Image Enhancement in the Spatial Domain

Jesus J. Caban

Outline

1. Need to know the papers you are presenting by 9/15/10
2. Questions about homework?
3. Today's Topic: Image Enhancement

Introduction: Image Enhancement

- Goals:
  - Noise Removal
  - Feature Enhancement
  - Visualization
- Common Techniques
  - Simple (conceptually and computationally)
  - Generally applicable (domain independent)
  - Often heuristic (difficult to define final image "quality")

Image Enhancement: Spatial Domain

- The spatial domain:
  - The image plane
  - For a digital image is a Cartesian coordinate system of discrete rows and columns. At the intersection of each row and column is a pixel. Each pixel has a value, which we will call intensity.
- The frequency domain :
  - A (2-dimensional) discrete Fourier transform of the spatial domain
- Enhancement :
  - To "improve" the usefulness of an image by using some transformation on the image.
  - Often the improvement is to help make the image "better" looking, such as increasing the intensity or contrast.

Image Enhancement

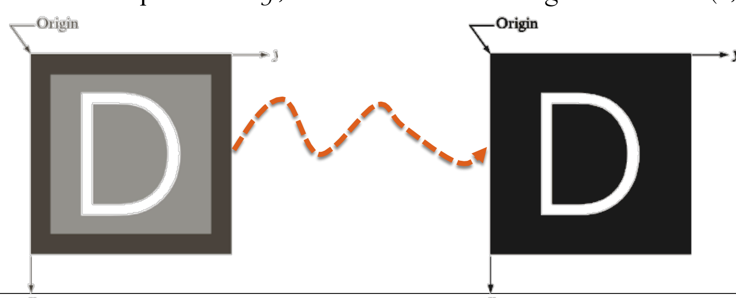
- A mathematical representation of spatial domain enhancement:

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

where $f(x, y)$: the input image

$g(x, y)$: the processed image

T: an operator on f, defined over some neighborhood of (x, y)



The Image Histogram

- The image histogram shows the frequency distribution of gray values in the image.

- Histogram discards spatial information and shows relative frequency of occurrence of gray values

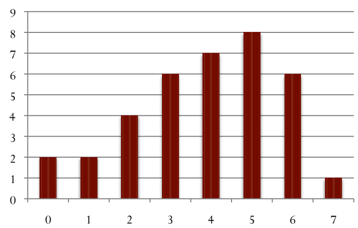
0	3	3	2	5	5
1	1	0	3	4	5
2	2	2	4	4	4
3	3	4	4	5	5
3	4	5	5	6	6
7	6	6	6	6	5

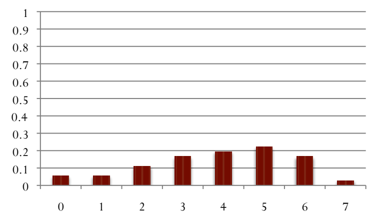
Image Histogram

0	3	3	2	5	5
1	1	0	3	4	5
2	2	2	4	4	4
3	3	4	4	5	5
3	4	5	5	6	6
7	6	6	6	6	5

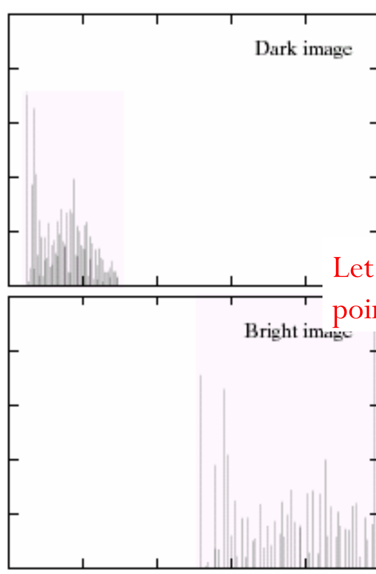
$$\sum_{i=I_{\min}}^{I_{\max}} H(i) = M * N$$

Image size: M * N





Histogram Processing



Let's start with point processing

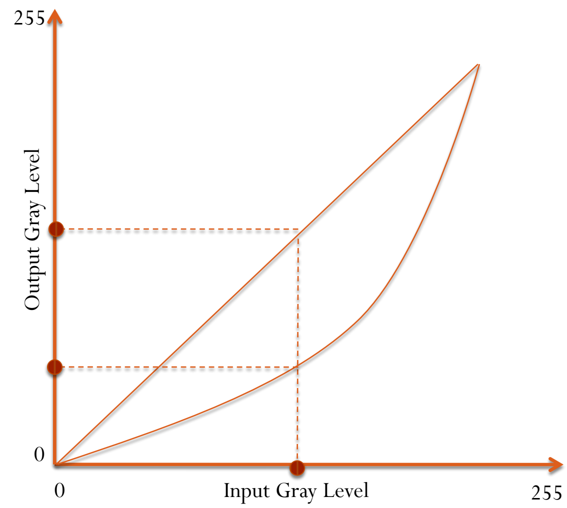
Enhancement through Point Processing

- Based on the intensity of a single pixel only as opposed to a neighborhood or region
- Straightforward methods typically used to globally adjust
  - Contrast
    - The difference in gray levels or luminance value over some area of the image
    - Contrast Range: $I_{max} - I_{min}$
    - Contrast Ratio: I_{max}/I_{min}
  - Color
  - Intensity

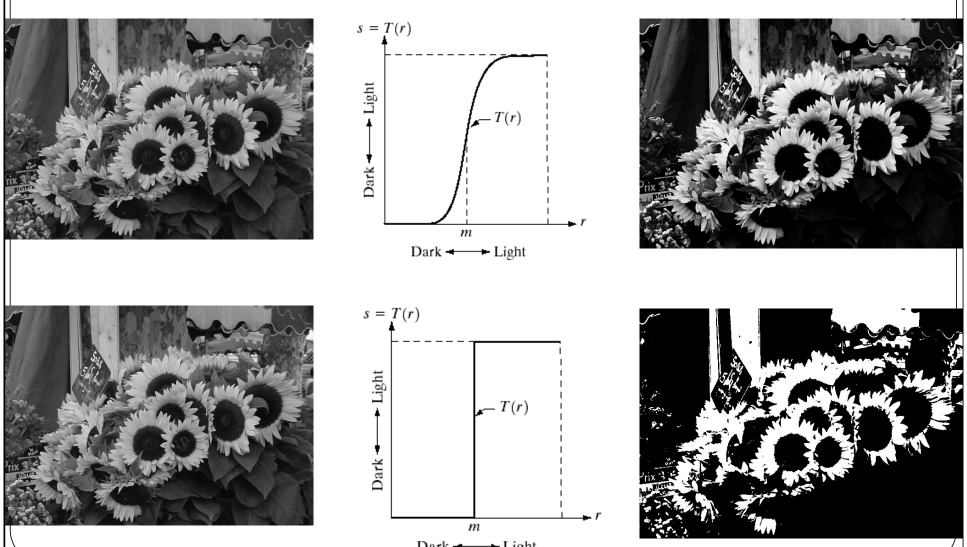
Contrast Enhancement: Scaling

- General Idea:
  - Apply a linear (non-linear) scaling transform to every point in the image to obtain a new (modified) image.

Gray-level Transformation



Gray-level Transformation



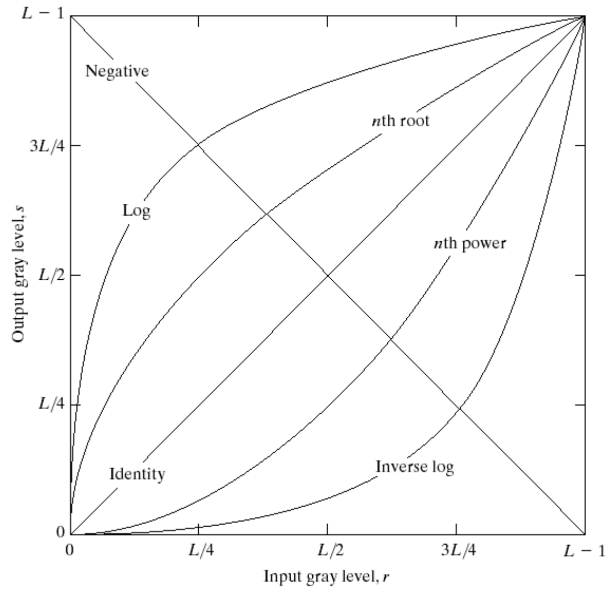
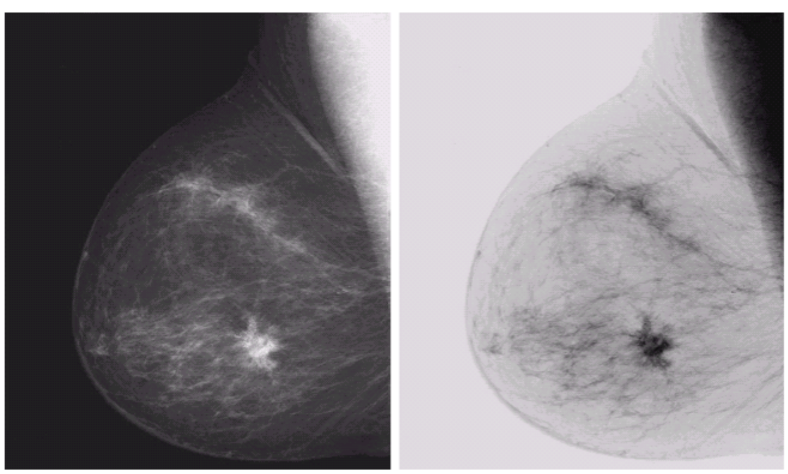


Image Negatives

- Let the range of gray level be [0, *L*-1], then

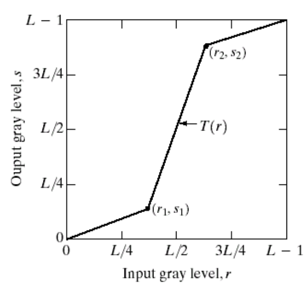
$$s = L - 1 - r$$

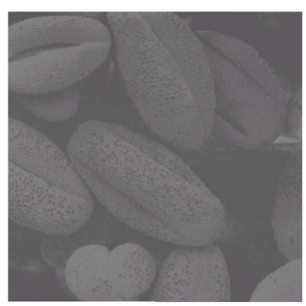


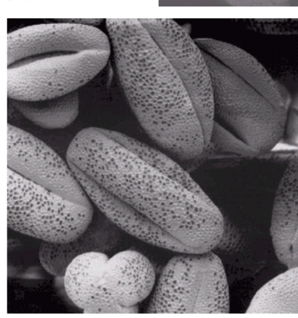
Gray-level : Manipulation

1) Contrast Stretching
2) Gray-level Slicing

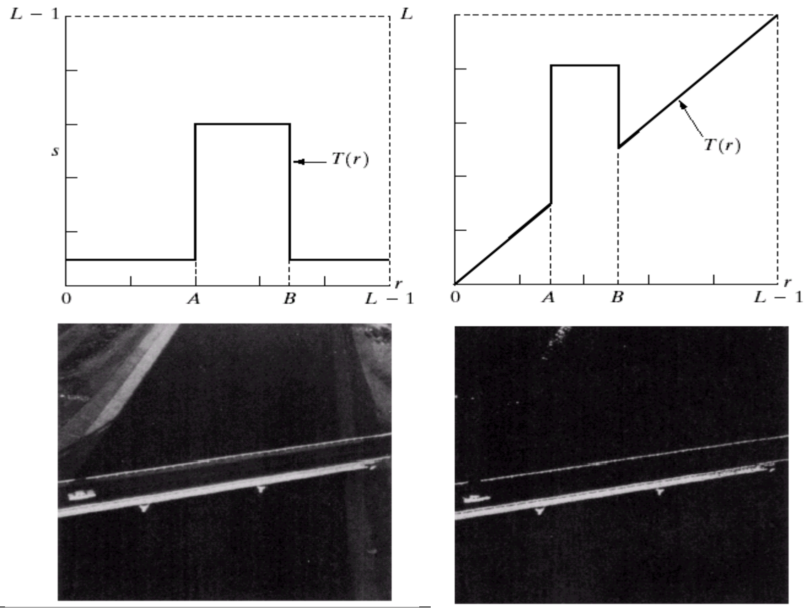
1) Contrast Stretching



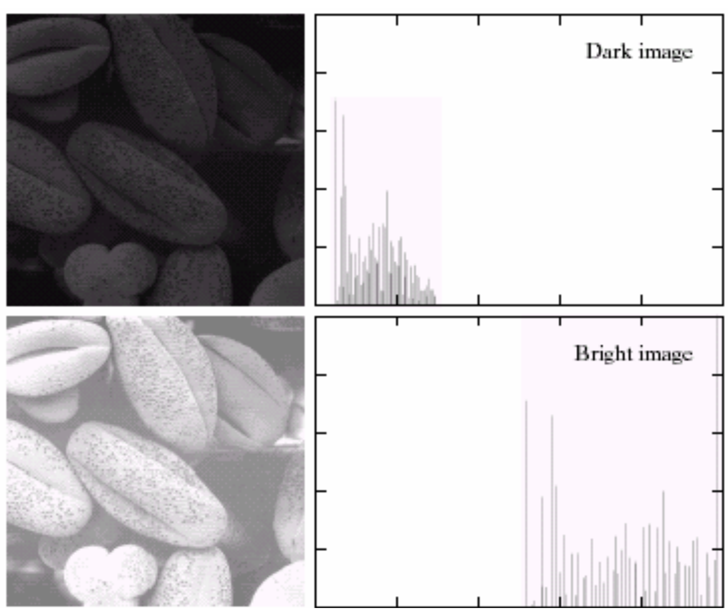




2) Gray-level Slicing

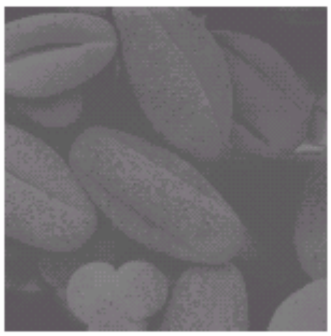


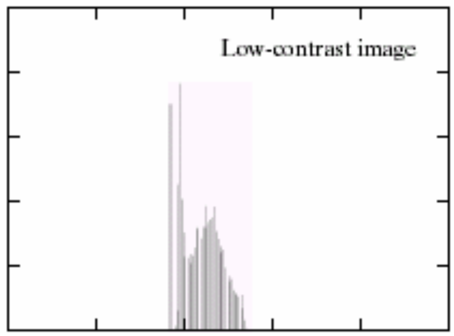
Contrast Stretching: How?



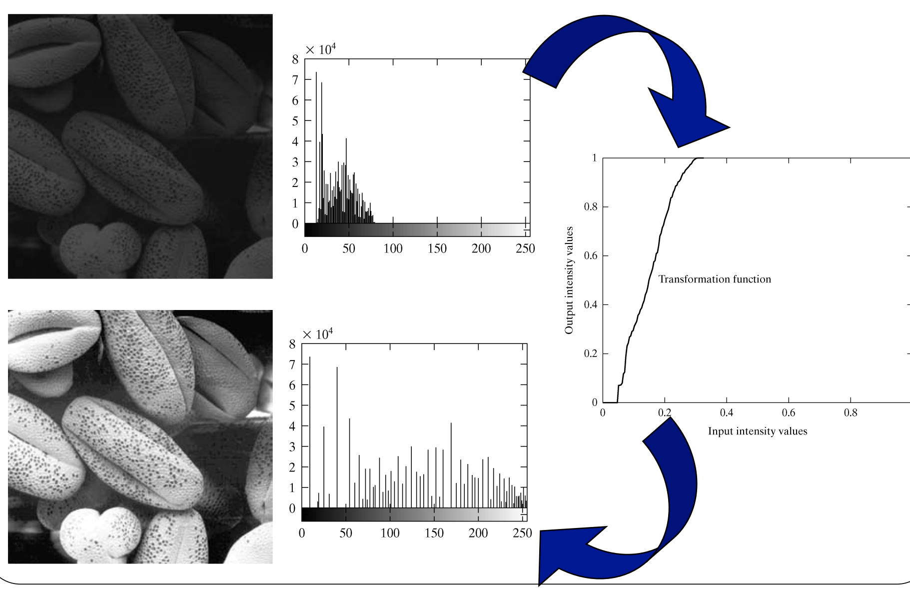
Histogram Equalization

- Histogram equalization:
  - To improve the contrast of an image
  - To transform an image in such a way that the transformed image has a nearly uniform distribution of pixel values





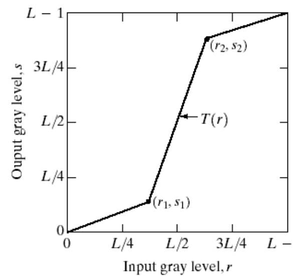
Equalization Transformation Function



Histogram Equalization

1) Probability of Occurrence

$$p_x(i) = p(x=i) = \frac{n_i}{n}, \quad 0 \le i < L$$



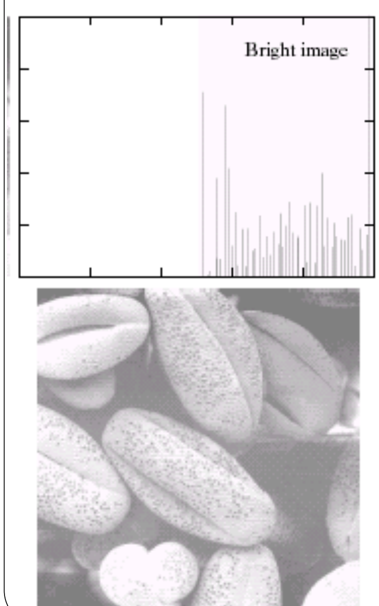
2) Create Cumulative Distribution Function

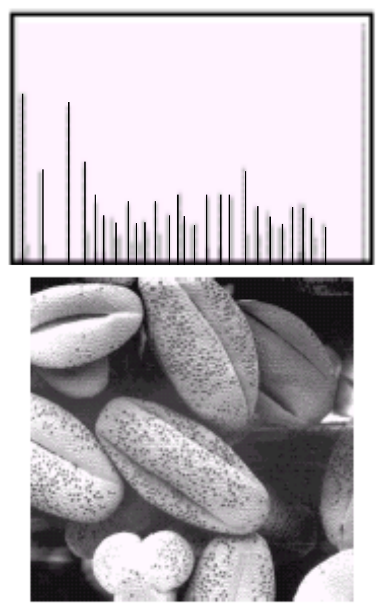
$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

3) Map old value to new value

$$r_k = round\left(\frac{cdf(s_k) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

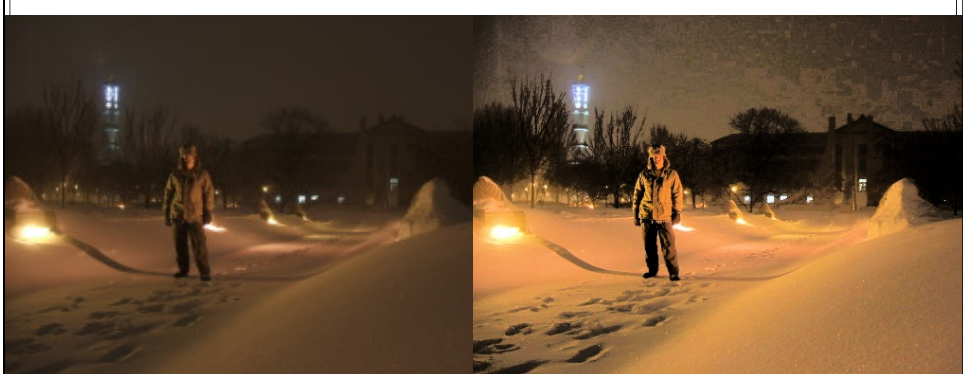
Equalization Examples





Histogram Equalization

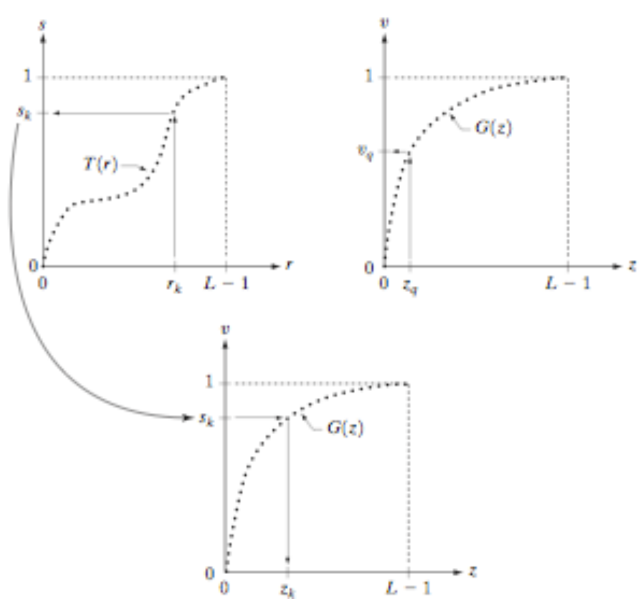
Histogram Equalization - Limitations



1) Histogram Matching

- Histogram matching is similar to histogram equalization, except that instead of trying to make the output image have a flat histogram, we would like it to have a histogram of a specified shape, say $p_z(z)$.

Histogram Matching



Histogram Matching

1. Obtain the histogram of the given image
2. Pre-compute a mapped level s_k for each level r_k using

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} \qquad K = 1, \ldots, L-1$$

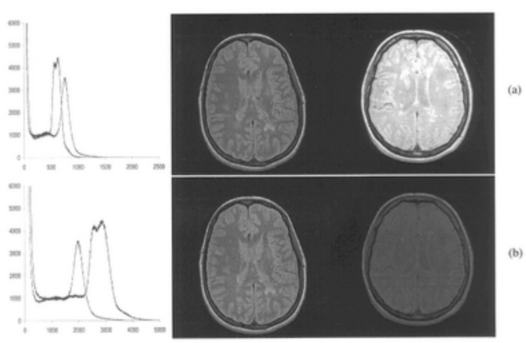
3. Obtain the transformation function G from the given $p_z(z)$ using

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

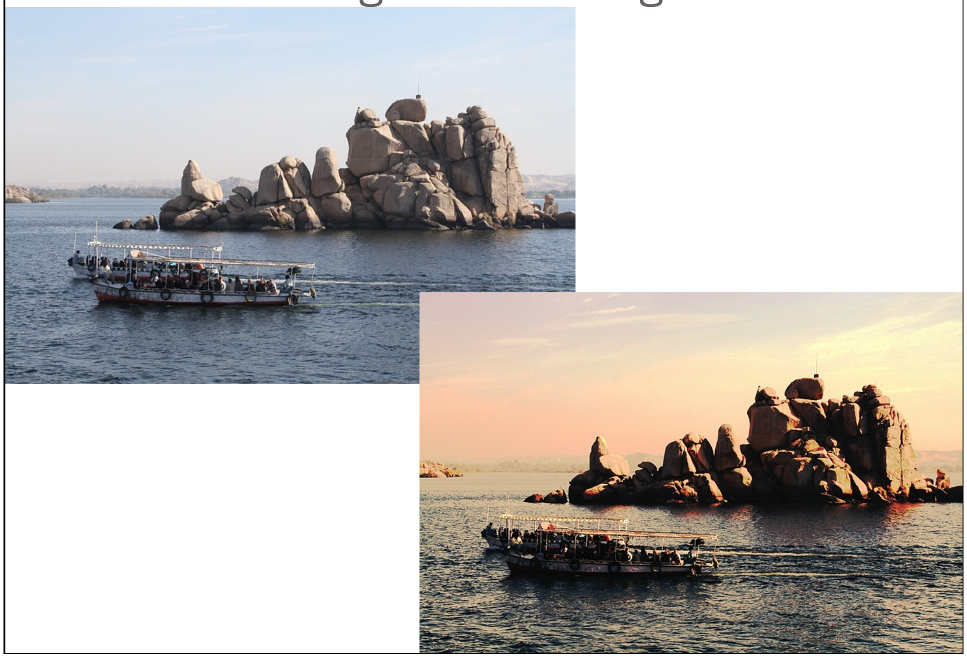
4. Pre-compute z_k for each value s_k using a iterative scheme

5. For each pixel in the original image, if the value of the pixel is r_k, map this value to its corresponding level s_k, then map level s_k into the final level z_k.

Histogram Matching

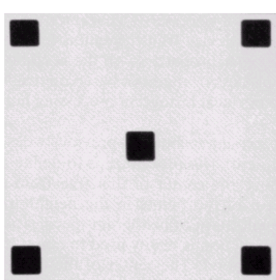


Histogram Matching

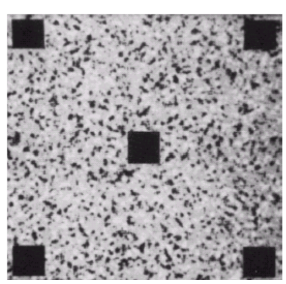


2) Local Enhancement

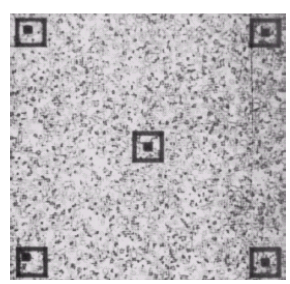
- The histogram processing methods discussed previously are global, in the sense that pixels are modified by a transformation function based on the gray-level content of an entire image.
- However, there are cases in which it is necessary to enhance details over small areas in an image.



original



global



Local (7x7)

Operations

Image Operations

- Just as with numbers, we can apply different operations to an image
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - And: &&
  - Or: | |
  - Derivative: $\frac{dx}{dy}$
  - Average
  - Mean

Enhancement Using Logic Operations

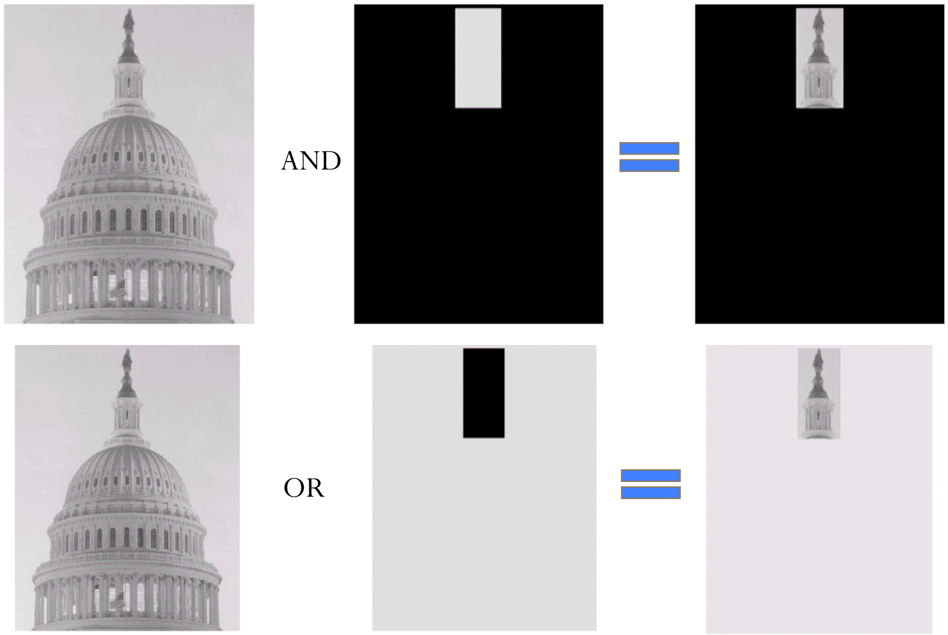


Image Subtraction

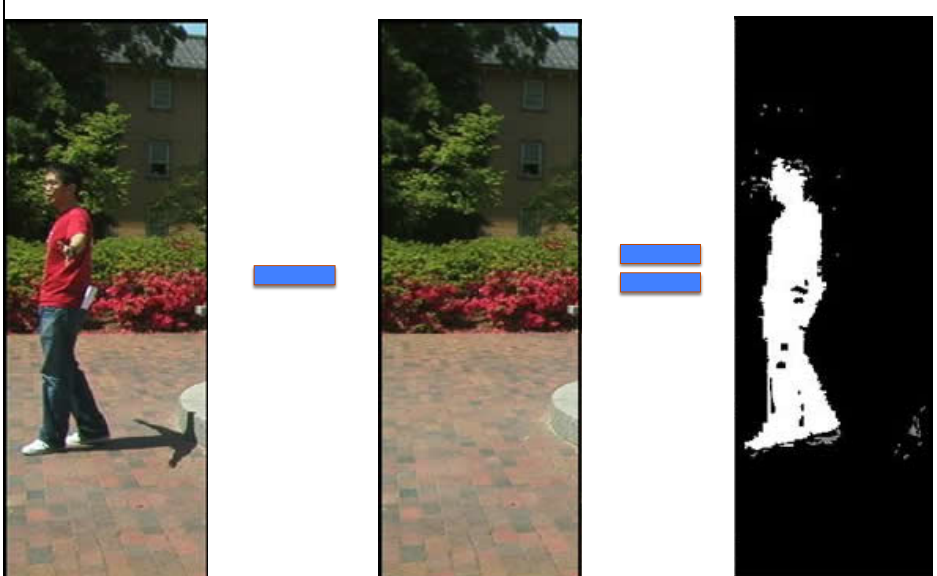


Image Averaging

- When taking pictures in reduced lighting (i.e., low illumination), image noise becomes apparent.
- A noisy image $g(x,y)$ can be defined by

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

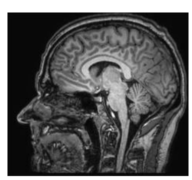
where $f(x,y)$: an original image
$\eta(x,y)$: the addition of noise

- One simple way to reduce this granular noise is to take several identical pictures and average them, thus smoothing out the randomness.

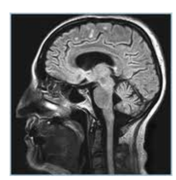
Image Averaging

AVG =

+



+



3

Basics of Spatial Filtering

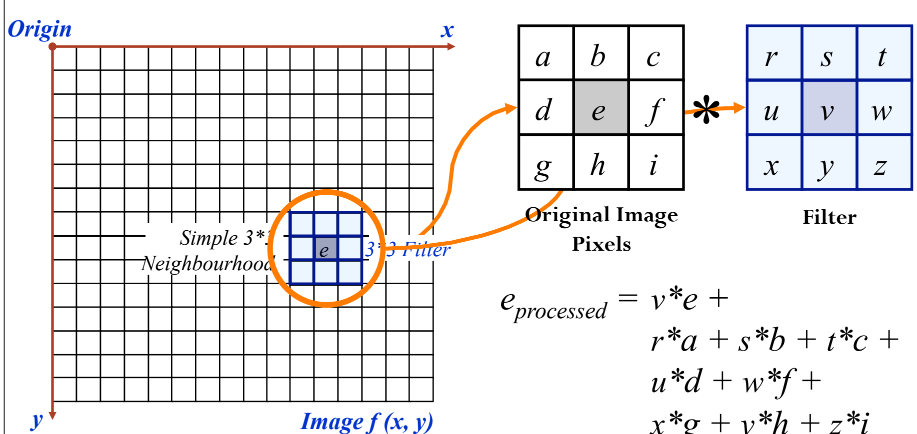
- In general, linear filtering of an image f of size MxN is given by

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

- This concept called convolution. Filter masks are sometimes called convolution masks or convolution kernels.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

The Spatial Filtering Process



$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the filtered image

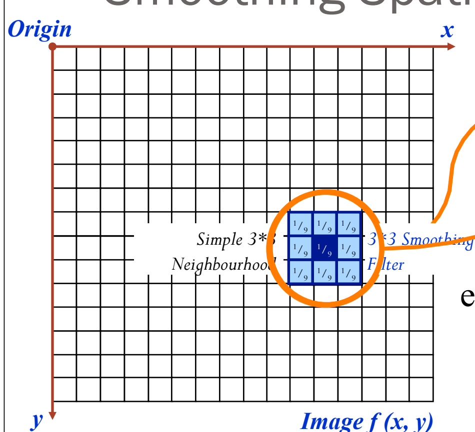
Smoothing Spatial Filters

- One of the simplest spatial filtering operations we can perform is a smoothing operation
  - Simply average all of the pixels in a neighbourhood around a central value
  - Especially useful in removing noise from images
  - Also useful for highlighting gross detail

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Simple averaging filter

Smoothing Spatial Filtering



104	*100*	*108*
99	*106*	*98*
95	*90*	*85*

Original Image Pixels

$*$

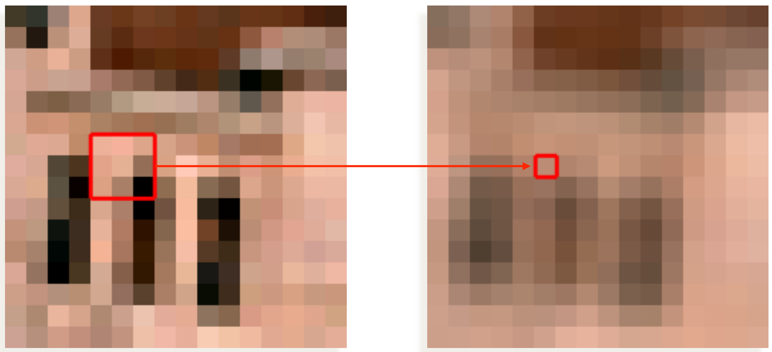
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Filter

$$\begin{aligned} e = {} & ^1/_9*106 + \\ & ^1/_9*104 + {}^1/_9*100 + {}^1/_9*108 + \\ & ^1/_9*99 + {}^1/_9*98 + \\ & ^1/_9*95 + {}^1/_9*90 + {}^1/_9*85 \\ = {} & 98.3333 \end{aligned}$$

The above is repeated for every pixel in the original image to generate the smoothed image

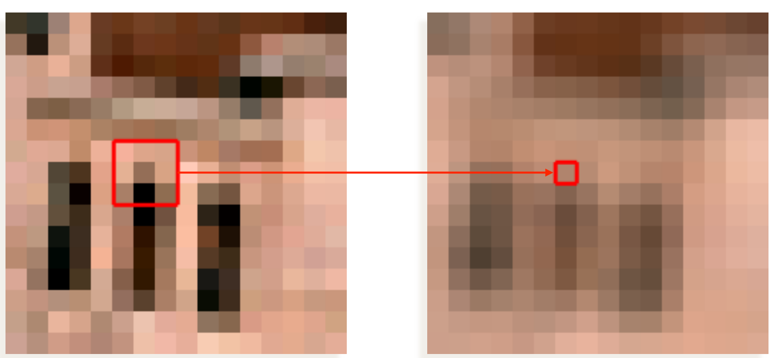
Moving Window Transform: Example



original

3x3 average

Moving Window Transform: Example



original

3x3 average

Image Smoothing Example

- The image at the top left is an original image of size 500*500 pixels
- The subsequent images show the image after filtering with an averaging filter of increasing sizes
  - 3, 5, 9, 15 and 35
- Notice how detail begins to disappear

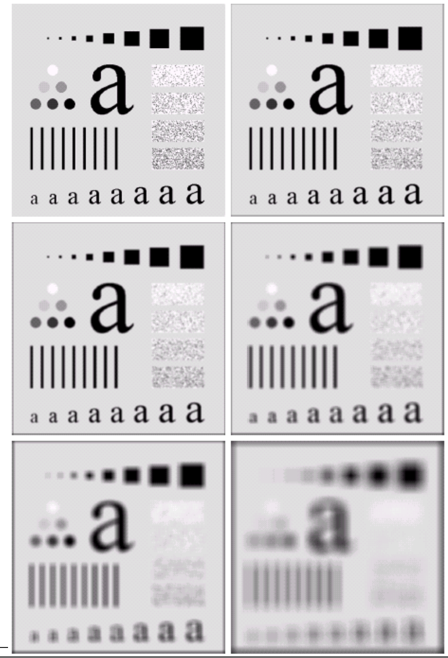
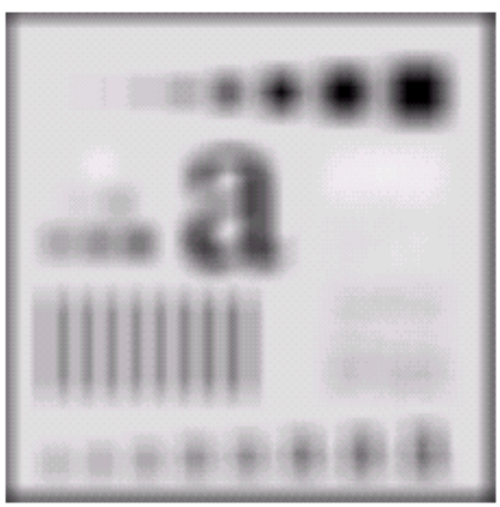


Image Smoothing Example



Basics of Spatial Filtering

- In general, linear filtering of an image *f* of size *M*x*N* is given by

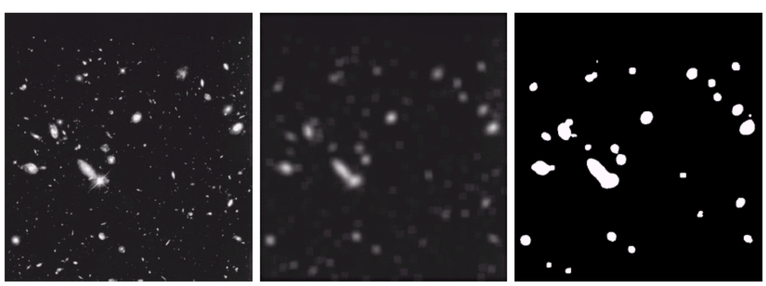
$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

- This concept called convolution. Filter masks are sometimes called convolution masks or convolution kernels.

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Another Smoothing Example

- By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



Original Image **Smoothed Image** **Thresholded Image**

Weighted Smoothing Filters

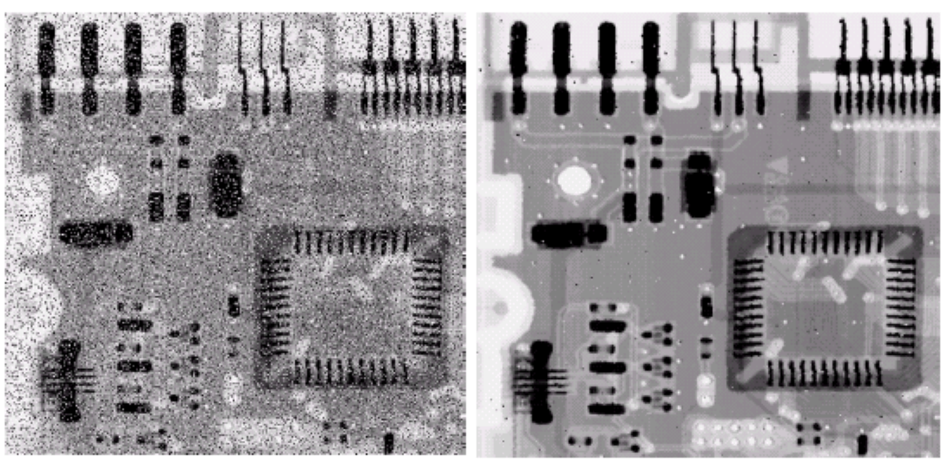
- More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function
  - Pixels closer to the central pixel are more important
  - Often referred to as a *weighted averaging*

$^1/_{16}$	$^2/_{16}$	$^1/_{16}$
$^2/_{16}$	$^4/_{16}$	$^2/_{16}$
$^1/_{16}$	$^2/_{16}$	$^1/_{16}$

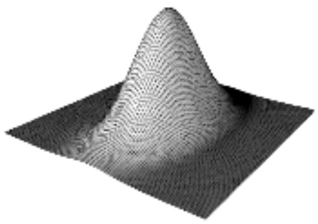
Weighted averaging filter

Order Statistic Filters

- Order-statistic filters
  - Median filter: to reduce impulse noise (salt-and-pepper noise)



Gaussian Smoothing



$$g(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2-\mu_x^2}{2\sigma^2}} e^{-\frac{y^2-\mu_{yx}^2}{2\sigma^2}}$$

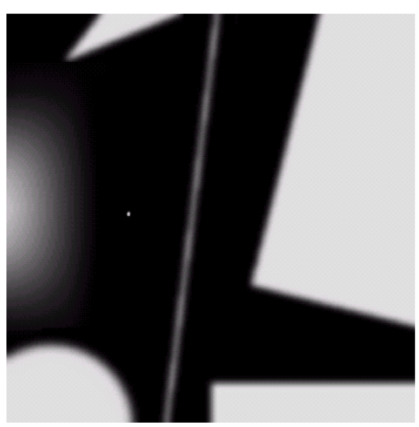
$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

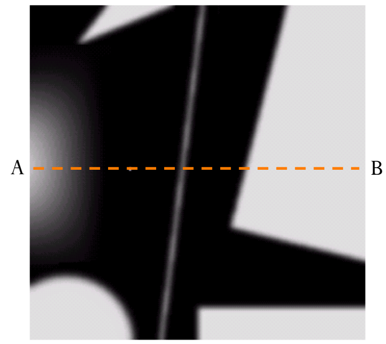
Derivatives

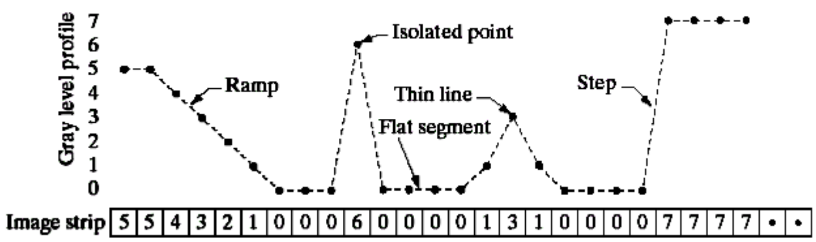
Differentiation measures the *rate of change* of a function

Let's consider a simple 1 dimensional example



Spatial Differentiation





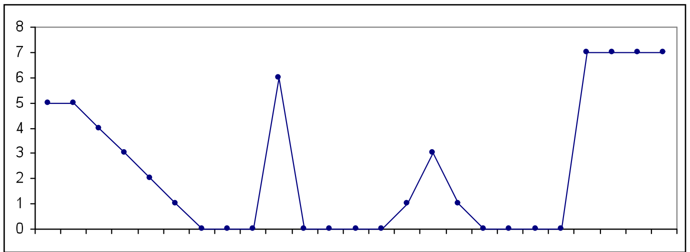
1st Order Derivative

The formula for the 1st derivative of a function is as follows:

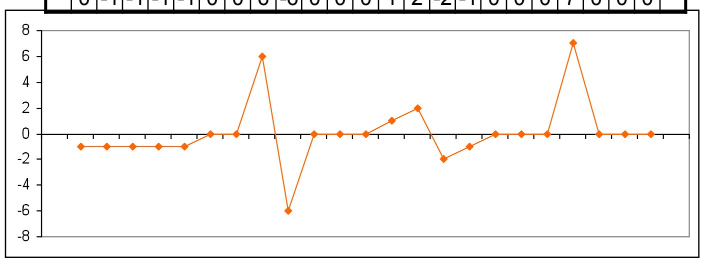
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

It's just the difference between subsequent values and measures the rate of change of the function

1st Order Derivative



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



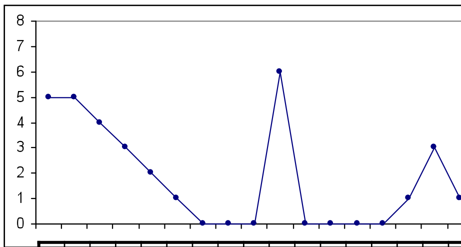
2nd Order Derivative

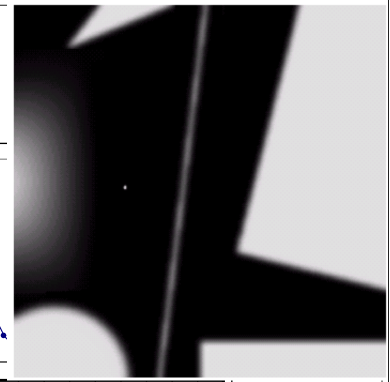
The formula for the 2nd derivative of a function is as follows:

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

Simply takes into account the values both before and after the current value

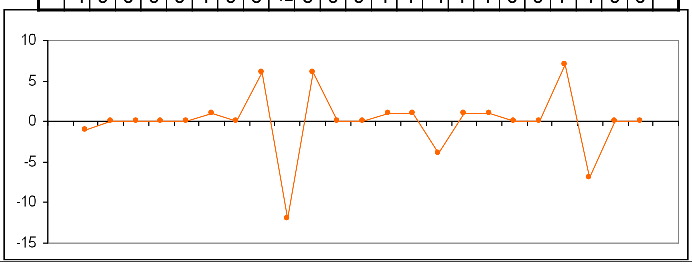
2nd Order Derivative





5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7

	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0	



Using Second Derivatives For Image Enhancement

The 2nd derivative is more useful for image enhancement than the 1st derivative

- Stronger response to fine detail
- Simpler implementation

The Laplacian

The Laplacian is defined as follows:

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

where the partial 1st order derivative in the x direction is defined as follows:

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

and in the y direction as follows:

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

The Laplacian

So, the Laplacian can be given as follows:

$$\begin{aligned}\nabla^2 f = [&f(x+1, y) + f(x-1, y) \\ &+ f(x, y+1) + f(x, y-1)] \\ &- 4f(x, y)\end{aligned}$$

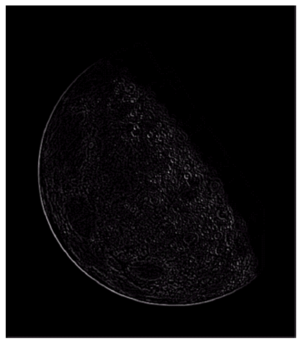
We can easily build a filter based on this

0	1	0
1	-4	1
0	1	0

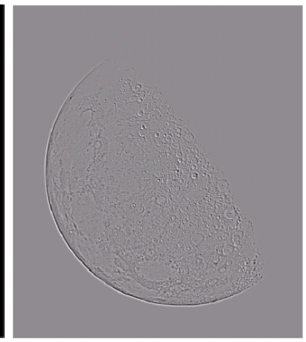
The Laplacian

Applying the Laplacian to an image we get a new image that highlights edges and other discontinuities

Original Image



Laplacian Filtered Image

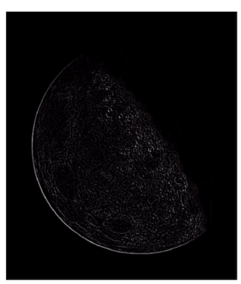


Laplacian Filtered Image Scaled for Display

Laplacian Image Enhancement

Original Image

-



Laplacian Filtered Image

=

Sharpened Image

In the final sharpened image edges and fine detail are much more obvious

$$g(x,y) = f(x,y) - \nabla^2 f$$

Laplacian Image Enhancement

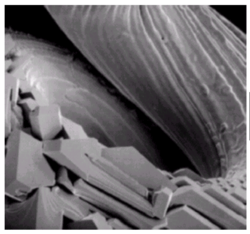
Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

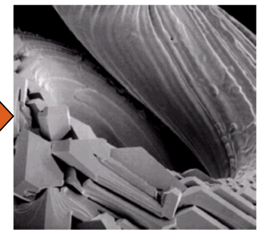
$$
\begin{aligned}
g(x,y) &= f(x,y) - \nabla^2 f \\
&= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\qquad + f(x,y+1) + f(x,y-1) \\
&\qquad - 4f(x,y)] \\
&= 5f(x,y) - f(x+1,y) - f(x-1,y) \\
&\qquad - f(x,y+1) - f(x,y-1)
\end{aligned}
$$

Simplified Image Enhancement (cont…)

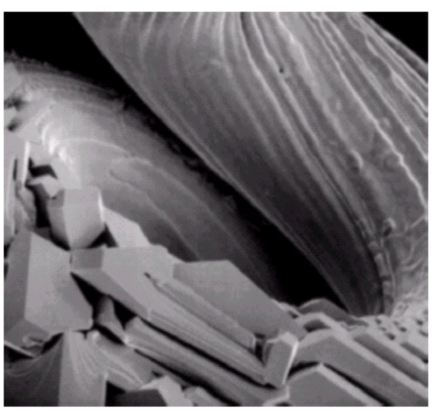
This gives us a new filter which does the whole job for us in one step

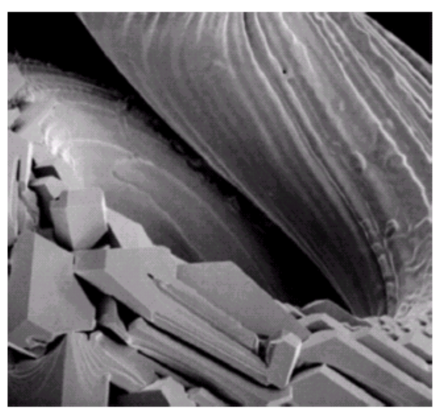


0	-1	0
-1	5	-1
0	-1	0



Simplified Image Enhancement





Variants On The Simple Laplacian

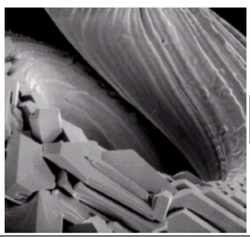
There are lots of slightly different versions of the Laplacian that can be used:

0	1	0
1	-4	1
0	1	0

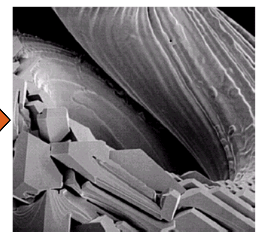
Simple Laplacian

1	1	1
1	-8	1
1	1	1

Variant of Laplacian



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



Acknowledgements

- Some of the images and diagrams have been taken from the Gonzalez et al, "Digital Image Processing" book.