ELIC 629

Digital Image Processing

Fall 2005

Image Enhancement in the Spatial Domain:

Histograms, Arithmetic/Logic Operators, Basics of Spatial Filtering, Smoothing Spatial Filters

Bill Kapralos

Monday, October 17 2005

Overview (1):

- **Before We Begin**
  - Administrative details
  - Review → some questions to consider
- **Histogram Processing**
  - Introduction
  - Examples
- **Arithmetic/Logic Operator Enhancement**
  - Image subtraction
  - Image averaging

Overview (2):

- **Spatial Filtering**
  - Introduction
  - Basics of 1D and 2D spatial filtering
- **Smoothing Spatial Filters**
  - Introduction
  - Smoothing linear filters

Before We Begin

Administrative Details (1):

- **Lab Two**
  - Lab report and assignment due today!
- **Lab Four**
  - Should be a rather straightforward lab to complete
  - No lab report required for this lab but there is a lab assignment due October 31 (two weeks from now)

Some Questions to Consider (1):

- What is a linear operator and a non-linear operator ?
- How do we show an operator is linear or non-linear ?
- What is the spatial domain ?
- What is image enhancement in the spatial domain ?
- Describe the identity and the negative operator
- List/describe some other operators
- What is a histogram ?
- Can you think of how to enhance an image by using its histogram ?

Histogram Processing

Histograms - Introduction (1):

- **What is a Histogram (H_f) of an Image ?**
  - A plot or graph of the frequency of occurrence of each gray-level in the image
    - 1D function with domain [0, … L-1] and range from 0 to total number of pixels in image
    - Basically, for each gray level n_i of an image, the histogram gives you a count of how many pixels have this particular gray level n_i
    - Mathematically:

$$H_f(k) = J$$

where $H_f \rightarrow$ histogram, $k \rightarrow$ gray level, $J \rightarrow$ number of occurrences of gray level "k" in image

Histograms - Introduction (2):

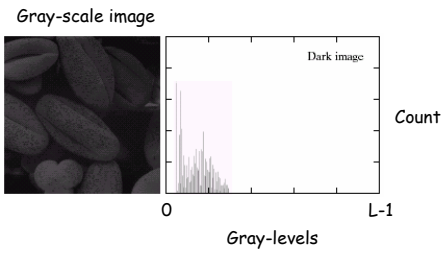
- **Histogram Normalization**
  - Common to normalize a histogram such that its return value is between 0 to 1.
    - Accomplished by dividing each of the histogram's values by the total number of pixels in the image
    - Basically, the normalized output gives the probability of the occurrence of the particular gray-level "k"
    - Sum of all histogram values is equal to 1
    - Think of it as providing a distribution of gray-levels in the image

Histograms - Introduction (3):

- **Histogram Properties**
  - Image histograms are a fundamental construct in image processing and widely used for image enhancement!
  - Simple to use and can accomplish good results, quickly, including for real-time applications
  - Important: histogram results in a reduction of dimensionality → cannot deduce image f from the values of the histogram – it does not provide us spatial information of the gray-levels only how many occur in the image!

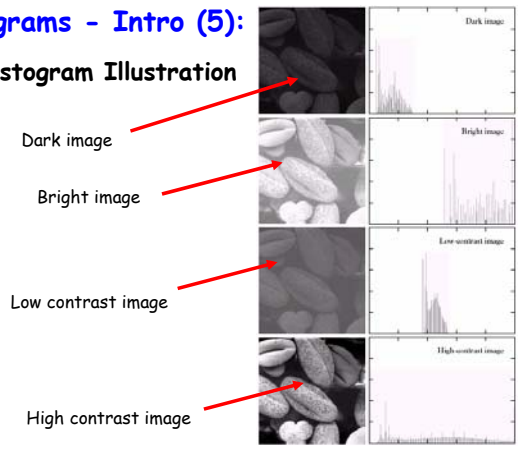
Histograms - Introduction (4):

- **Histogram Illustration**



Histograms - Intro (5):

- **Histogram Illustration**



Arithmetic and Logic Operators

Introduction (1):

- **Basics**
  - Arithmetic and logic operations are performed on a pixel-by-pixel basis between two or more images
  - Depending on hardware/software, arithmetic/logical operations can be performed sequentially (one at a time) or in parallel

Introduction (2):

- **Logical Operators**
  - Logic operations are only concerned with the three functionally complete operators since everything else can be implemented with them
    - AND, OR and NOT
  - Logic operations → pixel values are processed as strings of binary numbers
    - Logic operations performed on a bit-by-bit basis

Introduction (3):

- **Logic Operators**
  - NOT
    - Consider 8-bit gray-level p = 10001110
    - After performing NOT operations p = 01110001
    - When applied to entire image, produces negative transformation
  - AND and OR operators are used for masking
    - Selecting portions (sub-images) of an image
    - Typically used to isolate area of image for further processing
    - Light represents binary 1 and dark binary 0

Introduction (4):

- **Arithmetic Operators**
  - Of the four arithmetic operators (addition, subtraction, multiplication and division) addition and subtraction are the most useful
    - Division of two image is simply multiplication by the inverse of one image
  - Example: subtraction or addition of two images f and g to obtain new image h (pixel-by-pixel process)

$$\text{Addition: } h(x,y) = f(x,y) + g(x,y)$$

$$\text{Subtraction: } h(x,y) = f(x,y) - g(x,y)$$

Image Subtraction (1):

- **Mathematically**

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

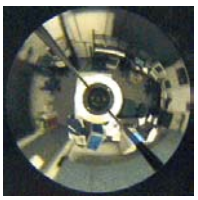
  - Results in a difference between the two images f and h $\rightarrow$ g is know as a difference image
  - Very useful for a variety of applications including surveillance!
    - Consider image of room (image h) without any person present. Then consider person in room and take another image (image f). Assuming everything else remains the same, the difference image will give us the difference between the two images e.g., the person!

Image Subtraction (2):

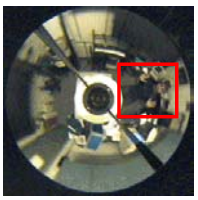
- **Finding Foreground**
  - Allows you to extract the foreground objects from the background provided you have an image of the background
  - Also know as change detection and useful for detecting moving objects
  - Many medical applications as well including mask mode radiography

Image Subtraction (3):

- **Graphical Illustration**
  - Real world application!



Reference image of empty room



Person present in room

Difference image

Image Subtraction (4):

- **Some Implementation Details**
  - Recall we are dealing typically with 8-bit images so that intensity (gray-levels) range from 0 – 255
    - Image subtraction can however lead to values outside of this range (e.g., negative values!)
    - Several solutions to this problem
      1. Add 255 to every pixel and divide by 2 → simple, fast but may lead to loss of accuracy due to division by 2
      2. Take the absolute value (e.g., ignore the minus sign)

Image Averaging (1):

- **Image Averaging**
  - Big application → removal of noise from an image
  - Consider image f(x,y) → due to various factors including sampling, optics of sensing device etc., f(x,y) will contain noise denoted by $\eta(x,y)$
  - We now model an image f(x,y) with noise as

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

  - We assume that noise for every pixel at location (x,y) is un-correlated (e.g., completely random noise)

Image Averaging (2):

- **Goal of Image Averaging**
  - Reduce the noise content by adding (averaging) a set of noisy images $\{g_i(x,y)\}$
  - This approach is actually very common in a many signal processing applications and not restricted to images! e.g., your signal processing course
  - Mathematically, average image $g_{avg}(x,y)$ is formulated as follows:

$$g_{avg} = \frac{1}{K} \times \sum_{1}^{K} g_i(x,y)$$

Image Averaging (3):

- **Goal of Image Averaging (cont…)**
  - As K increases we have the following:

$$E\{g_{avg}(x,y) = f(x,y)\}$$

$$\sigma^2 g_{avg}(x,y) = 1/K \times \sigma^2_{\eta}(x,y)$$

  - In words, as we increase the number of images being averaged, the expected value of the output image approaches the ideal input image (e.g., all noise removed and the standard deviation (and hence variance) is decreased

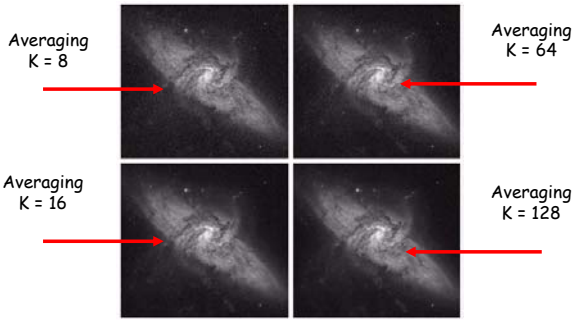
Image Averaging (4):

- **Applications of Image Averaging**
  - Astronomy → astronomical images typically very noisy due to very low light levels
    - Many times, single images are useless for analysis e.g., cannot determine any information!

Example of a noisy astronomical image

Image Averaging (5):

- **Applications of Image Averaging (cont…)**



Introduction to Spatial Domain Filtering

Introduction (1):

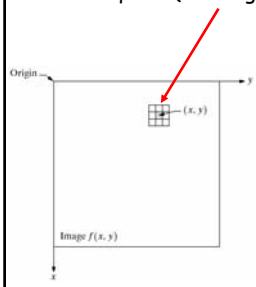
- **Spatial Domain**
  - The aggregate of pixels comprising an image
- **Spatial Domain Methods**
  - Procedures that operate directly on the image pixels
  - Denoted by the following expression

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

  - $f(x,y)$ → input image
  - $g(x,y)$ → output image
  - T → operator defined over some neighborhood of (x,y)

Introduction (2):

- **Defining a Neighborhood About x,y**
  - Square (rectangular) sub-image area centered at x,y



- Center of sub-image is moved from pixel to pixel
- Operator T is applied at each location x,y to yield output g
- Operator T utilizes only pixels in area of image f spanned by neighborhood

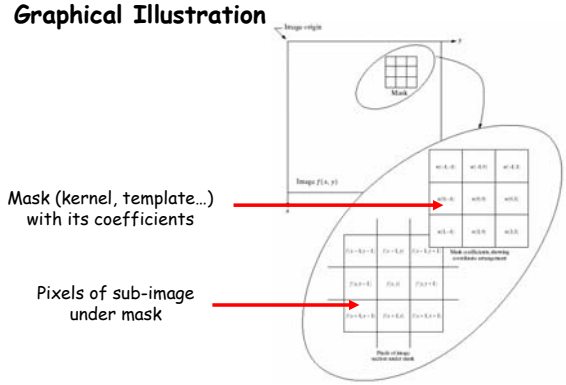
Introduction (3):

- **Defining a Neighborhood About x,y (cont.)**
  - Square (rectangular) sub-image area centered at x,y
    - Also known as a mask, template, filter or window
    - Each entry of the sub-image has its own value → each value known as a coefficient
  - Mask does not always have to be two-dimensional
    - Can also have one-dimensional mask (more later…)

Basics of Spatial Filtering

Introduction – 2D Masks (1):

- **Graphical Illustration**



Introduction – 2D Masks (2):

- **Template, Kernel, Mask**
  - Coefficients denoted by w
  - Origin is in the "middle"
  - Arbitrary size → dimensions do not need to be odd implying no "true" center (origin)

w(-1,-1)	w(-1,0)	w(-1,1)
w(0,-1)	w(0,0)	w(0,1)
w(1,-1)	w(-1,0)	w(1,1)

Example of a 3x3 template with its coefficients

Introduction – 2D Masks (3):

- **"Mechanics" of Spatial Filtering**
  - Moving the template over each pixel of the image
    - At each pixel (x,y) the response (e.g., output value) is determined using some pre-defined relationship
    - For linear spatial filtering, response is given by a sum of the products of the filter coefficients (denoted by w) and the corresponding image pixels "under" the area of the template
    - Mathematically, response g at (x,y) given as

$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots + w(0,0)f(x,y) + \ldots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1)$

Introduction 2D – Masks (4):

- **"Mechanics" of Spatial Filtering (cont…)**
  - General filtering expression

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

  - m → row dimension
  - n → column dimension
  - m = 2a+1 and n = 2b+1 and a,b are non-negative integers or a = (m-1)/2 and b = (n-1)/2
  - But this gives output for one pixel location (x,y) only!

Introduction - 2D Masks (5):

- **"Mechanics" of Spatial Filtering (cont…)**
  - To generate complete output image, the above equation (process) must be applied to each pixel of input image e.g., for each x = 0 – M-1 and y = 0 – N-1 where M,N are the number of rows and columns of the input image
  - Similar to a frequency domain concept called convolution (more on this in the future…)
    - Hence sometimes referred to as "*convolving a mask with an image*" and the mask is often called a convolution mask

Introduction – 2D Masks (6):

- **"Mechanics" of Spatial Filtering (cont…)**
  - When we are not interested in processing entire image but rather only a particular pixel (x,y) we can use the following (shorter) mathematical definition:

$R = w_1 z_1 + w_2 z_2 + \ldots + w_{mn} z_{mn} +$

$= \sum_{i=1}^{mn} w_i z_i$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

→ where, the w's are the filter coefficients and z's are the image gray-levels corresponding to the coefficients and mn is total number of coefficients in template

Introduction – 2D Masks (7):

- **Important Considerations**
  - What happens when kernel "placed over" a border pixel ?
  - Several approaches for dealing with this
    1. Ignore (don't handle) border pixels altogether or any pixels which lead to kernel (or portions of the kernel) "falling" out of image range
    2. "Wrap-around"
    3. "Zero-pad"
  - Keep in mind, some approaches may lead to an output image whose size is not equal to input image!

Introduction – 1D Masks (1):

- **Similar to 2D Case Except Now 1D**

w_1	w_2	w_3

Mask

- Size of mask can be odd or even
  - If even → how do you define "center" ?
  - Can convert to odd by prepending a "zero" entry
  - Now origin is middle entry

Smoothing Spatial Filters

Introduction (1):

- **Purpose of Smoothing Spatial Filters**
  - Used for blurring, particularly in pre-processing
    - Removal of small detail prior to extraction of large object(s) in image
    - Bridging ("closing") of small gaps in lines or curves
  - Also used for noise reduction
    - Can be achieved with a linear or non-linear filter

Smoothing Linear Filters (1):

- **Essentially an Averaging Filter**
  - Output of smoothing filter is simply the average of pixels in the neighborhood of filter mask (kernel)
  - Also known as a low pass filter
    - Eliminates high frequency components (we will describe this further in later lectures)
  - Idea of smoothing filter
    - Random noise typically consists of sharp transitions in gray levels

Smoothing Linear Filters (2):

- Idea of smoothing filter (cont…)
  - By replacing the value of every pixel by the average of its neighbors, we essentially reduce the sharp transitions in gray levels
- Most obvious application of a smoothing filter is noise reduction
- However, beware!
  - Not all sharp transitions are bad and un-wanted!
  - Edges which are typically very important and wanted features of an image are also defined as sharp transitions in gray levels

Smoothing Linear Filters (3):

- However, beware! (cont…)
  - Since smoothing filter removes sharp transitions in gray level, averaging (smoothing) filters blur edges!
- Several other applications of smoothing (averaging) filters in addition to noise reduction
  - Smoothing of false contours which result when not using a large number of gray levels
  - Removing irrelevant details in an image $\rightarrow$ pixel regions that are small in comparison to the size of the filter kernel

Smoothing Linear Filters (4):

- **Smoothing Filter Kernels**
  - Example of a 3 x 3 smoothing filter

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

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

  - Above kernel (filter) produces average of the pixels under the mask
  - Notice that coefficients are equal to 1 and not 1/9!
    - One division instead of nine → more efficient!

Smoothing Linear Filters (5):

- **Smoothing Filter Kernels (cont…)**
  - Another example of a 3 x 3 smoothing filter

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

$$R = \frac{1}{16}\sum_{i=1}^{9} z_i$$

  - This is an example of a weighted average filter
    - Coefficients are not all the same value!
    - Some pixels multiplied by higher values thus giving those pixels more importance in average

Smoothing Linear Filters (6):

- Another example of a 3 x 3 smoothing filter (cont…)

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

  - Center coefficient is highest meaning center pixel is given most importance
  - Other coefficients are reduced inversely as a function of distance from the center coefficient
    - Diagonal terms are further away than the "edge" neighbors and thus the corresponding pixels in image provide less importance to average

Smoothing Linear Filters (7):

- Another example of a 3 x 3 smoothing filter (cont…)
  - Many other types of coefficient masks are also available depending on application but typically try to keep the sum of the coefficients an integral power of 2 (e.g., 16 as in previous example)
- In general, hard to notice differences between images filtered by both these filter examples

Smoothing Linear Filters (8):

- **General Filter Implementation**
  - Recall the filtering expression for filtering M x N image with weighted averaging filter of size m x n

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

  - The general expression equation can now be stated as (again, given an N x M image with m x n filter where m,n are odd)

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

Smoothing Linear Filters (9):

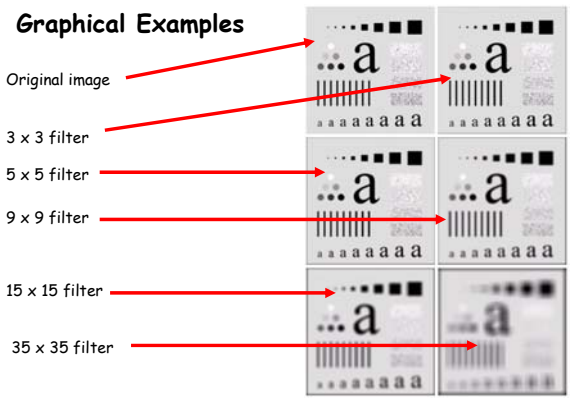
- **General Filter Implementation (cont…)**

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

  - Complete filtered image is obtained by applying above equation for each x = 0,1,2, …, M-1 and y = 0,1,2, …, N-1
  - Denominator is sum of the mask (kernel) coefficients and is constant (e.g., computed once!)
    - Typically division applied once to output image rather than at each stage (pixel output)

Smoothing Linear Filters (10):

- **Graphical Examples**



Smoothing Linear Filters (11):

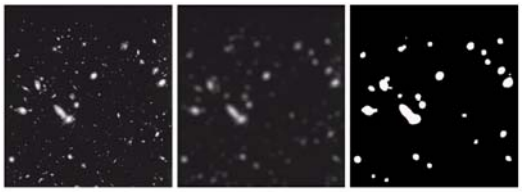
- **Some Notes Regarding Averaging Filters**
  - For small filter (e.g., 3 x 3) a slight general blurring of entire image occurs but details that are about same size of filter are affected considerably
  - Noise is less pronounced
  - Jagged borders are "pleasantly" smoothed
  - As filter size increases, blurring is more pronounced

Smoothing Linear Filters (12):

- **Image Blurring**
  - Important application of averaging filter is blurring to get "gross representation" of objects of interest
    - Smaller objects blend into the background
    - Larger objects become more "blob-like" and easy to detect
    - Size of mask (kernel) determines size of objects to be blended into background → smaller mask, smaller objects blended to background

Smoothing Linear Filters (13):

- **Image Blurring Graphical Example**
  - Image obtained with Hubble space telescope



Original image | Filtered with 15 x 15 averaging mask | Threshold image – small objects have disappeared