Politecnico di Milano
Master of Science program in Biomedical Engineering

Biomedical Image Processing Lab class

(5 credits)

Class V

April 20th 2012

Enrico Caiani, PhD

Filters for multiplicative noise
Homomorphic Filtering

- Digital images are created from optical images, consisting of a lighting component and a reflectance component.

$I(r,c) = L(r,c)R(r,c)$

where $L(r,c)$ represents the contribution of the lighting conditions

and $R(r,c)$ represents the contribution of the reflectance properties of the objects

- Homomorphic filtering attempts to enhance *R(r,c)* , while filtering out *L(r,c)*

Assumptions:

- Lighting component consists of primarily slow spatial changes and determines overall brightness range
- Reflectance properties of the objects consist of primarily of high spatial frequency information (edges/details) and creates the local contrast

Homomorphic Filtering (multiplicative Noise Filtering)

Noise Model:

Image	i(x,y)
Noise	n(x,y)
Brightness	$\boxed{f(x,y) = i(x,y) \cdot n(x,y)}$

Assumption: noise $\approx$ low frequencies.

Goal: Clean multiplicative noise
(suppress low frequencies associated with n(x,y))

However:

$$\boxed{\tilde{F}(i(x,y)\cdot n(x,y)) \neq \tilde{F}(i(x,y))\cdot \tilde{F}(n(x,y))}$$

Homomorphic Filtering - Example

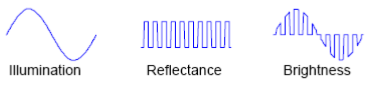
Reflectance Model:

Surface Reflectance	i(x,y)
Illumination	n(x,y)
Brightness	$\boxed{f(x,y) = i(x,y) \cdot n(x,y)}$

Assumptions:

Illumination changes "slowly" across scene
$\Longrightarrow$ Illumination $\approx$ low frequencies.

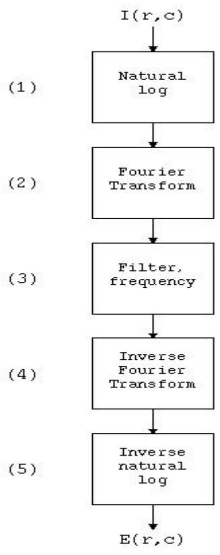
Surface reflections change "sharply" across scene
$\Longrightarrow$ reflectance $\approx$ high frequencies.



Goal: Determine i(x,y)

The homomorphic filtering process consists of five steps:

1) *A natural log transform* (base e): it decouples the *L(r,c)* and *R(r,c)* components
2) *Fourier transform:* it puts the image in the frequency domain
3) *Filtering*
4) *Inverse Fourier transform*
5) *Inverse log function:* the exponential, to put back the image in the spatial domain.



Perform:

$z(x,y) = \log(f(x,y)) = \log(i(x,y) \cdot n(x,y)) = \log(i(x,y)) + \log(n(x,y))$

$\downarrow \mathcal{F}$

$Z(u,v) = I(u,v) + N(u,v)$

Apply low attenuating filter $H(u,v)$ $\downarrow$

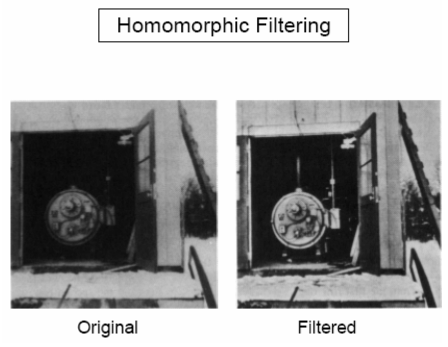
$S(u,v) = H(u,v) \cdot Z(u,v) = H(u,v) \cdot I(u,v) + H(u,v) \cdot N(u,v)$

$\downarrow \mathcal{F}^{-1}$

$s(x,y) = i'(x,y) + n'(x,y)$

$\downarrow$

$g(x,y) = \exp(s(x,y)) = \exp(i'(x,y)) \cdot \exp(n'(x,y))$

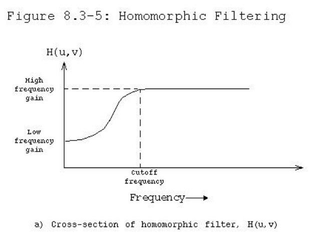


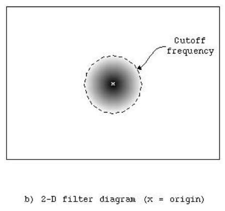
Homomorphic Filtering:

$$\begin{tikzcd}
{} \arrow[r, "\text{image}"] & \boxed{\log} \arrow[r] & \boxed{\text{FFT}} \arrow[r] & \boxed{H(u,v)} \arrow[r] & \boxed{\text{FFT}^{-1}} \arrow[r] & \boxed{\exp} \arrow[r, "\text{image}"] & {}
\end{tikzcd}$$

Caiani 20/04/2012

Low-pass filter for Homomorphic Filtering

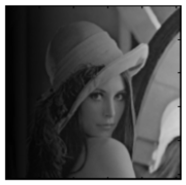


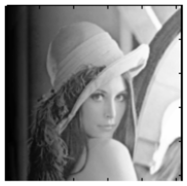


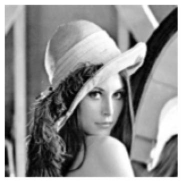
- Typically the high frequency gain is greater than 1, and the low frequency gain is less than 1 which provides us with the desired effect of boosting the *R(r,c)* components, while reducing the *L(r,c)* components
- The selection of the cutoff frequency is highly application-specific, and needs to be chosen so that no important information is lost

Caiani 20/04/2012

Homomorphic Filtering


Original


Histogram Equalized


Filtered

Caiani 20/04/2012

Segmentation

- Image segmentation represents the process by which the image is subdivided into regions or objects.
- The segmentation of complex images represents one of the most diffucult tasks in the image processing field. The accuracy and precision of segmentation results into the success of failure of the computerized analysis procedures.
- Image segmentation methods will look for objects that either have some measure of homogeneity within themselves, or have some measure of contrast with the objects on their border
- The homogeneity and contrast measures can include features such as gray level, color, and texture
- Image segmentation techniques can be divided into four main categories:
  - Histogram thresholding
  - Region growing
  - Clustering methods
  - Boundary detection

There is no universally applicable segmentation technique that will work for all images. No segmentation technique is perfect!!!

Boundary detection methods

Subdivide the image based on sharp changes in videointensity (edges).

Point detection

To identify isolated points surrounded by homogeneous background, it is possible to use this filter mask.
Its performance is the following: the greater output is associated when the mask is centered into an isolated discontinuity, while null values are associated elsewhere.

−1	−1	−1
−1	8	−1
−1	−1	−1

Create an image of 128 x 128 pixels, class double, which elements have a homogeneous value (but not 0 or 1). Create inside the image several isolated points with different intensity. Apply the filter mask for point detection. Extract the coordinates of the isolated points and plot on the original image, in correspondence of the coordinates, a red dot.

Caiani 20/04/2012

Line detection

Specific masks give higher output when overimposed to lines of one pixel thckness, with a specific orientation. The sensitive direction is associated to the higher coefficients, and their sum is equal to zero.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

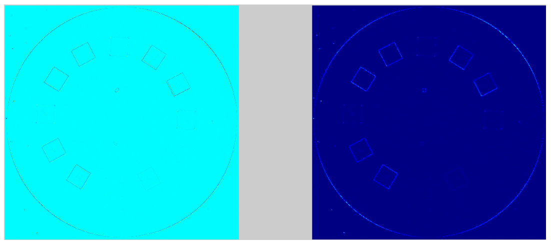
−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

Load "disco1.jpg'. Implement these masks and filter the image with each of them, visualizing the result in the same image using []. Also, visualize the absolute value.



Caiani 20/04/2012

Edge detection

This approach is based on the searching for discontinuity using the first (gradient) and the second (Laplacian) derivatives.

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

A fundamental property of the gradient vector is that it is oriented towards the direction of the maximal variation of f in the coordinates (x,y).
Such angle is given by:

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_x}{G_y}\right)$$

The Laplacian is not commonly utilized by itself as edge detector, as it is very sensitive to noise. Moreover, it results in a double contour and it is not able to indicate the edge direction. However, it is utilized together with other edge detectors, to localize potential edges:

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

One of the key problems is how to estimate the directional derivatives as digital approximations.

Caiani 20/04/2012

<u>DERIVATIVE FILTERS in Matlab</u>

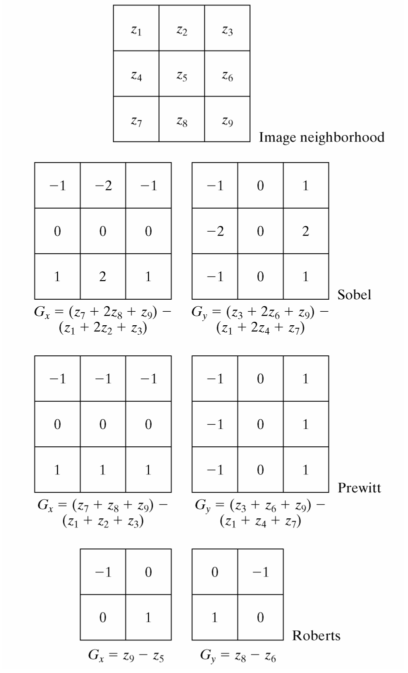
The **edge** function is applied to an intensity image I (uint8 or double) and results into a binary image BW (uint8) of the same size, with 1 in correspondence of the detected edge and 0 elsewhere.
Six different edge detection methods are available.

BW = **edge**(I,‘TYPEOFFILTER',THRESH,DIRECTION)

Using **edge** it is possible to fix a threshold THRESH with whom to compare the resulting gradient amplitude: if it is >THRESH, we are in presence of a true contour, and the corresponding pixel position in BW is set to 1, otherwise not. If not specified, the parameter is selected automatically.
By the parameter DIRECTION, it is possible to select the direction along with to search for the boundaries: ‘horizontal’, ‘vertical’ or ‘both’ (default).

Another way to proceed is, as seen, to generate a filter mask (also using **fspecial**) and to use **imfilter**. In this case, Gx and Gy are computed separately, from which to compute their amplitude to obtain the gradient to be thresholded.

Caiani 20/04/2012



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Image neighborhood

Sobel

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$ $G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$

Prewitt

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$ $G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$

Roberts

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$G_x = z_9 - z_5$ $G_y = z_8 - z_6$

➢SOBEL (default)
BW = **edge**(I,'sobel',THRESH,DIRECTION).
Each filter takes the derivative in one direction and smoothes in the orthogonal direction using a 1D version of a *triangular* filter.

➢PREWITT
BW = **edge**(I,'prewitt',THRESH,DIRECTION)
It is based on the gradient approximation according to Prewitt (max gradient).
Each filter takes the derivative in one direction and smoothes in the orthogonal direction using a uniform filter.

➢ROBERTS
BW = **edge**(I,'roberts',THRESH)
It is based on the gradient approximation according to Roberts.

*Create these masks with **fspecial** and visualize thier frequency responce with **freqz2**. Load an image and filter it both using **edge** and **imfilter**.*

Compass Masks

- The *Kirsch* and *Robinson* edge detection masks are called compass masks since they are defined by taking a single mask and rotating it to the eight major compass orientations: North, Northwest, West, Southwest, South, Southeast, East, and Northeast

Kirsch masks

$$k_0\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad k_1\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad k_2\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad k_3\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$k_4\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad k_5\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad k_6\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad k_7\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

Robinson masks

$$r_0\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \quad r_1\begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix} \quad r_2\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \quad r_3\begin{bmatrix}2 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -2\end{bmatrix}$$

$$r_4\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \quad r_5\begin{bmatrix}0 & -1 & -2\\ 1 & 0 & -1\\ 2 & 1 & 0\end{bmatrix} \quad r_6\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \quad r_7\begin{bmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{bmatrix}$$

They are easier to be implemented, as they rely only on coefficients of 0, 1 and 2, and are symmetrical about their directional axis (the axis with the zeros which corresponds to the line direction)

Edge detection with Compass Masks

- The edge magnitude is defined as the maximum value found by the convolution of each of the masks with the image
- The edge direction is defined by the mask that produces the maximum magnitude
- For instance, k_0 corresponds to a edge along the horizontal direction, whereas k_5 corresponds to a diagonal edge in the Northeast/Southwest direction
- In general, any of the edge detection masks can be extended by rotating them in a manner like the compass masks, which allows us to extract explicit information about edges in any direction

Load "disco1.jpg" and apply the filtering with Robinson compass masks, observing the different results obtained with each mask. Repeat the process after having added Gaussian noise (zero mean, variance 128) to the image.

Caiani 20/04/2012