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

Tuesday, February 7 2006

ELIC 629, Winter 2006, Bill Kapralos

Overview (1):

- **Before We Begin**
  - Administrative details
  - Review → some questions to consider
- **Basic Gray-Level Transformations**
  - Introduction
  - Image negatives
  - Log transformations
  - Power law transformations
  - Piece-wise linear transformations

Overview (2):

- **Histogram Processing**
  - Introduction
  - Examples
- **Arithmetic/Logic Operator Enhancement**
  - Image subtraction
  - Image averaging
- **Spatial Filtering**
  - Introduction

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 ?

Basic Gray-Level Transformations

(Cont. From Last Lecture)

Introduction (1):

- ***Simplest Image Enhancement Techniques***
  - Recall → values of pixels before processing known as "r", after processing "s" and related by s = T(r)
  - Three common types of gray-level transformations
    1. Linear (negative and identity transformations)
    2. Logarithmic (log and inverse-log transformations)
    3. Power-law (n^{th} power & n^{th} root transformations)

Introduction (2):

- **Linear, Logarithmic & power Law Operators**

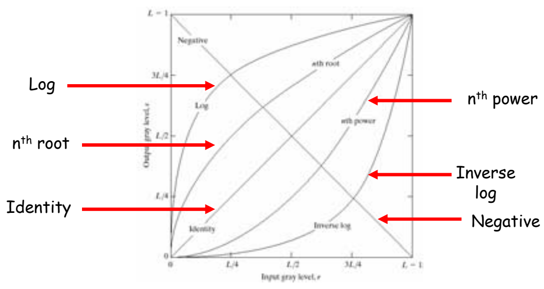


Image Negatives (1):

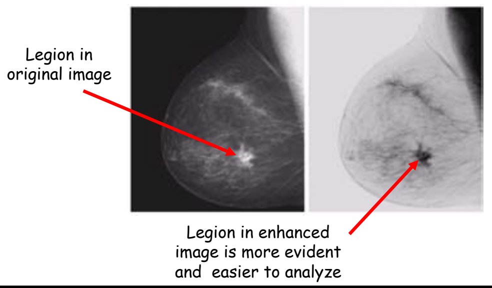
- **Equivalent to a Photographic Negative**

$$s = L - 1 - r$$

  - L → maximum intensity value (e.g., 255)
  - Reversing the intensity levels of an image
  - Primary use → enhancing white or gray detail embedded in dark regions of image especially when these dark regions are largest
    - Medical imaging → can detect problems (lesions etc.)

Image Negatives (2):

- **Mammogram Example**
  - Small legion in original easier to analyze in negative!


Legion in original image
Legion in enhanced image is more evident and easier to analyze

Log Transformations (1):

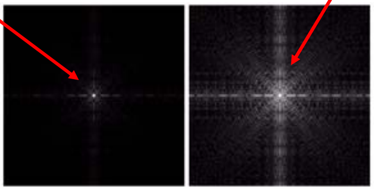
- **Expand Values of Dark Pixels in Image, Compress Higher Level Values**
  - Map narrow range of low gray-levels into wider range of output values
  - Map higher range of high gray-levels into narrow region of output values
  - Particularly useful for compressing dynamic range of images with large variations in pixel values
    - Fourier transform → high dynamic range ($0 - 10^6$) can't be represented on display device → devices cannot faithfully reproduce such a wide range

Log Transformations (2):

- **Graphical Illustration**

Fourier spectrum with intensity values in range 0 – 1.5 x 10^6 – no detail evident!

After log transformation – detail now evident



- Most Fourier spectra we look at will be log transformed

Power Law Transformations (1):

- **Mathematically**

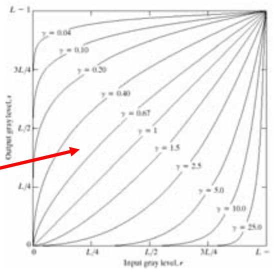
(1) $s = cr^{\gamma}$ or

(2) $s = c(r + \varepsilon)^{\gamma}$

- $c > 0, \gamma > 0$ ($\gamma \rightarrow$ "gamma")
- (2) is used as an "offset" when input is zero (e.g., when input is zero, output will be not zero!)
- As with log transform, fractional values of γ (e.g., $\gamma < 1$) map narrow range of dark input values to wider range of output values and higher input levels to narrow range of output values

Power Law Transformations (2):

- **Mathematically (cont…)**
  - When $\gamma > 1 \rightarrow$ opposite effect!



Power law transform for various values of γ

Power Law Transformations (3):

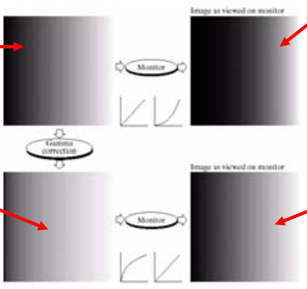
- **Power Law is Everywhere!**
  - Although you may not be aware of it, a large number of devices including digital cameras, printers, monitors etc. respond to their input via a power law
  - Automatically "adjust" their input so that it is scaled following a power law
  - We typically have to account for this scaling and apply inverse power law to output of device
    - This is known as gamma correction

Power Law Transformations (4):

- **Gamma Correction**
  - The process used to account (correct) for the power law scaling
  - Example $\rightarrow$ cathode ray tubes (CRTs) intensity-to-voltage response is a power function with gamma values ranging from 1.8 to 2.5
    - Produces images that are darker than we would expect!
    - Can easily account for this by applying an inverse power law process $\rightarrow s = r^{1/\gamma}$ where γ is original gamma value

Power Law Transformations (5):

- **Gamma Correction (cont…)**
  - Graphical illustration



Gray-scale image viewed on monitor

Original gray-scale image

Gamma corrected image viewed on monitor

Gamma correction via inverse power law transformation

Power Law Transformations (6):

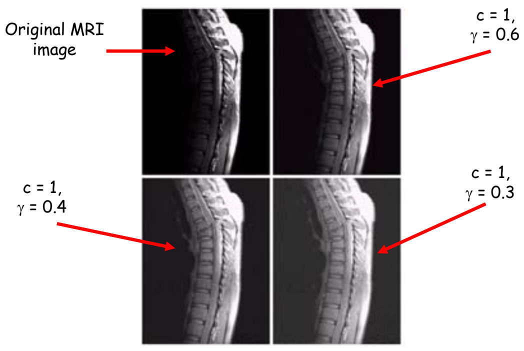
- **Gamma Correction (cont...)**
  - Important if you need to display image accurately on a computer screen e.g., medical imaging or color based applications
  - If not applied properly then
    - Image can appear too dark or "washed out"
  - Particularly relevant over the last several years given images on the internet with the many types of displays used to view them
    - Typically use an average gamma to process them

Power Law Transformations (7):

- **Gamma Correction (cont...)**
  - In addition to gamma correction, also used for modifying contrast in general
    - Useful in medical imaging $\rightarrow$ allows for details to be seen which might not have otherwise been visible

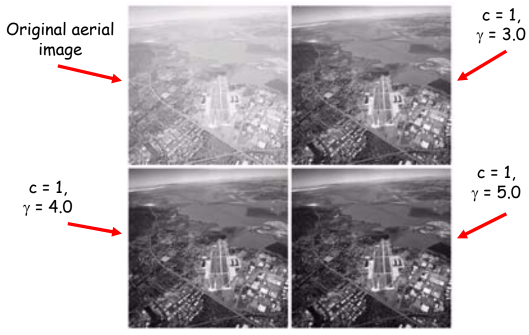
Power Law Transformations (8):

- **Gamma Correction (gamma < 1)**



Power Law Transformations (9):

- **Gamma Correction (gamma > 1)**



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 k_i of an image, the histogram gives you a count of how many pixels have this particular gray level k_i
    - Mathematically:

$$H_f(k_i) = J$$

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

Histograms - Introduction (2):

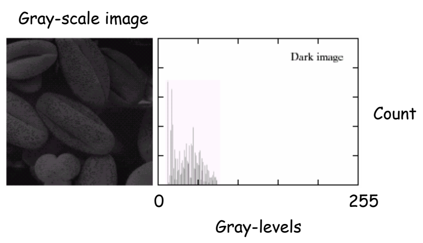
- **Histogram Normalization**
  - Common to normalize a histogram such that its return value is between 0 to 1.
    - Accomplished by dividing each of the histogram's values by total number of pixels in image
    - Basically, the normalized output gives the probability of the occurrence of the particular 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!
  - 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
    - Provides only a count of the occurrence of a particular gray level in the image

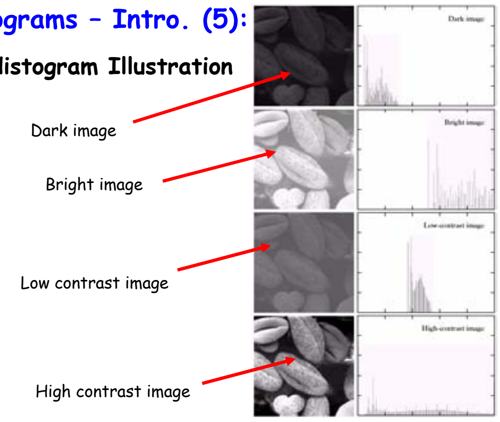
Histograms - Introduction (4):

- **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 $\rightarrow$ 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)

Addition: $h(x,y) = f(x,y) + g(x,y)$

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

Image Subtraction (1):

- **Mathematically**

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

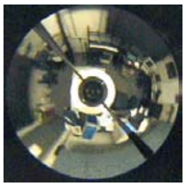
  - Results in a difference between the two images f and h $\rightarrow$ g is know as a difference image
  - Very useful for a variety of applications including surveillance!
    - Consider image of room (image h) without any person present. Then consider person in room and take another image (image f).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

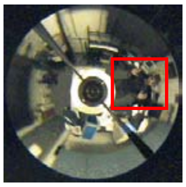
Image Subtraction (2):

- **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

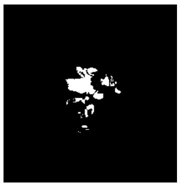

Difference image

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

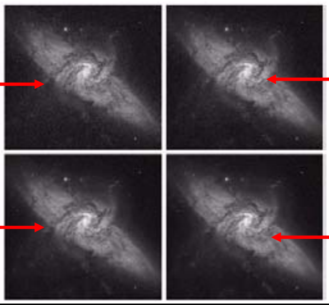
Image Averaging (4):

- **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):

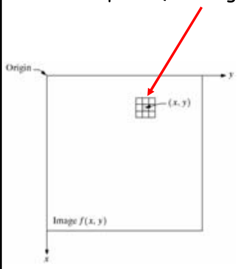
- **Spatial Domain**
  - The aggregate of pixels comprising an image
- **Spatial Domain Methods**
  - Procedures that operate directly on the image pixels
  - Denoted by the following expression

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

  - $f(x,y) \rightarrow$ input image
  - $g(x,y) \rightarrow$ output image
  - $T \rightarrow$ 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

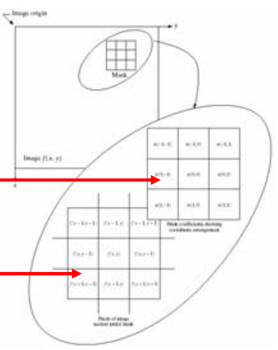
Introduction (3):

- **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 $\rightarrow$ 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**



Mask (kernel, template…) with its coefficients

Pixels of sub-image under mask

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

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

  - 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!