CS 534: Computer Vision
Texture

Spring 2005
Ahmed Elgammal
Dept of Computer Science
Rutgers University

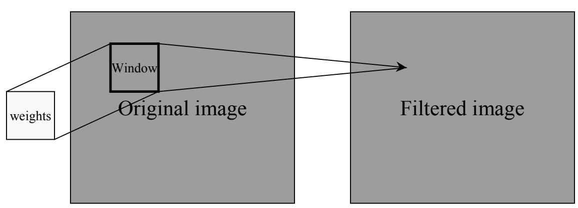
Outlines

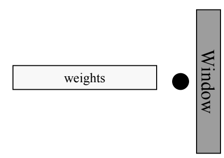
- Finding templates by convolution
- What is Texture
- Co-occurrence matrecis for texture
- Spatial Filtering approach
- Multiresolution processing, Gaussian Pyramids and Laplacian Pyramids
- Gabor filters and oriented pyramids
- Texture Synthesis

Convolution as a Dot Product

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Convoluting an image with a filter is equivalent to taking the dot product of the filter with each image window.

$$R_{ij} = \sum_{u,v} G_{i-u,j-v} H_{uv} = g \cdot h$$

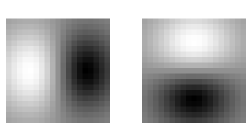


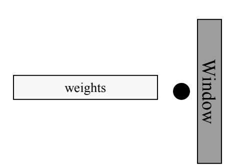


- Largest value when the vector representing the image is parallel to the vector representing the filter
- Filter responds most strongly at image windows that looks like the filter.
- Filter responds stronger to brighter regions! (drawback)

Insight:

- filters look like the effects they are intended to find
- filters find effects they look like



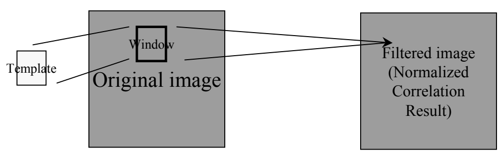


Ex: Derivative of Gaussian used in edge detection looks like edges

Normalized Correlation

- Convolution with a filter can be used to find templates in the image.
- Normalized correlation output is filter output, divided by root sum of squares of values over which filter lies
- Consider template (filter) *M* and image window *N*:

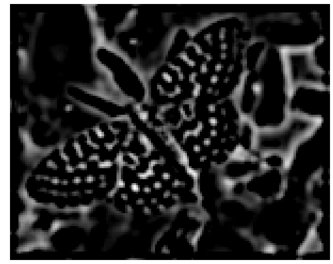
$$C = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} M(i,j)N(i,j)}{[\sum_{i=1}^{n}\sum_{j=1}^{n} M(i,j)^2 \sum_{i=1}^{n}\sum_{j=1}^{n} N(i,j)^2]^{1/2}}$$



Normalized Correlation

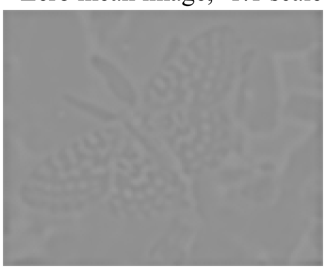
$$C = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} M(i,j)N(i,j)}{[\sum_{i=1}^{n}\sum_{j=1}^{n} M(i,j)^2 \sum_{i=1}^{n}\sum_{j=1}^{n} N(i,j)^2]^{1/2}}$$

- This correlation measure takes on values in the range [0,1]
- it is 1 if and only if N = cM for some constant c
- so N can be uniformly brighter or darker than the template, M, and the correlation will still be high.
- The first term in the denominator, $\Sigma\Sigma M^2$ depends only on the template, and can be ignored
- The second term in the denominator, $\Sigma\Sigma N^2$ can be eliminated if we first normalize the grey levels of *N* so that their total value is the same as that of *M* - just scale each pixel in N by $\Sigma\Sigma\, M / \Sigma\Sigma\, N$

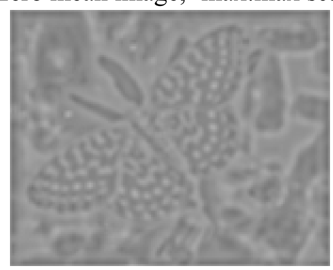


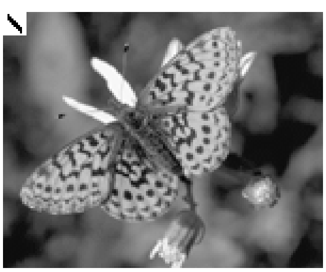
Positive responses

Zero mean image, -1:1 scale



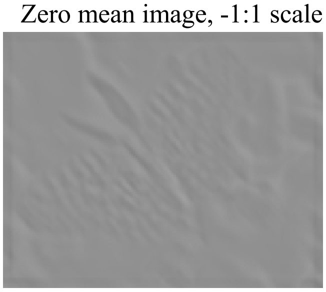
Zero mean image, -max:max scale





Positive responses

Zero mean image, -1:1 scale



Zero mean image, -max:max scale

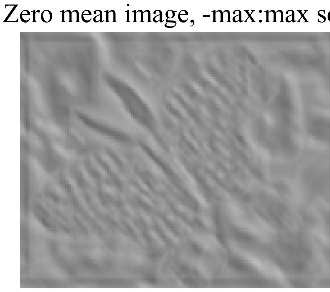
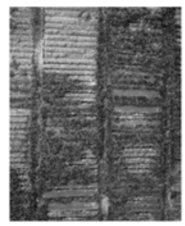


Figure from “Computer Vision for Interactive Computer Graphics,” W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

Texture

- What is texture ? Easy to recognize hard to define
  - Views of large number of small objects: grass, foliage, brush, pebbles, hair
  - Surfaces with patterns: spots, stripes , wood, skin
- Texture consists of organized patterns of quite regular subelements.
- Whether an effect is referred to as texture or not depends on the scale at which it is viewed.

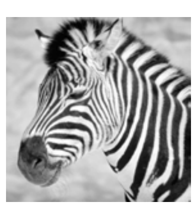
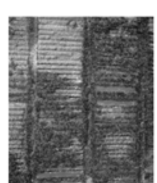





Texture

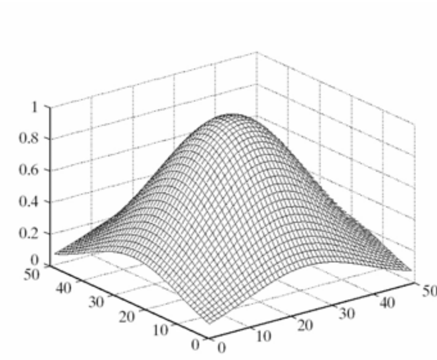
Problems related to Texture:

- Texture analysis: how to represent and model texture
- Texture segmentation: segmenting the image into components within which the texture is constant
- Texture synthesis: construct large regions of texture from small example images
- Shape from texture: recovering surface orientation or surface shape from image texture.


Shape from Texture

- Texture looks different depending on the viewing angle.
- Texture is a good cue for shape and orientation
- Humans are very good at that.



Representing Texture

- What we should look for ?
- Texture consists of organized patterns of quite regular subelements. “Textons”
- Find the subelements, and represent their statistics
- Reason about their spatial layout.
- Problem: There is no known canonical set of textons.

Texture Analysis

Different approaches:

- Co-occurrence matrices (classical)
- Spatial Filtering
- Random Field Models

Co-occurrence Matrix Features

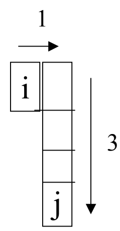
Objective: Capture spatial relations

A co-occurrence matrix is a 2D array C in which

- Both the rows and columns represent a set of possible image values
- $C_d(i,j)$ indicates how many times value i co-occurs with value j in a particular spatial relationship d.
- The spatial relationship is specified by a vector d = (dr,dc).

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

gray-tone image



d = (3,1)

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 3 \\ 1 & 2 & 0 & 2 \\ 2 & 0 & 0 & 1 \end{array} \quad C_d$$

co-occurrence matrix

From C_d we can compute N_d , the normalized co-occurrence matrix, where each value is divided by the sum of all the values.

Co-occurrence Features

From Co-occurrence matrices extract some quantitative features:

$$Energy = \sum_i \sum_j N_d^2(i,j) \quad (7.7)$$

$$Entropy = -\sum_i \sum_j N_d(i,j) log_2 N_d(i,j) \quad (7.8)$$

$$Contrast = \sum_i \sum_j (i-j)^2 N_d(i,j) \quad (7.9)$$

$$Homogeneity = \sum_i \sum_j \frac{N_d(i,j)}{1+|i-j|} \quad (7.10)$$

$$Correlation = \frac{\sum_i \sum_j (i-\mu_i)(j-\mu_j) N_d(i,j)}{\sigma_i \sigma_j} \quad (7.11)$$

where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the row and column sums.

Disadvantages:

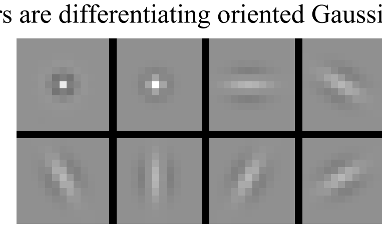
- Computationally expensive
- Sensitive to gray scale distortion (co-occurrence matrices depend on gray values)
- May be useful for fine-grain texture. Not suitable for spatially large textures.

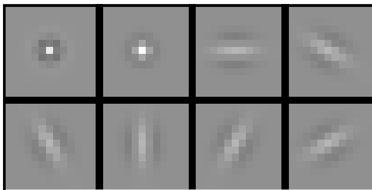
Spatial Filtering Approaches

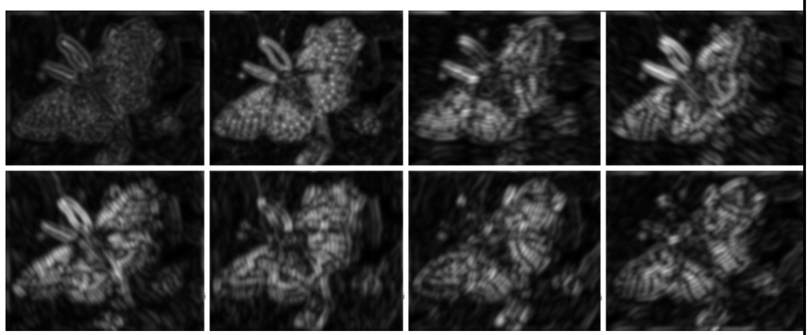
- Look for the subelements
- But what are the subelements, and how do we find them?
- Find subelements by applying filters, looking at the magnitude of the response
- Spots and bars detectors at various scales and orientations.

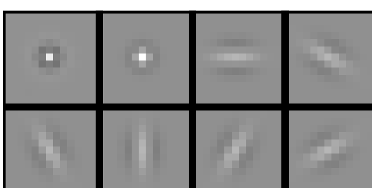
Typically:

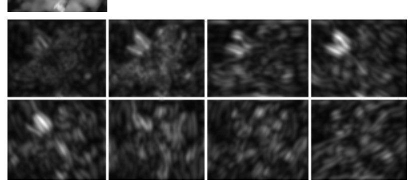
- "Spot" filters are Gaussians or weighted sums of concentric Gaussians.
- "Bar" filters are differentiating oriented Gaussians



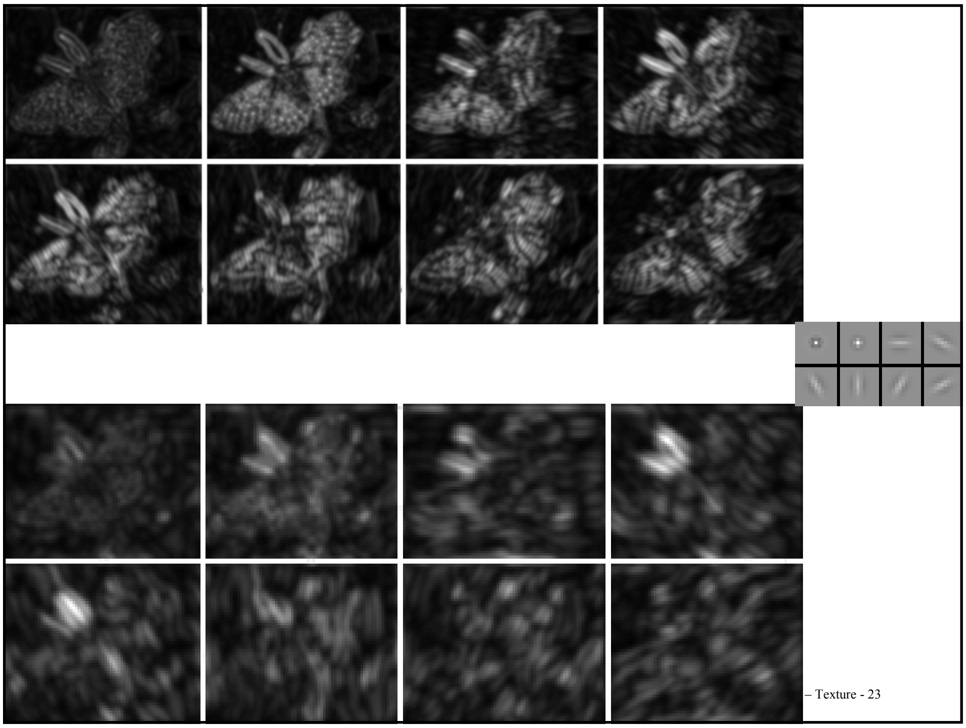




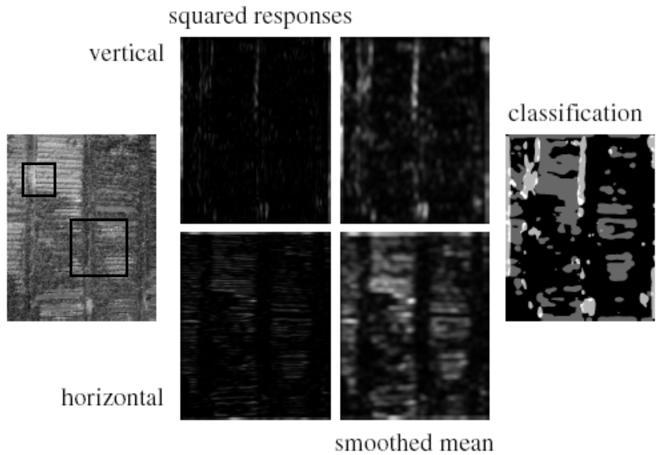




We look at the squared magnitude response. Why ?

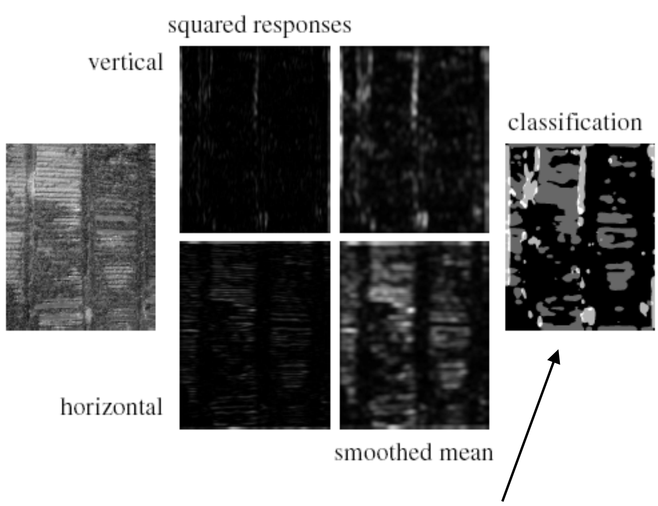


- How many filters and at what orientations ?
- Filter responses are not unique
- Tradeoff: using more filters leads to a more detailed and more redundant representation of the image
- How to control the amount of redundant information?
- At what scale?
- There are two scales:
  – The scale of the filter
  – The scale over which we consider the distributions of the filters.
- What statistics should be collected from filters responses.



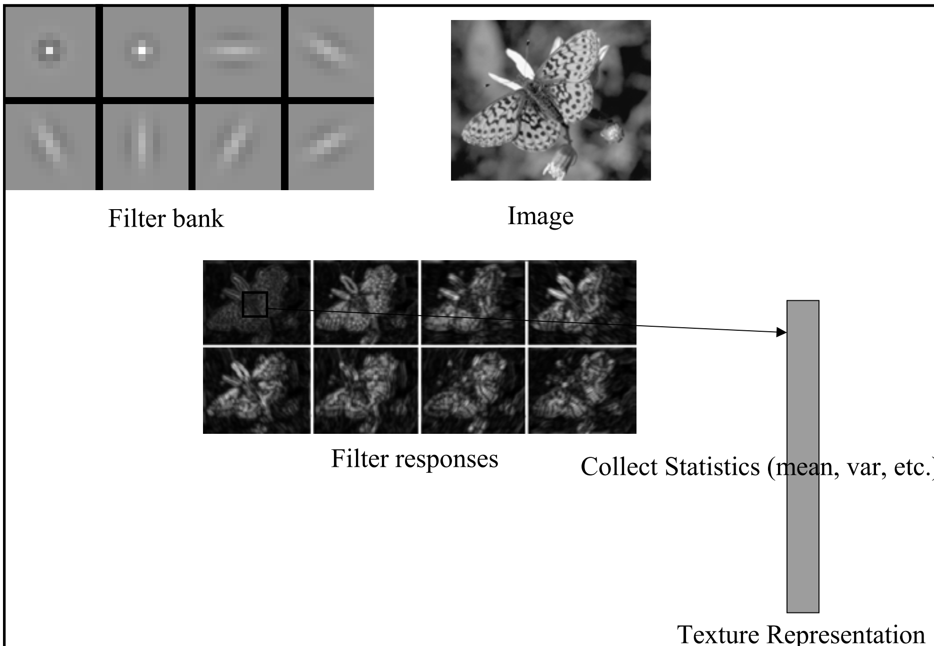
There are two scales:

- The scale of the filter
- The scale over which we consider the distributions of the filters.



Classification:

- Texture is classified into four categories: vertical, horizontal, both or neither.



Scaled representations: Multiresolution

Use a multiresolution representation (Image Pyramid)

- Search over scale
- Spatial Search
- Feature Tracking

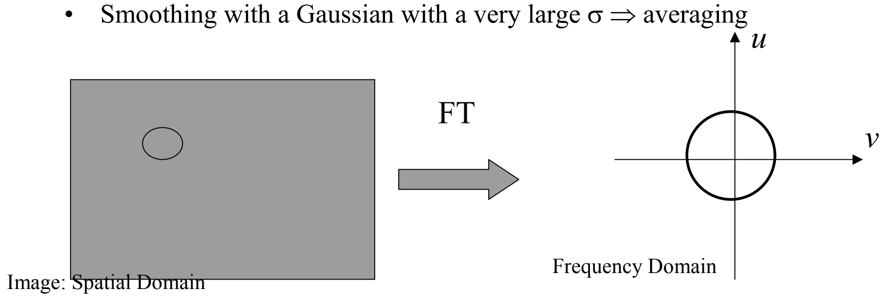
Examples:

- Search for correspondence
  - look at coarse scales, then refine with finer scales
- Edge tracking
  - a “good” edge at a fine scale has parents at a coarser scale
- Control of detail and computational cost in matching
  - e.g. finding stripes
  - terribly important in texture representation

Gaussian Filter and Smoothing

Gaussian Filter is Low-Pass Filter:

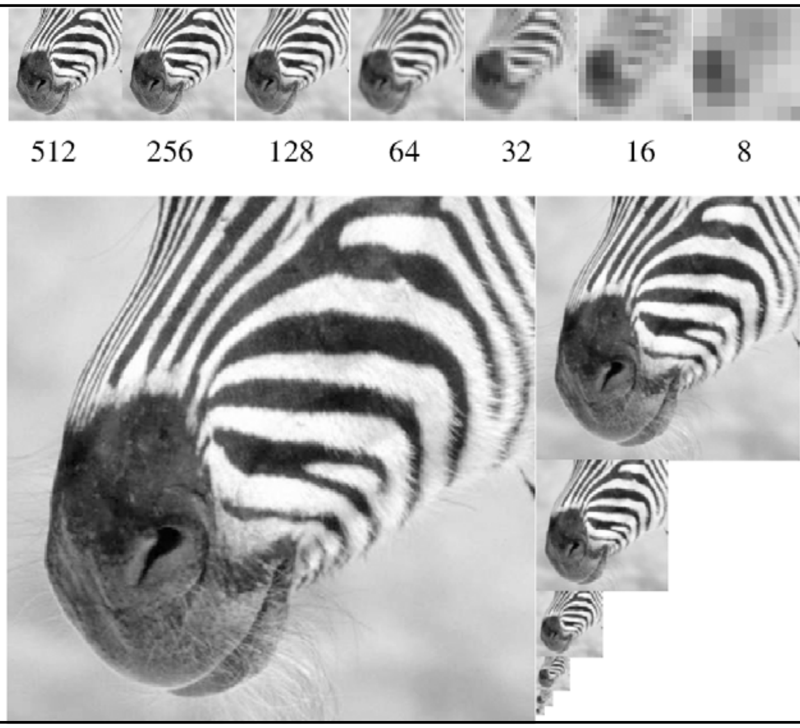
- Recall: Convolution in the image domain is equivalent to multiplication in the Frequency domain.
- Recall: FT of a Gaussian with sd=σ is a Gaussian with sd=1/σ
- Therefore, convolving an image with a Gaussian with sd=σ is equivalent to multiplying it's FT with a Gaussian with sd=1/σ
- Therefore we will get rid of high frequencies.
- Smoothing with a Gaussian with a very small σ ⇒ get rid of highest spatial frequencies
- Smoothing with a Gaussian with a very large σ ⇒ averaging



The Gaussian pyramid

- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Forming a Gaussian Pyramid:
  - Set the finest scale layer to the image
  - For each layer going up (coarser)
    - Obtain this layer by smoothing the previous layer with a Gaussian and subsampling it

$$P_{\text{Gaussian}}(I)_{n+1} = S^{\downarrow}(G_{\sigma} * P_{\text{Gaussian}}(I)_n)$$
$$P_{\text{Gaussian}}(I)_1 = I$$



The Laplacian Pyramid

- Gaussians are low pass filters, so response is redundant
- A coarse level layer of the Gaussian pyramid predicts the appearance of the next finer layer
- Laplacian Pyramid
  - preserve differences between upsampled Gaussian pyramid level and Gaussian pyramid level
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other levels

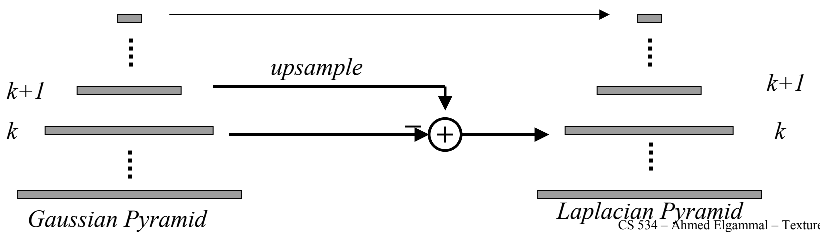
$$P_{\text{Laplacian}}(I)_m = P_{\text{Gaussian}}(I)_m$$

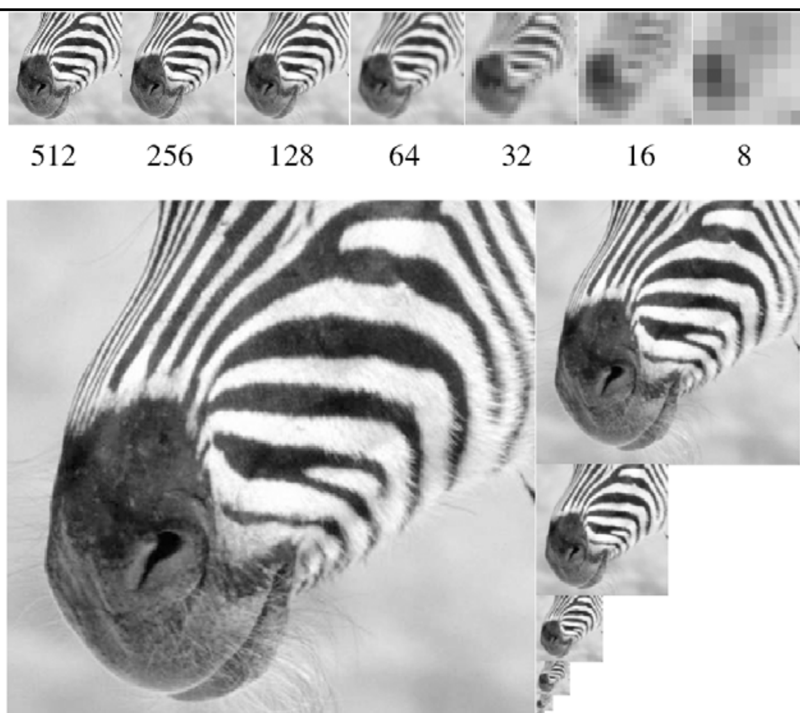
$$P_{\text{Laplacian}}(I)_k = P_{\text{Gaussian}}(I)_k - S^{\uparrow}(P_{\text{Gaussian}}(I)_{k+1})$$

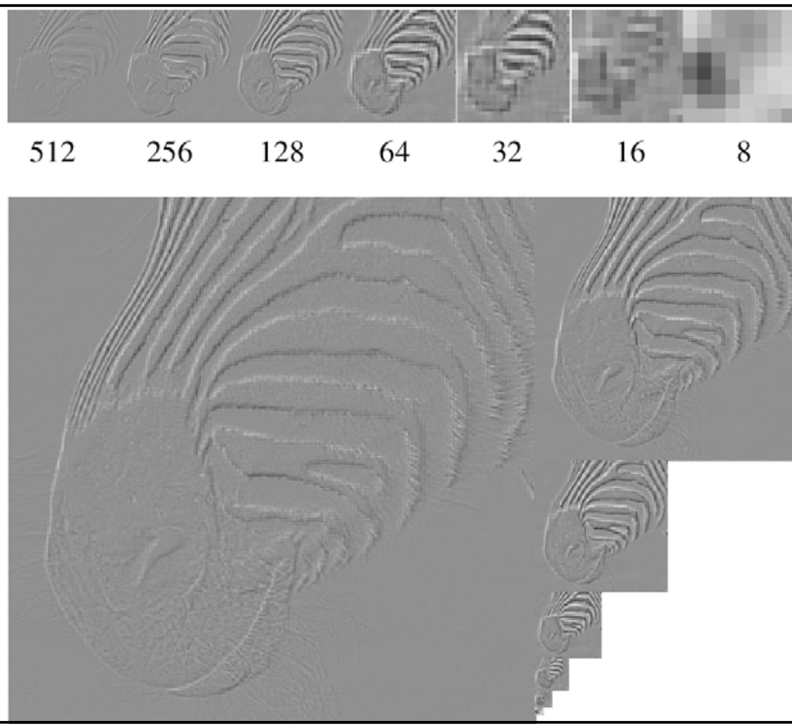
Laplacian Pyramid

- Building a Laplacian Pyramid:
  - Form a Gaussian pyramid
  - Set the coarsest layer of the Laplacian pyramid to be the coarsest level of the Laplacian pyramid
  - For each layer going from next to coarsest to finest (top to bottom):
    - Obtain this layer by upsampling the coarser layer and subtracting it from this layer of the Gaussian pyramid.

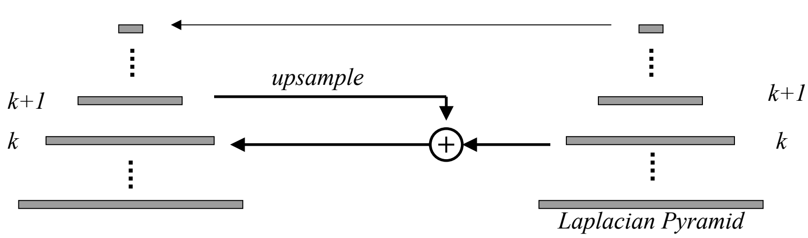
$$P_{\text{Laplacian}}(I)_m = P_{\text{Gaussian}}(I)_m$$
$$P_{\text{Laplacian}}(I)_k = P_{\text{Gaussian}}(I)_k - S^{\uparrow}(P_{\text{Gaussian}}(I)_{k+1})$$



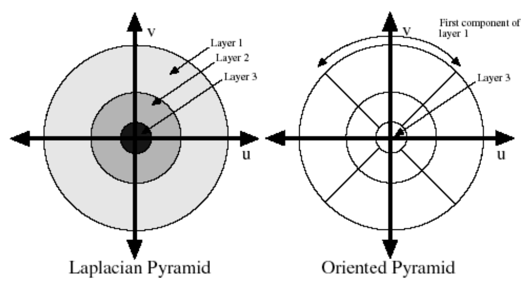




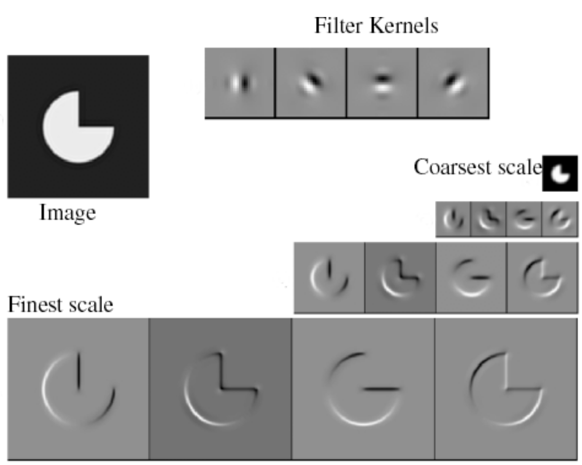
- Synthesis: Obtaining an Image from a Laplacian Pyramid:
  - Start at the coarsest layer
  - For each layer from next to coarsest to finest
    - Upsample the current image and add the current layer to the result

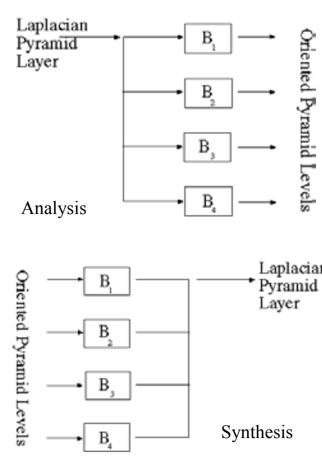


- Laplacian pyramid layers are band-pass filters.
- Laplacian pyramid is orientation independent
- Apply an oriented filter to determine orientations at each layer
- Look into spatial frequency domain:



Oriented Pyramids

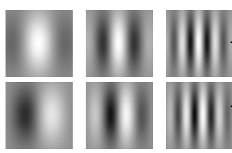




Reprinted from "Shiftable MultiScale Transforms," by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE

Gabor Filters

- Fourier coefficients depend on the entire image (Global): We lose spatial information.
- Objective: Local Spatial Frequency Analysis
- Gabor kernels: look like Fourier basis multiplied by a Gaussian
  - The product of a symmetric Gaussian with an oriented sinusoid
  - Gabor filters come in pairs: symmetric and antisymmetric
  - Each pair recover symmetric and antisymmetric components in a particular direction.
  - (k_x,k_y) :the spatial frequency to which the filter responds strongly
  - σ: the scale of the filter. When σ = infinity, similar to FT
- We need to apply a number of Gabor filters at different scales, orientations, and spatial frequencies.



$$G_{symmetric}(x,y) = \cos(k_x x + k_y y)\exp-\left\{\frac{x^2+y^2}{2\sigma^2}\right\}$$

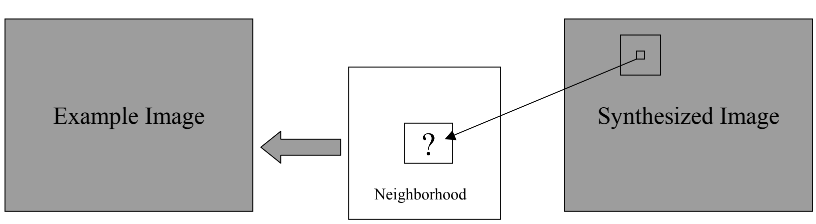
$$G_{antisymmetric}(x,y) = \sin(k_x x + k_y y)\exp-\left\{\frac{x^2+y^2}{2\sigma^2}\right\}$$

Final texture representation

- Form an oriented pyramid (or equivalent set of responses to filters at different scales and orientations).
- Square the output
- Take statistics of responses
  - e.g. mean of each filter output (are there lots of spots)
  - std of each filter output
  - mean of one scale conditioned on other scale having a particular range of values (e.g. are the spots in straight rows?)

Texture synthesis

- Variety of approaches.
- Example: Synthesis by Sampling Local Models: Efros and Leung 1999 (Nonparametric texture matching)
  - Use image as a source of probability model
  - Choose pixel values by matching neighborhood, then filling in



Find Matching Image Neighborhood and chose value uniformly randomly from these matches

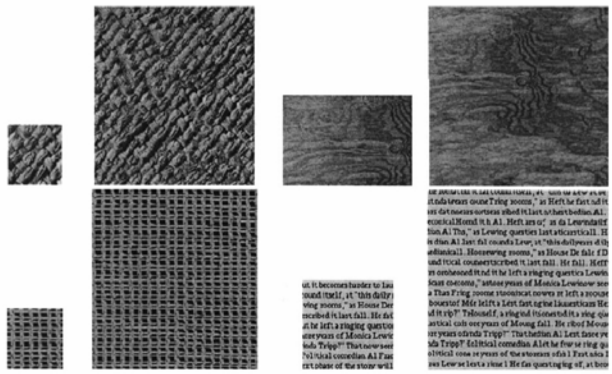


Figure from Texture Synthesis by Non-parametric Sampling, A. Efros and T.K. Leung, Proc. Int. Conf. Computer Vision, 1999 copyright 1999, IEEE

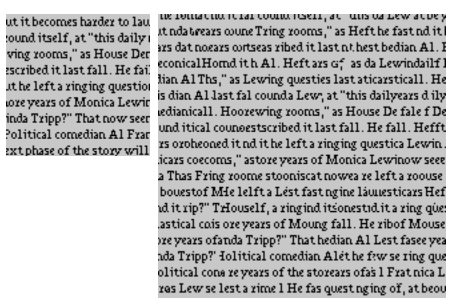
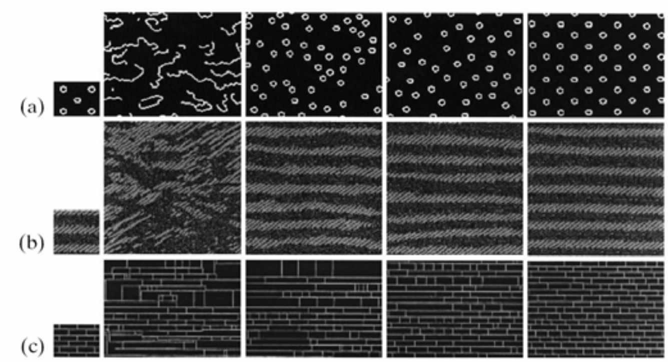


Figure from Texture Synthesis by Non-parametric Sampling, A. Efros and T.K. Leung, Proc. Int. Conf. Computer Vision, 1999 copyright 1999, IEEE

- The size of the image neighborhood to be matched makes a significant difference

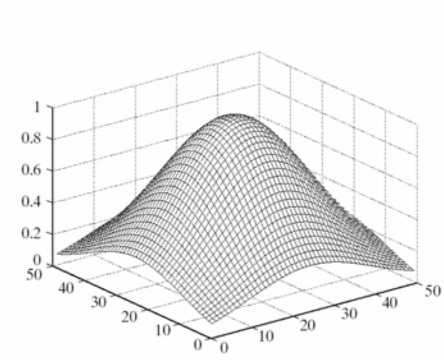


Variations

- Texture synthesis at multiple scales
- Texture synthesis on surfaces
- Texture synthesis by tiles
- "Analogous" texture synthesis

Shape from Texture

- Texture looks different depending on the viewing angle.
- Texture is a good cue for shape and orientation
- Humans are very good at that.



Sources

- Forsyth and Ponce, Computer Vision a Modern approach: chapter 9.
- L. G. Shapiro and G. C. Stockman “Computer Vision”, Prentice Hall 2001.
- R. Gonzalez and R.E. Woods, “Digital Image Processing”, 2002.
- Slides by
  - D. Forsyth @ UC Berkeley
  - G.C. Stockman @MSU