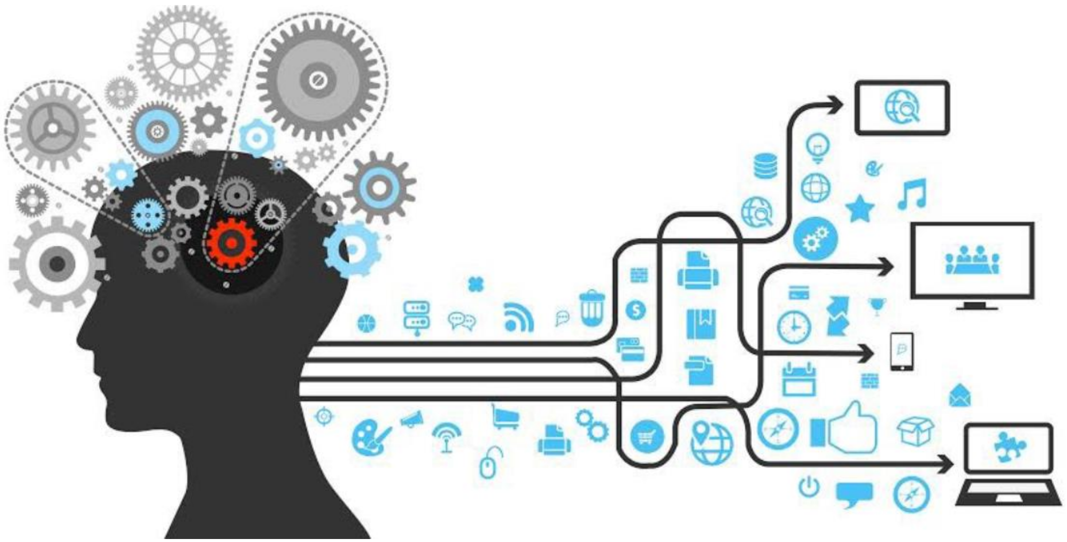
Computer Vision

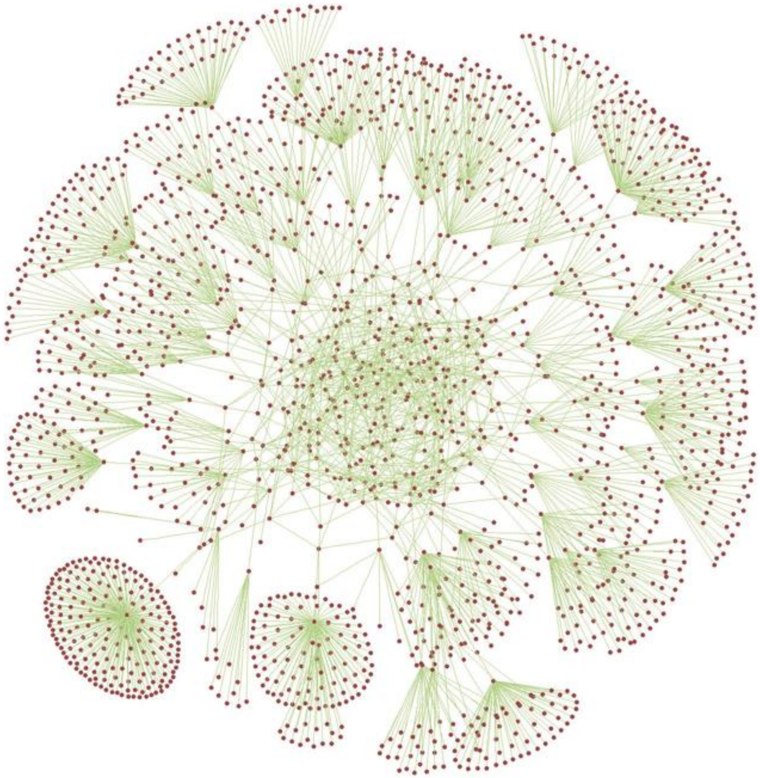
Early vision: Just one image

School of Electronic &
Electrical Engineering

Sungkyunkwan University

Hyunjin Park

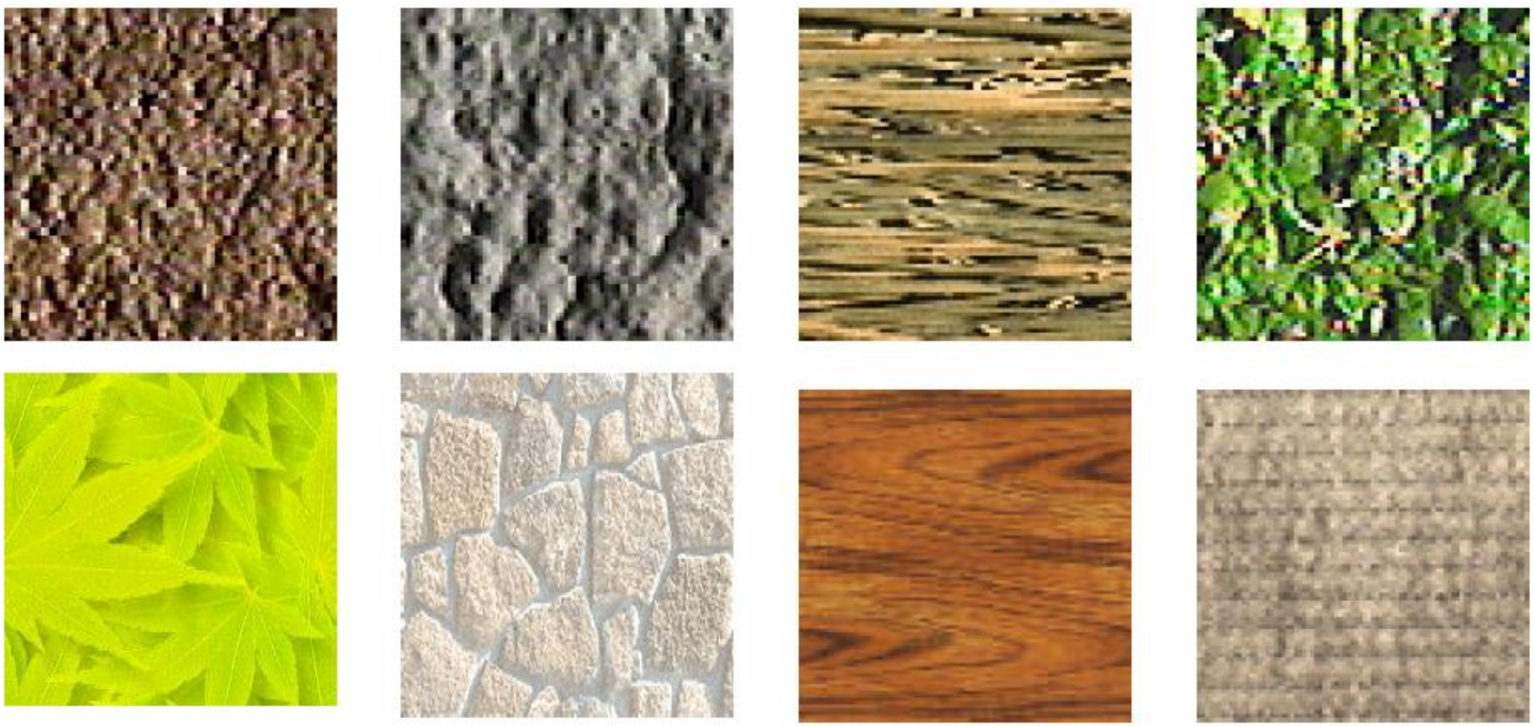




Texture

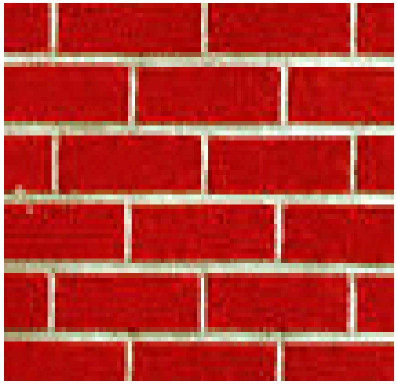
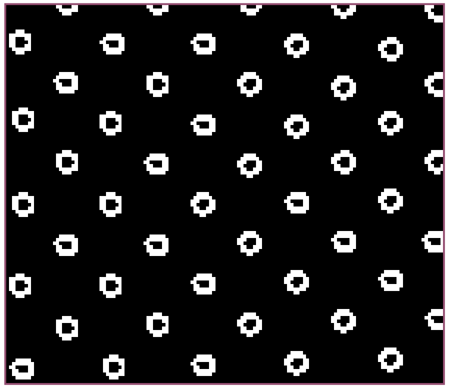
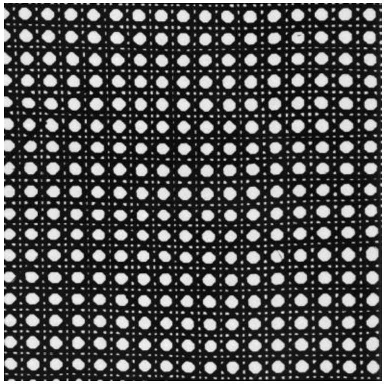
Texture

- Images of most natural objects exhibit visual texture
- Texture provides important visual cue



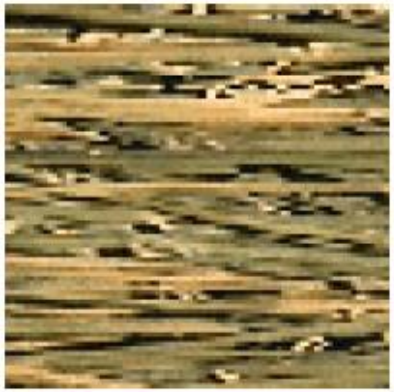
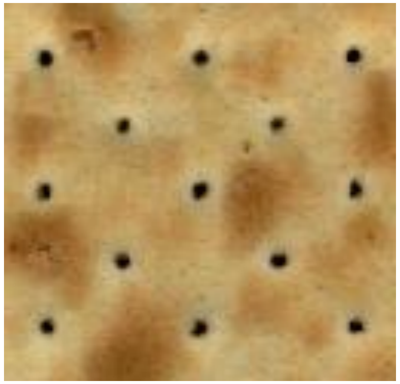
Texture

- Regular patterns

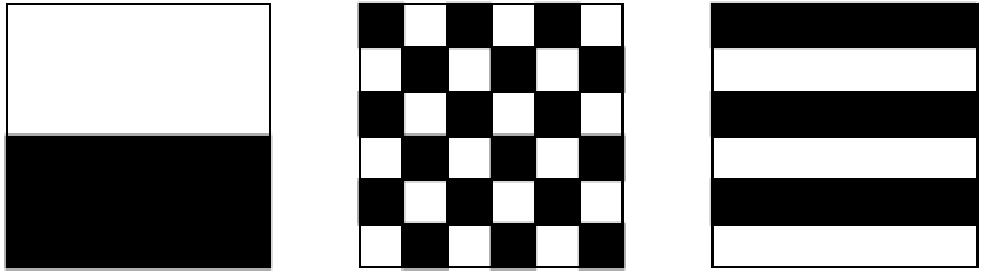
- Random patterns



Texture

- Texture
  - A ***feature*** used to partition images into regions of interest and to classify those regions
  - Provides information in the ***spatial arrangement*** of colors or intensities
  - Characterized by the ***spatial distribution*** of intensity levels in a neighborhood
  - A repeating pattern of ***local variations*** in image intensity

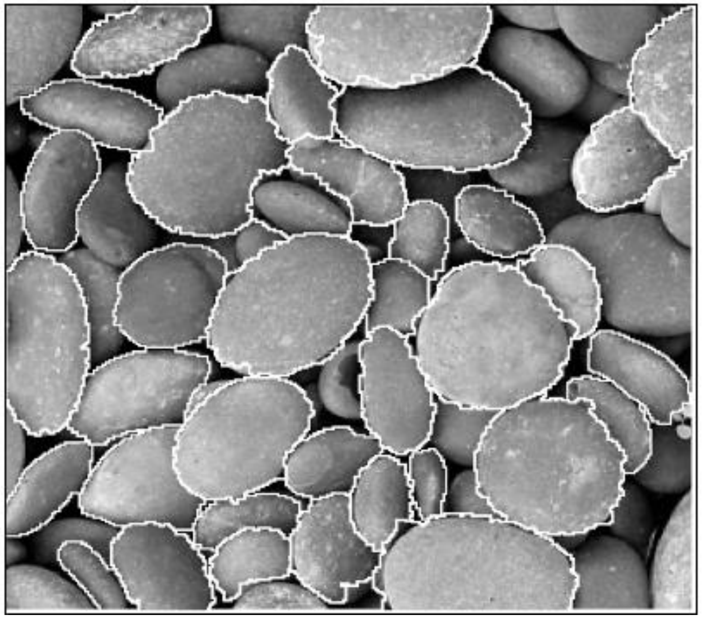


block pattern checkerboard striped pattern

Three different images with the same intensity distribution,
but with different textures

Texture

- Approaches for defining texture
  - ***Structural*** approach
    - Texture is a set of primitive texels in some regular or repeated relationship
    - Work well for man-made, regular patterns

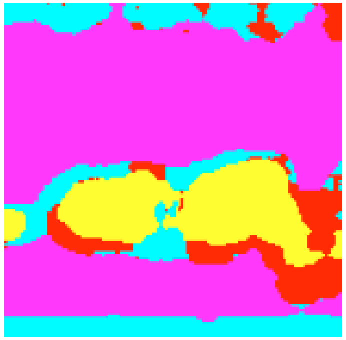


Texture

- Approaches for defining texture
  - ***Statistical*** approach
    - Texture is a quantitative measure of the arrangement of intensities in a region → Set of measurements is called a "feature vector"
    - More general and easier to compute
    - Used more often in practice

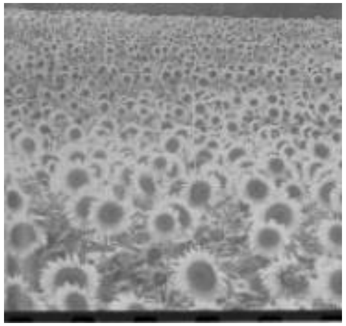
Texture

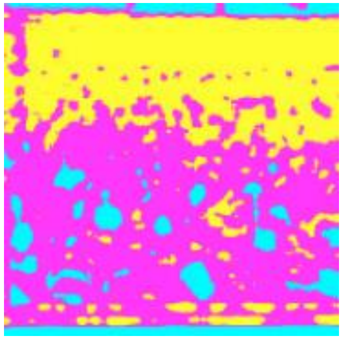
- Key issue: Representing texture
- Standard problems
  - Texture segmentation
    - Breaking an image into components within which the texture is constant



segmentation into 4 clusters

original image



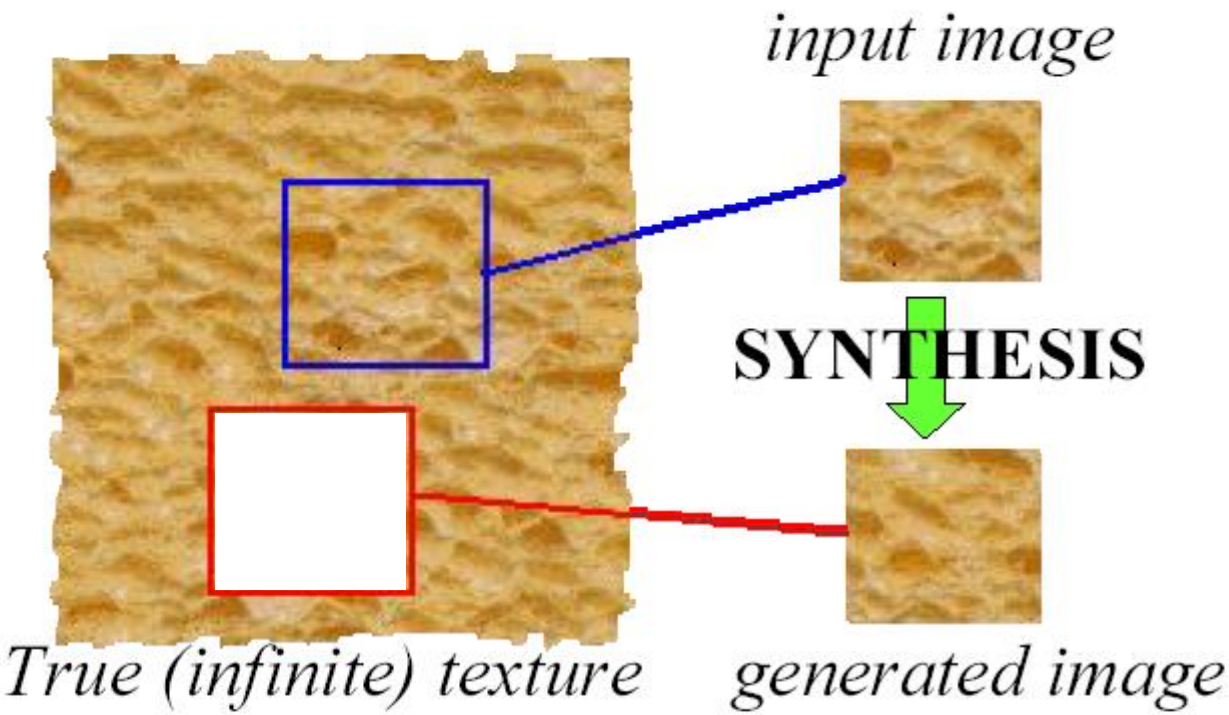


segmentation into 3 clusters

original image

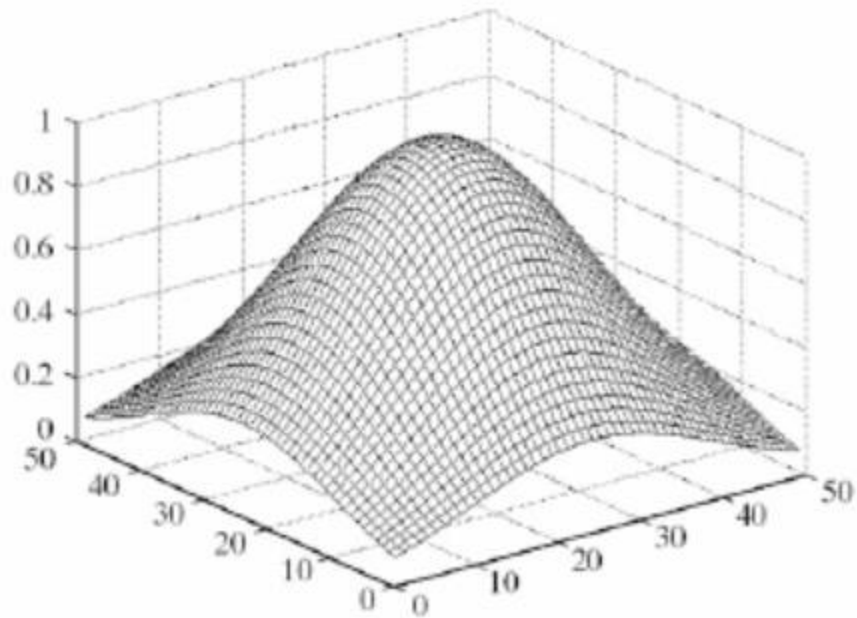
Texture

- Key issue: Representing texture
- Standard problems
  - Texture synthesis
    - Construct large regions of texture from small example images



Texture

- Key issue: Representing texture
- Standard problems
  - Shape from texture
    - Recover surface orientations or surface shape from image texture



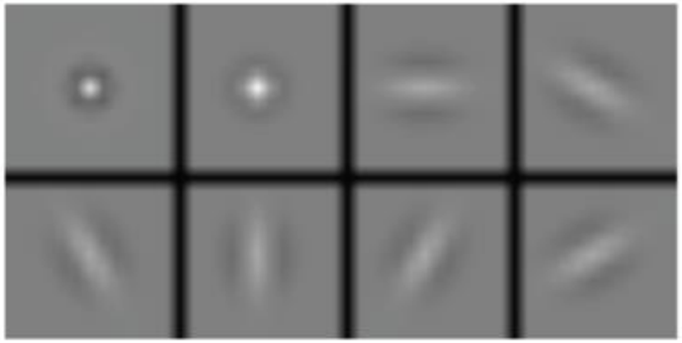
Representing textures

- Textures
  - Made up of quite stylized subelements, repeated in meaningful ways
- Representation
  - Find the subelements and represent their statistics
- What are the subelements and how do we find them?
  - Apply filters and look at the magnitude of the response

Spatial filtering approach

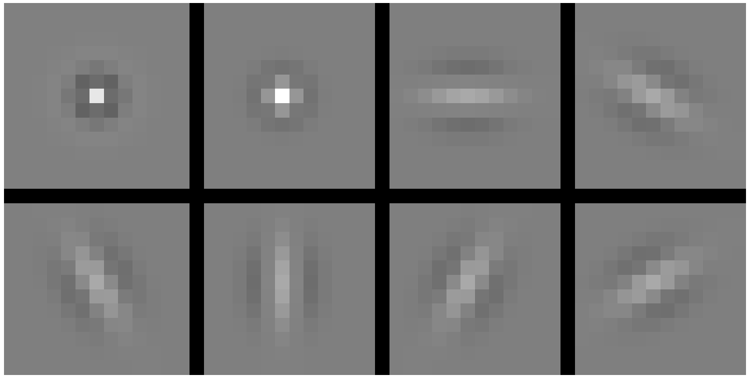
- Spot filters
  - Gaussian or weighted sums of concentric Gaussians
  - Respond strongly to small regions that differ from their neighbors
  - Detect non-oriented structure

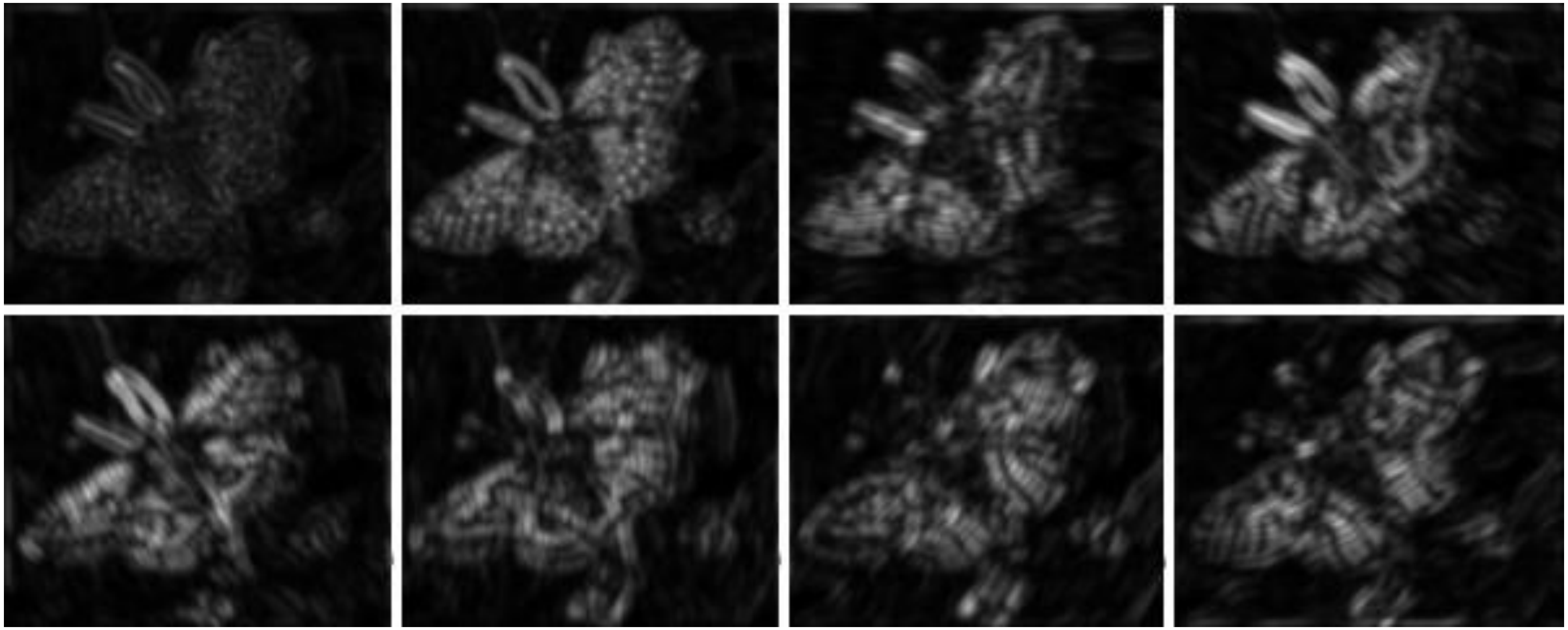
- Bar filters
  - Differentiating oriented Gaussians
  - Collection of oriented bar filters, at different orientations, scales, and phases
  - Tend to respond to oriented structure



Spatial filtering approach

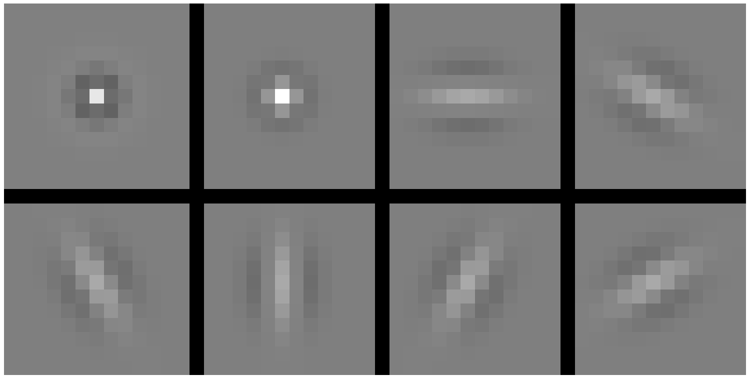
- Spot and bar filters: Fine scale

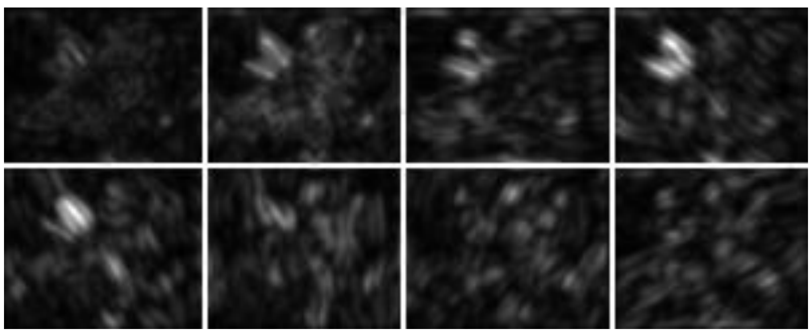




Spatial filtering approach

- Spot and bar filters: Coarse scale





The size of the filter has stayed fixed, but the image is half the original size

Spatial filtering approach

- How many filters?
  - Using more filters leads to a more detailed representation of the image
  - But, computationally expensive

- Number of orientations?
  - Varies from application to application and does not seem to matter much, as long as there are at least about six orientations

- At what scale?
  - Overall distribution of texture elements depends on the scale of the filter
  - Small window on a zebra: contain constant black/white region
  - Large window: contain some background as well as the relevant texture

Choice of scale

- Steps for choosing the scale
  - Start with a small window at the point of interest
  - Increase the window size until it does not cause a significant change in certain criterion

- May not result in unique scale
  - Depends on the initial window size

Scaled representation

- Use a multiresolution representation: Image Pyramid

- Gaussian pyramid
  - Low-pass filter → Response is redundant
  - A coarse level layer of the Gaussian pyramid predicts the appearance of the next fine layer

- Laplacian pyramid
  - Preserve differences between upsampled Gaussian pyramid level and Gaussian pyramid level
  - Band-pass filter → Each level represent spatial frequencies (largely) unrepresented at other levels

Laplacian pyramid

- Building a Laplacian pyramid from an image
  - Form a Gaussian pyramid
  - Set the coarsest layer of the Laplacian pyramid to be the coarsest level of the Gaussian pyramid

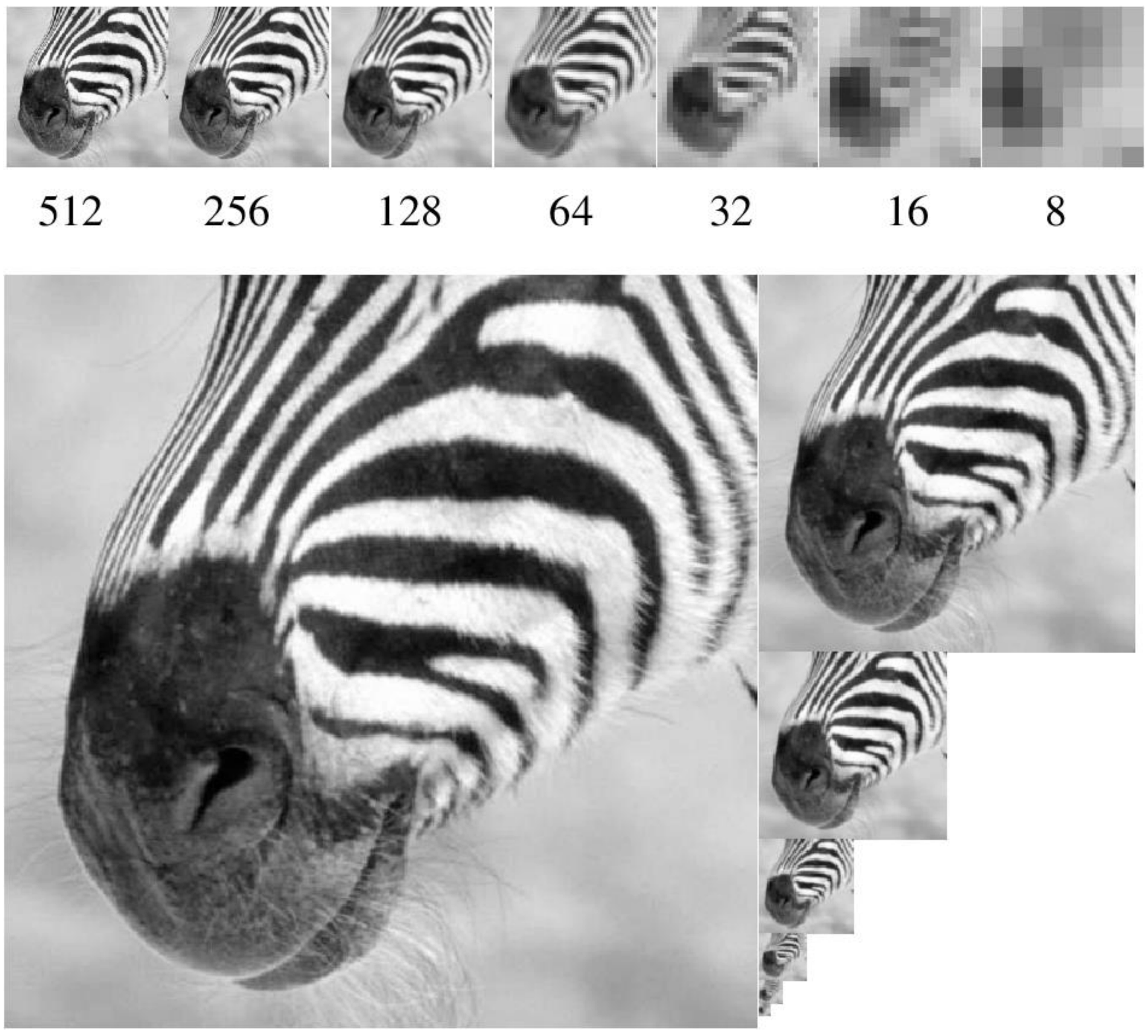
$$P_{Laplacian}(I)_m = P_{Gaussian}(I)_m,\ m\text{: coarsest level}$$

  - For each layer, going from next to coarsest to finest
    - Obtain this layer of the Laplacian pyramid by upsampling the next coarser layer, and subtracting it from this layer of the Gaussian pyramid

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

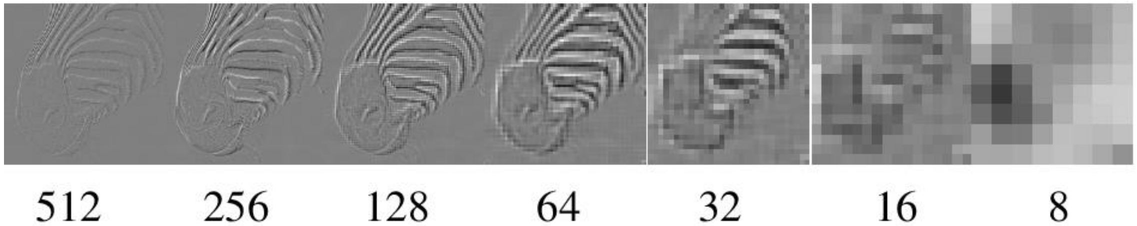
Laplacian pyramid

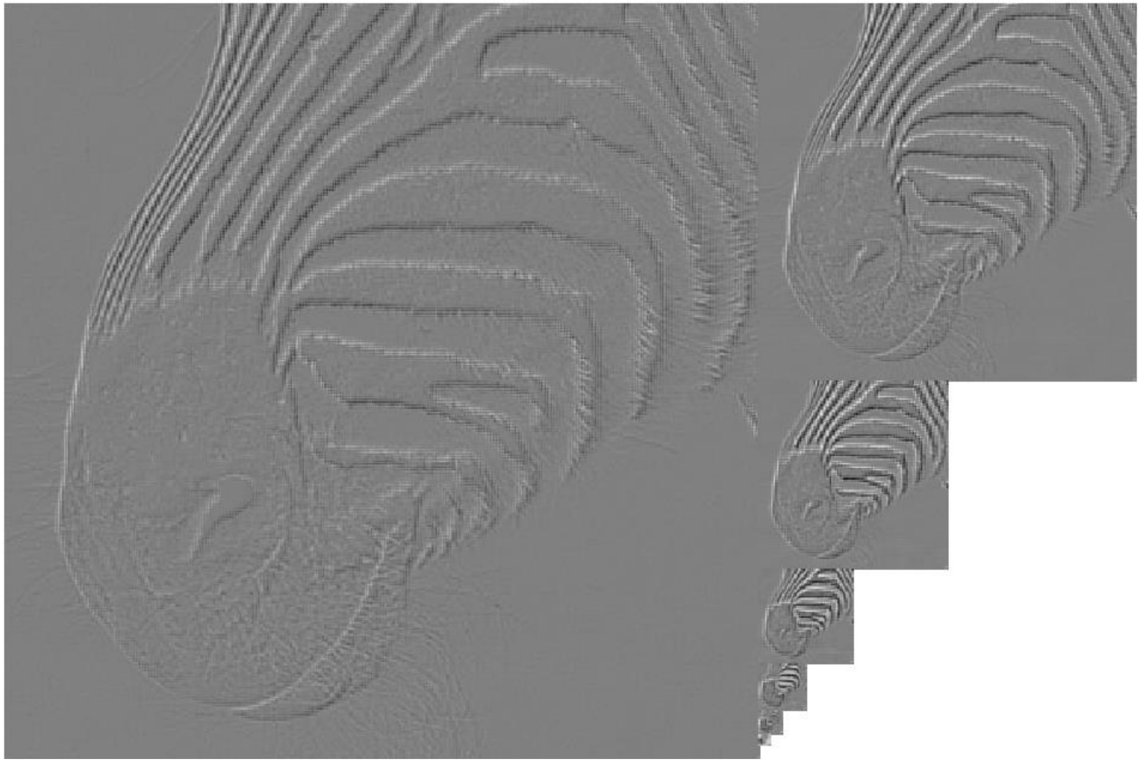
- Gaussian pyramid



Laplacian pyramid

- Laplacian pyramid



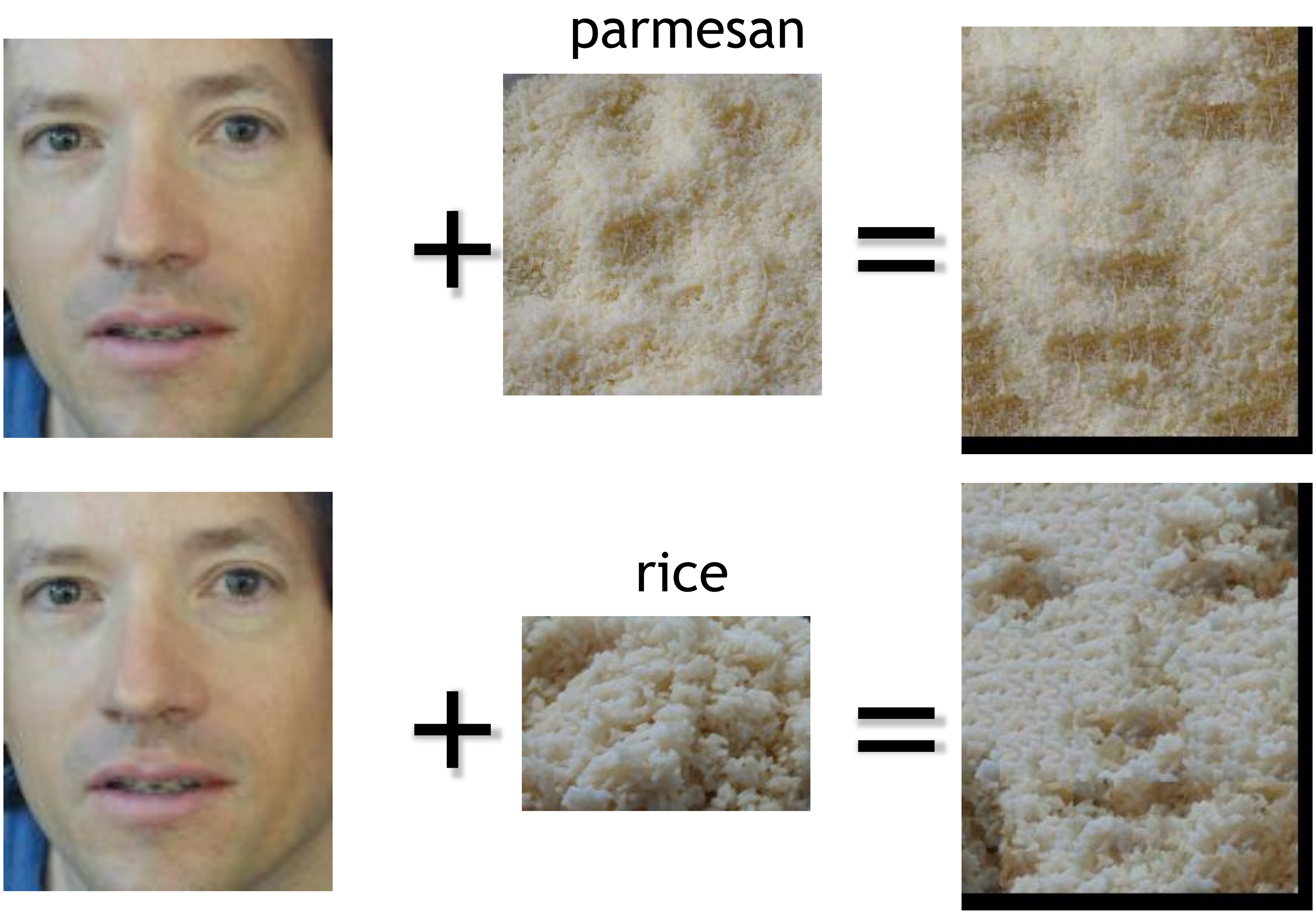


Synthesis

- Synthesis: Obtaining an image from its Laplacian pyramid
- Recover Gaussian pyramid from the Laplacian and then take the finest scale of the Gaussian pyramid
  - Start with the coarsest scale
  - Next to the coarsest scale of the Gaussian pyramid
    - Take the coarsest scale and upsample it
    - Add the next to the coarsest scale of the Laplacian pyramid
  - And so on up the scales

Texture transfer

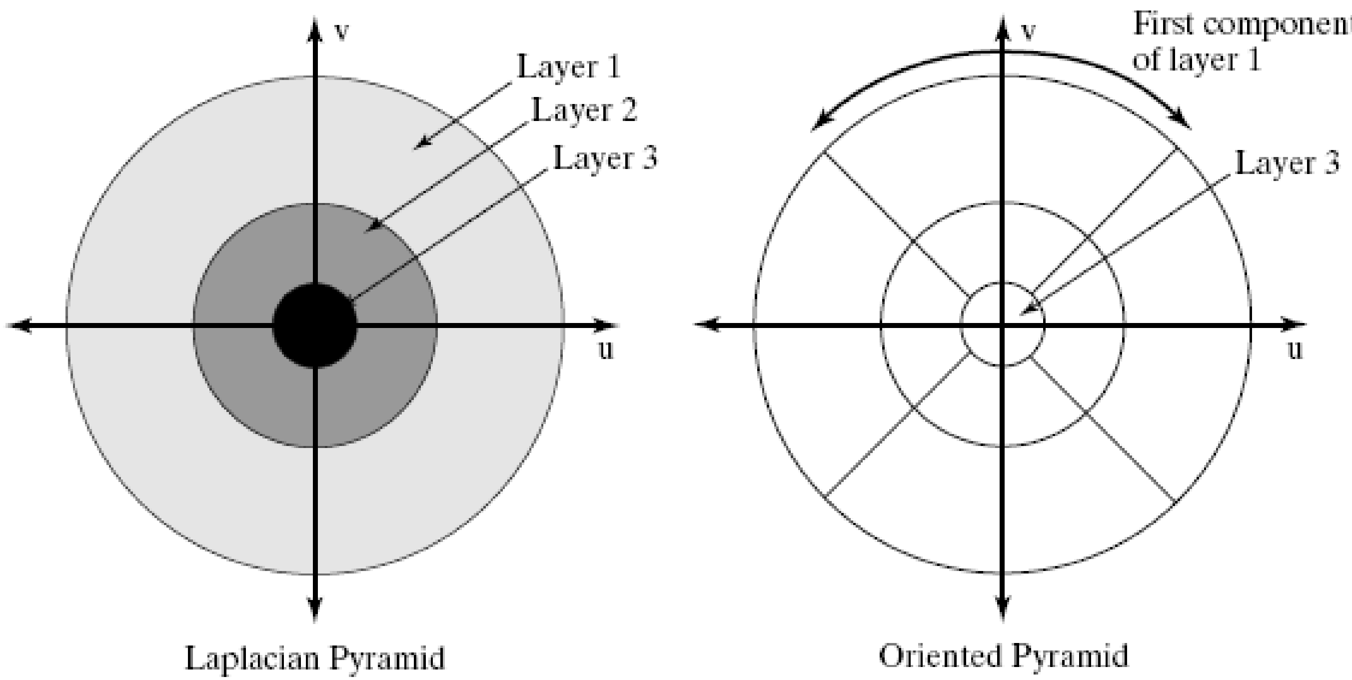
- Take the texture from one object and paint it onto another object



Pyramids in FT space

- Recall FT: $F\big(g(x,y)\big)(u,v) = \iint_{R^2} g(x,y)e^{-2\pi(ux+vy)}dxdy$

$$frequency = \sqrt{u^2+v^2}, \qquad \theta = \tan^{-1}\left(\frac{v}{u}\right)$$



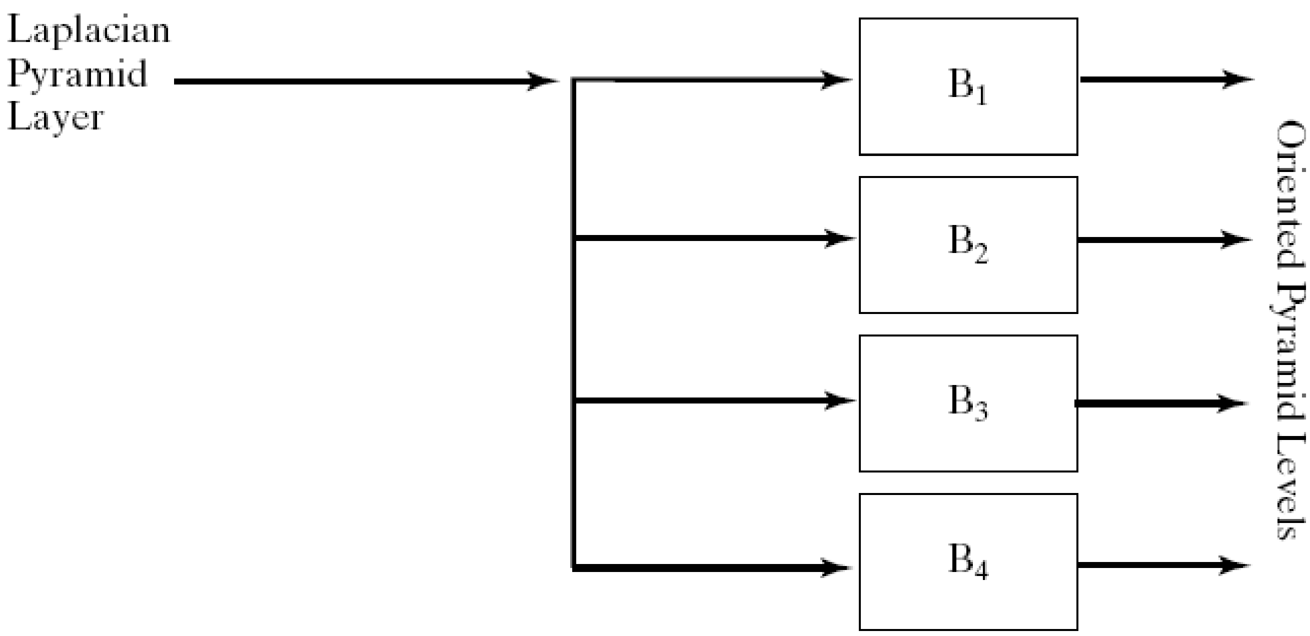
Oriented pyramids

- Laplacian pyramid is orientation independent
- Apply an oriented filter to determine orientations at each layer

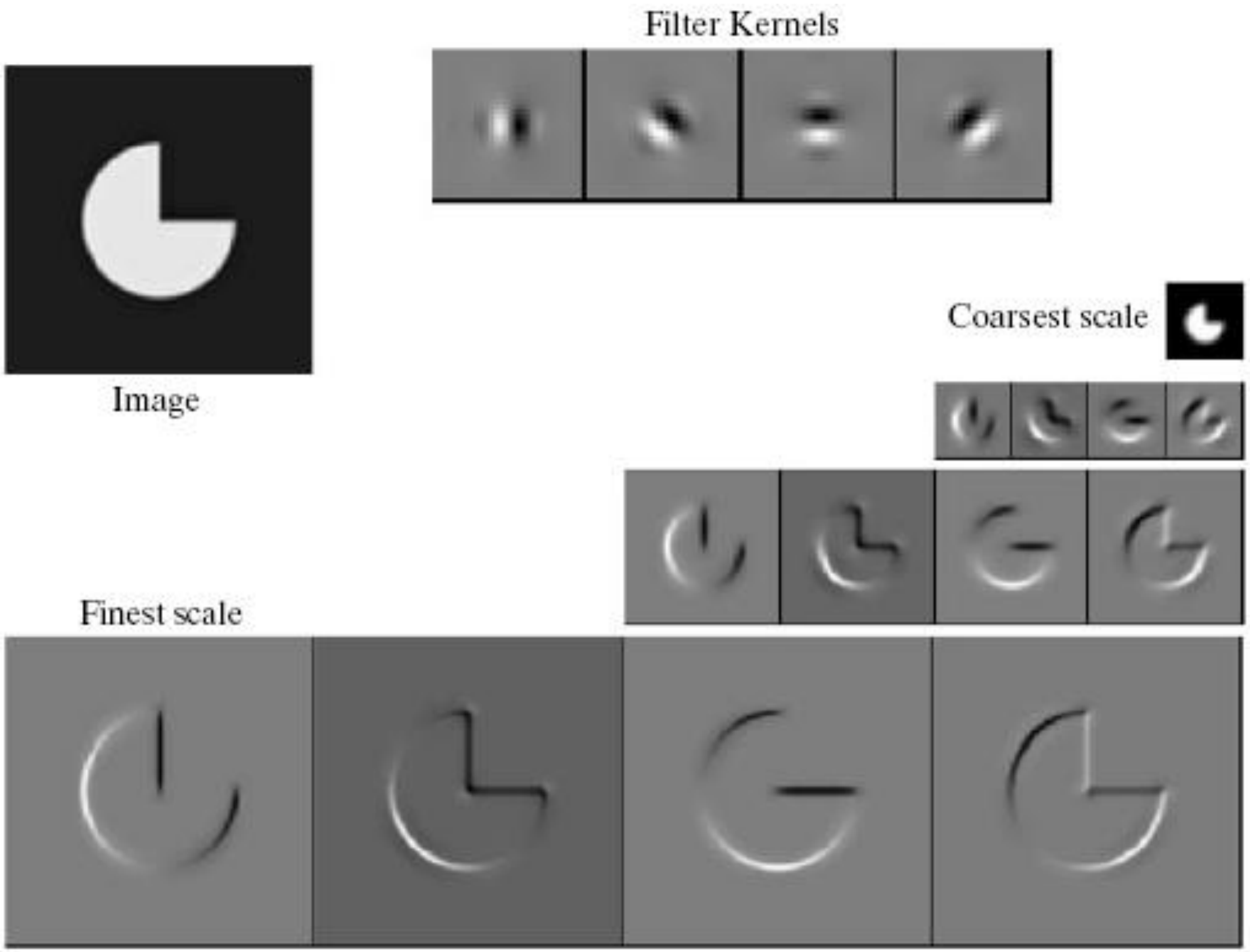
- Gabor filters
  - Product of a symmetric Gaussian with an oriented sinusoid
    - $G_{symmetric}(x, y) = \cos\left(k_x x + k_y y\right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
    - $G_{antisymmetric}(x, y) = \sin\left(k_x x + k_y y\right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
    - $\left(k_x, k_y\right)$: determines the spatial frequency to which the filter responds most strongly
    - σ: scale of the filter

Oriented pyramids

- A Laplacian pyramid does not contain enough information to reason about image texture, because there is no explicit representation of orientation
- Apply an oriented filter to determine orientations at each layer



Oriented pyramids

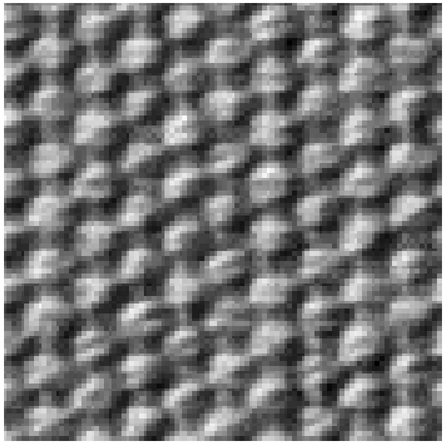


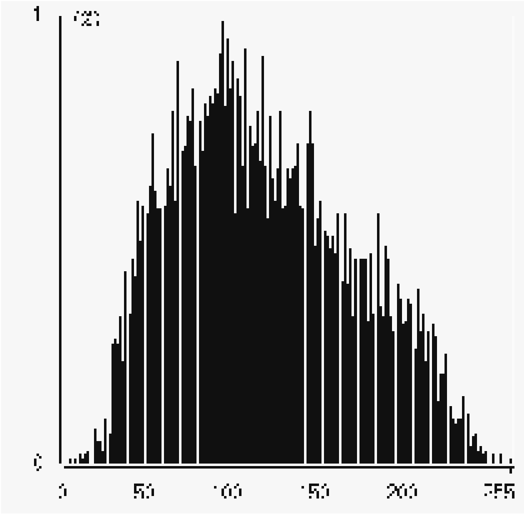
Final texture representation

- Form an oriented pyramid
- Square the output
- Take statistics of response
  - E.g.)
    - Mean of each filter output
    - Standard deviation of each filter output
    - Mean of one scale conditioned on other scale having a particular range of values

Final texture representation

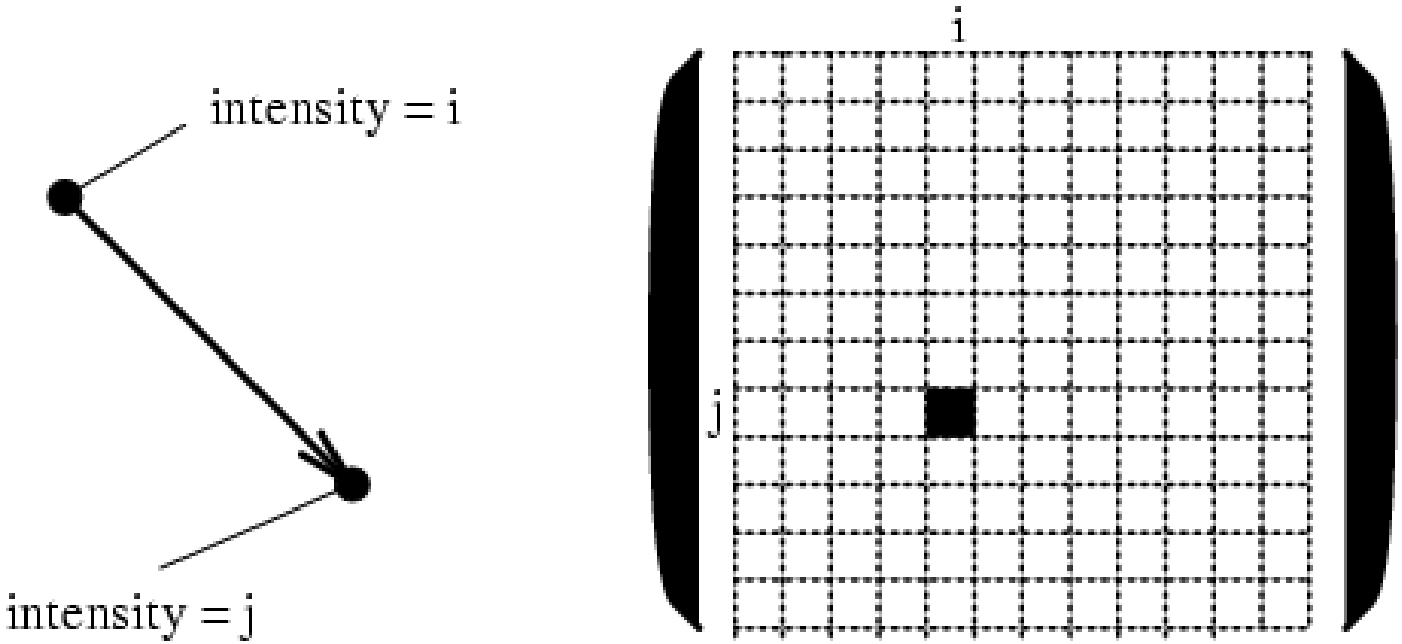
- Histograms
  - Intensity probability distribution
  - Captures global brightness information in a compact, but incomplete way
  - Doesn't capture spatial relationships
- Example
  - mean, standard deviation, median, range, variance, skewness, kurtosis,…





Final texture representation

- Co-occurrence matrices ($\mathbf{P_d}$)
  - Probability distributions for intensity pairs
  - Contains information on some aspects of the spatial configurations



Final texture representation

- Co-occurrence matrices ($\mathbf{P_d}$)
  - Illustration with a 4 x 4 image I and three different spatial configurations

$$\begin{array}{|cccc|}\hline 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ \hline\end{array}$$

Image I

$$C_{(0,1)}: \quad \begin{array}{c|ccc|} i \backslash j & 0 & 1 & 2 \\ \hline 0 & 4 & 0 & 2 \\ 1 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ \hline \end{array}$$

i	j

$$C_{(1,0)}: \quad \begin{array}{c|ccc|} i \backslash j & 0 & 1 & 2 \\ \hline 0 & 4 & 0 & 2 \\ 1 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ \hline \end{array}$$

i
j

$$C_{(1,1)}: \quad \begin{array}{c|ccc|} i \backslash j & 0 & 1 & 2 \\ \hline 0 & 2 & 0 & 2 \\ 1 & 2 & 1 & 1 \\ 2 & 0 & 0 & 1 \\ \hline \end{array}$$

i	
	j

Final texture representation

- Co-occurrence matrices ($\mathbf{P_d}$)
  - The elements of $P_d[i,j]$ can be normalized by dividing each entry by the total number of pixel pairs
    - Normalized co-occurrence matrix: $N[i,j] = \frac{P[i,j]}{\sum \sum P[i,j]}$
  - Standard features derivable from a normalized co-occurrence matrix, N_d
    - $Energy = \sum_i \sum_j N_d^2[i,j]$
    - $Entropy = -\sum_i \sum_j N_d[i,j] \log_2(N_d[i,j])$
    - $Contrast = \sum_i \sum_j (i-j)^2 N_d[i,j]$
    - $Homogeneity = \sum_i \sum_j \frac{N_d[i,j]}{1+|i-j|}$
    - $Correlation = \frac{\sum_i \sum_j (i-\mu_i)(j-\mu_j) N_d[i,j]}{\sigma_i \sigma_j}$