite Difference Second Order Derivative in x

Finite Difference Second Order Derivative in y

Second Image Derivatives

- Laplacian Zero Crossing

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

- Used for edge detection
  (alternative to computing Gradient extrema)

http://homepages.inf.ed.ac.uk/rbf/HIPR2/zeros.htm

Neighborhood Processing (filtering)

Laplacian Filtering

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I$$

-Zero on uniform regions
-Positive on one side of an edge
-Negative on the other side
-Zero at some point in between
on the edge itself
➔ band-pass filter (Suppresses both high and low frequencies)

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Laplacian of a Gaussian (LoG)

- Smooth before differentiation
  (remember associative property of convolution)

$$\Delta * G = LoG$$

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

$$g'(x) = -\frac{1}{2\sigma^2} 2x e^{-\frac{x^2}{2\sigma^2}}$$

$$LoG(x, y) = \frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$g''(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}}$$

LoG "Mexican Hat"

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Neighborhood Processing (filtering)

Laplacian of a Gaussian (LoG)

Suppresses both high and low frequencies

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Neighborhood Processing (filtering)

Laplacian of a Gaussian (LoG)

Raw zero-crossings (no contrast thresholding)

LoG sigma = 2, zero-crossing

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Laplacian of a Gaussian (LoG)

Raw zero-crossings (no contrast thresholding)

LoG sigma = 4, zero-crossing

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Laplacian of a Gaussian (LoG)

- Can be approximated by a difference of two Gaussians (DoG)

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

1D example

Best approximation when:
$\sigma_1 = \frac{\sigma}{\sqrt{2}}$, $\sigma_2 = \sqrt{2}\sigma$

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

DoG vs. LoG

- Separable (product decomposition) ➔ more efficient
- Can explain band-pass filter since Gaussian is a low-pass filter.

Neighborhood Processing (filtering)

LoG for Blob Detection

- Cross correlation with a filter can be viewed as comparing a little "picture" of what you want to find against all local regions in the image.

- Scale of blob (size ; radius in pixels) is determined by the sigma parameter of the LoG filter.

Maximum response:
dark blob on light background
Minimum response:
light blob on dark background

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Neighborhood Processing (filtering)

LoG for Blob Detection

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6p

Histogram Equalization

r_k	n_k	$p_r(r_k) = n_k/MN$	$7 * \sum_u p(u)$	v
$r_0 = 0$	790	0.19	1.33	1
$r_1 = 1$	1023	0.25	3.08	3
$r_2 = 2$	850	0.21	4.55	5
$r_3 = 3$	656	0.16	5.67	6
$r_4 = 4$	329	0.08	6.23	6
$r_5 = 5$	245	0.06	6.65	7
$r_6 = 6$	122	0.03	6.86	7
$r_7 = 7$	81	0.02	7.00	7