南京大学
NANJING UNIVERSITY

计算机视觉表征与识别
Chapter 4: Template, Pyramid, and Filter Banks

王利民

媒体计算课题组

http://mcg.nju.edu.cn/

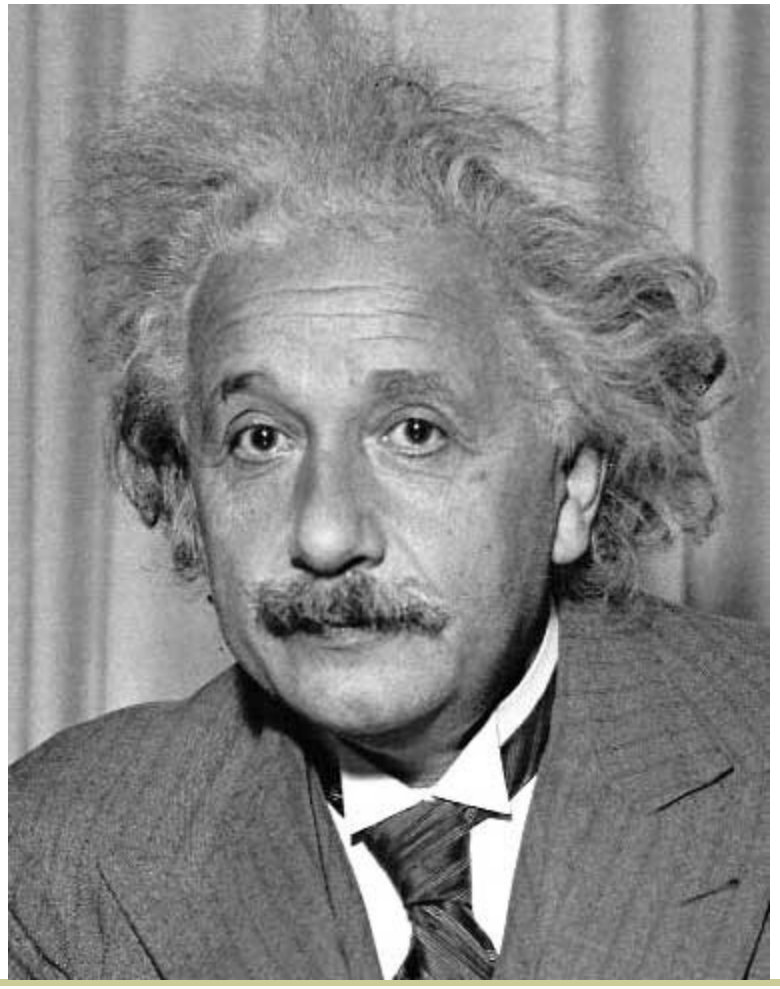
Today's class

- Template matching
- Gaussian pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian pyramids
  - Hybrid images
  - Application for image blending
- Steerable pyramids
  - Steerable filters
  - Orientation analysis
- Human vision system
  - Visual perception
- Filter Banks and Texture Analysis

Template matching

- Goal: find  in image

- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation

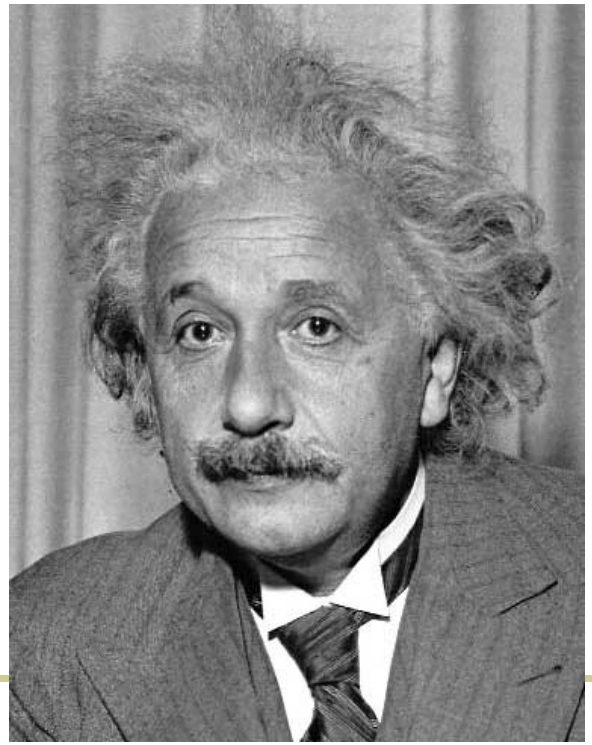


Matching with filters

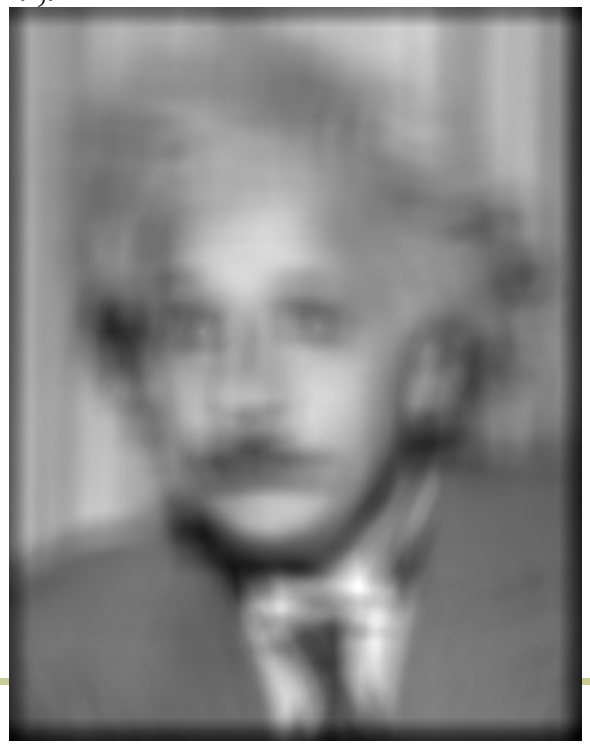
- Goal: find  in image
- Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

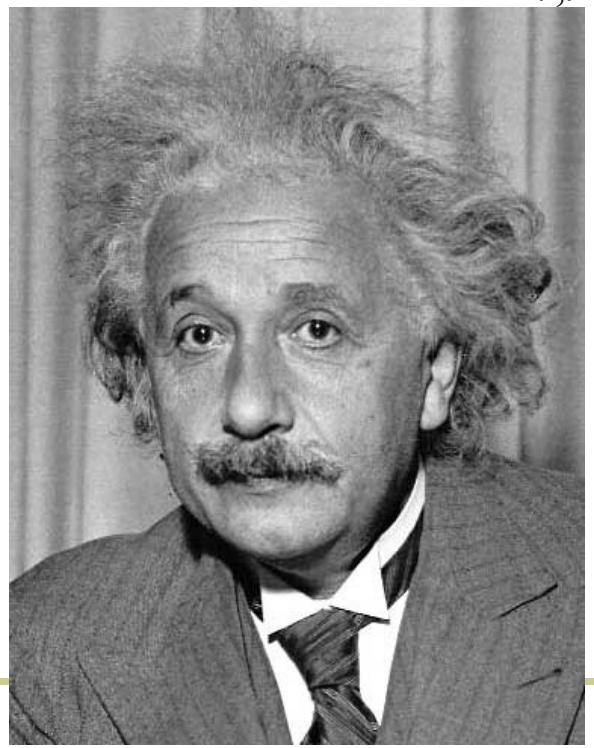
What went wrong?

Matching with filters

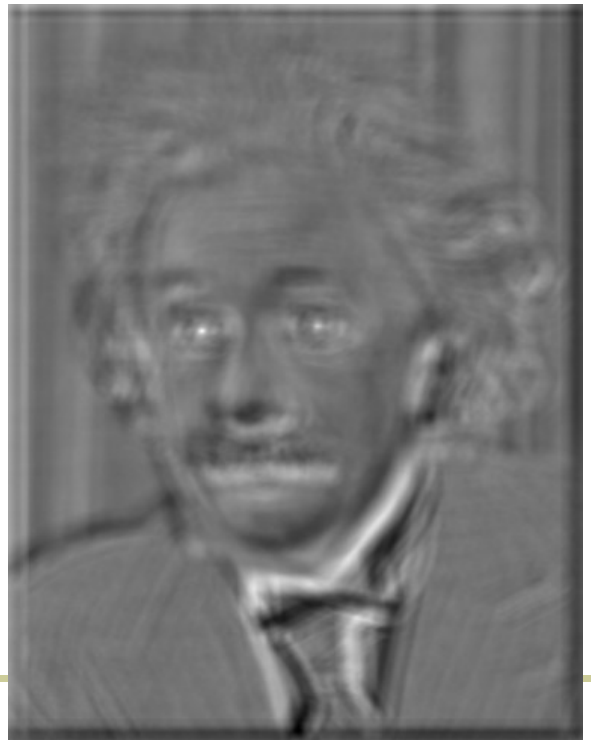
- Goal: find  in image
- Method 1: filter the image with zero-mean eye

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g})\,(f[m+k,n+l])$$

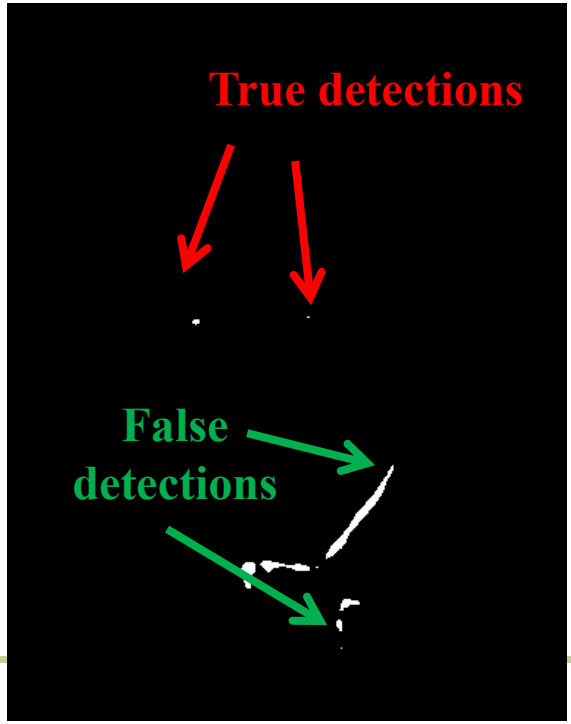
mean of template g



Input



Filtered Image (scaled)

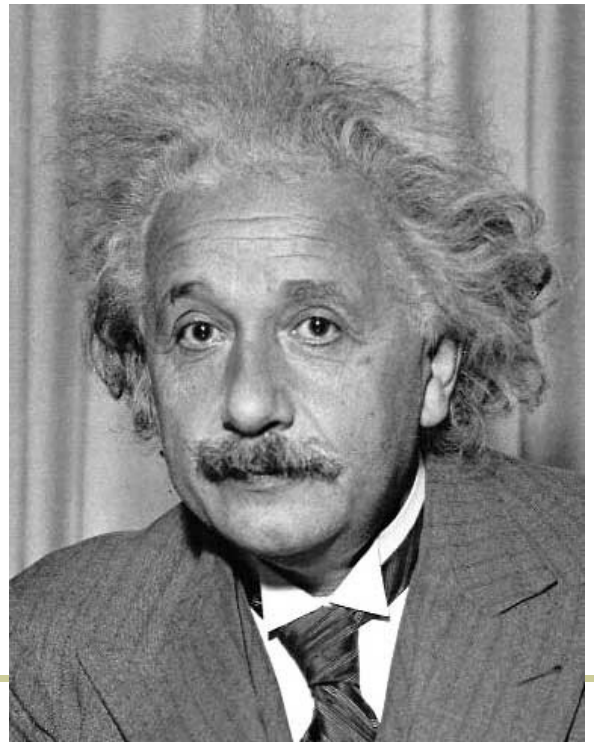


Thresholded Image

Matching with filters

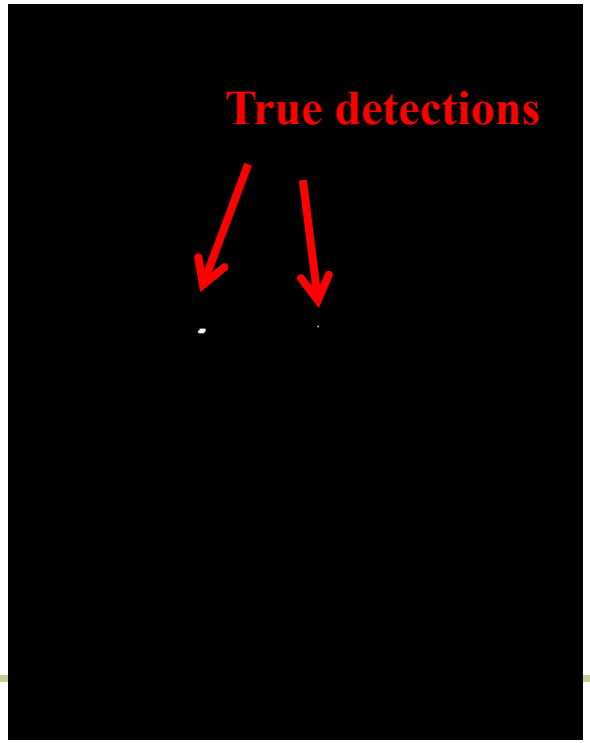
- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input

1- sqrt(SSD)



Thresholded Image

Matching with filters

Can SSD be implemented with linear filters?

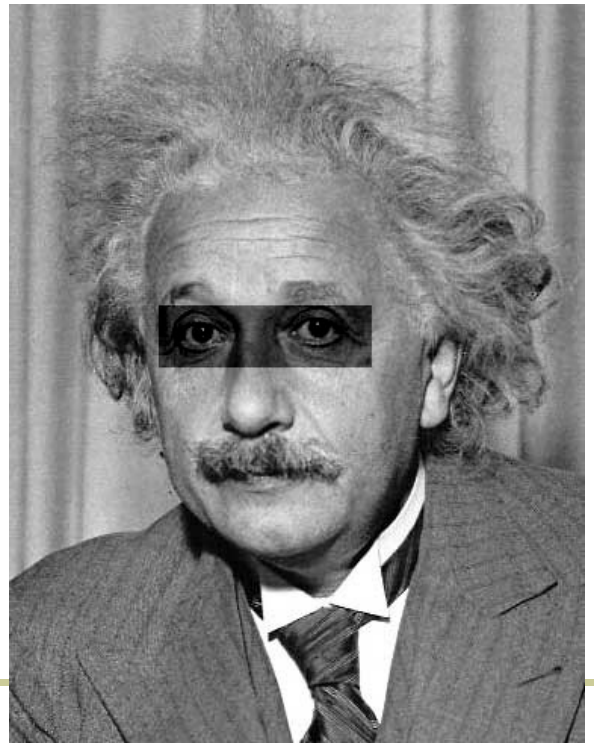
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

Matching with filters

- Goal: find  in image
- Method 2: SSD

What's the potential downside of SSD?

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input

1- sqrt(SSD)

Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation

mean template

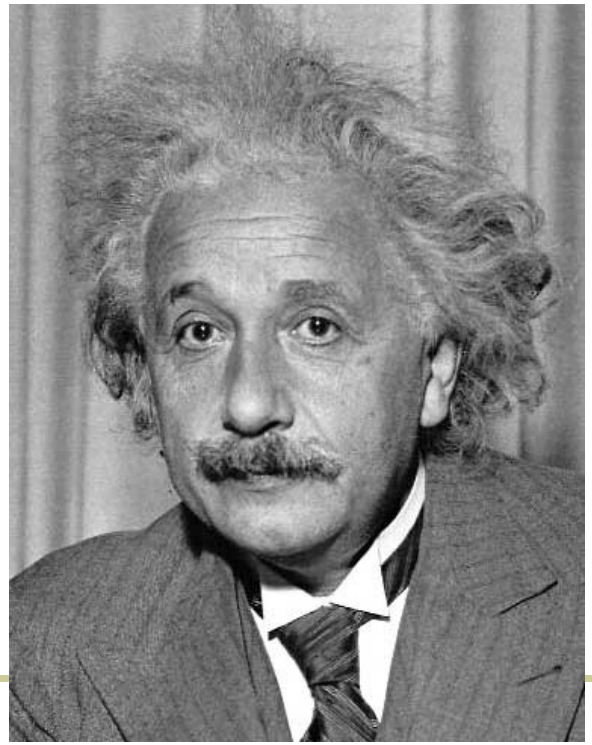
mean image patch

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

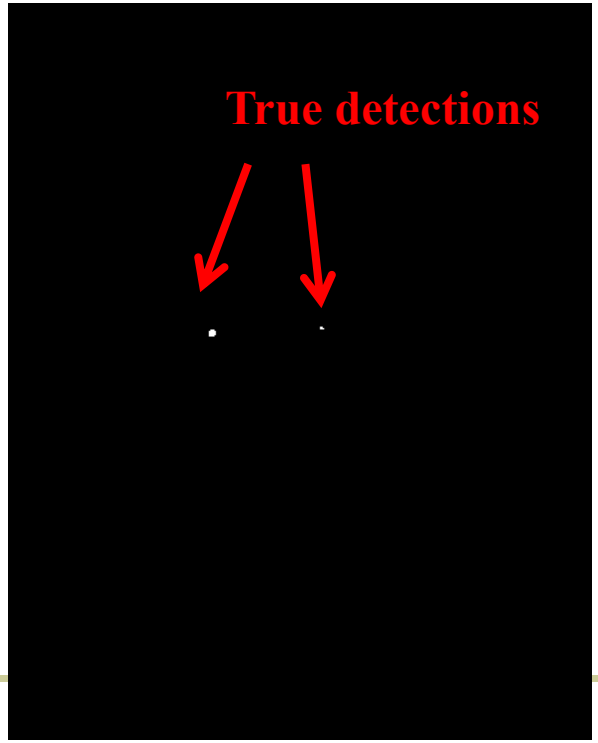
Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

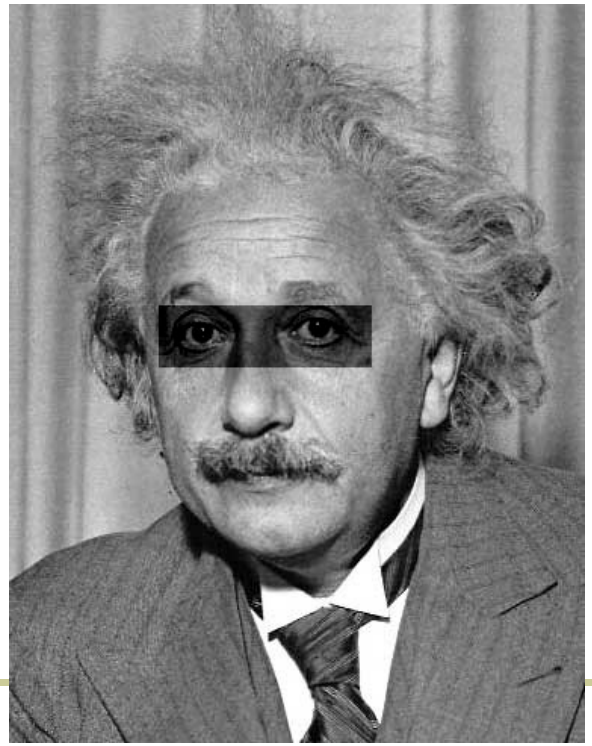
Normalized X-Correlation



Thresholded Image

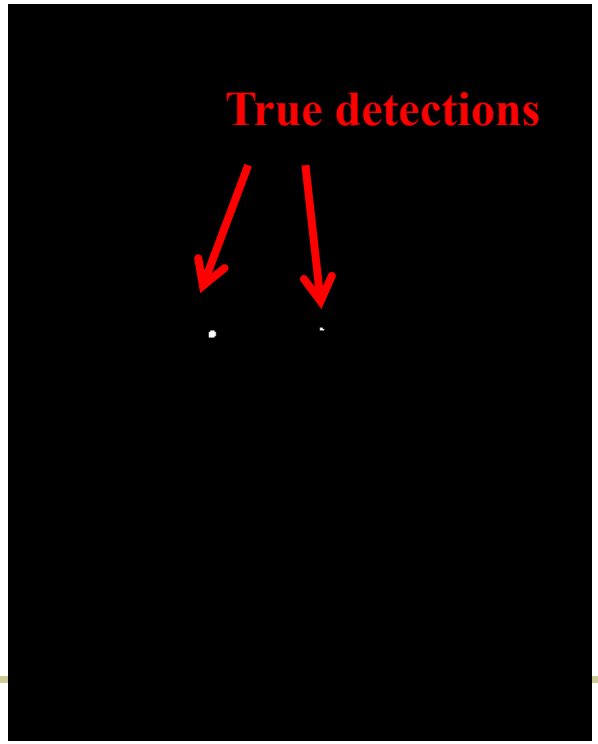
Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



Thresholded Image

Q: What is the best method to use?

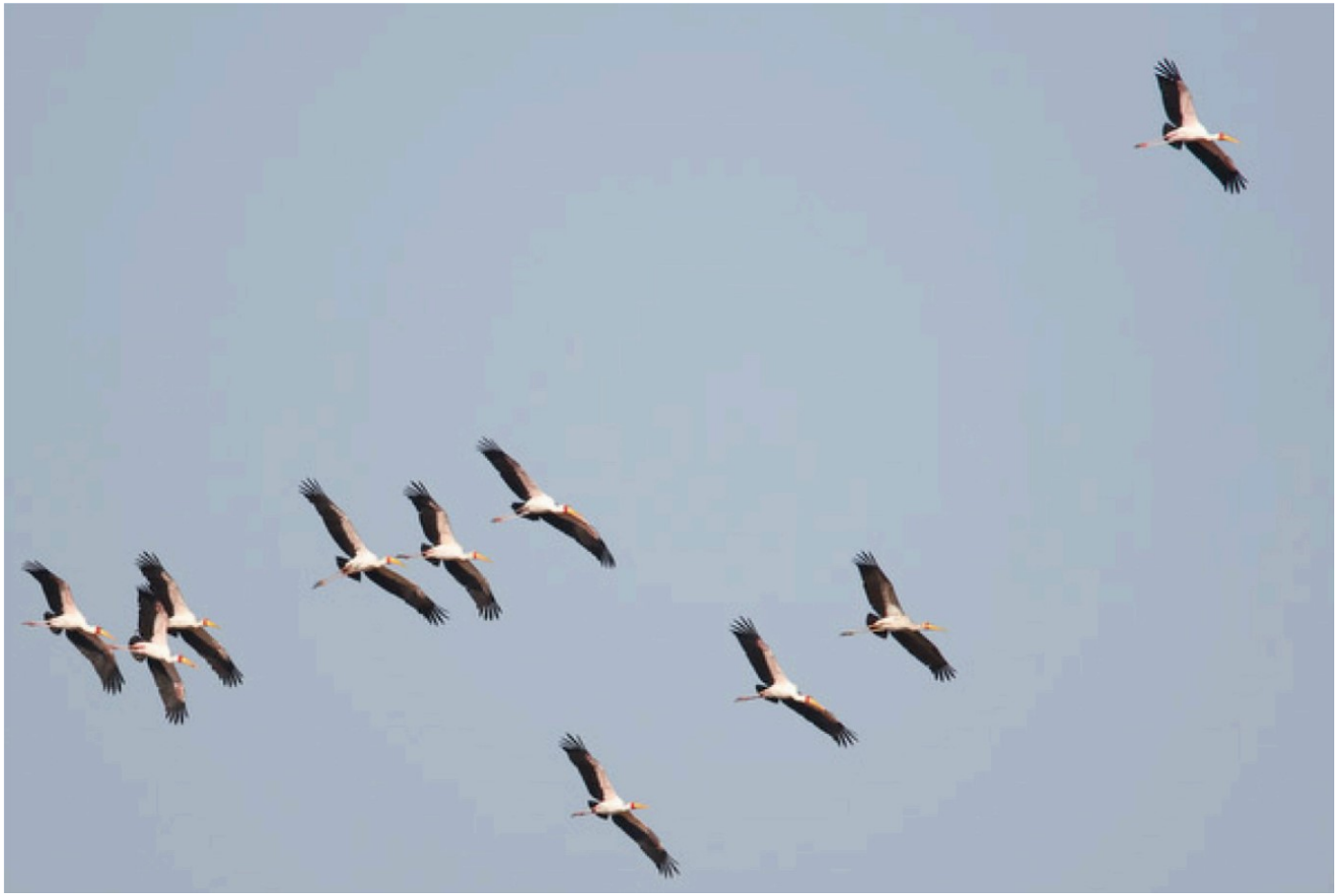
A: Depends

- Zero-mean filter: fastest but not a great matcher
- SSD: next fastest, sensitive to overall intensity
- Normalized cross-correlation: slowest, invariant to local average intensity and contrast

Today's class

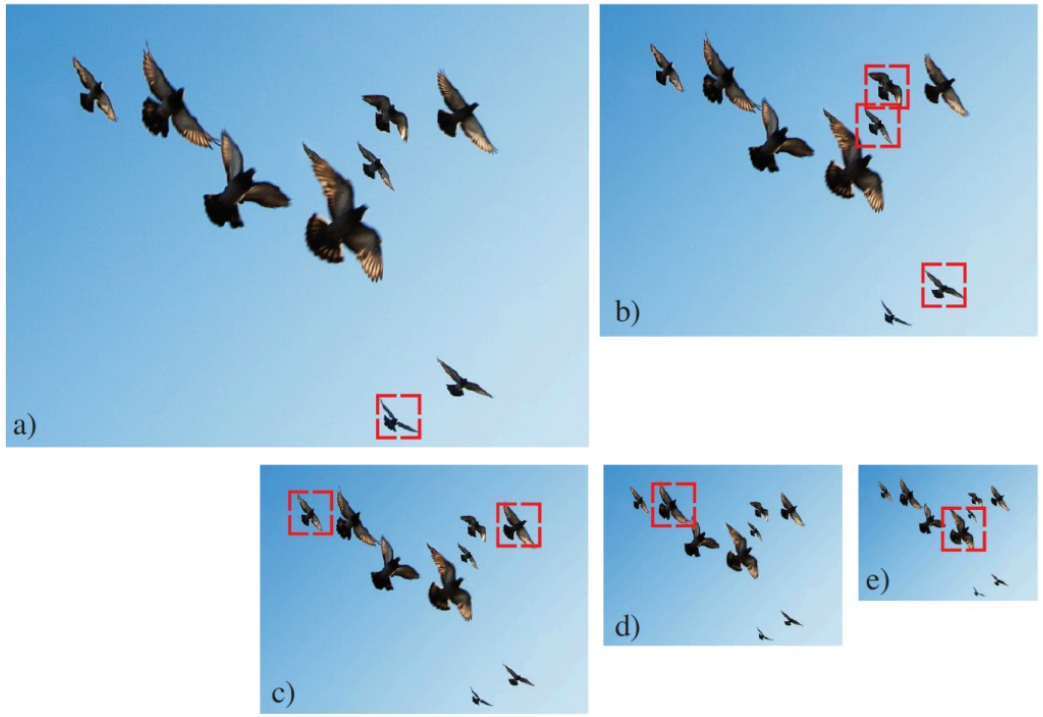
- Template matching
- **Gaussian pyramids**
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian pyramids
  - Hybrid images
  - Application for image blending
- Steerable pyramids
  - Steerable filters
  - Orientation analysis
- Human vision system
  - Visual perception
- Filter Banks and Texture Analysis

Translation invariance

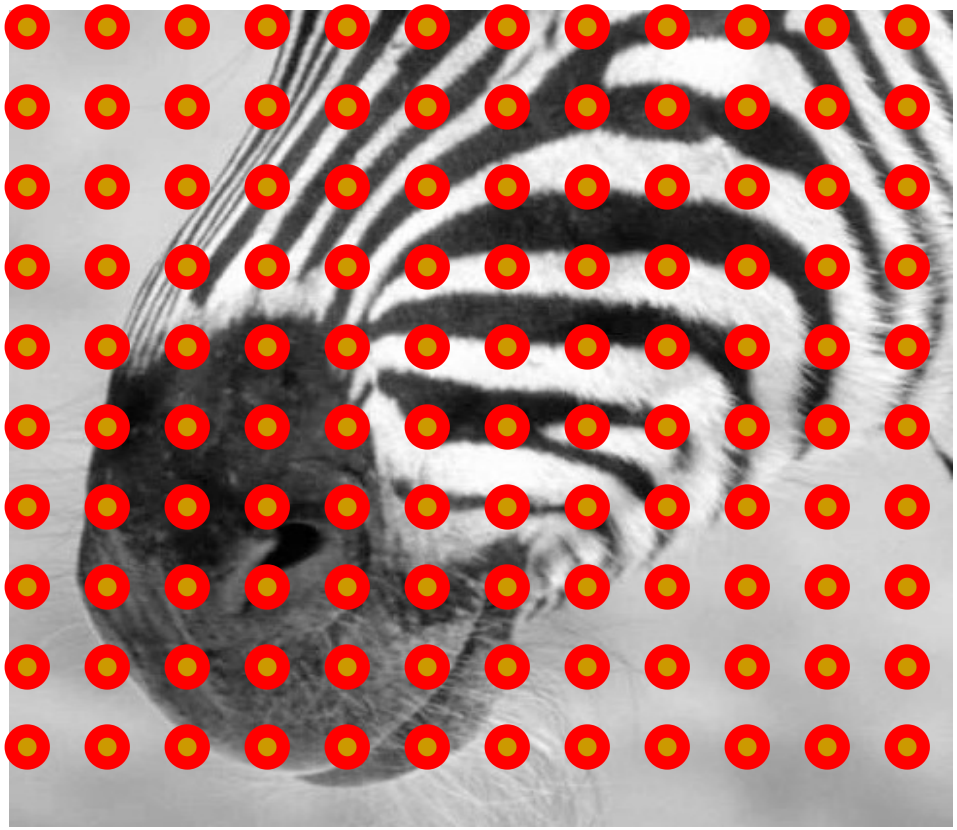


We need translation and scale invariance

Image pyramids

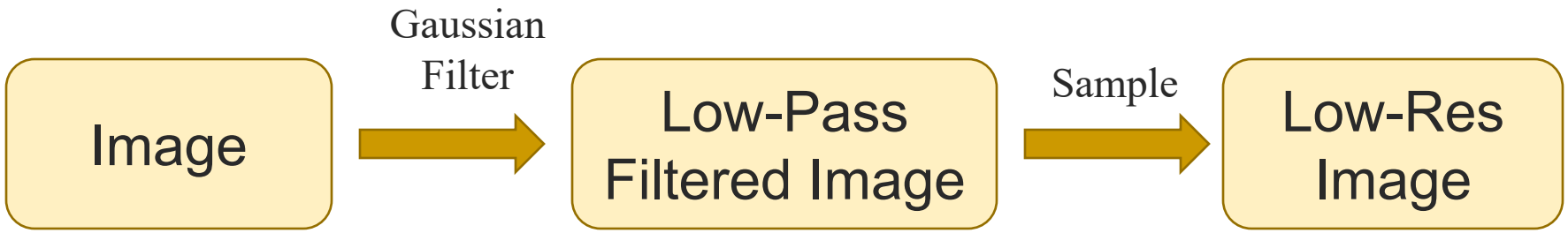


Subsampling by a factor of 2



Throw away every other row and column
to create a 1/2 size image

Recall: sampling



E. H. Adelson | C. H. Anderson | J. R. Bergen | P. J. Burt | J. M. Ogden

Pyramid methods in image processing

http://persci.mit.edu/pub_pdfs/RCA84.pdf

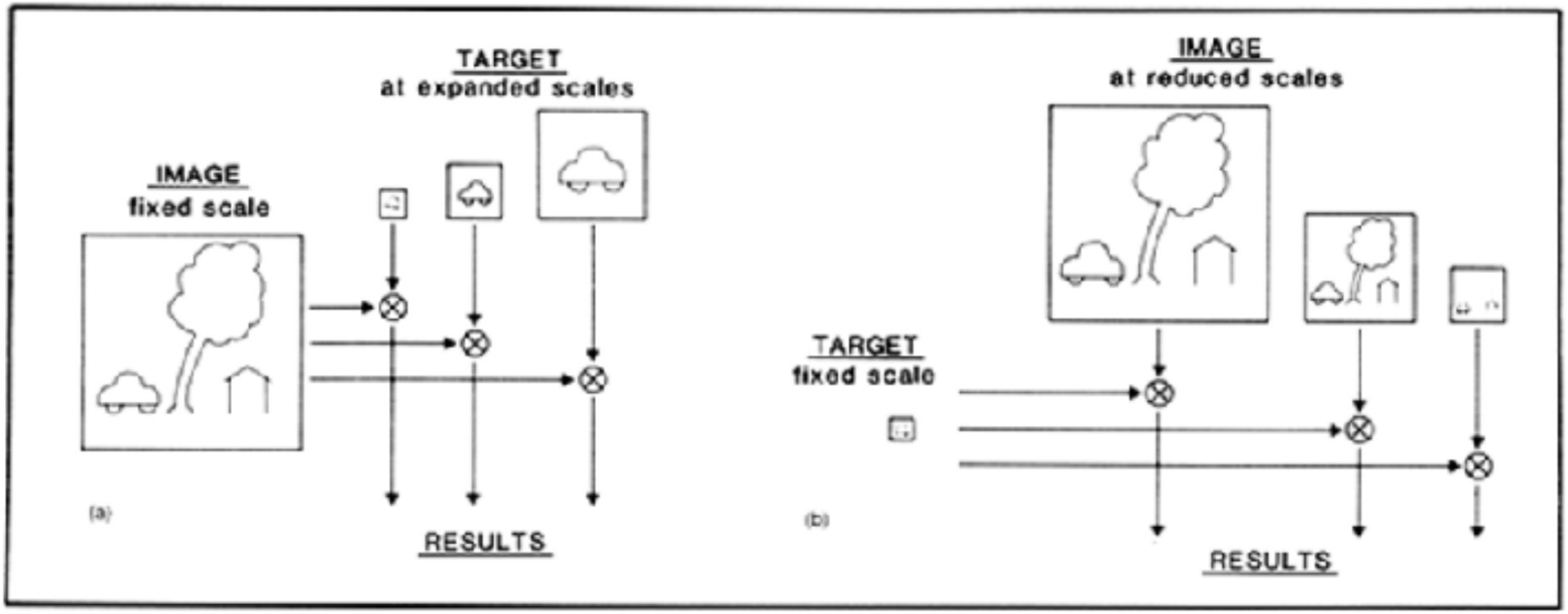
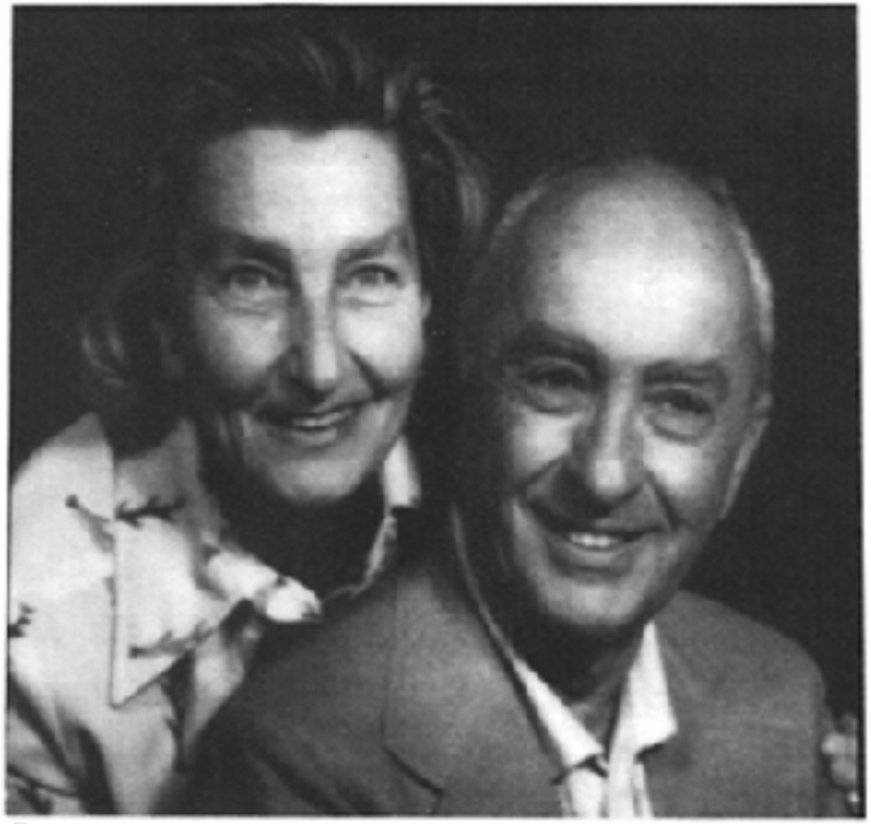


Fig. 1. Two methods of searching for a target pattern over many scales. In the first approach, (a), copies of the target pattern are constructed at several expanded scales, and each is convolved with the original image. In the second approach, (b), a single copy of the target is convolved with copies of the image reduced in scale. The target should be just large enough to resolve critical details. The two approaches should give equivalent results, but the second is more efficient by the fourth power of the scale factor (image convolutions are represented by 'O').



G_0

Fig. 2a. *The Gaussian pyramid. The original image, G_0, is repeatedly filtered and subsampled to generate the sequence of reduced resolution image G_1, G_2, etc. These comprise a set of lowpass-filtered copies of the original image in which the bandwidth decreases in one-octave steps.*

G_1

G_2

G_3

G_4

Fig. 2b. *Levels of the Gaussian pyramid expanded to the size of the original image. The effects of lowpass filtering are now clearly apparent.*

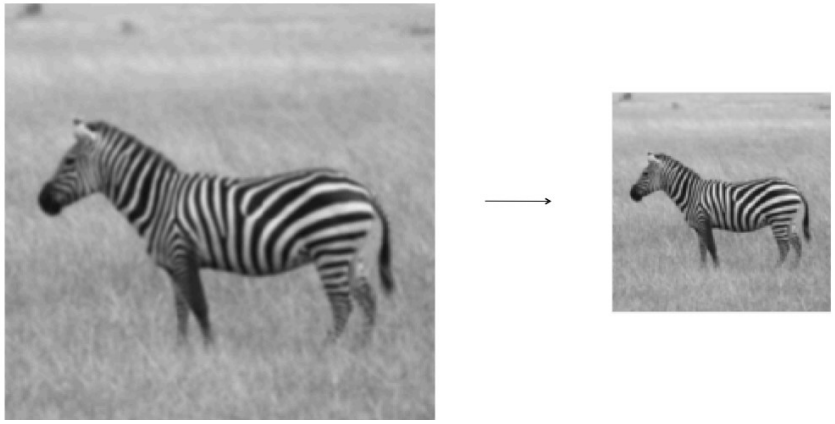
512 256 128 64 32 16 8

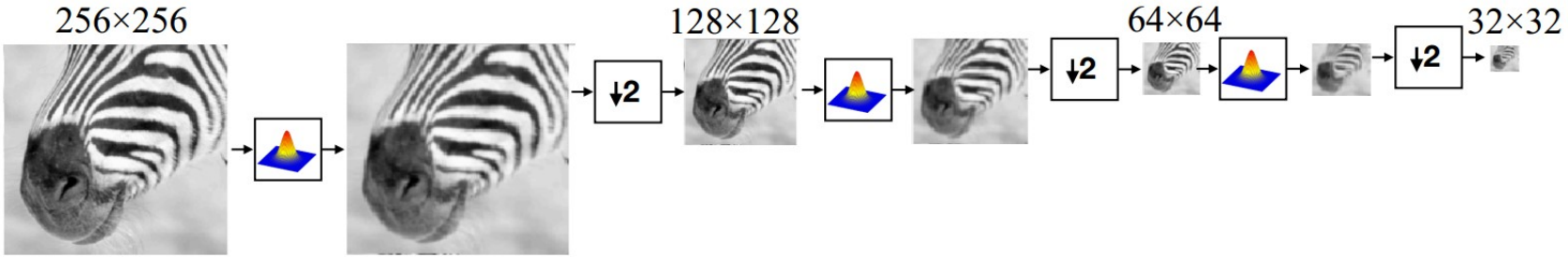
Gaussian Pyramid

Gaussian Pyramid

For each level

1. Blur input image with a Gaussian filter
2. Downsample image





Downsampling & Upsampling

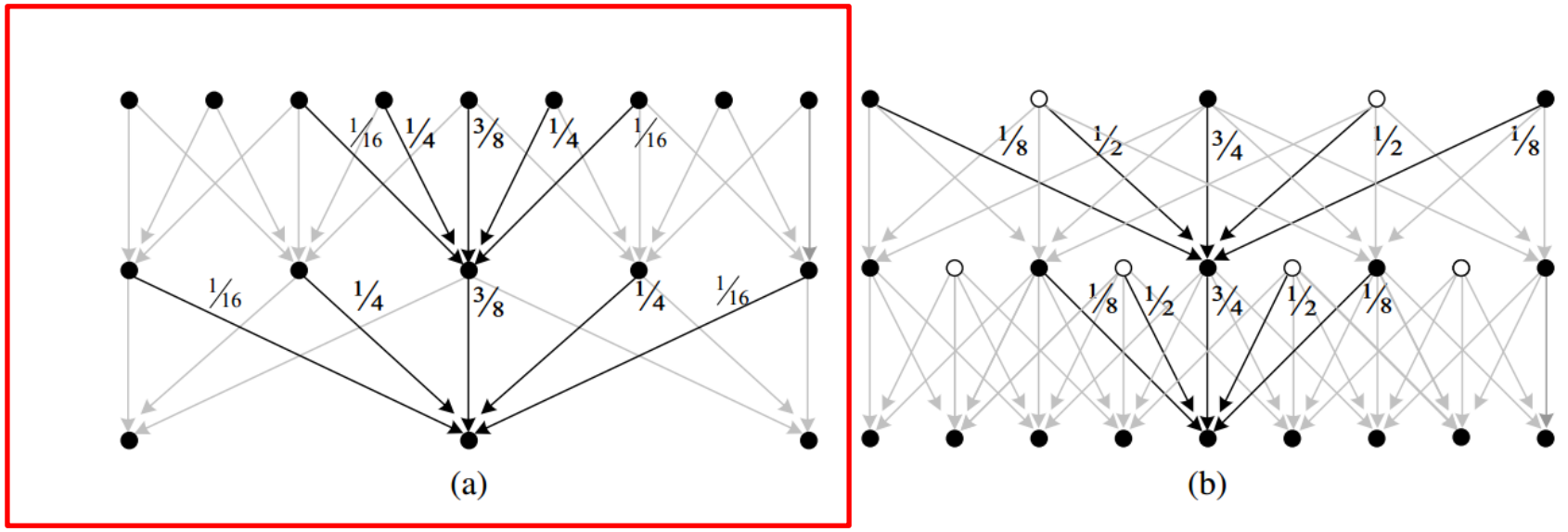
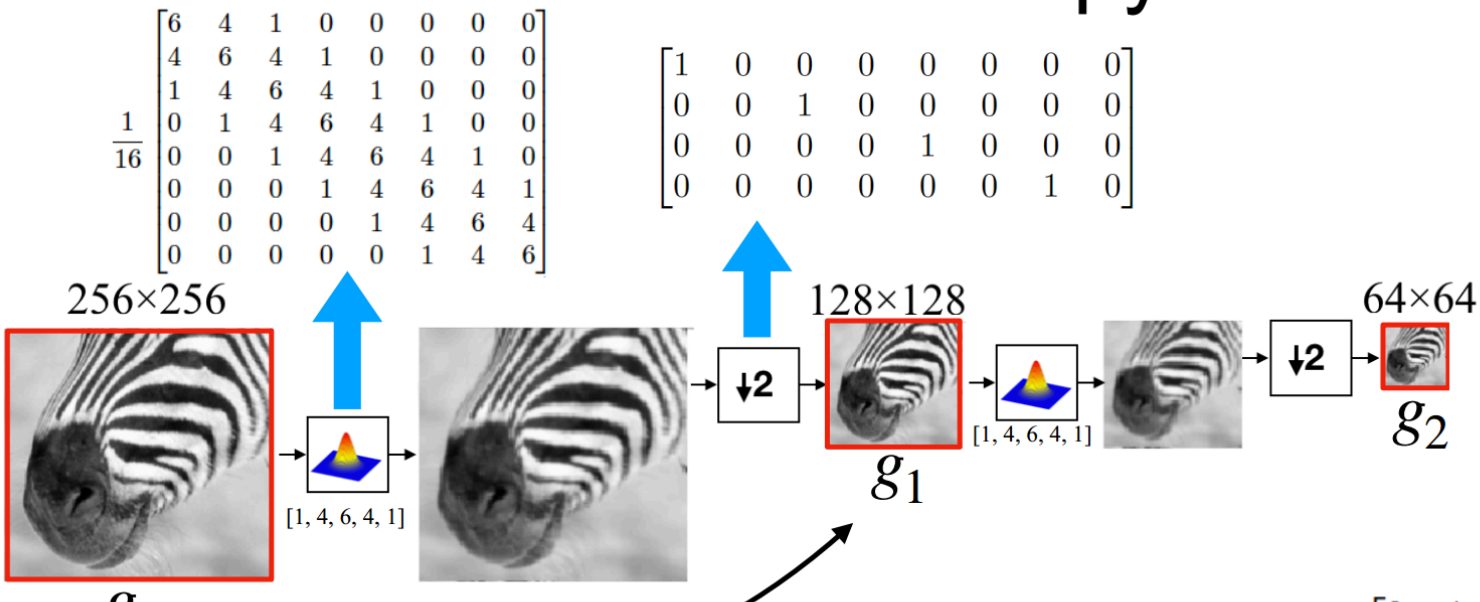


Figure 3.33 The Gaussian pyramid shown as a signal processing diagram: The (a) analysis and (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the $\uparrow 2$ upsampling operation. Notice how the reconstruction filter coefficients are twice the analysis coefficients. The computation is shown as flowing down the page, regardless of whether we are going from coarse to fine or *vice versa*.

The Gaussian pyramid

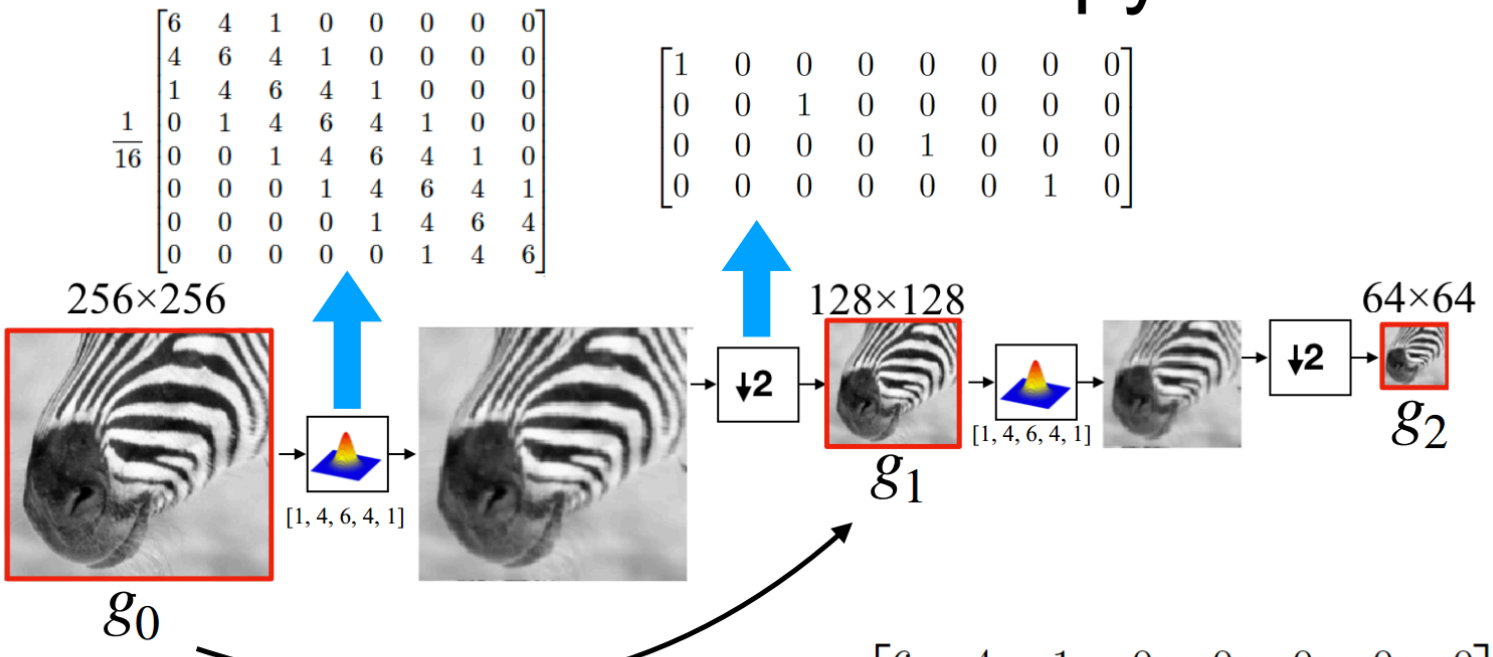


$$\frac{1}{16}\begin{bmatrix}6&4&1&0&0&0&0&0\\4&6&4&1&0&0&0&0\\1&4&6&4&1&0&0&0\\0&1&4&6&4&1&0&0\\0&0&1&4&6&4&1&0\\0&0&0&1&4&6&4&1\\0&0&0&0&1&4&6&4\\0&0&0&0&0&1&4&6\end{bmatrix}$$

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

$$g_1 = G_0 g_0 \qquad G_0 = \begin{bmatrix}1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0\end{bmatrix}\frac{1}{16}\begin{bmatrix}6&4&1&0&0&0&0&0\\4&6&4&1&0&0&0&0\\1&4&6&4&1&0&0&0\\0&1&4&6&4&1&0&0\\0&0&1&4&6&4&1&0\\0&0&0&1&4&6&4&1\\0&0&0&0&1&4&6&4\\0&0&0&0&0&1&4&6\end{bmatrix}$$

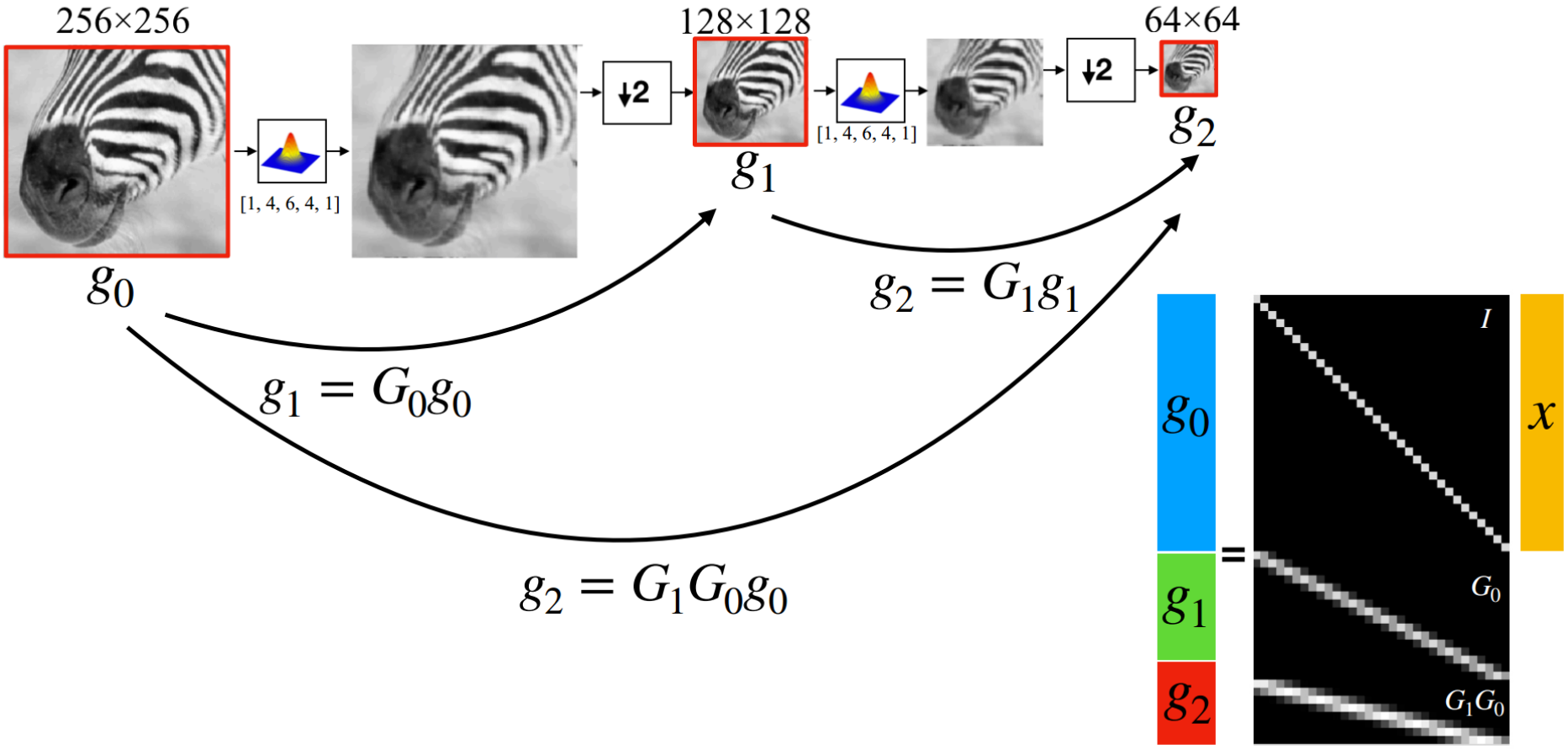
The Gaussian pyramid



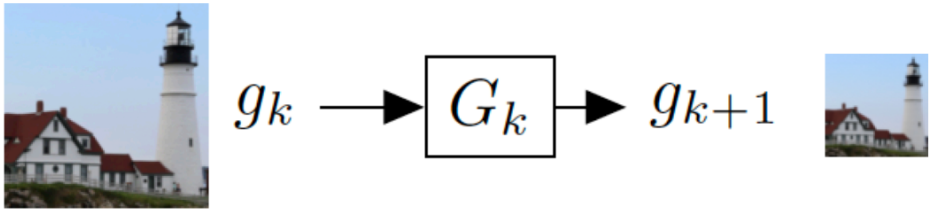
$$\frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$g_1 = G_0 g_0 \qquad G_0 = \frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \end{bmatrix}$$

The Gaussian pyramid



Gaussian Pyramid



For each level

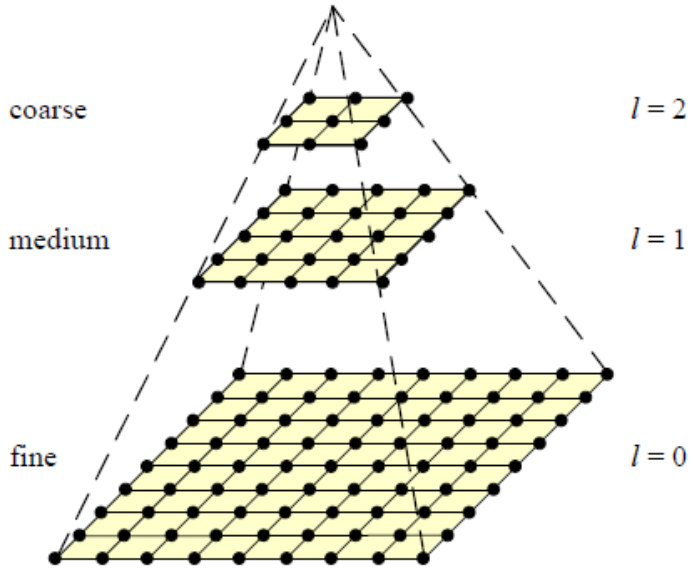
1. Blur input image with a Gaussian filter
2. Downsample image

Gaussian pyramids used for

- Up or down sample images
- Multi-resolution image analysis
  - Look for an object over various spatial scales
  - Coarse-to-fine image regestration: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat
  - Often a successful strategy for avoiding local minima in complicated estimation tasks

Coarse-to-fine Image Registration

1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range



Why is this faster?

Are we guaranteed to get the same result?

Template Matching with Image Pyramids

Input: Image, Template

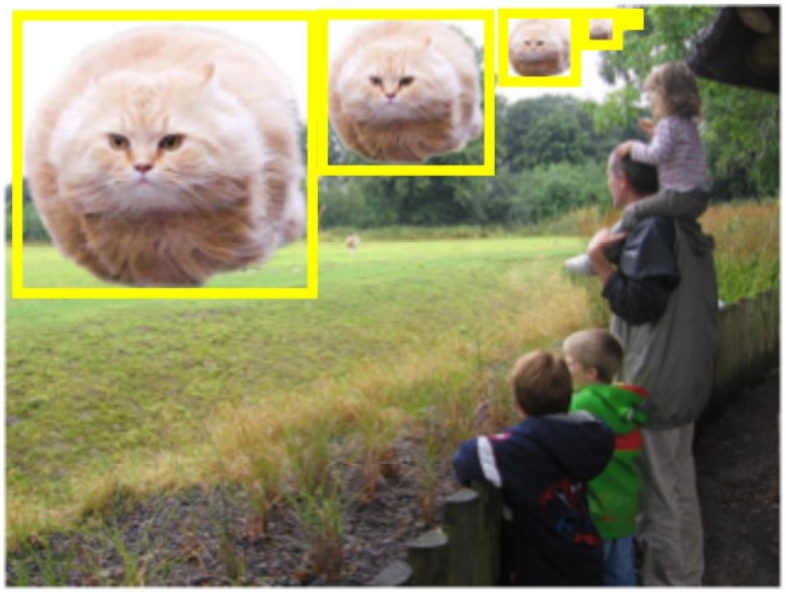
1. Match template at current scale

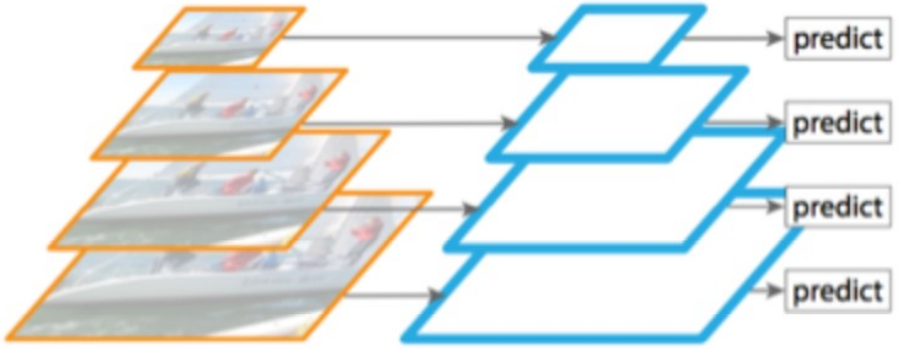
2. Downsample image
   - In practice, scale step of 1.1 to 1.2

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with non-maxima suppression

From Image Pyramid to Feature Pyramid



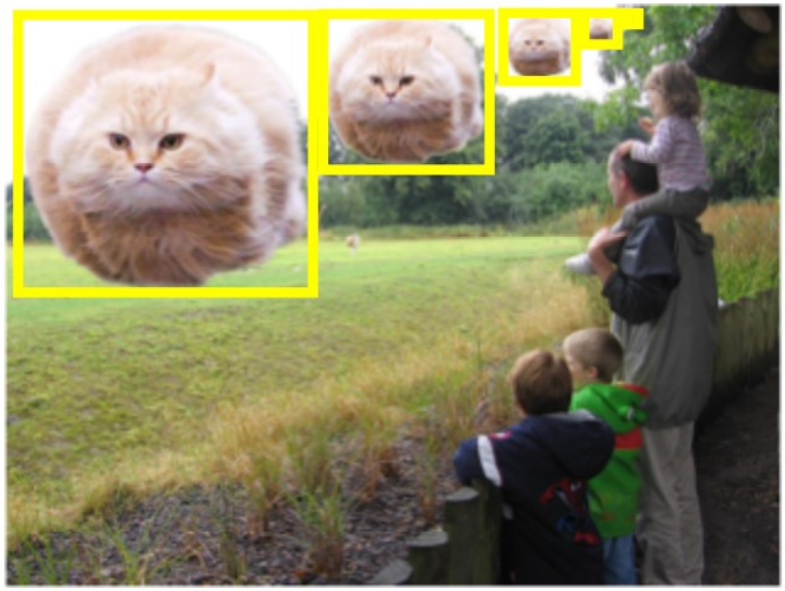


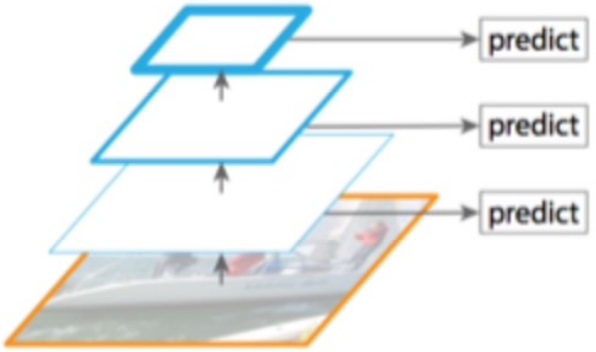
(a) Featurized image pyramid

Standard solution – *slow!*

(E.g., Viola & Jones, HOG, DPM, SPP-net, multi-scale Fast R-CNN, …)

Feature Pyramid Networks



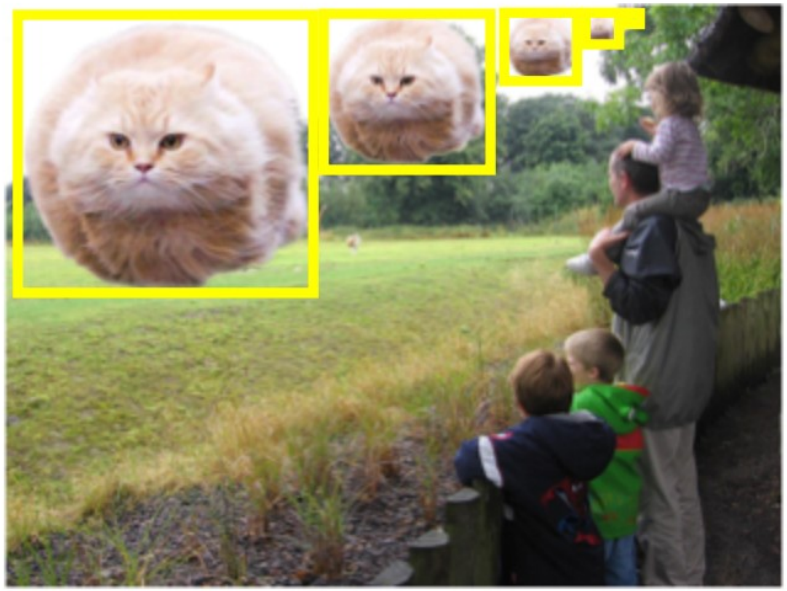


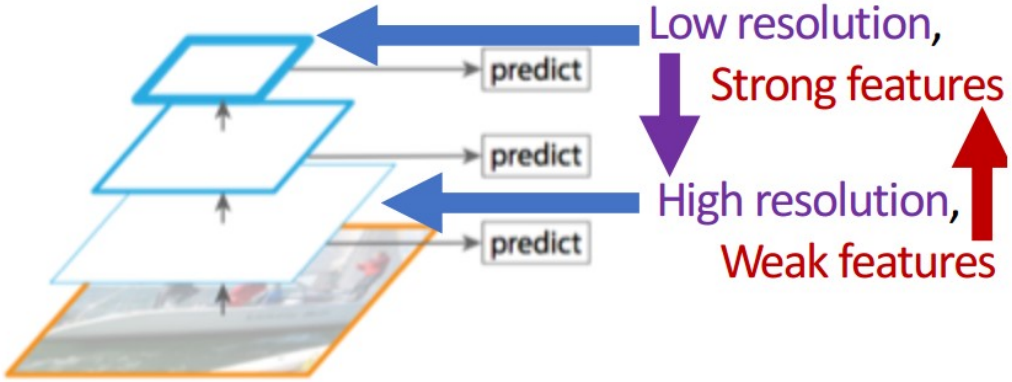
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Feature Pyramid Networks



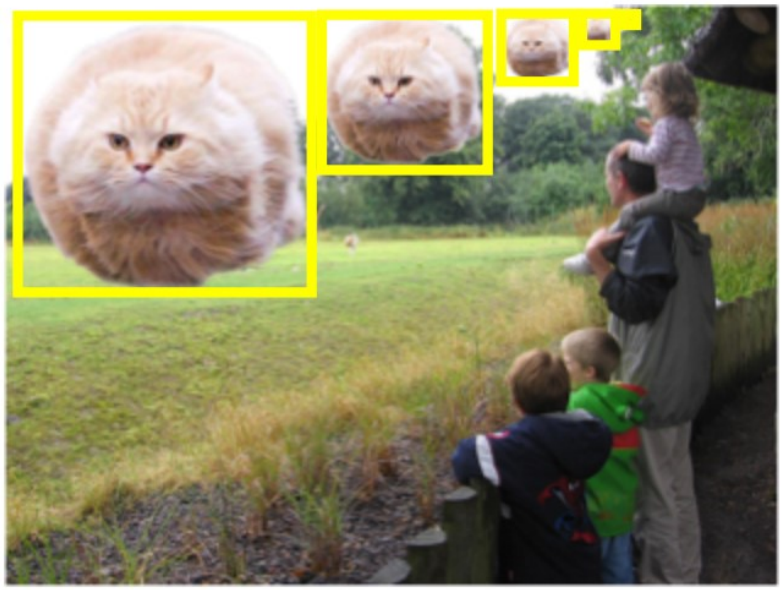


(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Feature Pyramid Networks

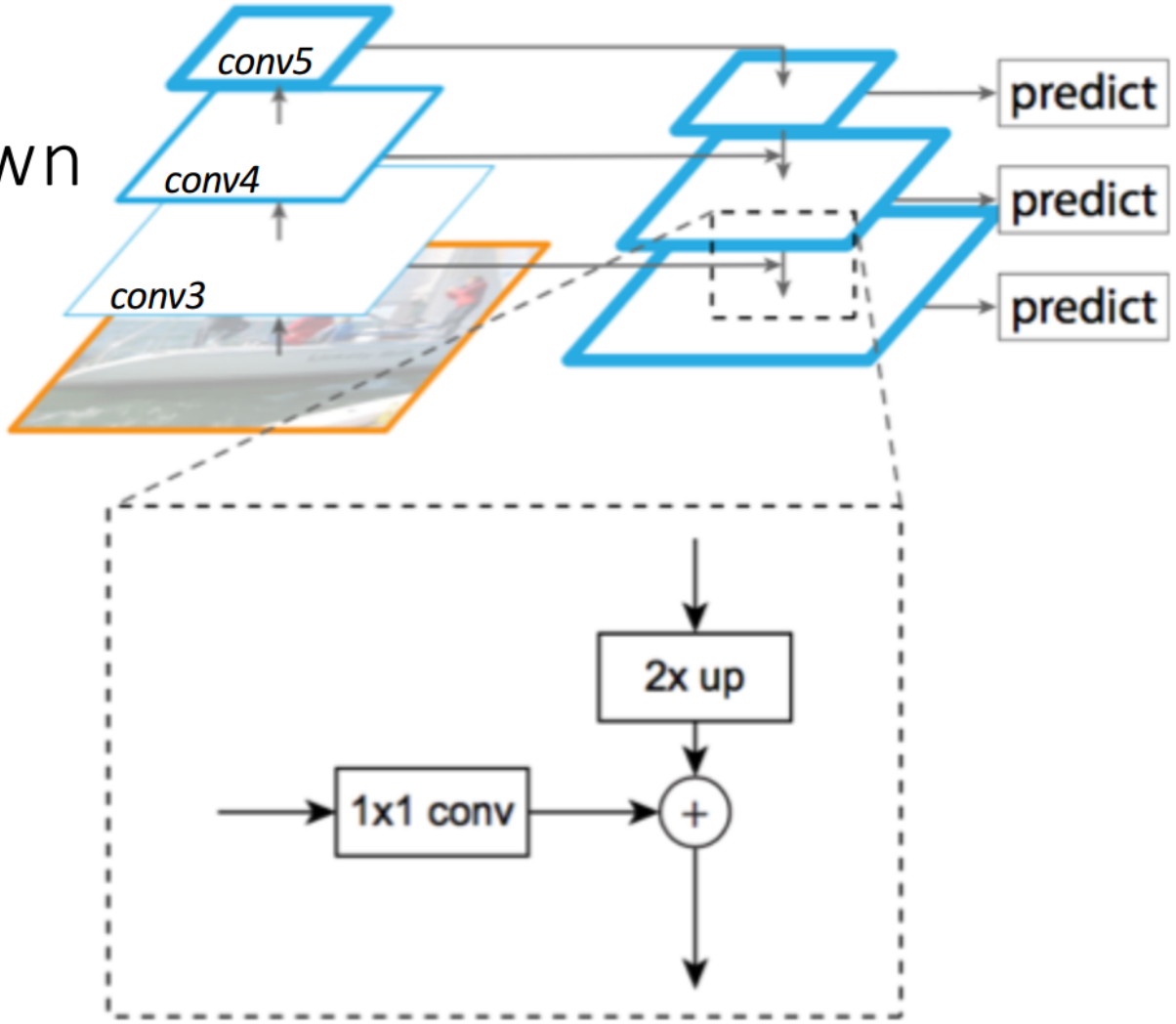


(d) Feature Pyramid Network

Top-down enrichment of high-res features –
fast, less suboptimal

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

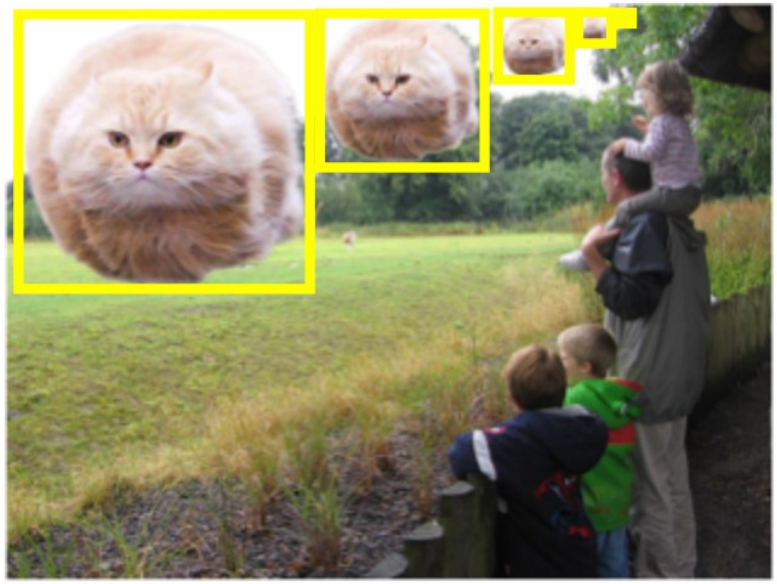
FPN Top-down Refinement Module

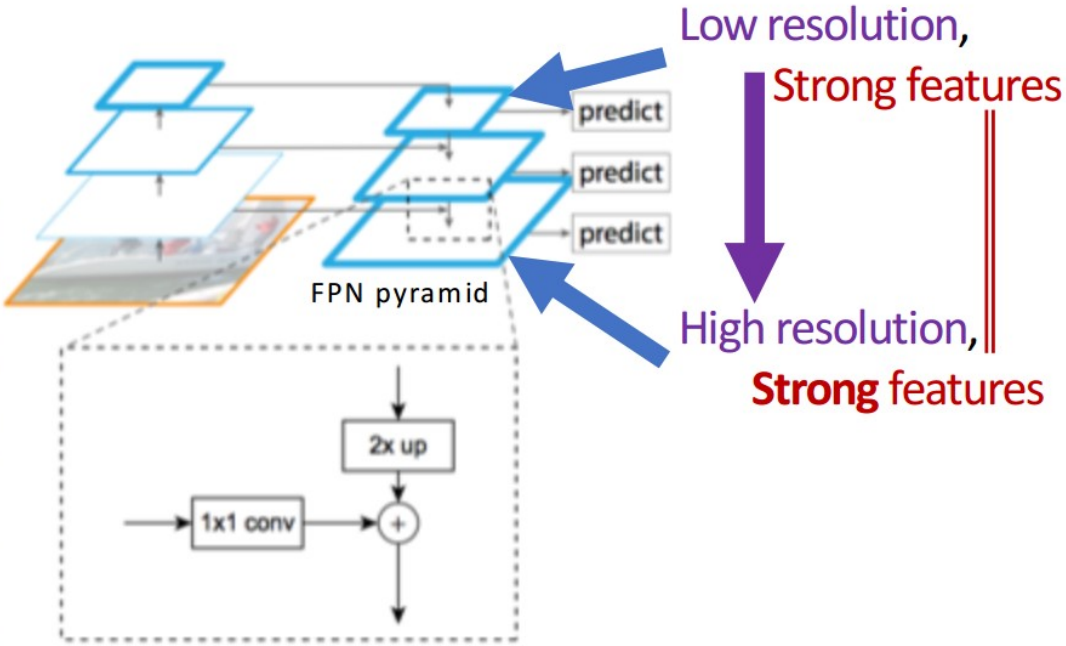


Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

Feature Pyramid Networks

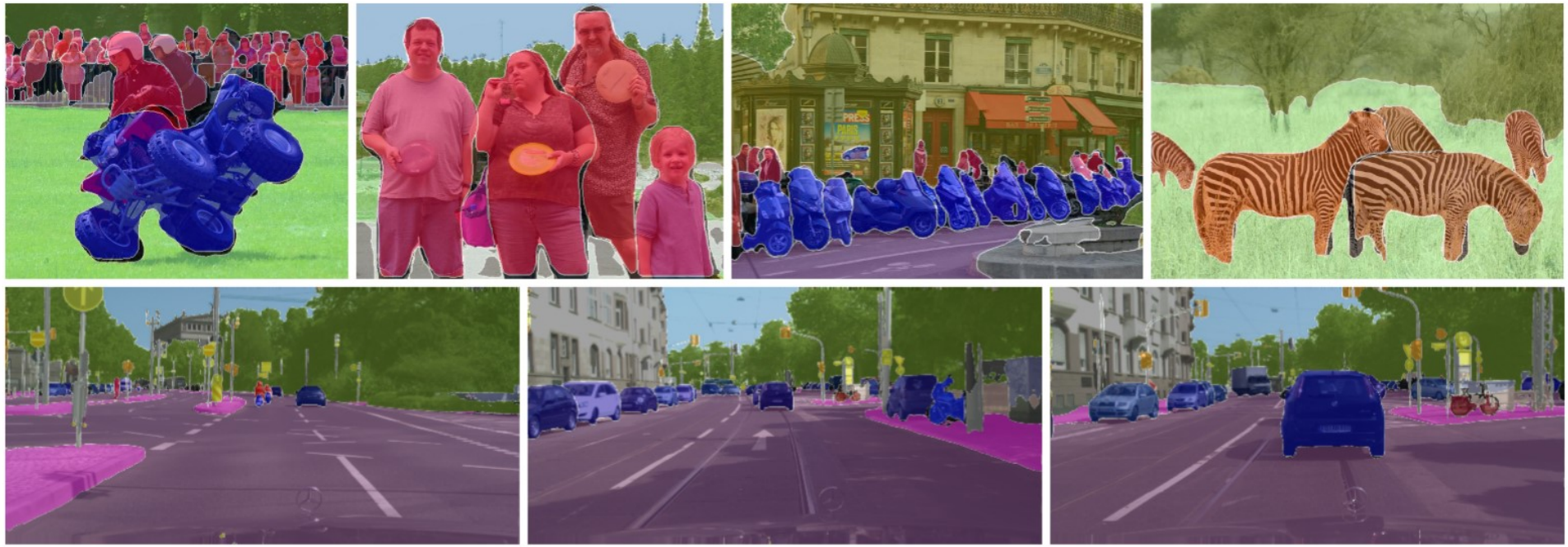
No Compromise on Feature Quality, still Fast



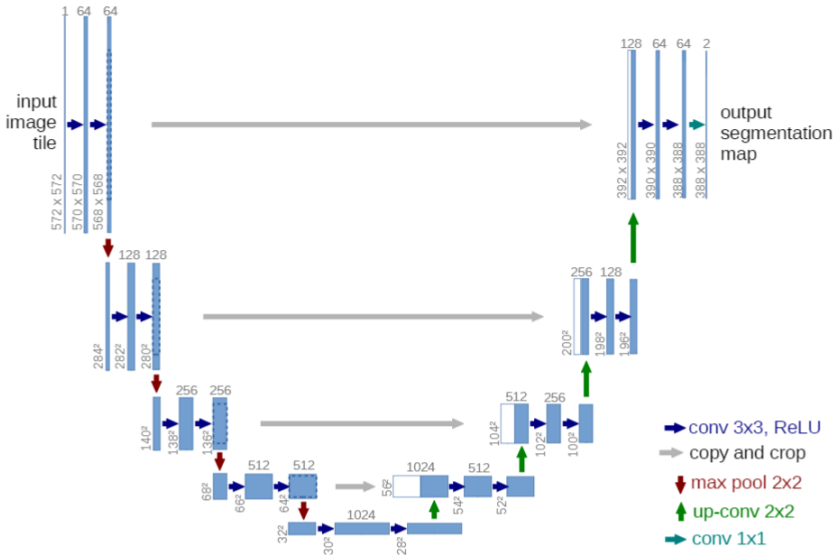


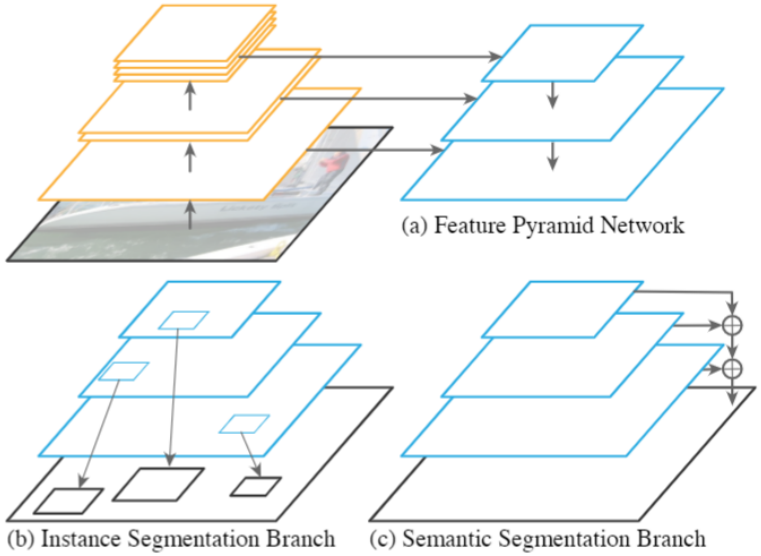
Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017. See also: Shrivastava's TDM.

Segmentation



Pyramid in segmentation

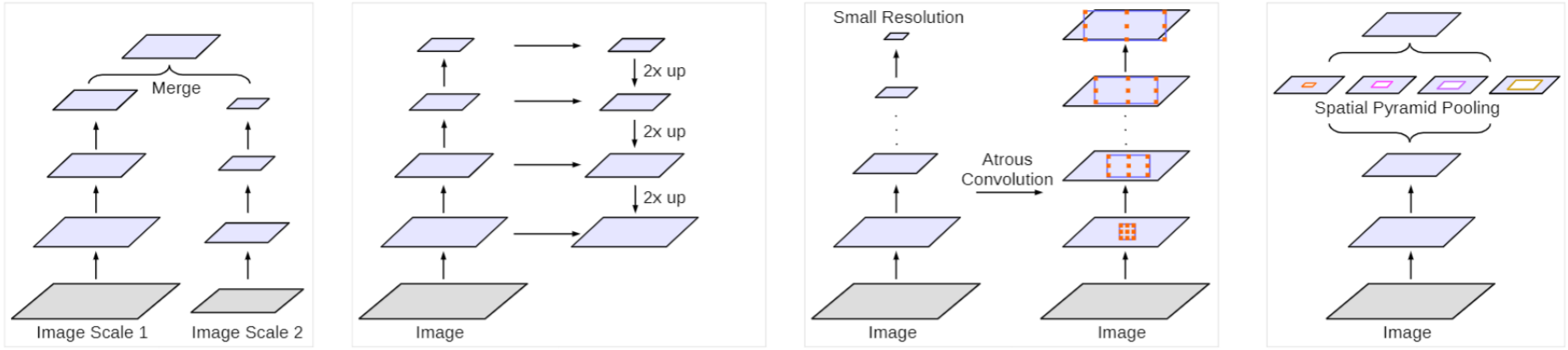




U-Net: Convolutional Networks for Biomedical Image Segmentation

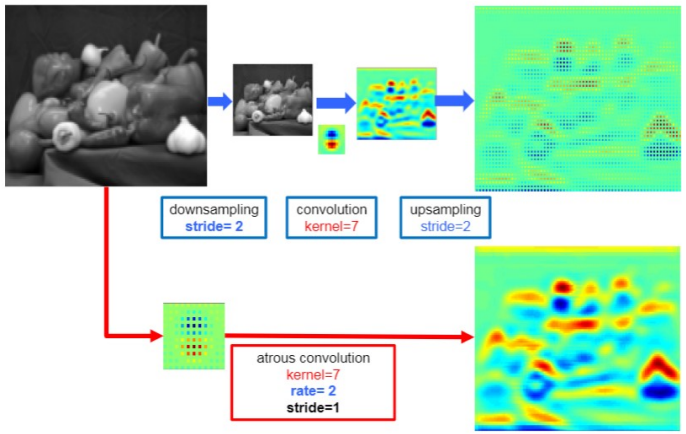
Panoptic Feature Pyramid Networks

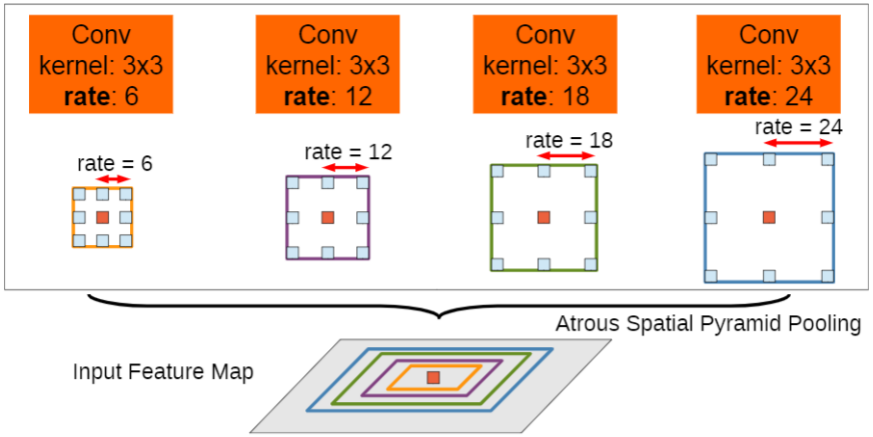
Atrous Convolution



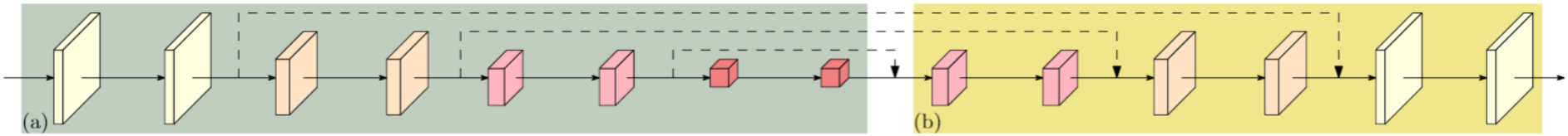
(a) Image Pyramid (b) Encoder-Decoder (c) Deeper w. Atrous Convolution (d) Spatial Pyramid Pooling

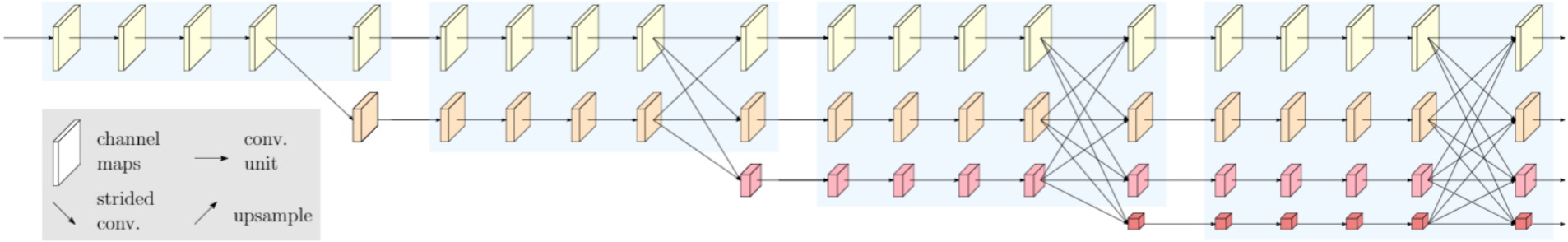
Figure 2. Alternative architectures to capture multi-scale context.



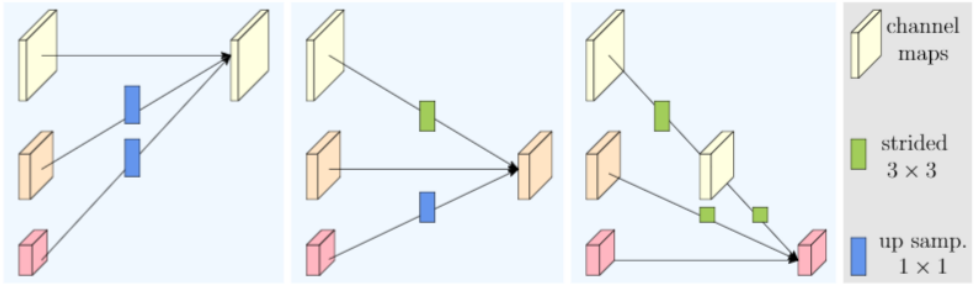


HRNet



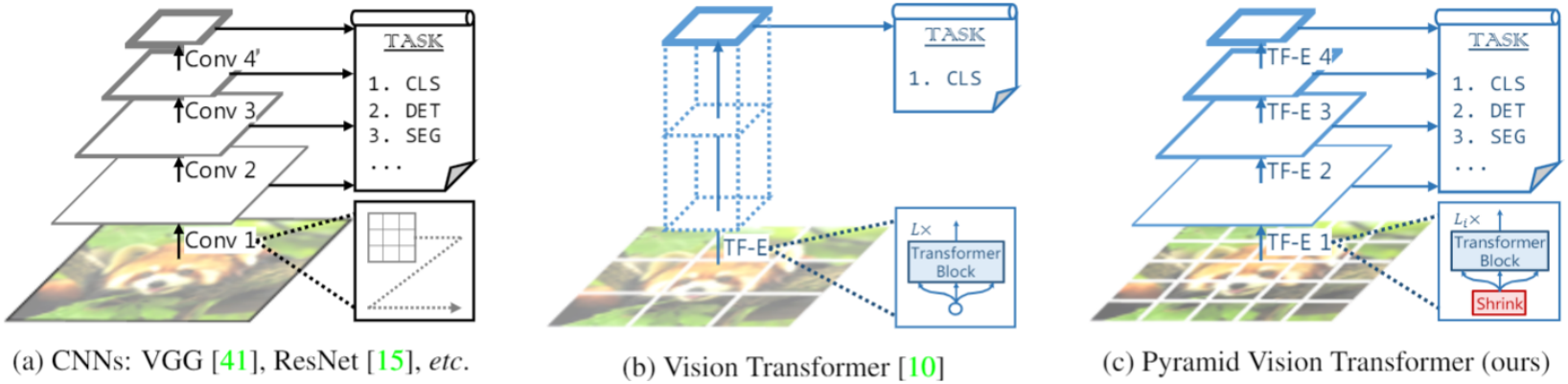


$$\begin{array}{ccccccc} \mathcal{N}_{11} & \rightarrow & \mathcal{N}_{21} & \rightarrow & \mathcal{N}_{31} & \rightarrow & \mathcal{N}_{41} \\ & \searrow & \mathcal{N}_{22} & \rightarrow & \mathcal{N}_{32} & \rightarrow & \mathcal{N}_{42} \\ & & & \searrow & \mathcal{N}_{33} & \rightarrow & \mathcal{N}_{43} \\ & & & & & \searrow & \mathcal{N}_{44}, \end{array}$$



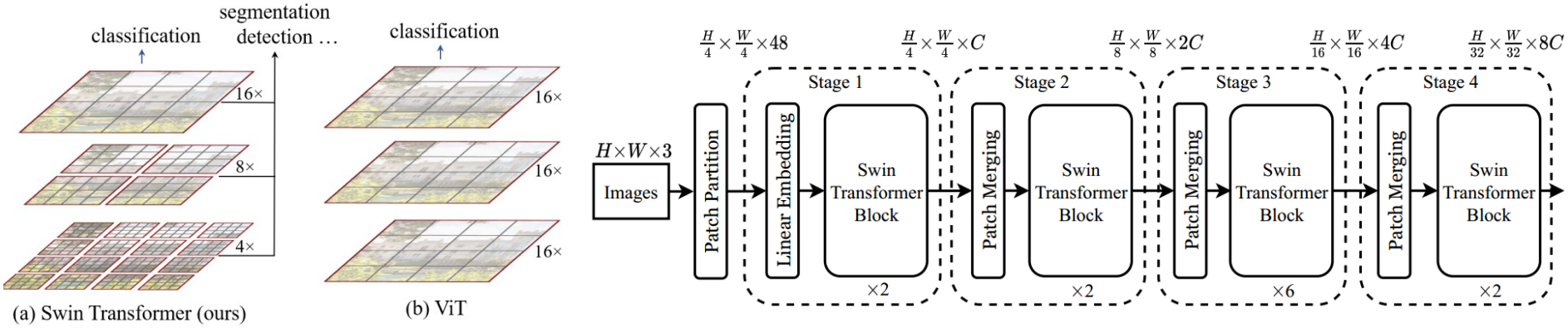
Deep High-Resolution Representation Learning for Visual Recognition

Pyramid in Transformer



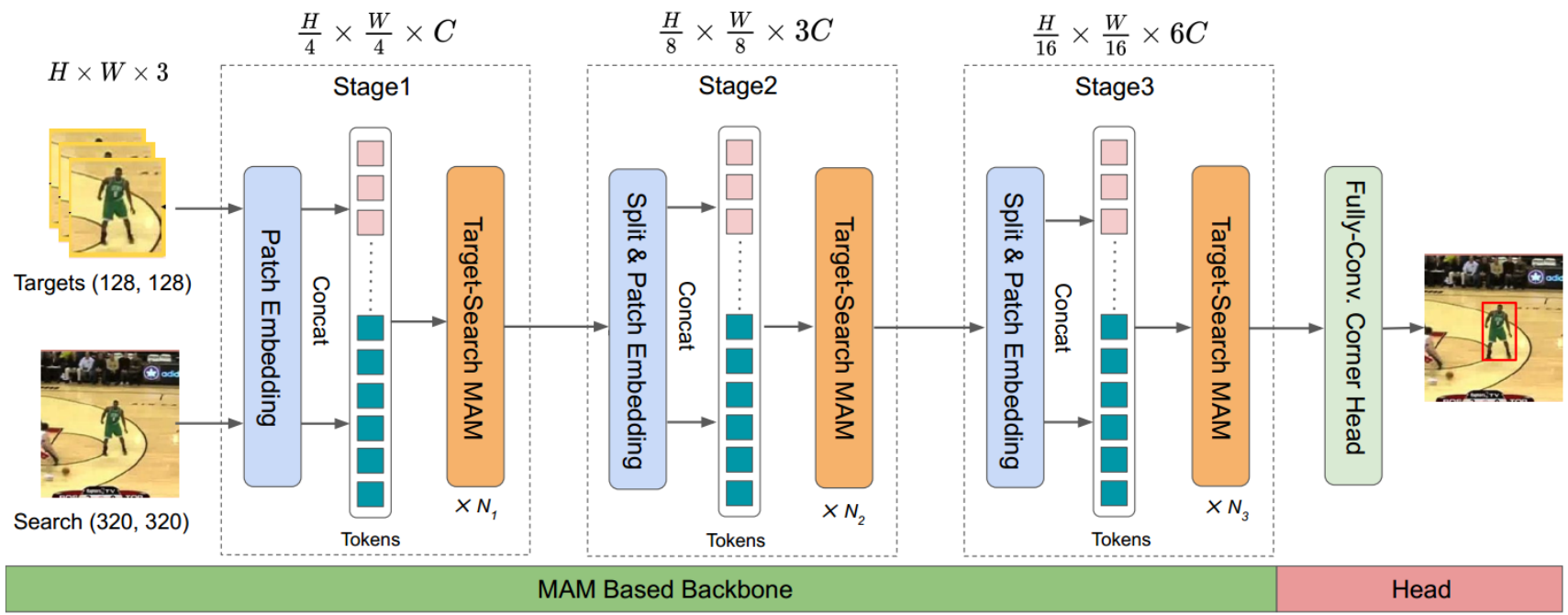
(a) CNNs: VGG [41], ResNet [15], *etc.* (b) Vision Transformer [10] (c) Pyramid Vision Transformer (ours)

Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions

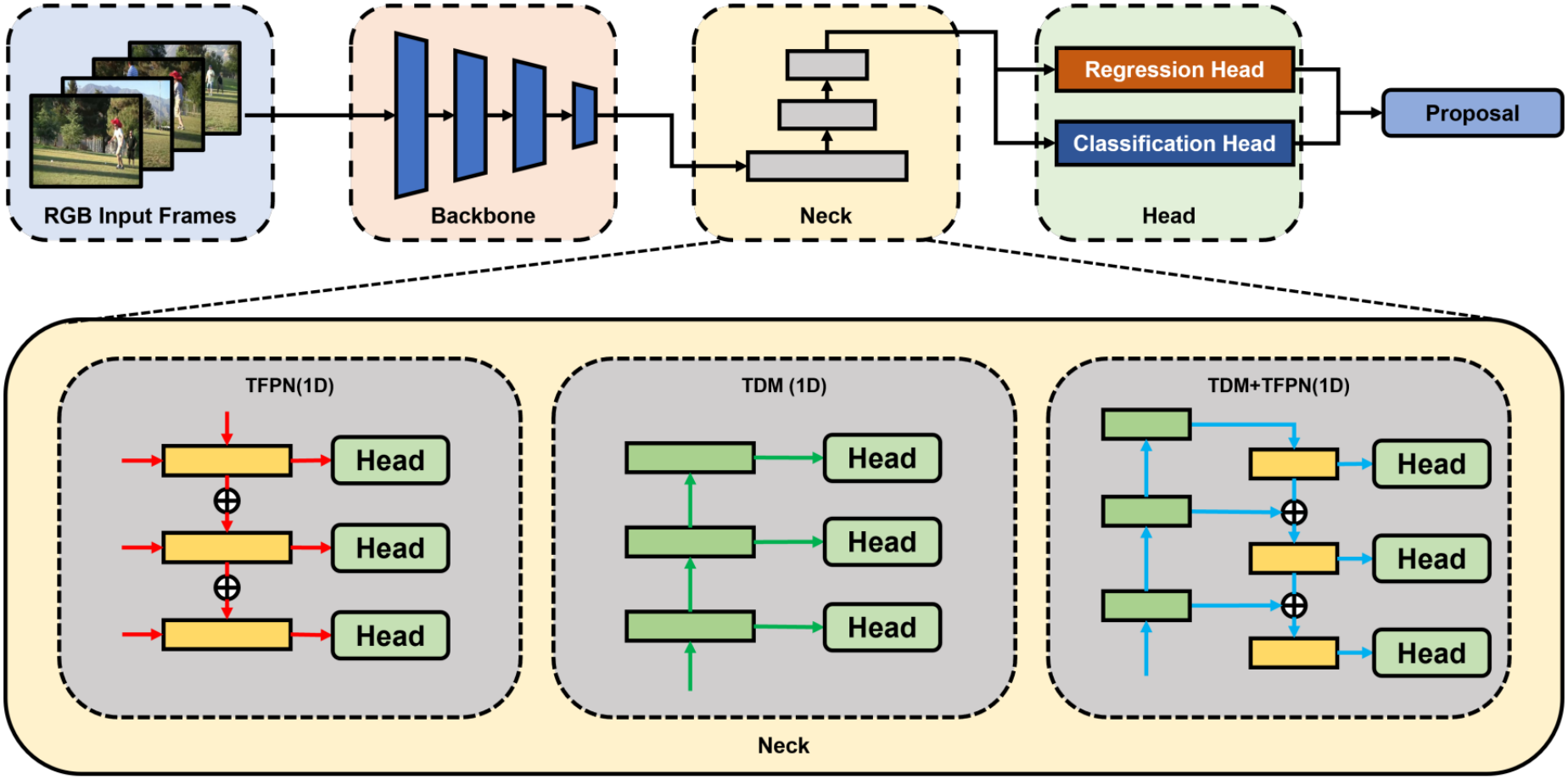


Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

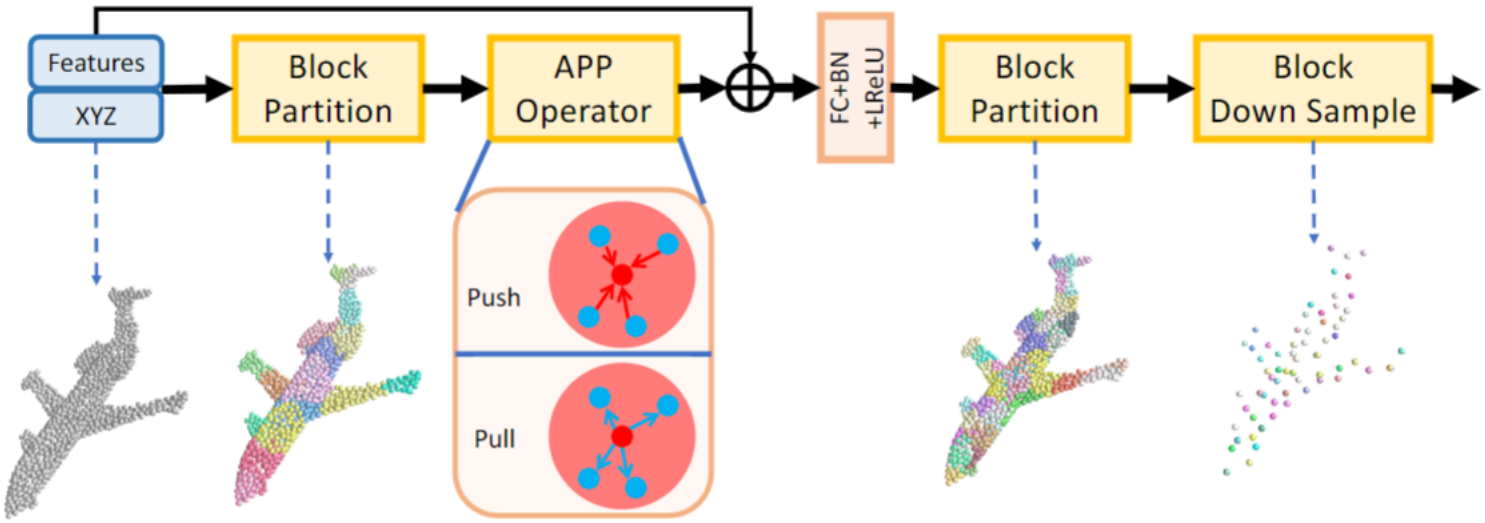
MixFormer in Tracking



Temporal Pyramid: Image to Video



Pyramid in Point Cloud



Today's class

- Template matching
- Gaussian pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- **Laplacian pyramids**
  - Hybrid images
  - Application for image blending
- Steerable pyramids
  - Steerable filters
  - Orientation analysis
- Human vision system
  - Visual perception
- Filter Banks and Texture Analysis

Fig. 4a. *The Laplacian pyramid. Each level of this band-pass pyramid represents the difference between successive levels of the Gaussian pyramid.*

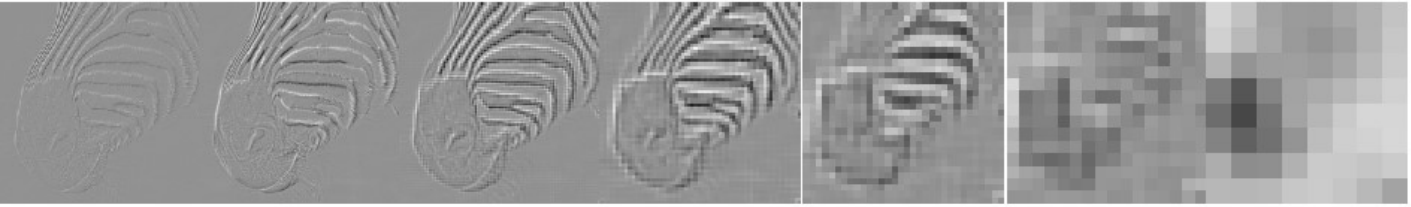
L_0 L_1 L_2 L_3 L_4

$L_{0.0}$

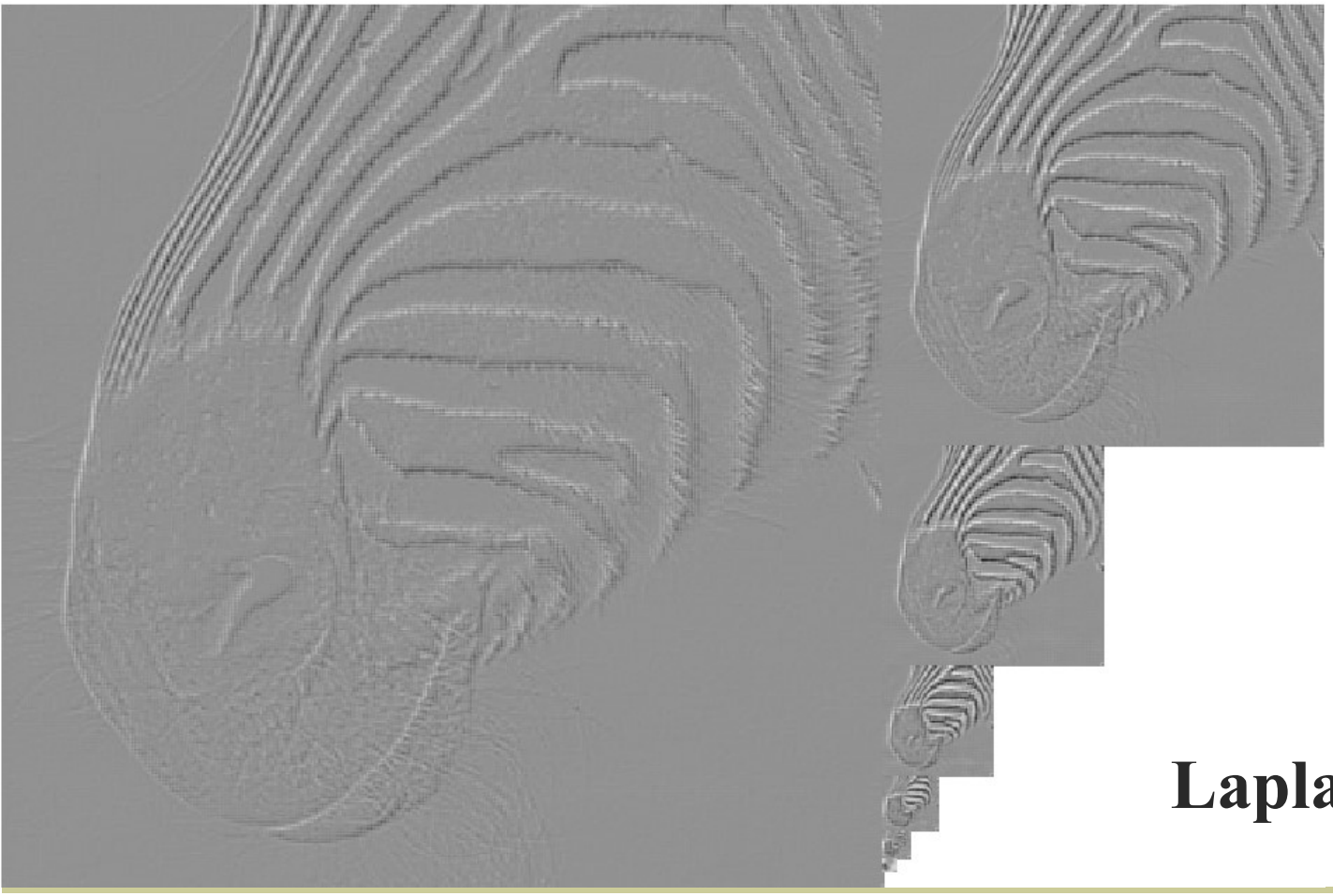
$L_{1.0}$

$L_{2.2}$

Fig. 4b. *Levels of the Laplacian pyramid expanded to the size of the original image. Note that edge and bar features are enhanced and segregated by size.*

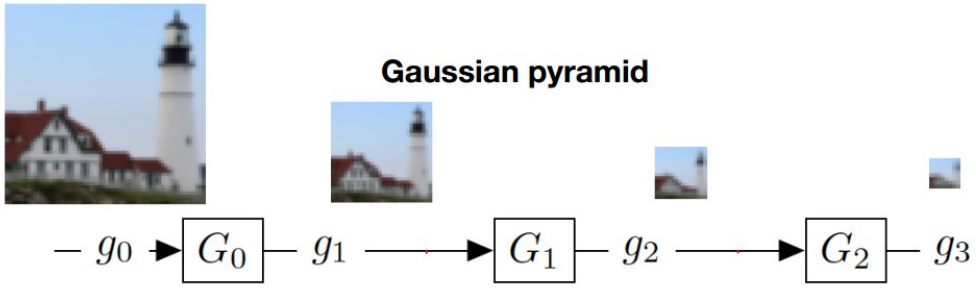


512 256 128 64 32 16 8

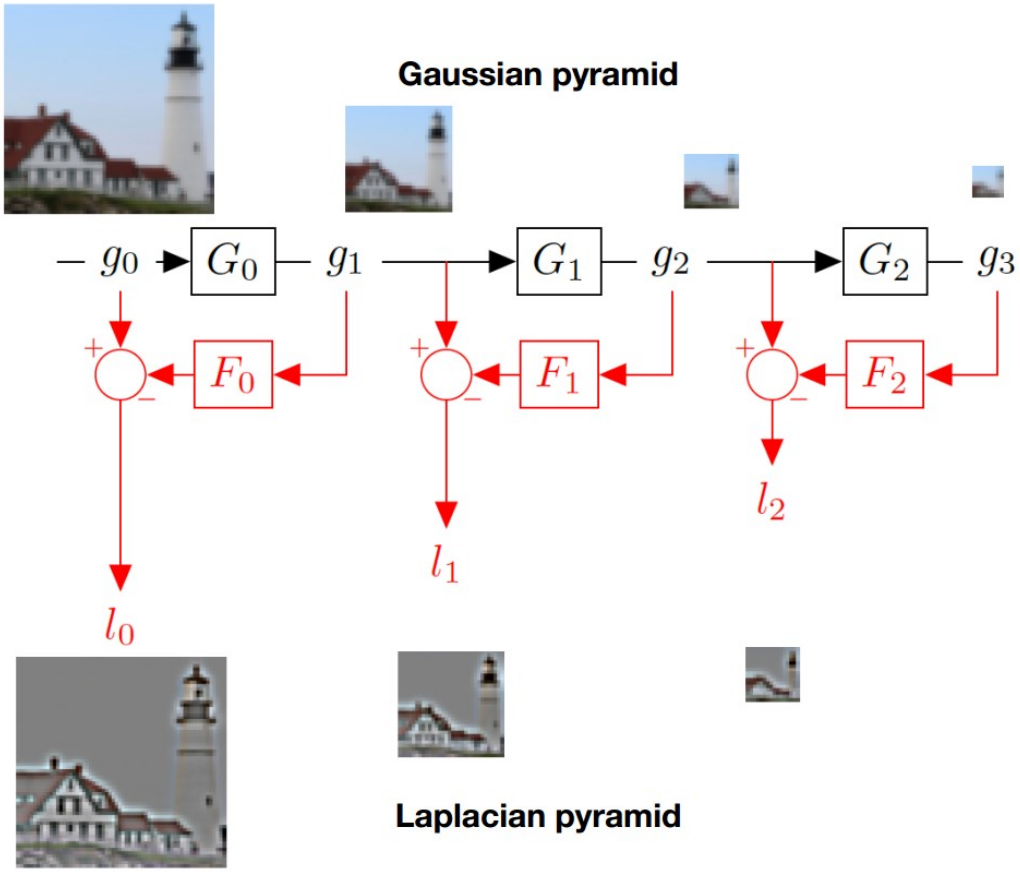


Laplacian Pyramid

The Laplacian Pyramid

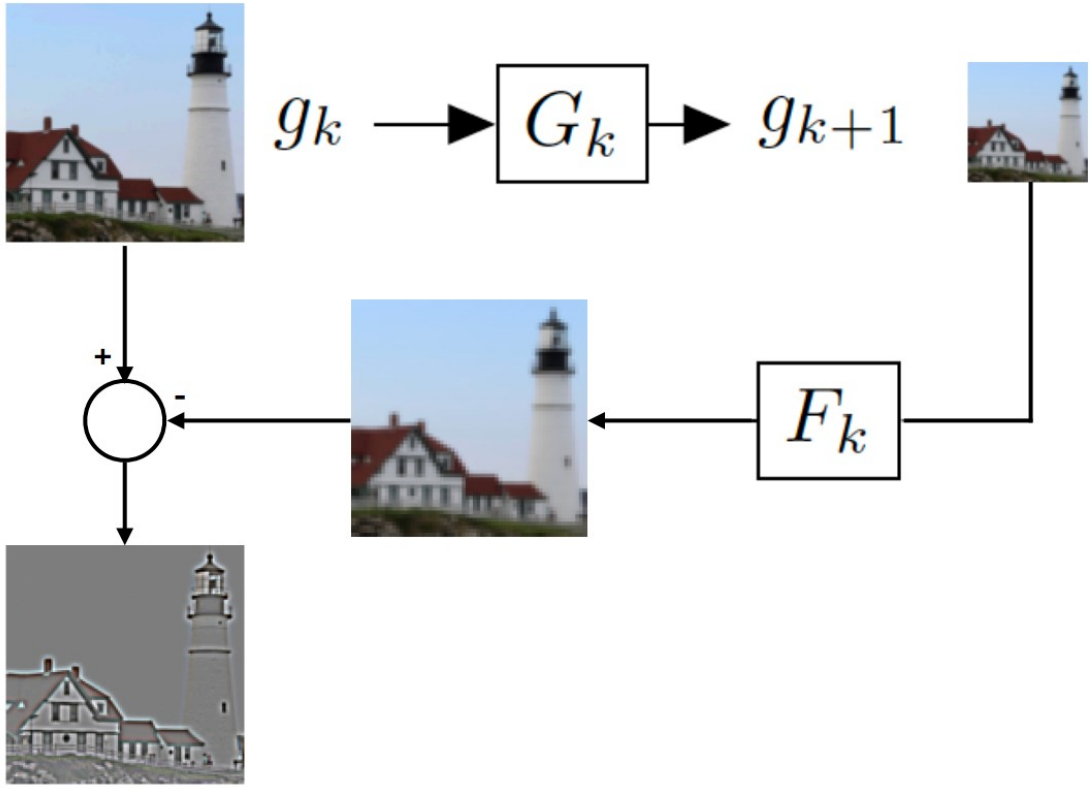


The Laplacian Pyramid

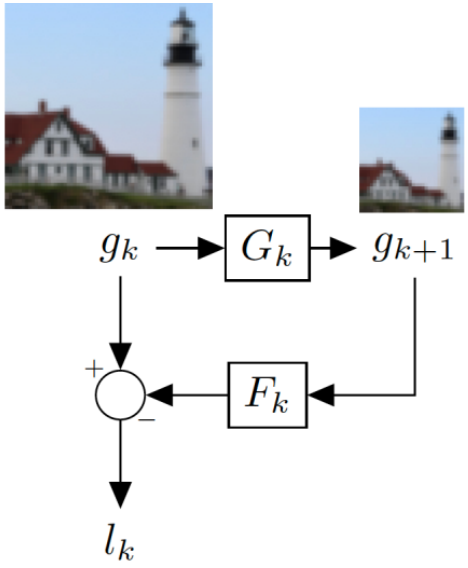


The Laplacian Pyramid

Compute the difference between upsampled Gaussian pyramid level k+1 and Gaussian pyramid level k.



The Laplacian Pyramid



Blurring and downsampling:

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(Downsampling by 2)

$$\frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

(blur)

Upsampling and blurring:

$$F_0 =$$

Downsampling & Upsampling

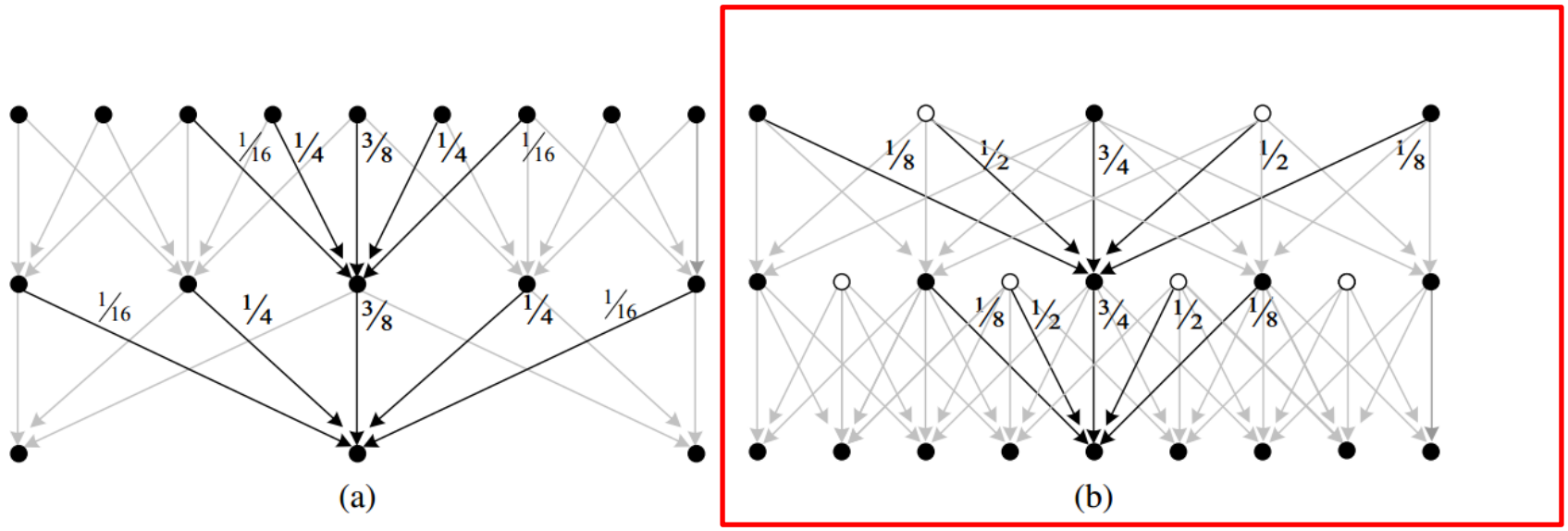
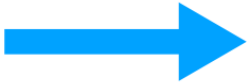


Figure 3.33 The Gaussian pyramid shown as a signal processing diagram: The (a) analysis and (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the $\uparrow 2$ upsampling operation. Notice how the reconstruction filter coefficients are twice the analysis coefficients. The computation is shown as flowing down the page, regardless of whether we are going from coarse to fine or *vice versa*.

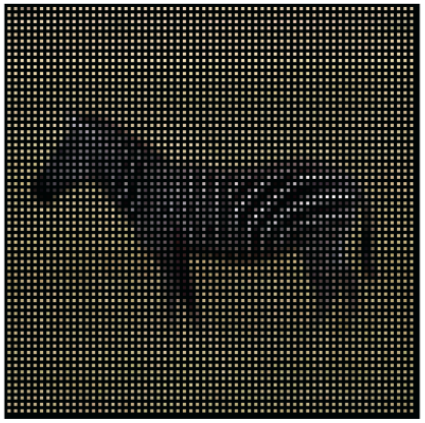
Upsampling

64x64

Insert zeros

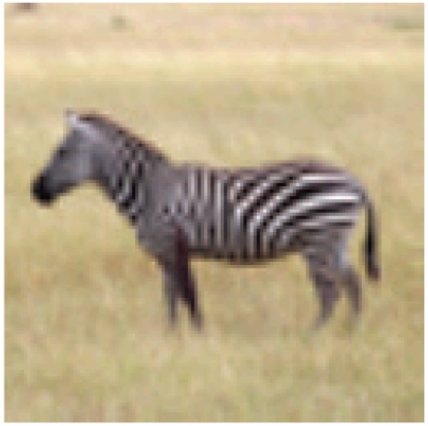


128x128

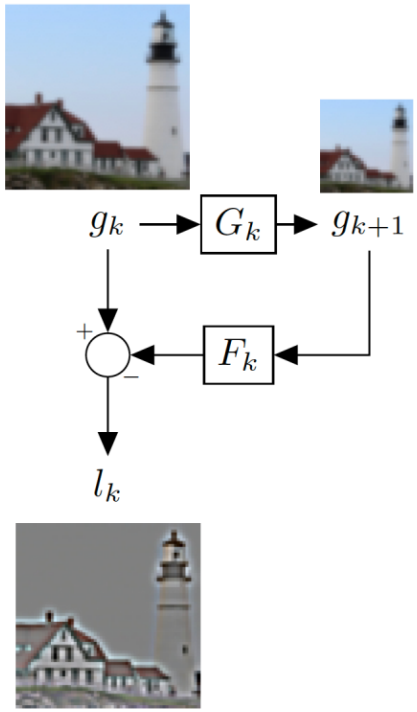


$$\circ \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} =$$

128x128



The Laplacian Pyramid



Blurring and downsampling:

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{16} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

(Downsampling by 2) (blur)

Upsampling and blurring:

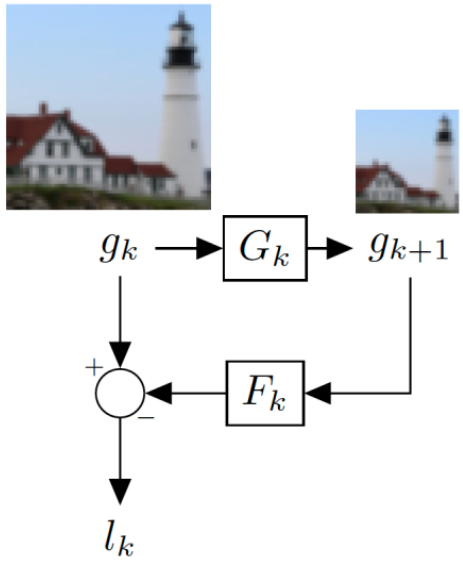
$$F_0 = \frac{1}{8} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(blur) (Upsampling by 2)

$$l_0 = (I_0 - F_0 G_0) g_0$$

22

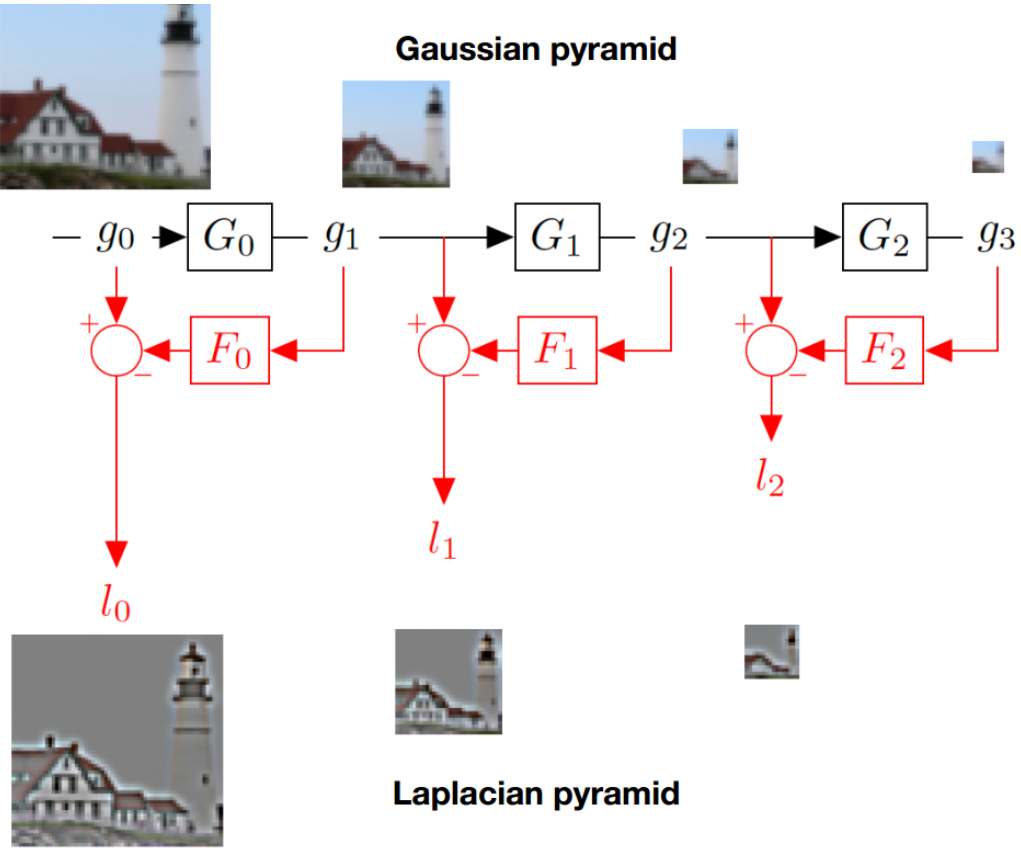
The Laplacian Pyramid

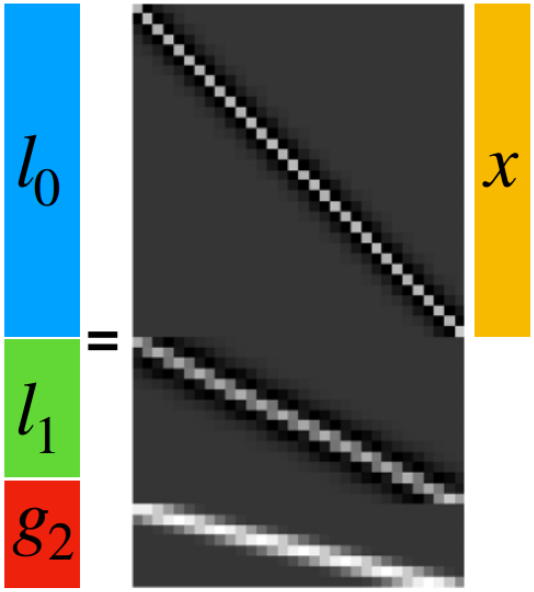


$$l_0 = (I_0 - F_0 G_0) g_0$$

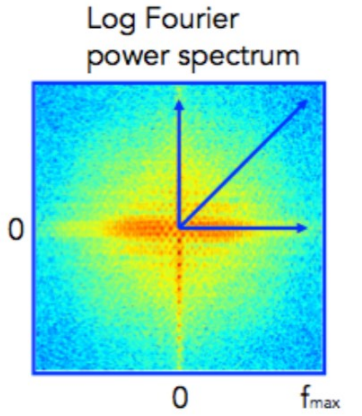
$$= \frac{1}{256} \begin{bmatrix} 182 & -56 & -24 & -8 & -2 & 0 & 0 & 0 \\ -56 & 192 & -56 & -32 & -8 & 0 & 0 & 0 \\ -24 & -56 & 180 & -56 & -24 & -8 & -2 & 0 \\ -8 & -32 & -56 & 192 & -56 & -32 & -8 & 0 \\ -2 & -8 & -24 & -56 & 180 & -56 & -24 & -8 \\ 0 & 0 & -8 & -32 & -56 & 192 & -56 & -32 \\ 0 & 0 & -2 & -8 & -24 & -56 & 182 & -48 \\ 0 & 0 & 0 & 0 & -8 & -32 & -48 & 224 \end{bmatrix} x$$

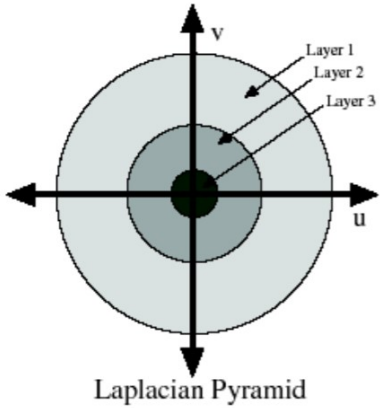
The Laplacian Pyramid





The Laplacian Pyramid

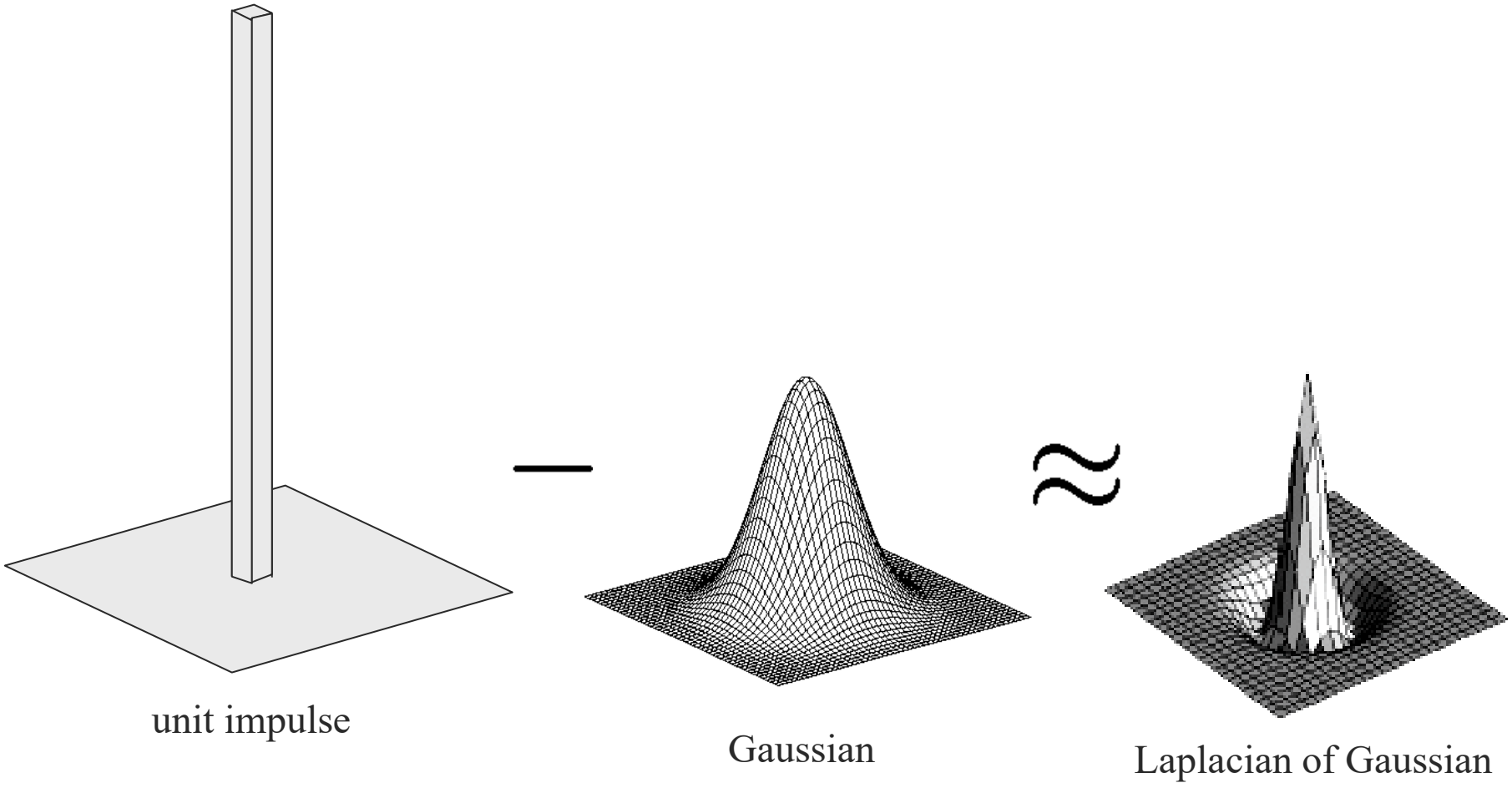




- Each layer of a laplacian pyramid consists in the elements of a smoothed and resampled image.
- **Band pass filters** - each level represents spatial frequencies (largely) unrepresented at other levels
- The fourier transform of each layer is an annulus
- Laplacian pyramid is orientation independent

SubbandDoGLaplacian/subbanddecomposition.m

Laplacian filter



LoG vs. DoG

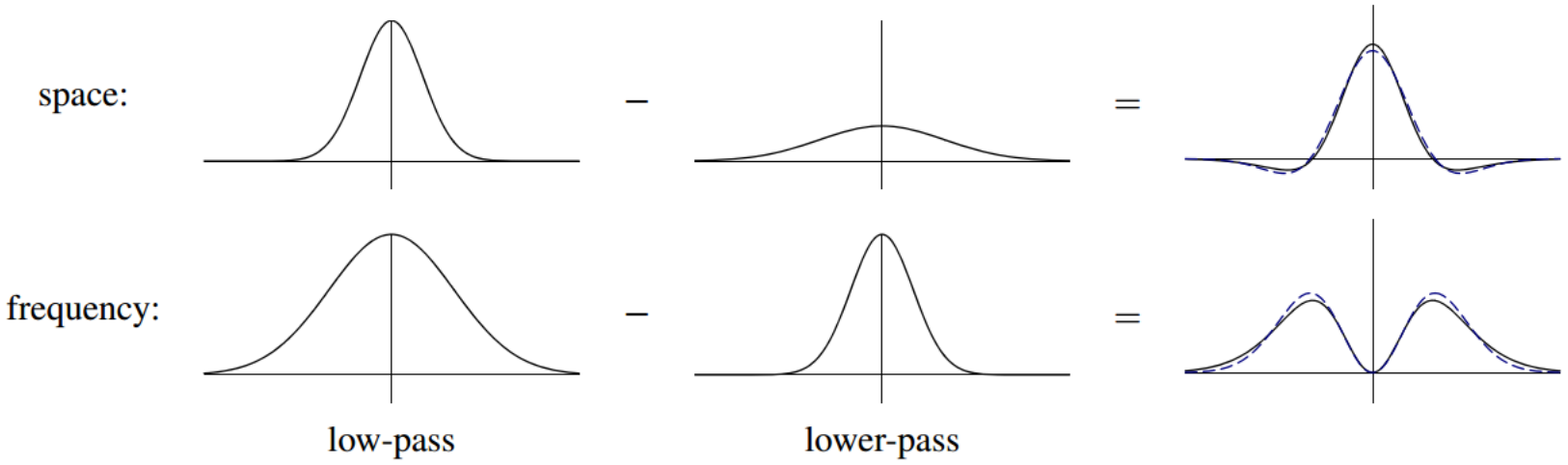


Figure 3.35 The difference of two low-pass filters results in a band-pass filter. The dashed blue lines show the close fit to a half-octave Laplacian of Gaussian.

$$\mathrm{DoG}\{I;\sigma_1,\sigma_2\} = G_{\sigma_1} * I - G_{\sigma_2} * I = (G_{\sigma_1} - G_{\sigma_2}) * I.$$

$$\mathrm{LoG}\{I;\sigma\} = \nabla^2(G_\sigma * I) = (\nabla^2 G_\sigma) * I,$$

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

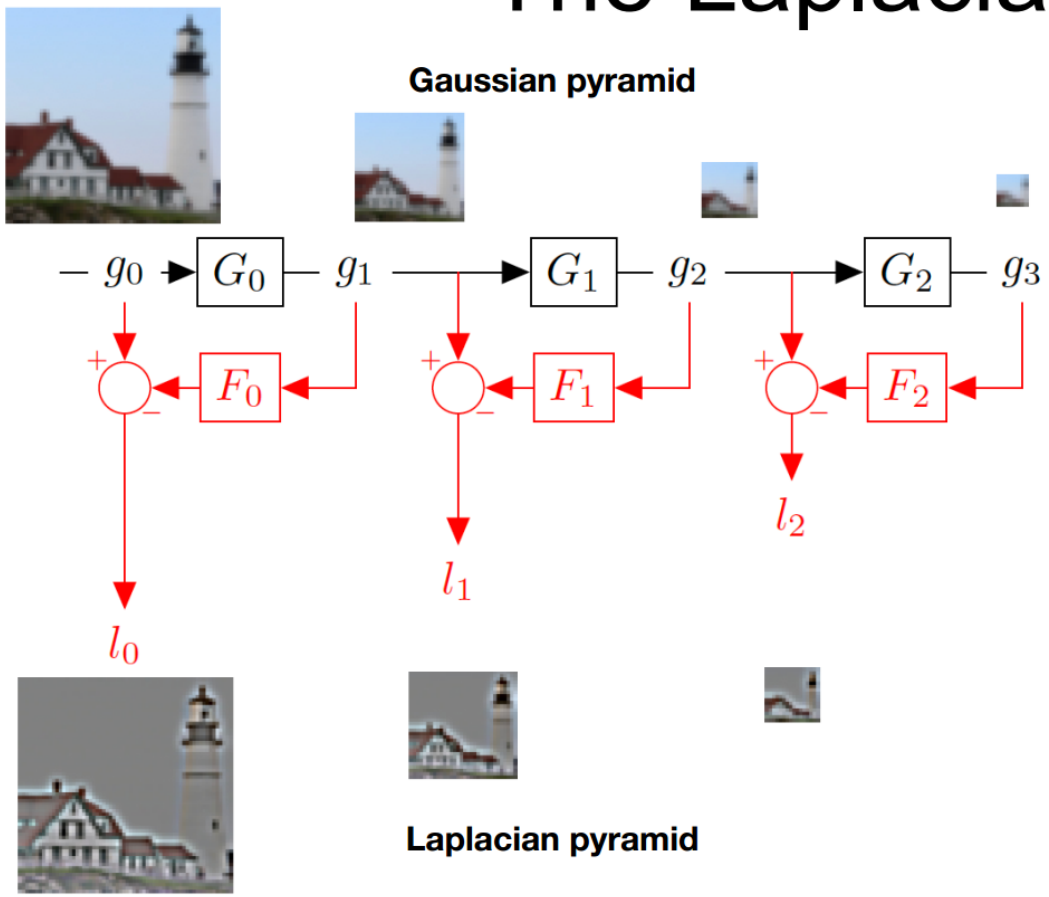
The Laplacian Pyramid

Gaussian residual

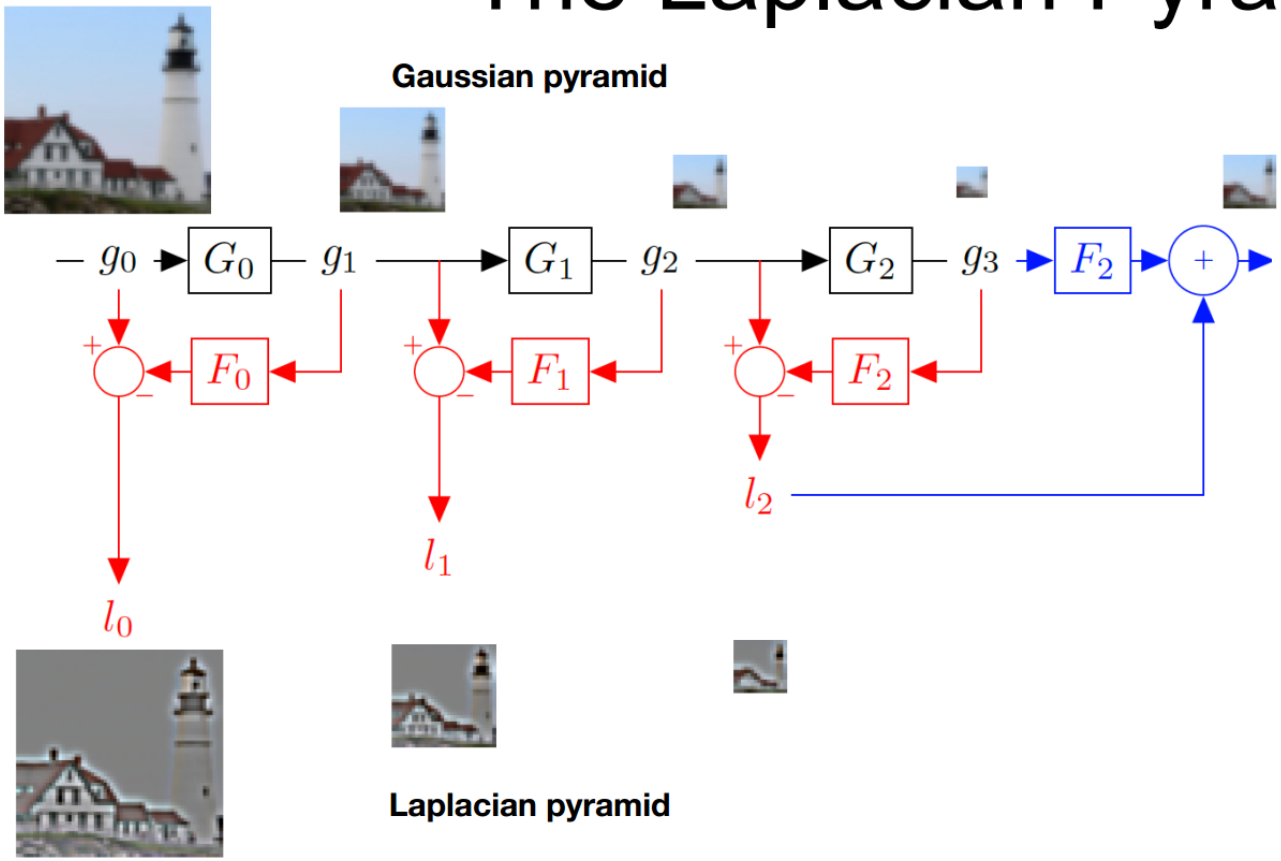
Laplacian pyramid

Can we invert the Laplacian Pyramid?

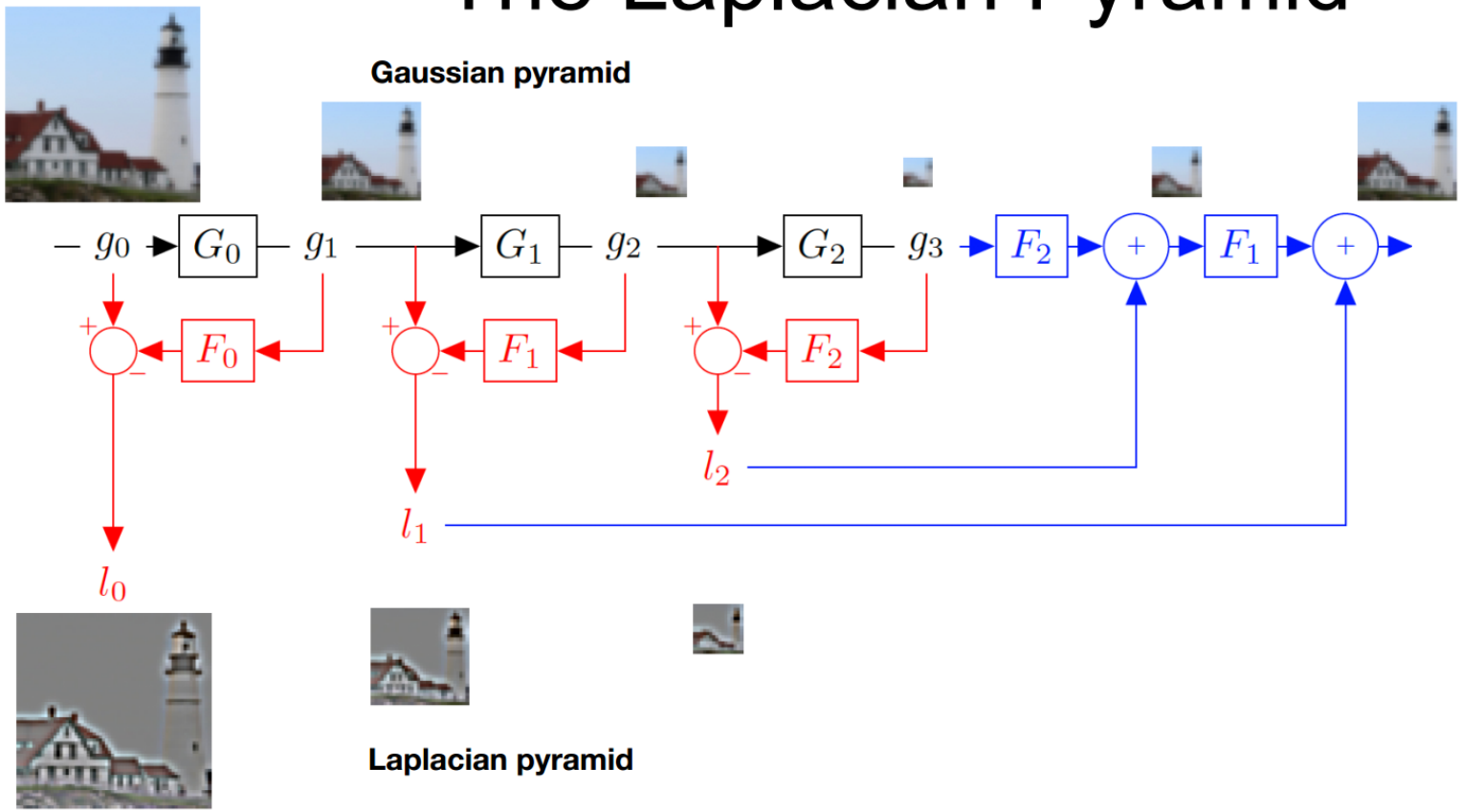
The Laplacian Pyramid



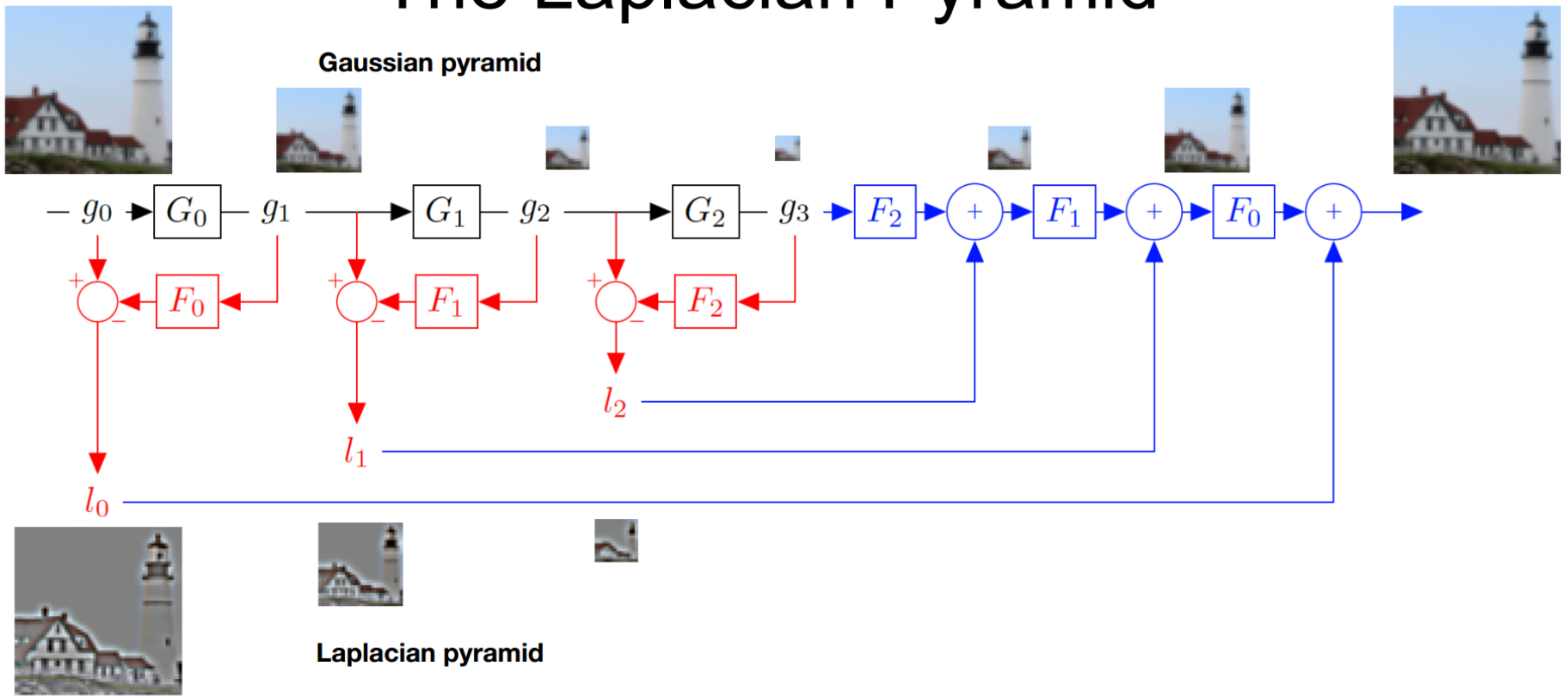
The Laplacian Pyramid



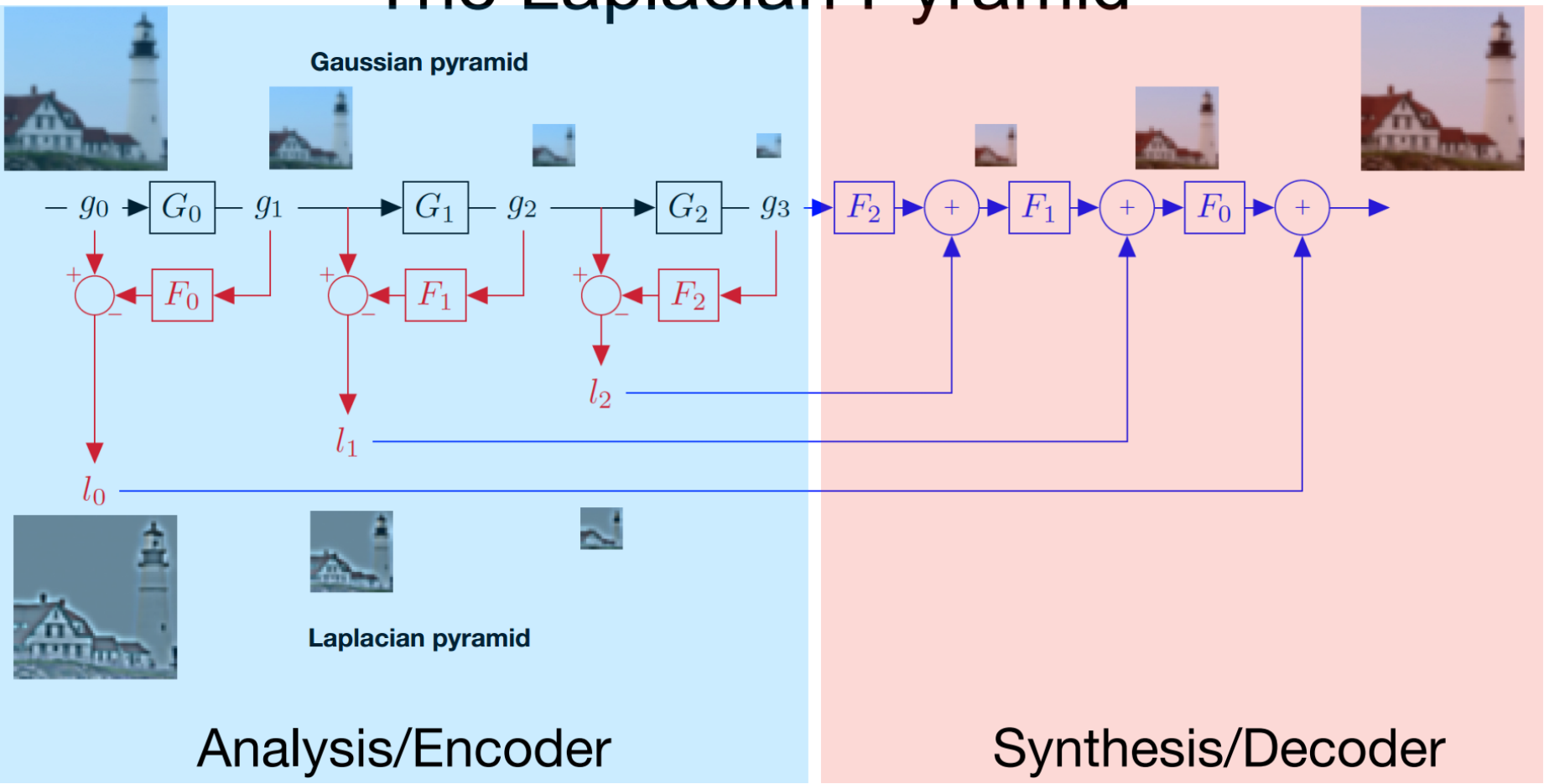
The Laplacian Pyramid



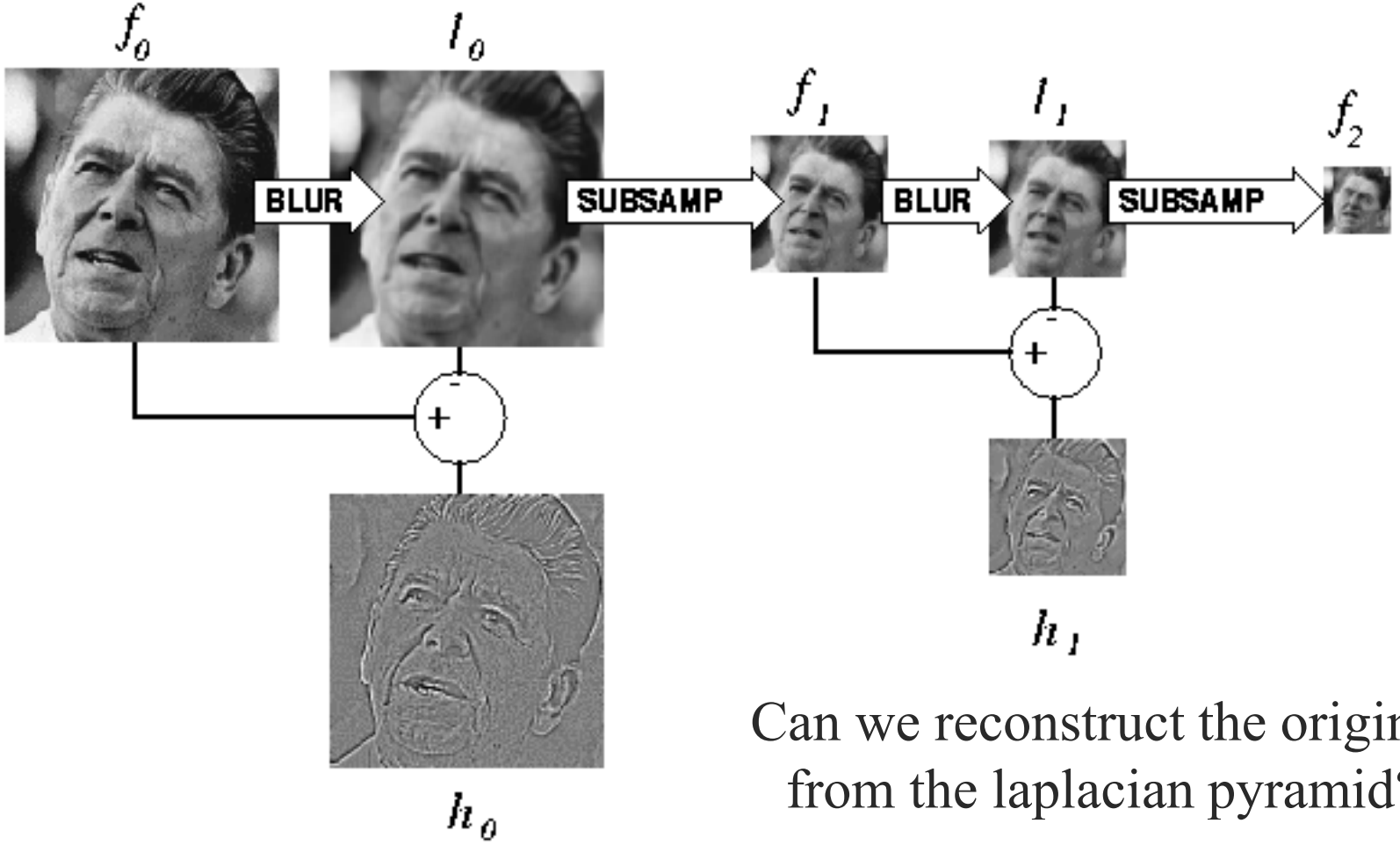
The Laplacian Pyramid



The Laplacian Pyramid



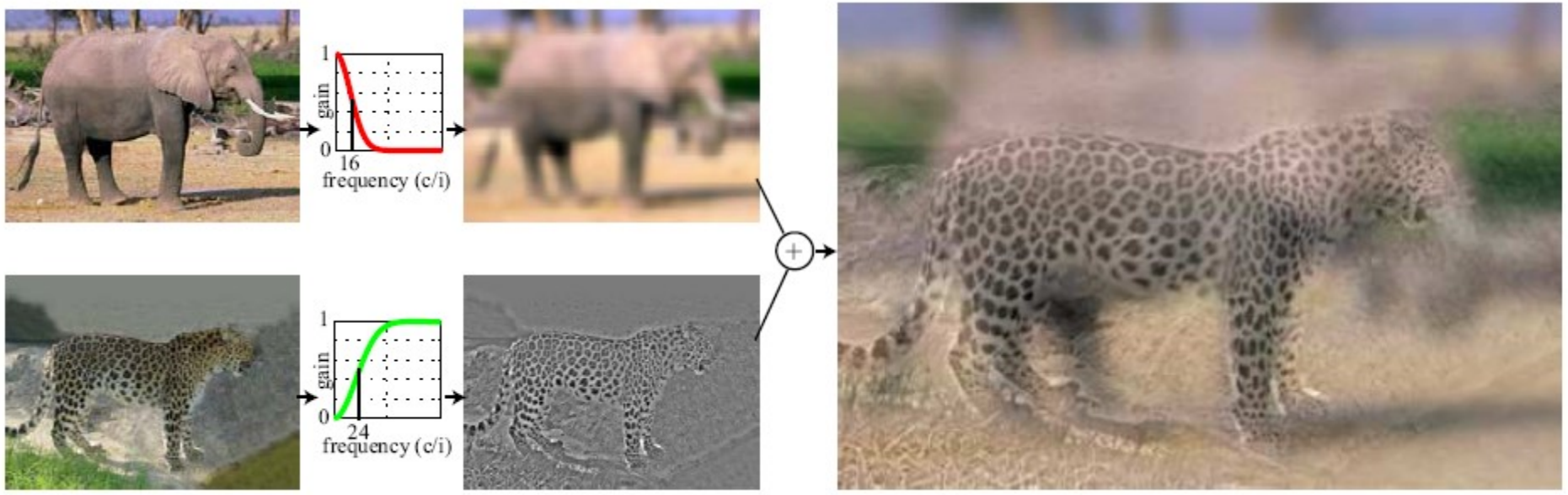
Computing Gaussian/Laplacian Pyramid



Can we reconstruct the original from the laplacian pyramid?

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Hybrid Image

Creating the Gaussian/Laplacian Pyramid

Image = G_1

Smooth, then downsample

Downsample (Smooth(G_1))

G_2

Downsample (Smooth(G_2))

G_3

$\ldots G_N = L_N$

G_1 - Smooth(Upsample(G_2))

L_1

L_2

G_2 - Smooth(Upsample(G_3))

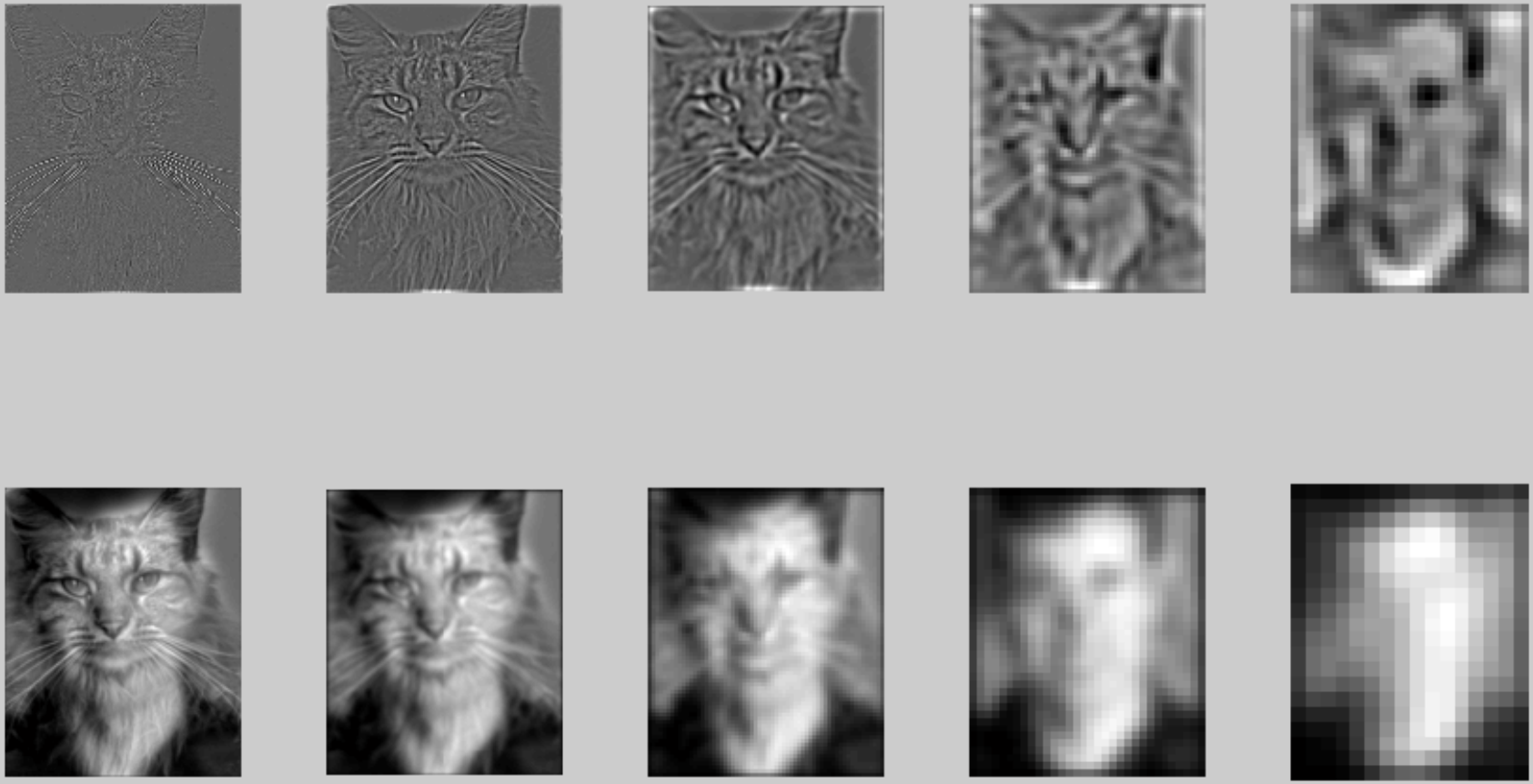
L_3

G_3 - Smooth(Upsample(G_4))

- Use same filter for smoothing in each step (e.g., Gaussian with $\sigma = 2$)
  - Downsample/upsample with "nearest" interpolation

Hybrid Image in Laplacian Pyramid

High frequency → Low frequency



Reconstructing image from Laplacian pyramid

Image = L_1 + Smooth(Upsample(G_2))

$G_2 = L_2$ + Smooth(Upsample(G_3))

$G_3 = L_3$ + Smooth(Upsample(L_4))

L_4

L_1

L_2

L_3

- Use same filter for smoothing as in deconstruction
  - Upsample with “nearest” interpolation
  - Reconstruction will be nearly lossless

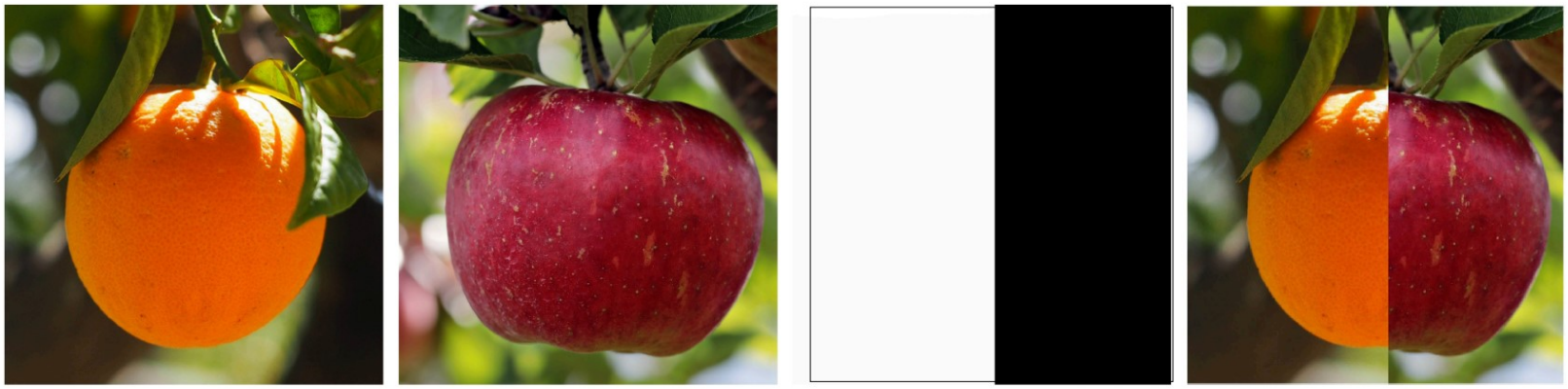
Laplacian pyramid applications

- Texture synthesis
- Image compression
- Noise removal
- Computing image features (e.g., SIFT)
- Image Blending…

Image Blending

Image Blending

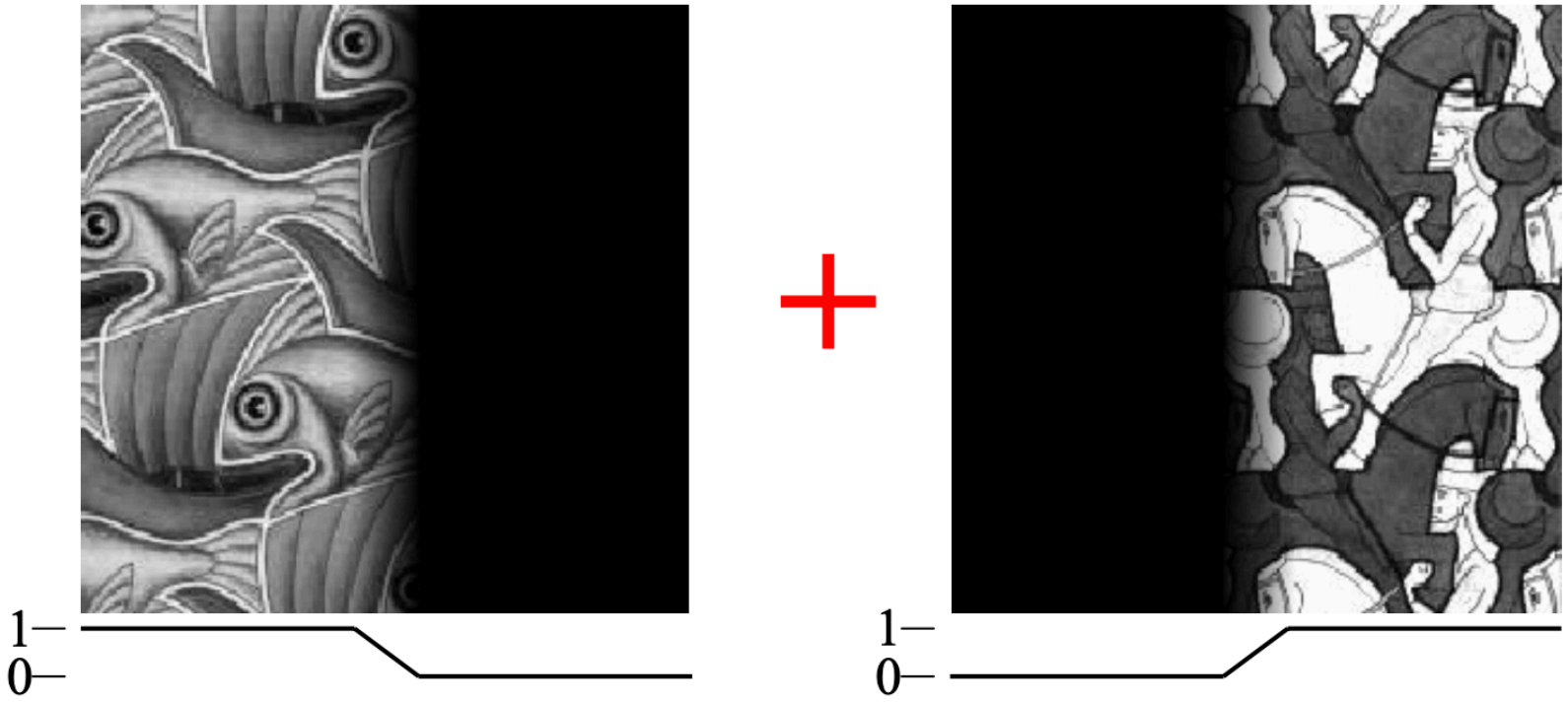
Image Blending



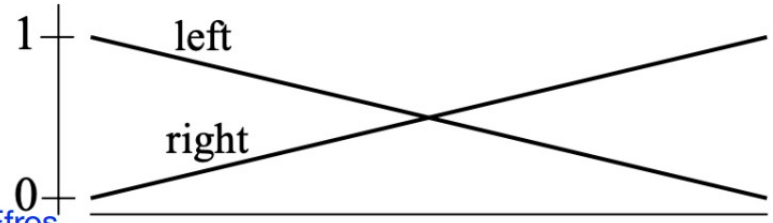
I^A I^B m I

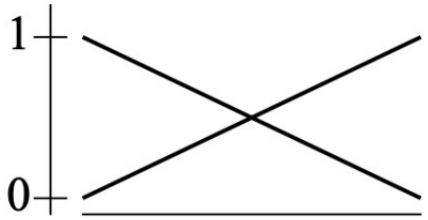
$$I = m * I^A + (1 - m) * I^B$$

Feathering



Affect of window size





Slide by A. Efros

Affect of window size

Slide by A. Efros

Good Window Size

"Optimal" Window: smooth but not ghosted

What is the Optimal Window

To avoid seams

- window >= size of largest prominent feature

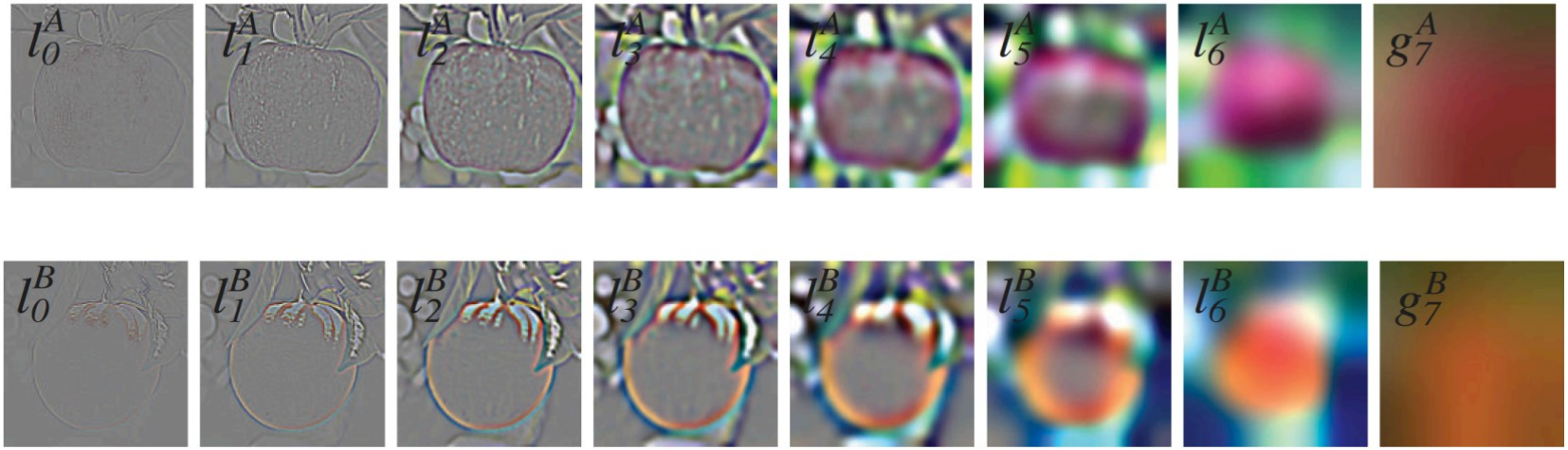
To avoid ghosting

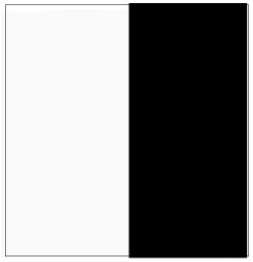
- window <= 2*size of smallest prominent feature

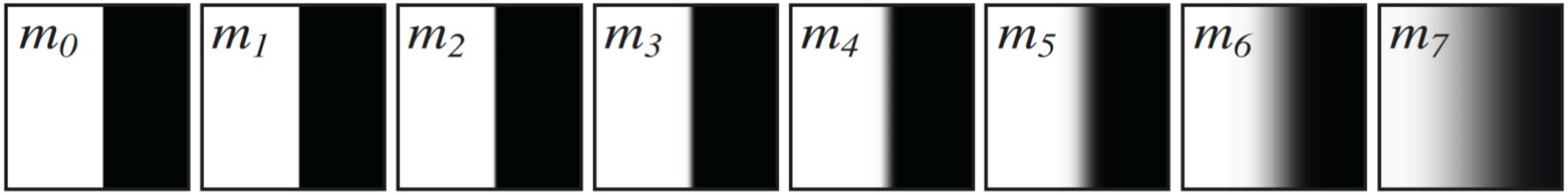
Natural to cast this in the *Fourier domain*

- largest frequency <= 2*size of smallest frequency
- image frequency content should occupy one “octave” (power of two)

Image Blending with the Laplacian Pyramid





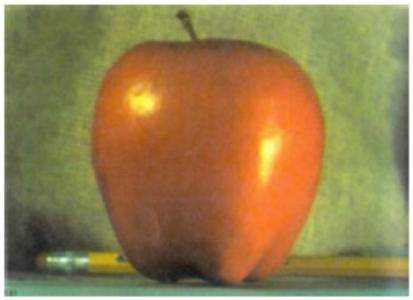


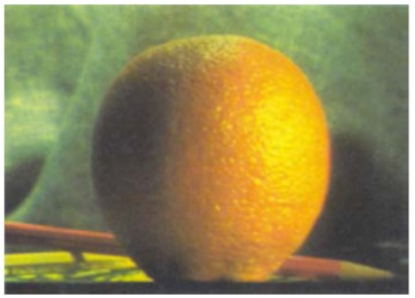
$$l_k = l_k^A * m_k + l_i^B * (1 - m_k)$$

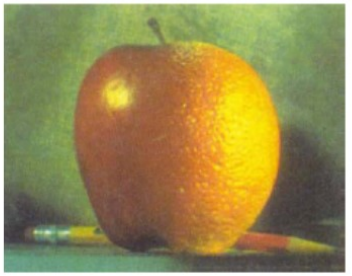
Image Blending with the Laplacian Pyramid

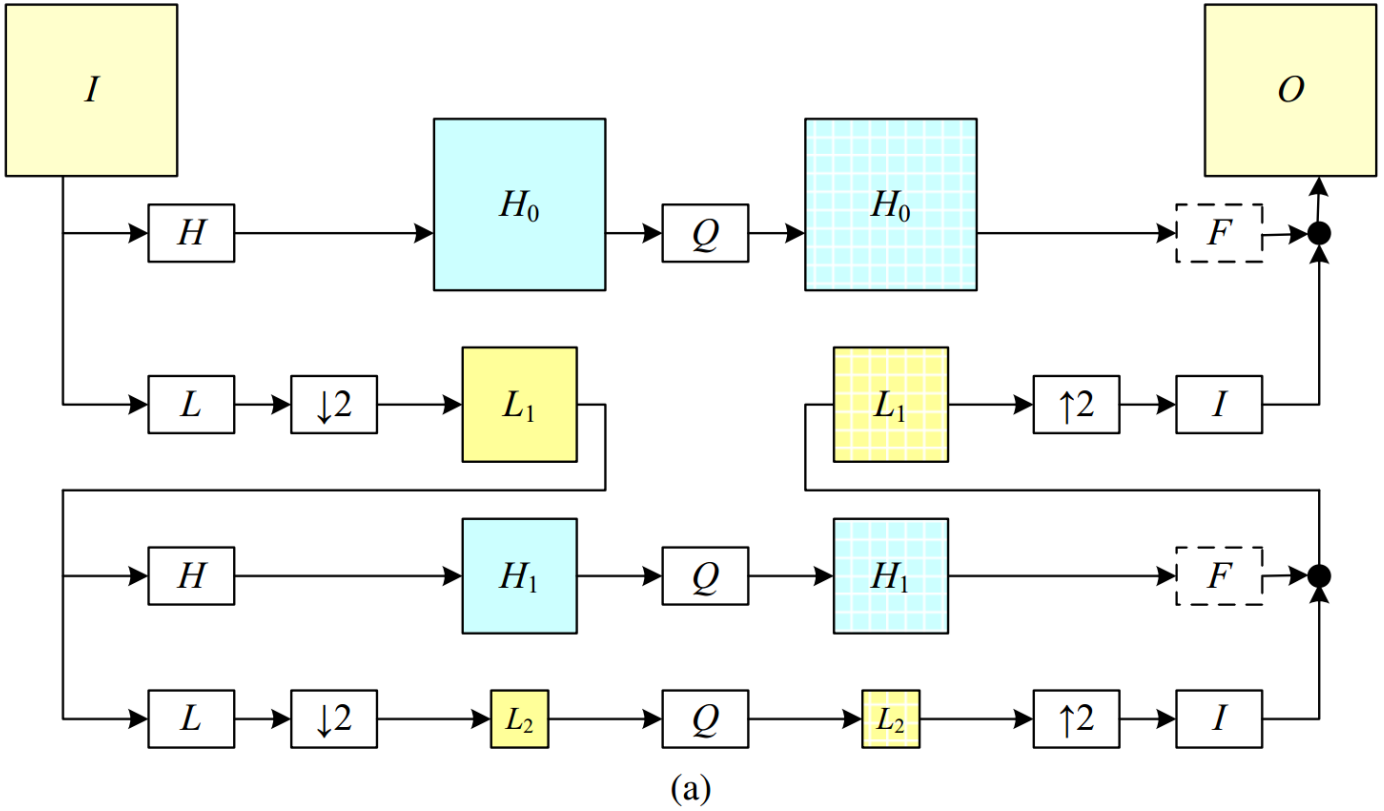
Image Blending with the Laplacian Pyramid

- Build Laplacian pyramid for both images: LA, LB
- Build Gaussian pyramid for mask: G
- Build a combined Laplacian pyramid:
- Collapse L to obtain the blended image

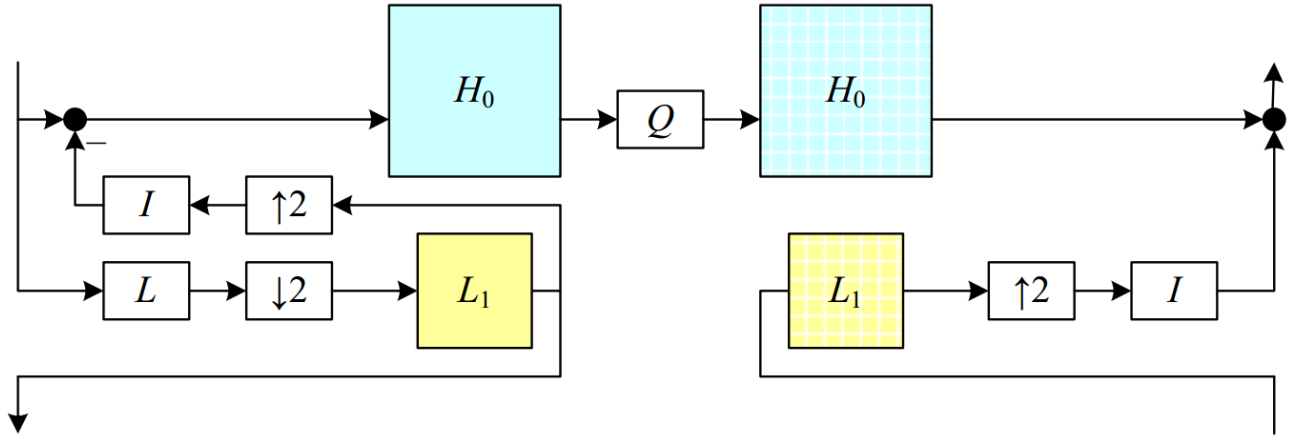








(a)



(b)

Image pyramids

- Gaussian

Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian

Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

Today's class

- Template matching
- Gaussian pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian pyramids
  - Hybrid images
  - Application for image blending
- **Steerable pyramids**
  - Steerable filters
  - Orientation analysis
- Human vision system
  - Visual perception
- Filter Banks and Texture Analysis

Orientations

Steerable Pyramid

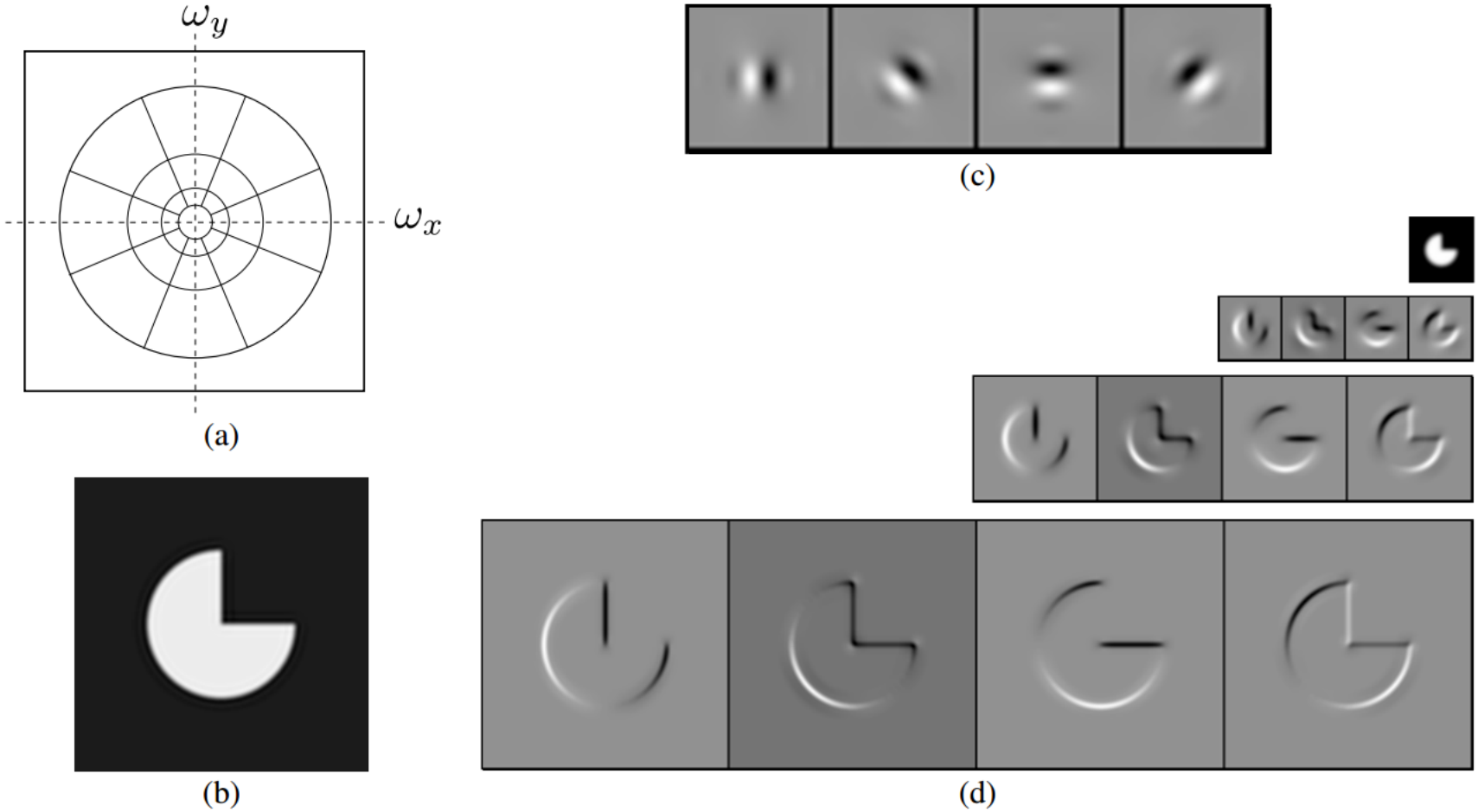
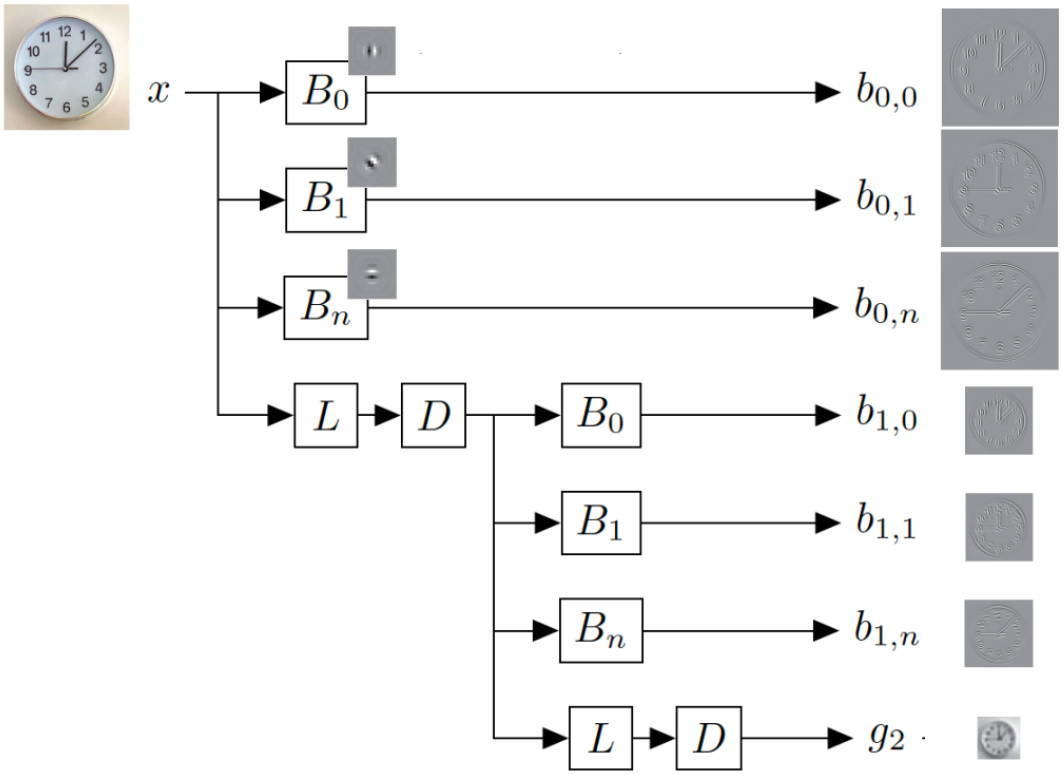
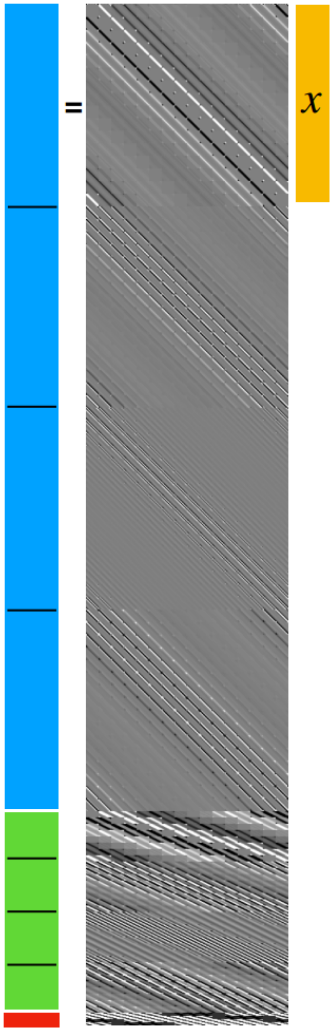


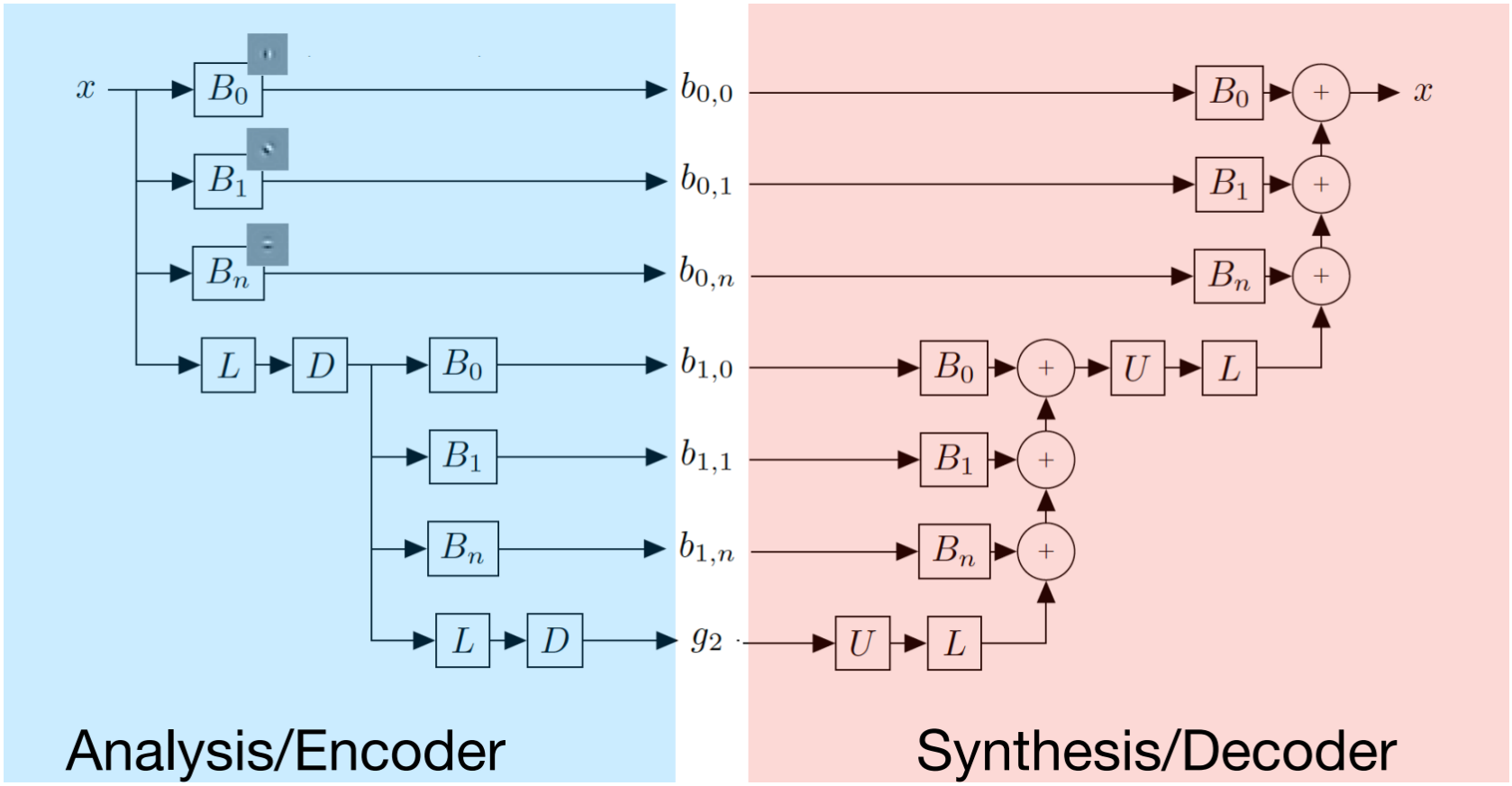
Figure 3.40 Steerable shiftable multiscale transforms (Simoncelli, Freeman, Adelson *et al.* 1992) © 1992 IEEE: (a) radial multi-scale frequency domain decomposition; (b) original image; (c) a set of four steerable filters; (d) the radial multi-scale wavelet decomposition.

Steerable Pyramid





Steerable Pyramid



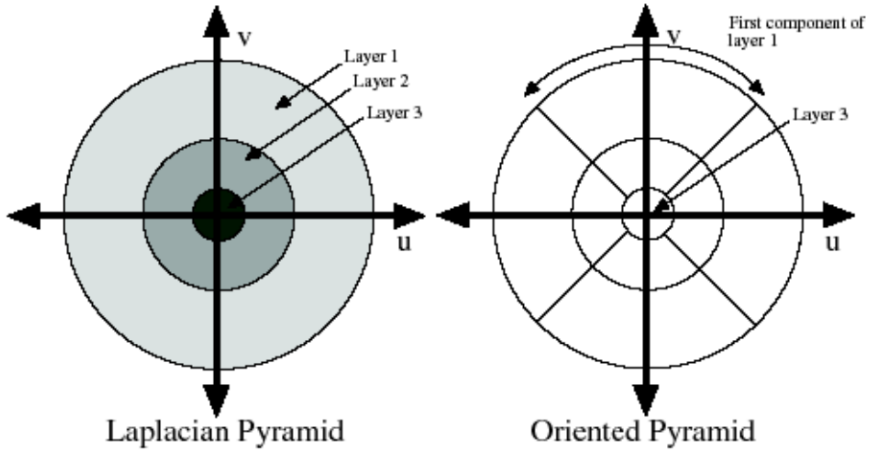
The steerable pyramid

-Preserves all image information (we can go back to the image)

-Provides more independent channels of information than pixel values (we can mess with each band independently)

Oriented Pyramids in spectral space

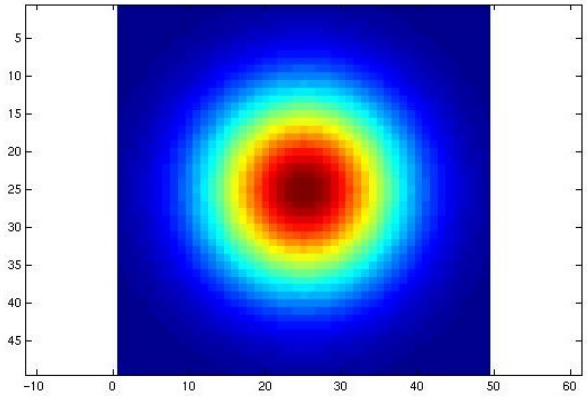
- Apply an oriented filter to determine orientations at each layer
  - Important for texture synthesis
  - this represents image information at a particular scale and orientation, similar to V1 cells in the human brain



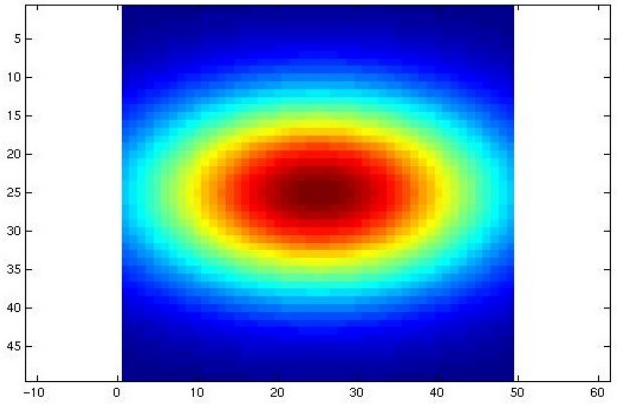
An oriented pyramid cut each annulus into a set of wedges. If (u,v) fourier space is represented in polar coordinates, each wedge corresponds to an interval of radius values and an interval of angle values.

Multivariate Gaussian

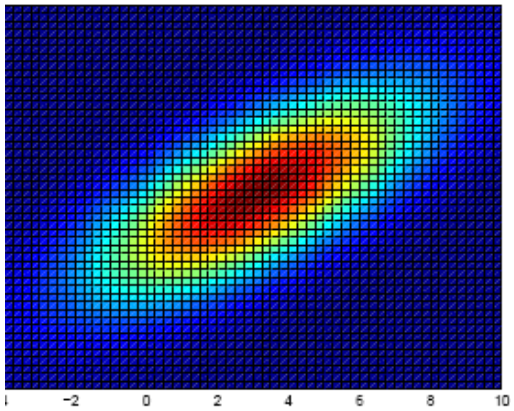
$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$



$$\Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix}$$



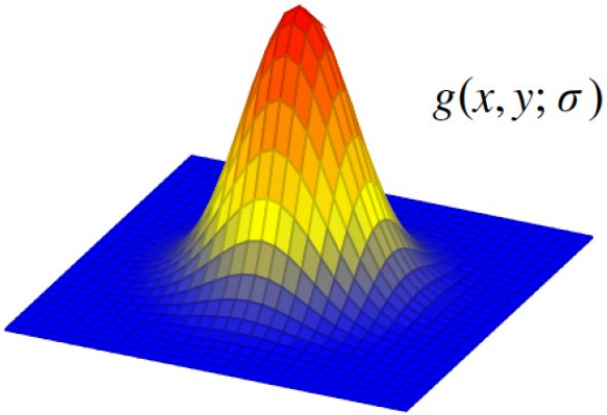
$$\Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

Slide credit:
Kristen Grauman

Gaussian Derivative



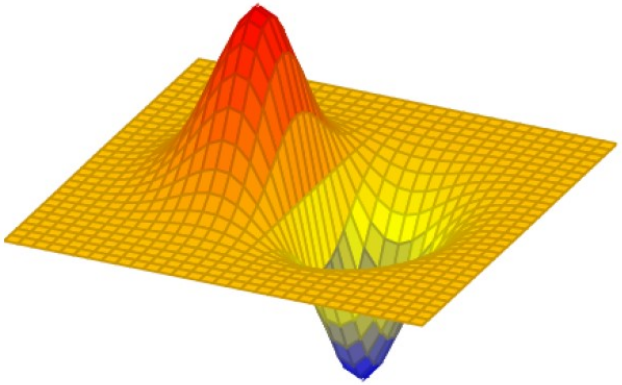
$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

The continuous derivative is:

$$g_x(x, y; \sigma) = \frac{\partial g(x, y; \sigma)}{\partial x} =$$

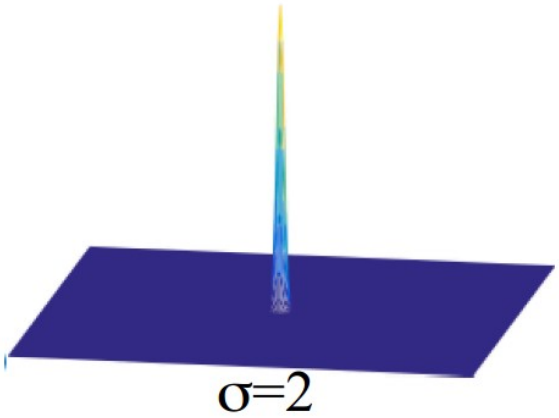
$$= \frac{-x}{2\pi\sigma^4} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

$$= \frac{-x}{\sigma^2} g(x, y; \sigma)$$

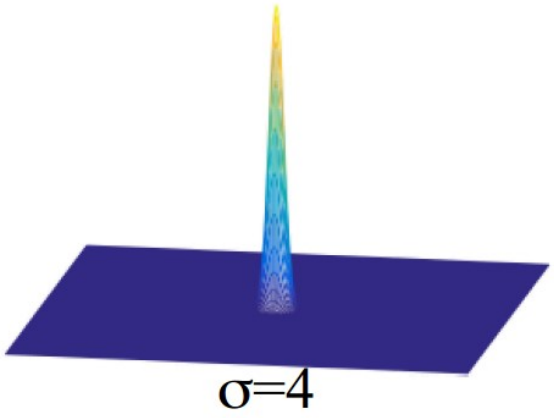


104

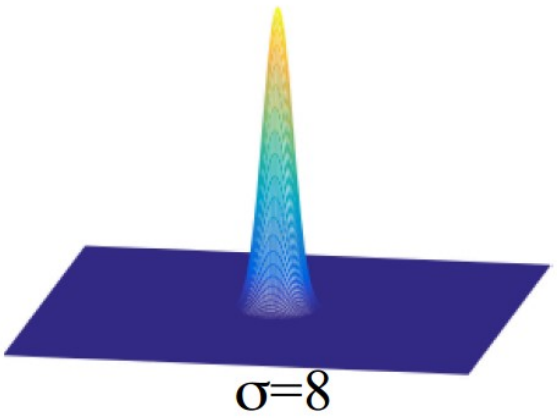
Gaussian Scale



σ=2

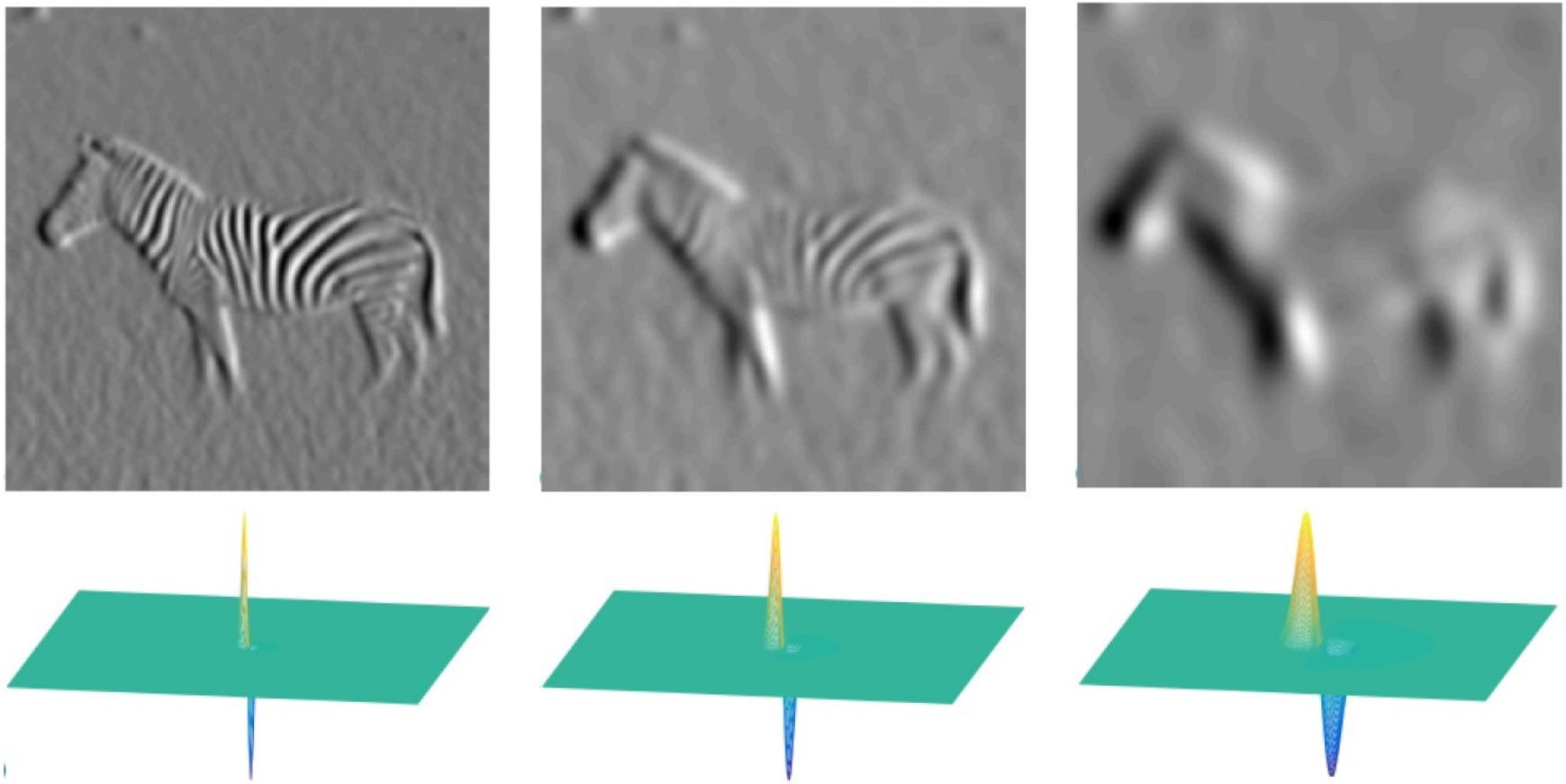


σ=4



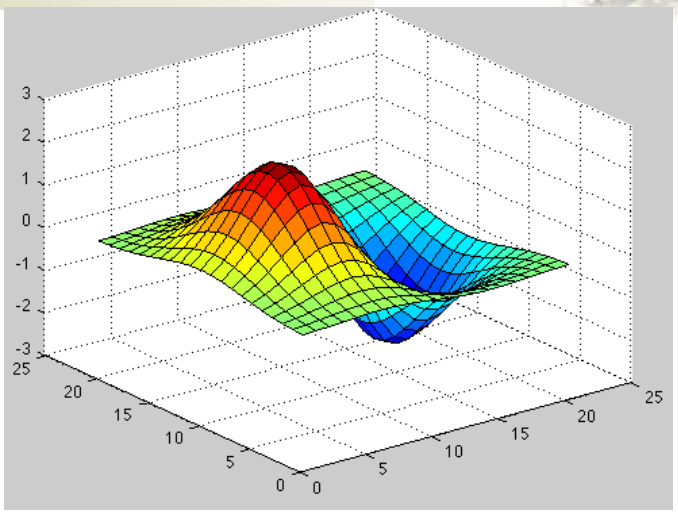
σ=8

Derivatives of Gaussian: Scale

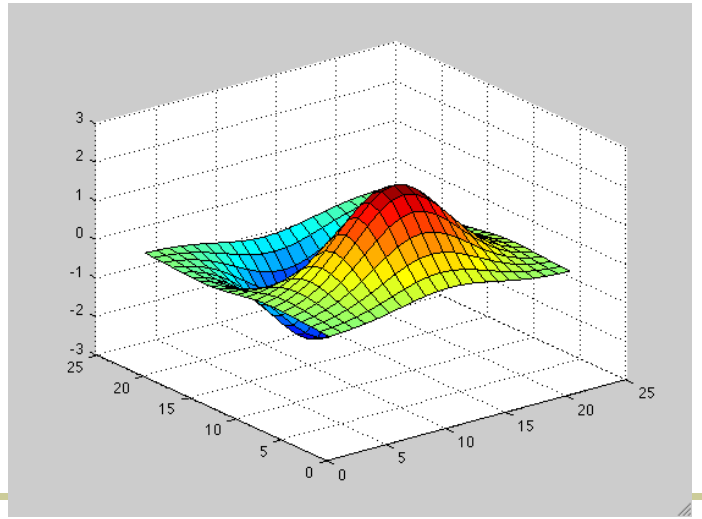


Orientation

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

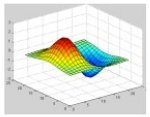


$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

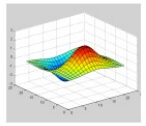


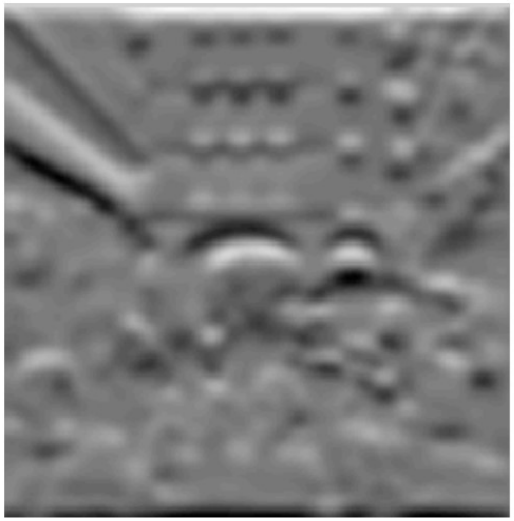
Orientation

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





What about other orientations not axis aligned?

Why are they steerable filter

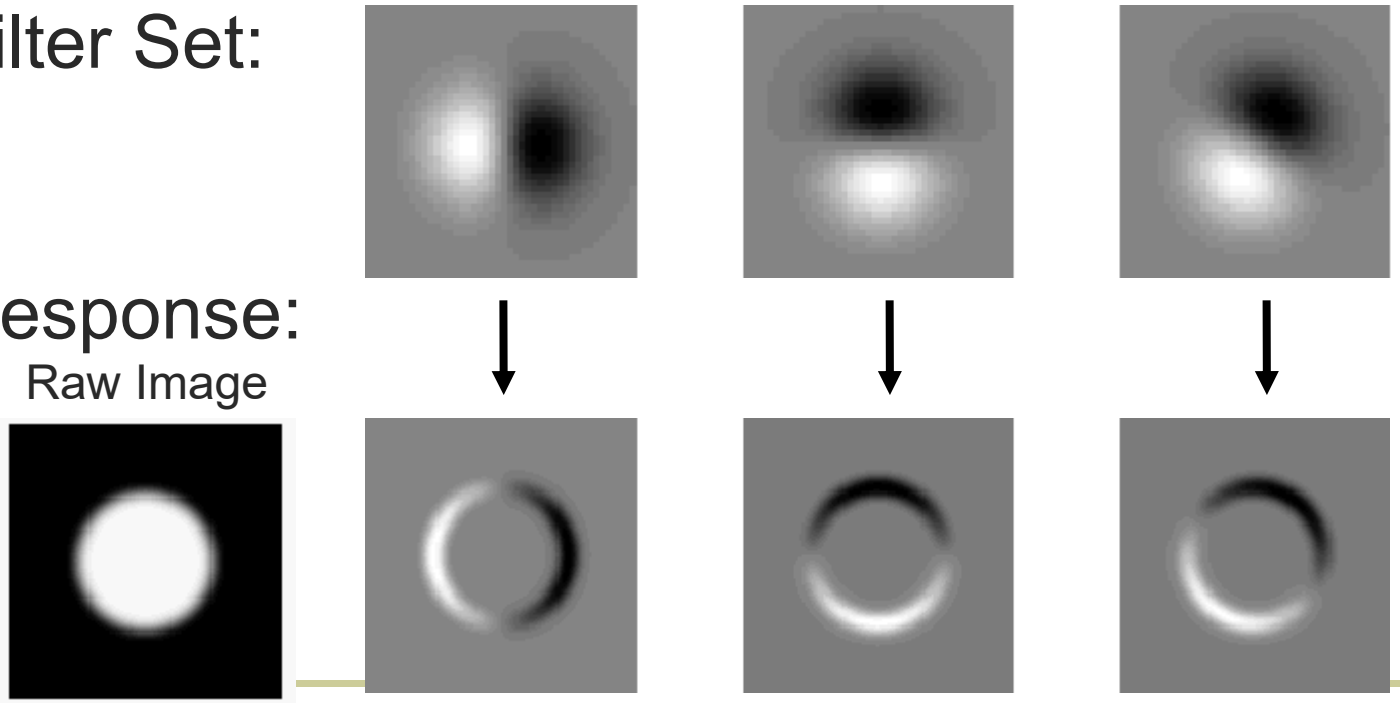
"Steerability"-- the ability to synthesize a filter of any orientation from a linear combination of filters at fixed orientations.

$$G_\theta^1 = \cos(\theta)G_0^1 + \sin(\theta)G_{90}^1$$

Filter Set:

Response:

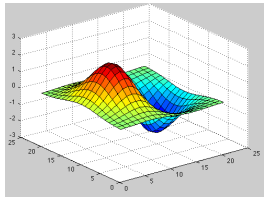
Raw Image



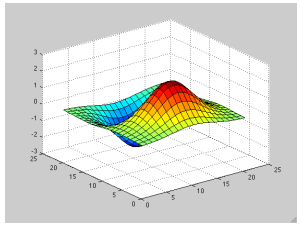
Taken from:
W. Freeman, T. Adelson, "The Design and Use of Sterrable Filters", IEEE Trans. Patt, Anal. and Machine Intell., vol 13, #9, pp 891-900, Sept 1991

Steerable filters

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

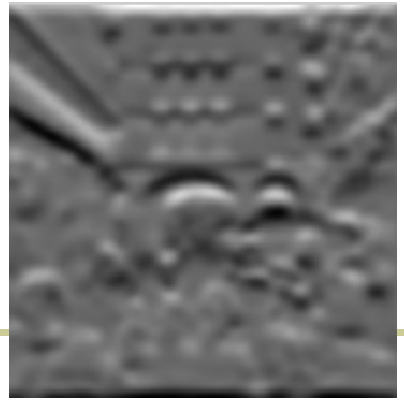


The smoothed directional gradient is a linear combination of two kernels

$$u^T \nabla g \otimes I = \left(\cos(\alpha) g_x(x,y) + \sin(\alpha) g_y(x,y)\right) \otimes I(x,y) =$$

Any orientation can be computed as a linear combination of two filtered images

$$= \cos(\alpha) g_x(x,y) \otimes I(x,y) + \sin(\alpha) g_y(x,y) \otimes I(x,y) =$$

$= \cos(\alpha)$  $+\sin(\alpha)$  $=$ 

Steereability of gaussian derivatives, Freeman & Adelson 92

Orientation analysis

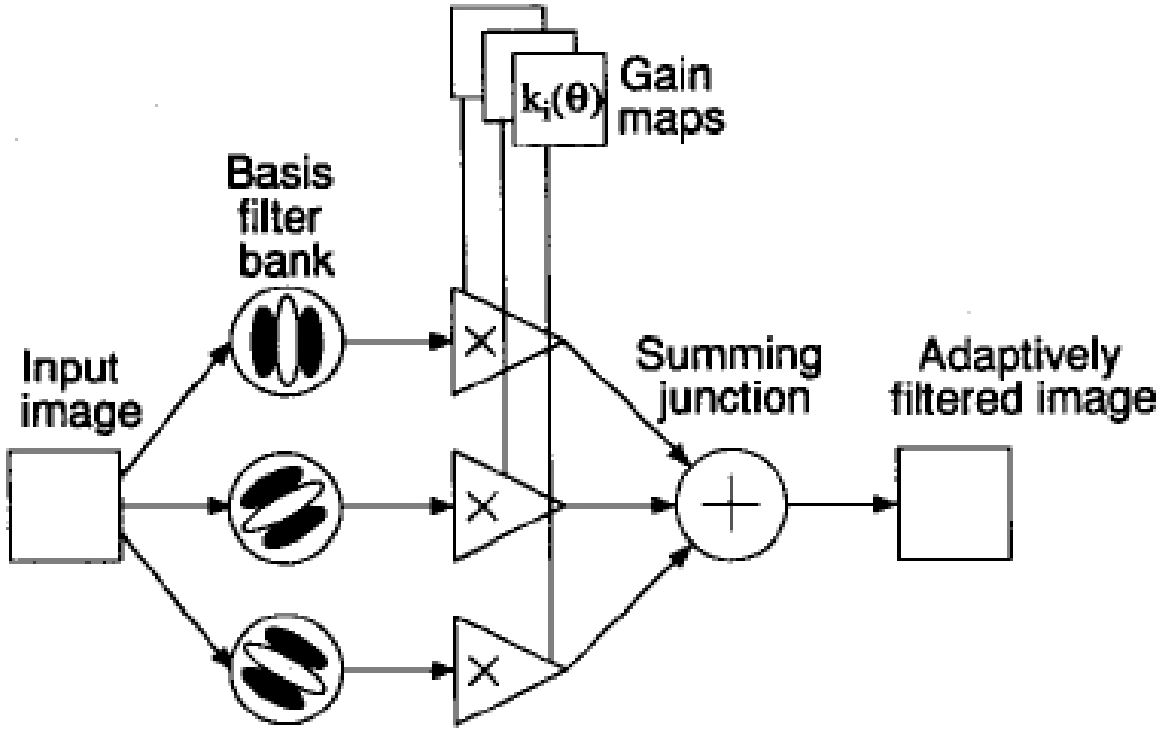
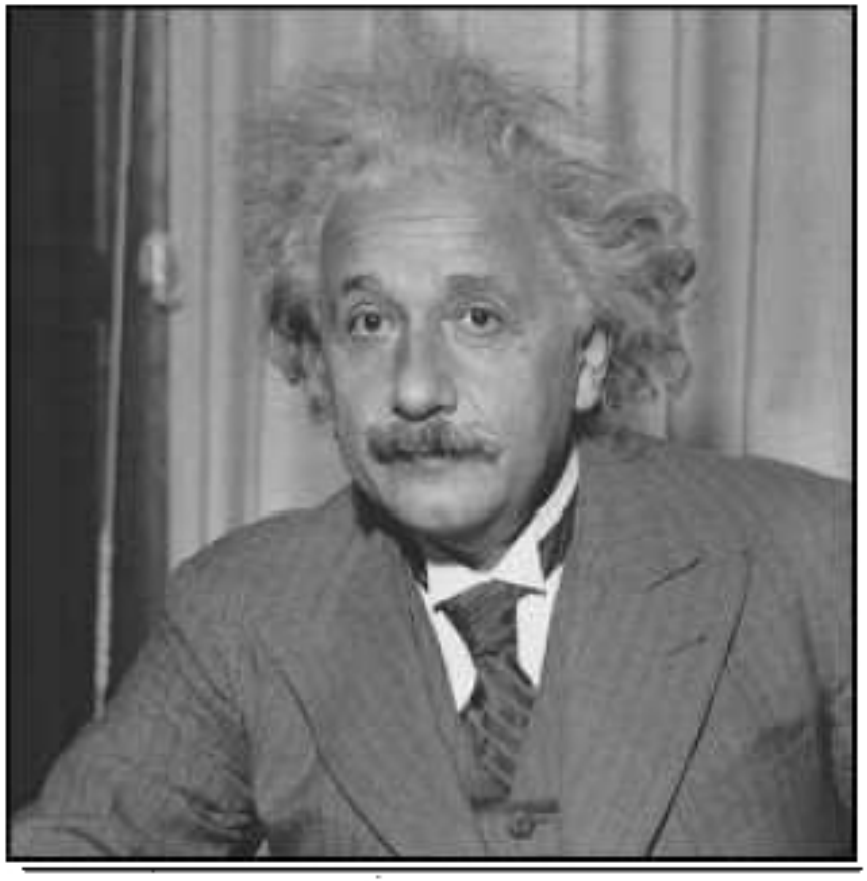


Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.

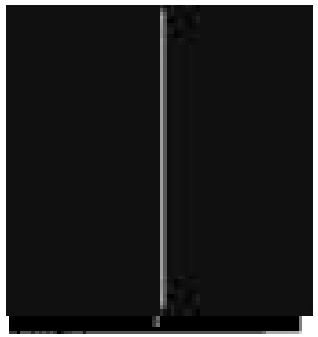
Orientation analysis



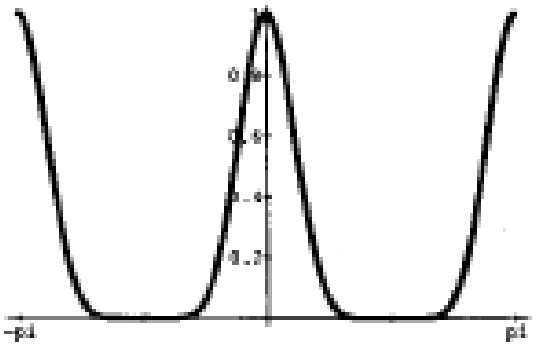
(a)

(b)

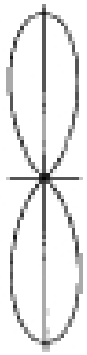
Orientation analysis



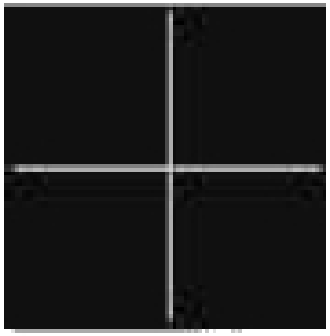
(a)



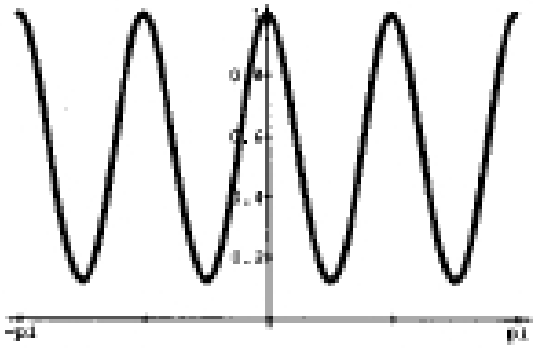
(c)



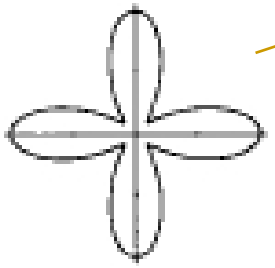
(e)



(b)



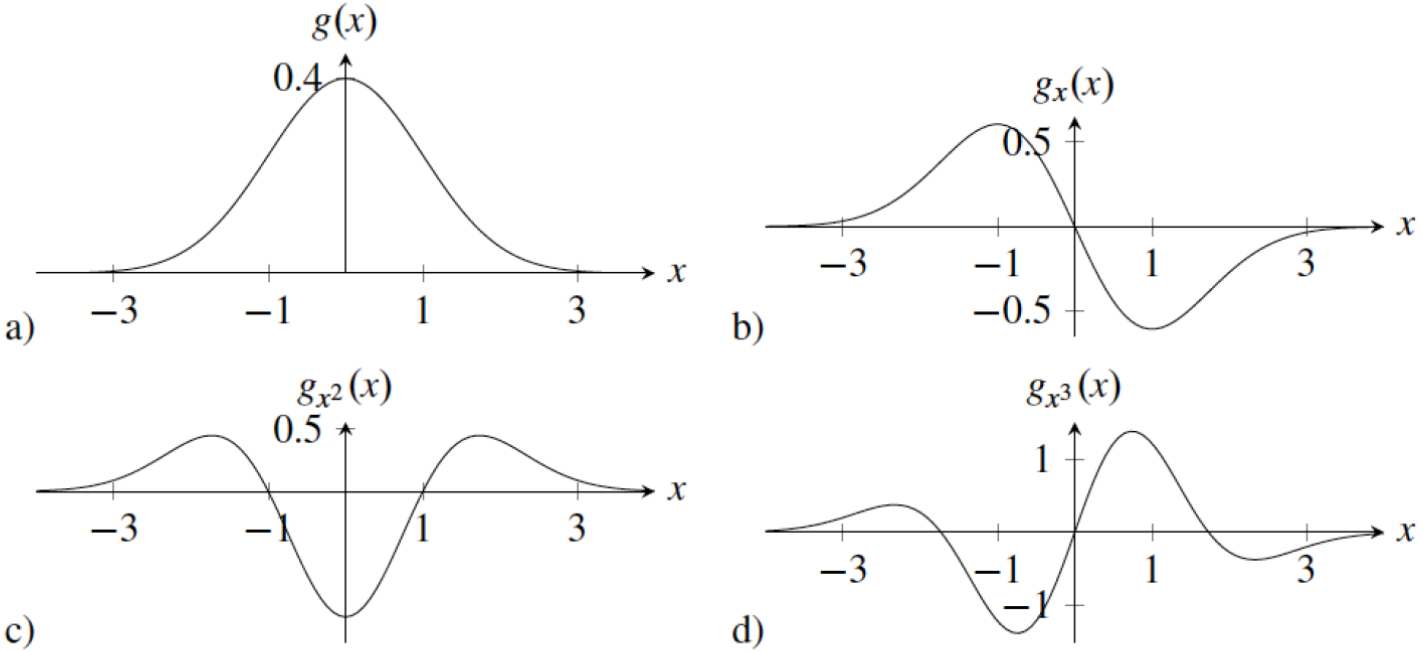
(d)



(f)

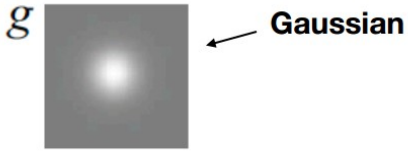
High resolution in orientation requires many oriented filters as basis (high order gaussian derivatives or fine-tuned Gabor wavelets).

N-th order Gaussian derivatives



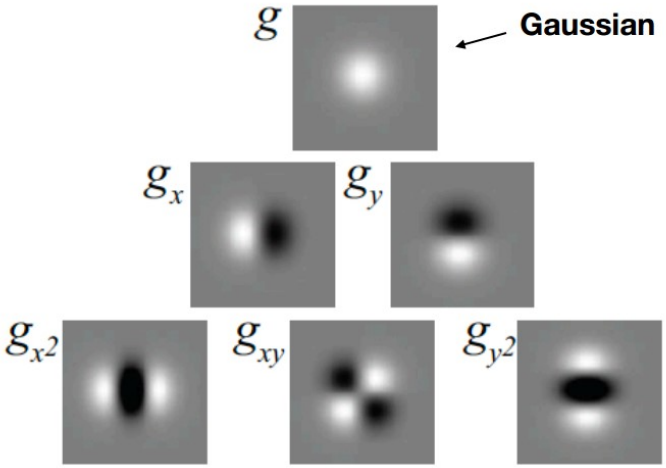
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

N-th order Gaussian derivatives



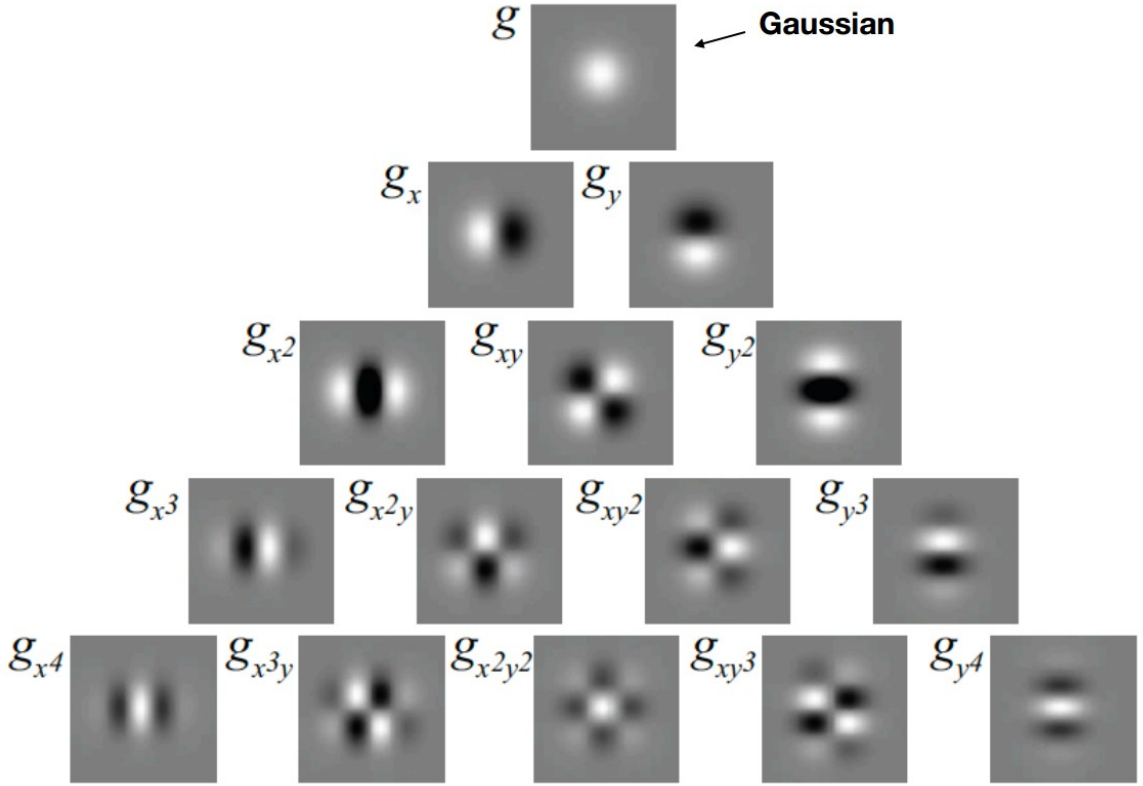
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

N-th order Gaussian derivatives



$$g_{x^n y^m}(x, y; \sigma) = \frac{\partial^{n+m} g(x, y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x, y; \sigma)$$

N-th order Gaussian derivatives



$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

Steerable pyramid applications

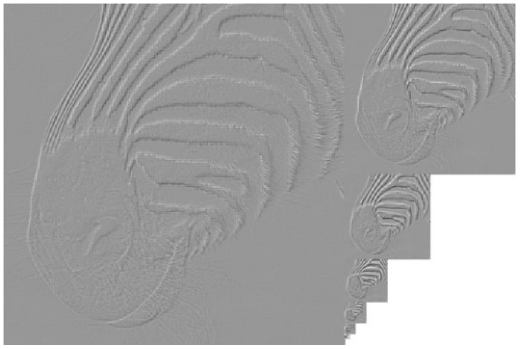
- Texture synthesis
- Noise removal
- Motion analysis
- Motion synthesis, motion magnification

Summary: Image pyramids

- Gaussian

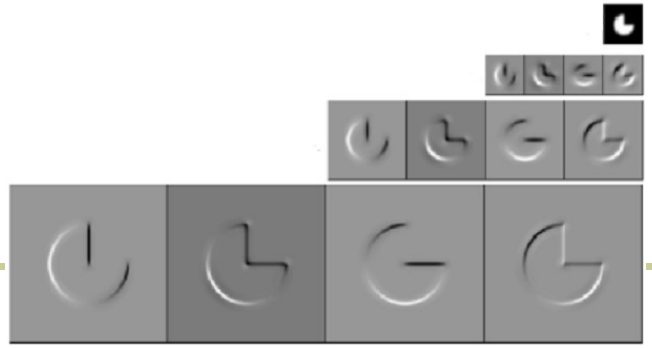
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian



Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Steerable pyramid



Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis.

Linear Image Transforms

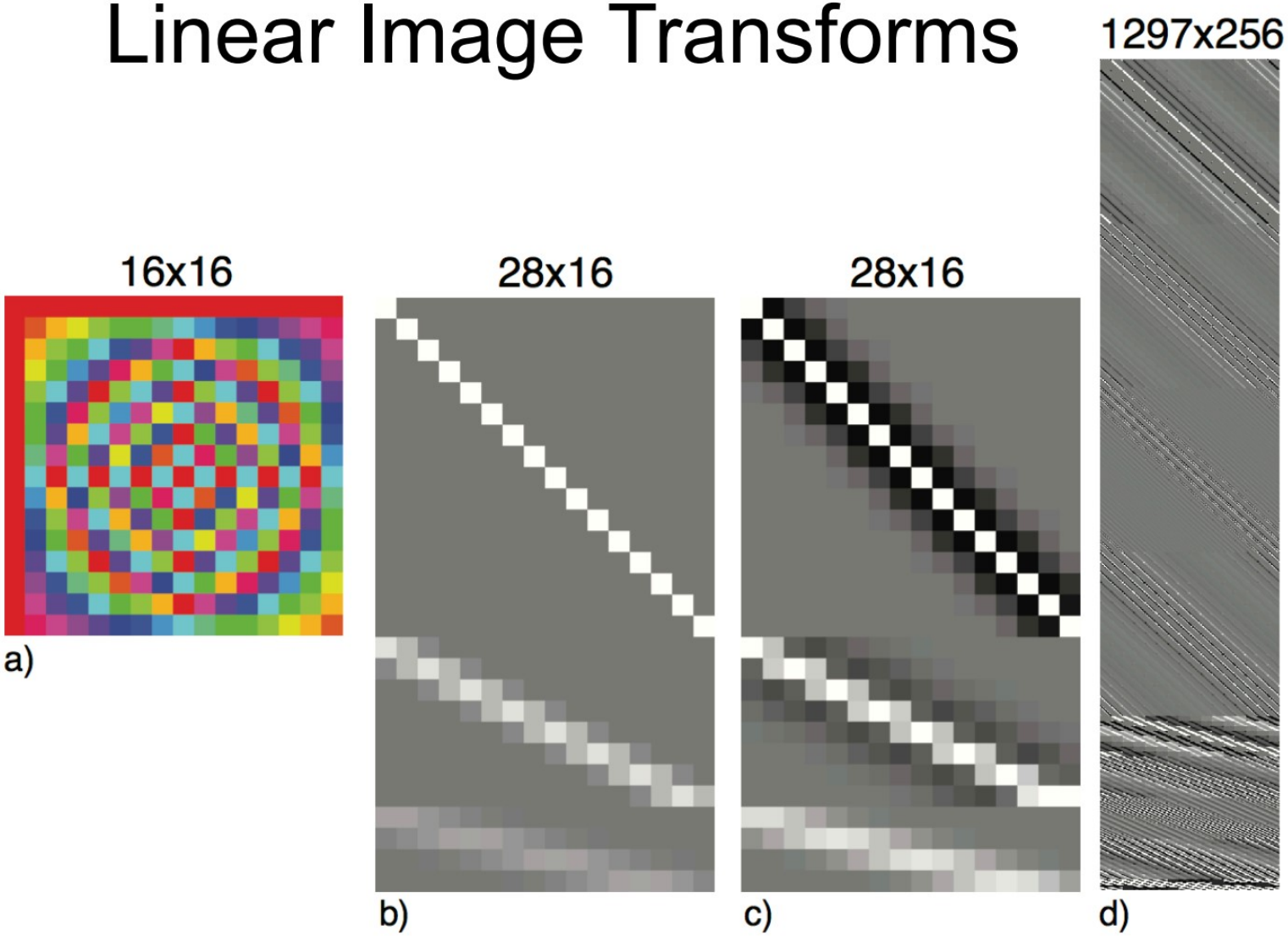


Image representation

- Pixels: great for spatial resolution, poor access to frequency
- Fourier transform: great for frequency, not for spatial information
- Pyramids/filter banks: balance between spatial and frequency information

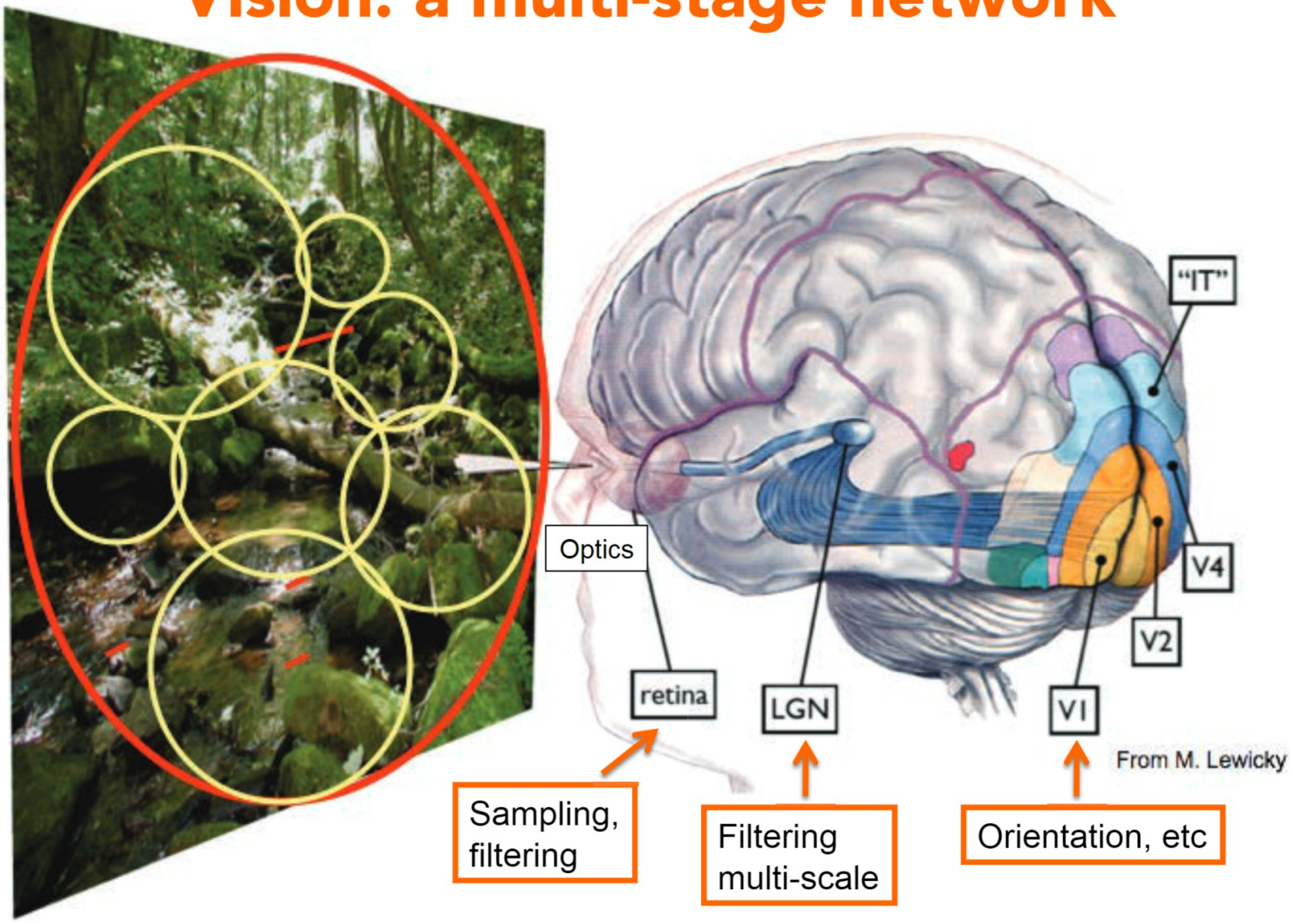
Why use these representations?

- Handle real-world size variations with a constant-size vision algorithm.
- Remove noise
- Analyze texture
- Recognize objects, scenes
- Label image features

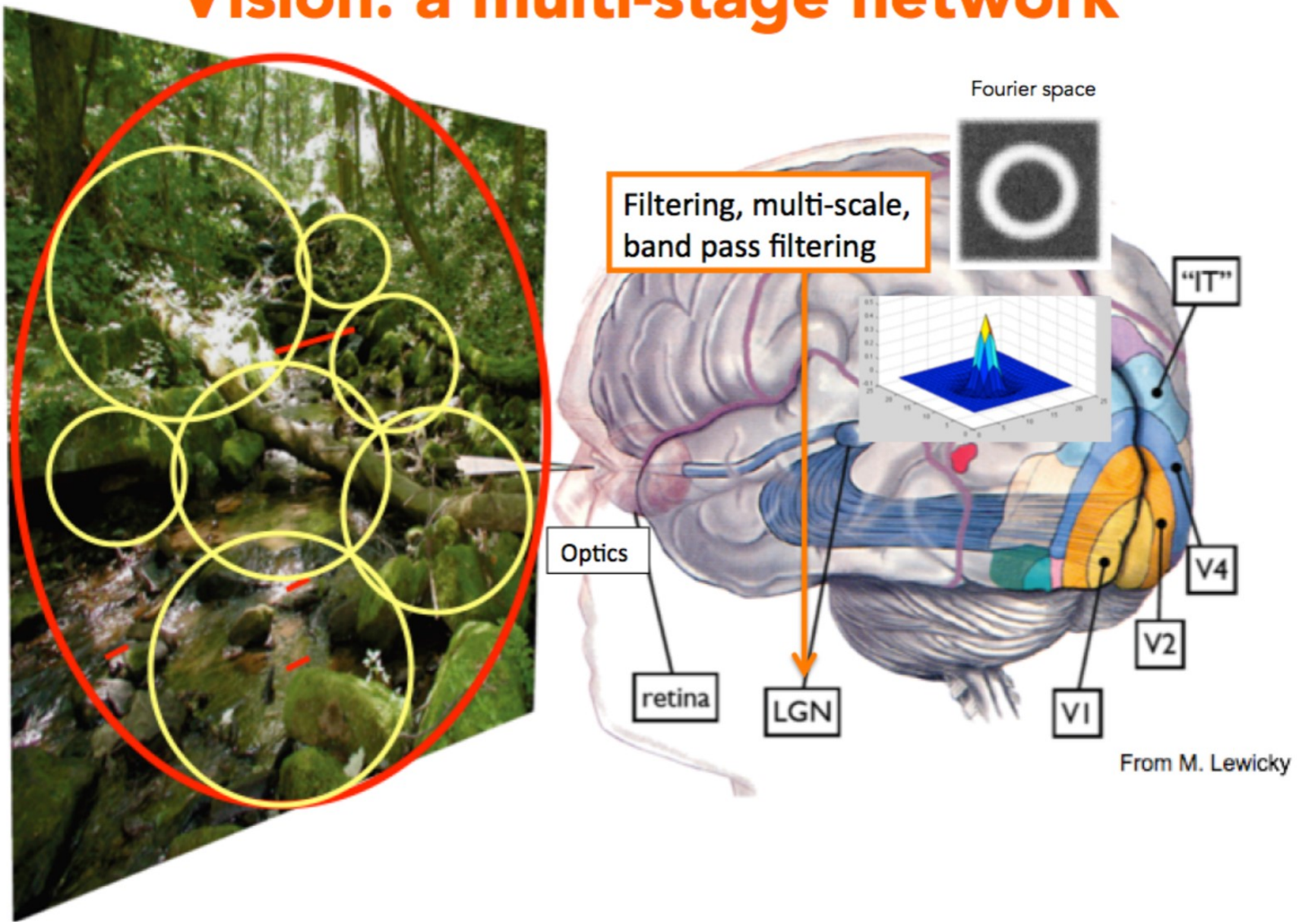
Today's class

- Template matching
- Gaussian pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian pyramids
  - Hybrid images
  - Application for image blending
- Steerable pyramids
  - Steerable filters
  - Orientation analysis
- **Human vision system**
  - Visual perception
- Filter Banks and Texture Analysis

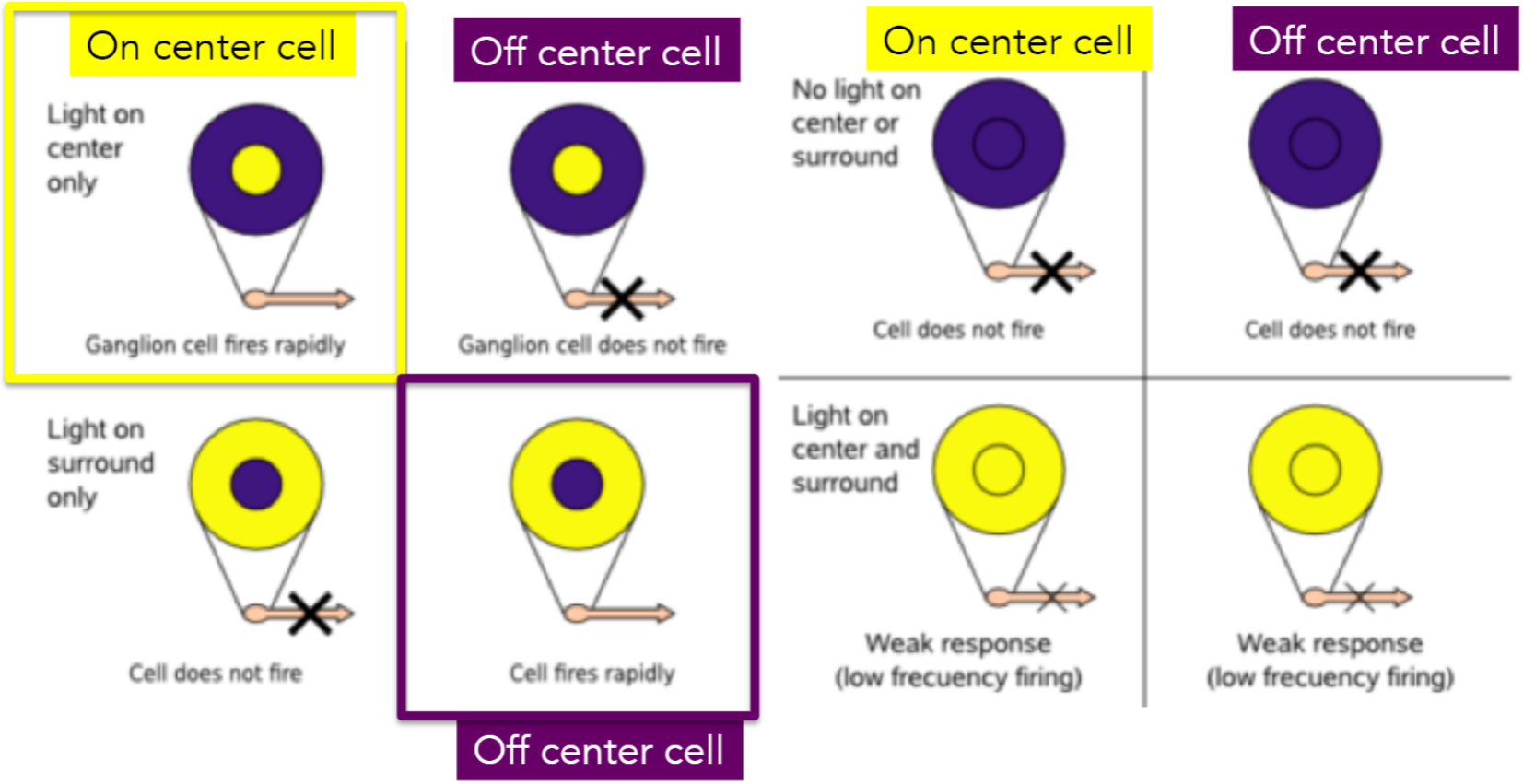
Vision: a multi-stage network



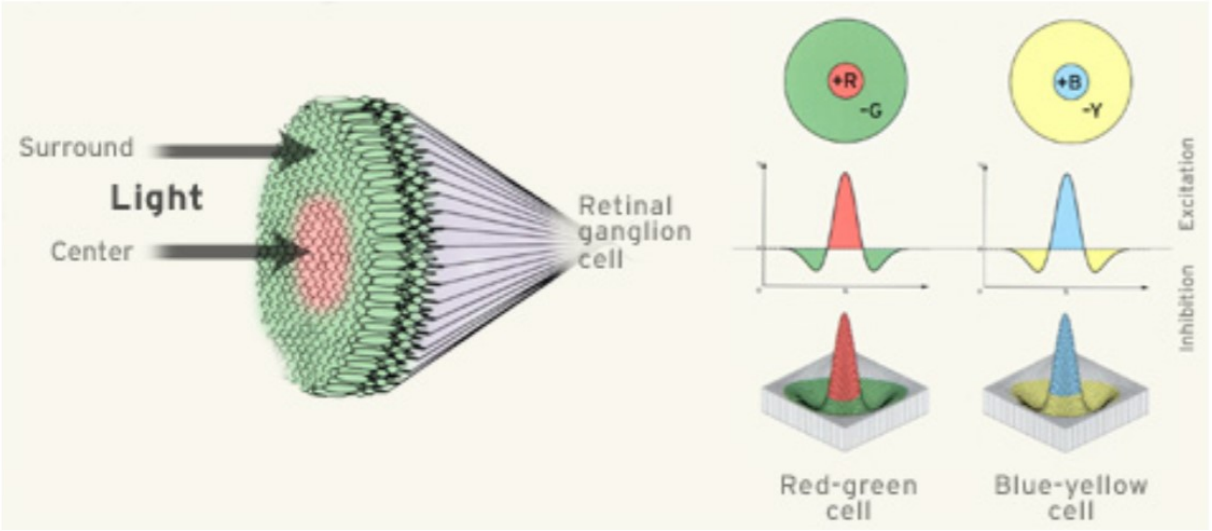
Vision: a multi-stage network

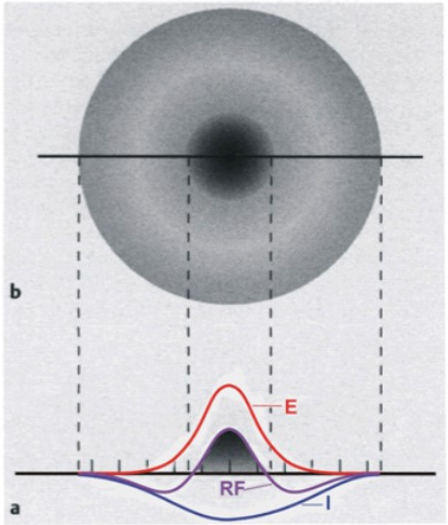


Model of Retinal and LGN cells



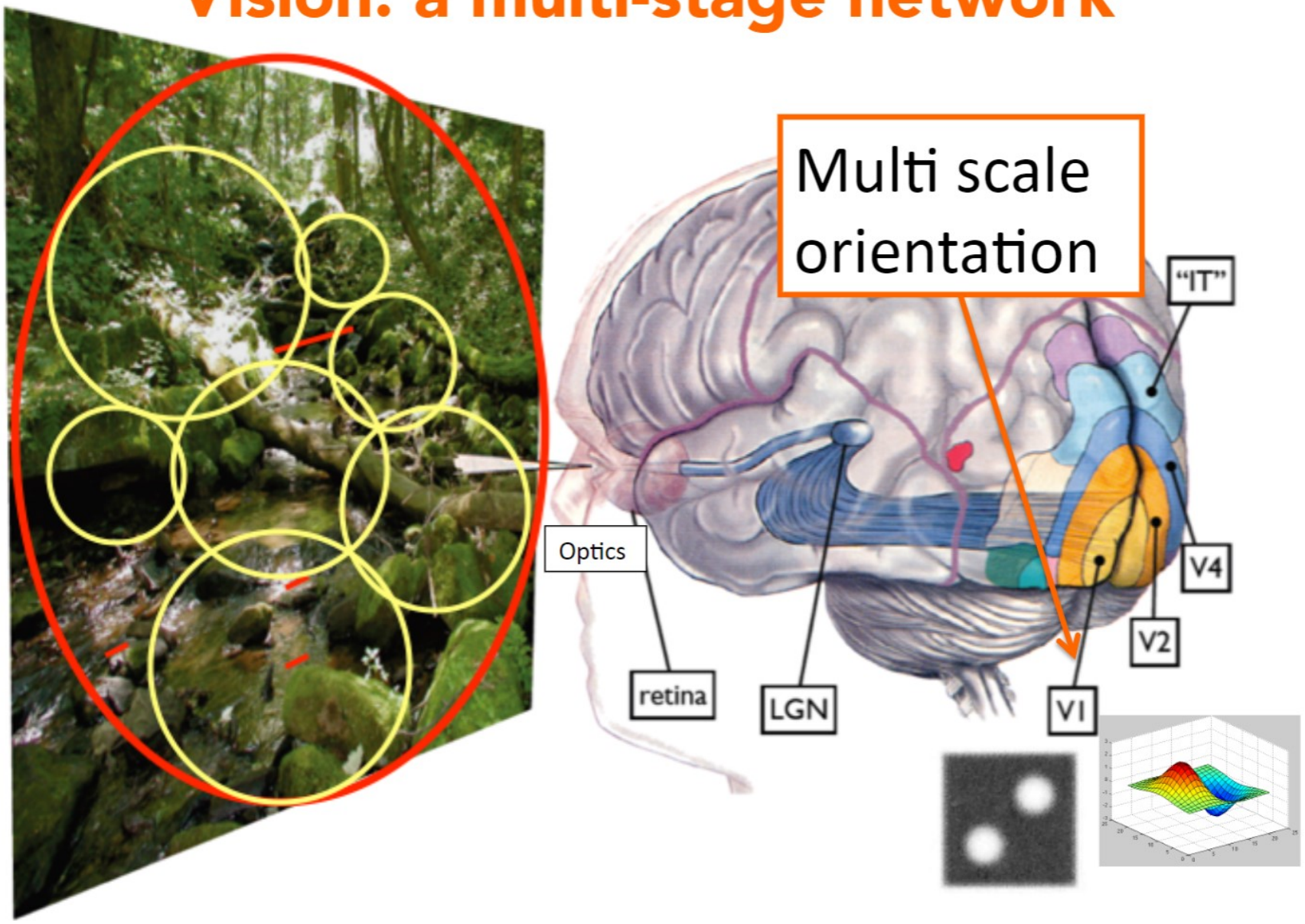
Difference of Gaussian Model





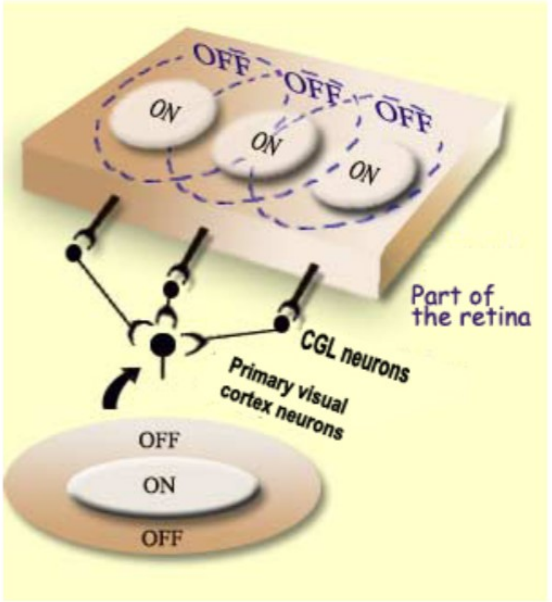
- The output of a retinal ganglion cell is a weighted sum of its inputs. You can treat the cell as a shift-invariant linear system
- **Difference of Gaussian (DoG):** A model of the operation performed by retinal and ganglion cells (LGN). The DoG model supposes that the neural response results from the combined signal of two separates mechanisms
- LGN and retinal neurons have circular receptive fields: **they respond equally well to all stimulus orientations, and at different spatial frequency (scale)**
- Multiple DoG is called a **Laplacian Pyramid**

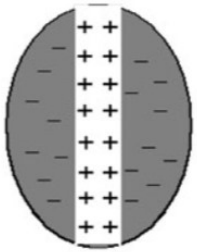
Vision: a multi-stage network



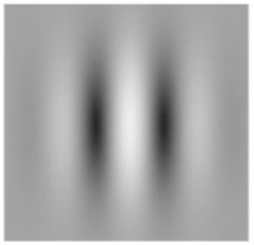
Oriented representation in V1

- Simple-cell receptive fields (RFs) are constructed from the output of LGN cells
- They are selective to **oriented contours** and edges





RECEPTIVE FIELD STRUCTURE

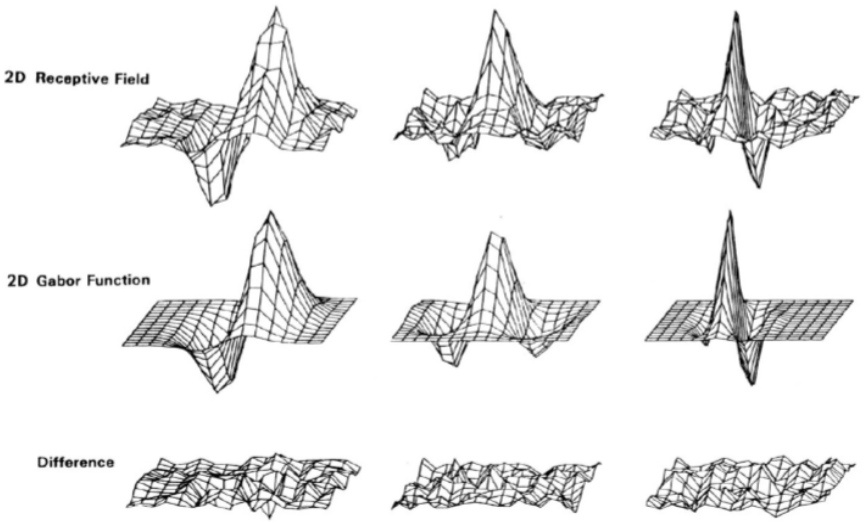


GABOR PATCH

V1 cells as Gabor Filters

The receptive fields of simple cells in V1 reflect the orientation and spatial frequency preferences of neurons. One way to model this, is to use a Gabor function, which is basically a two-dimensional gaussian modulated by a sinusoid.

2D Receptive fields in primary visual cortex



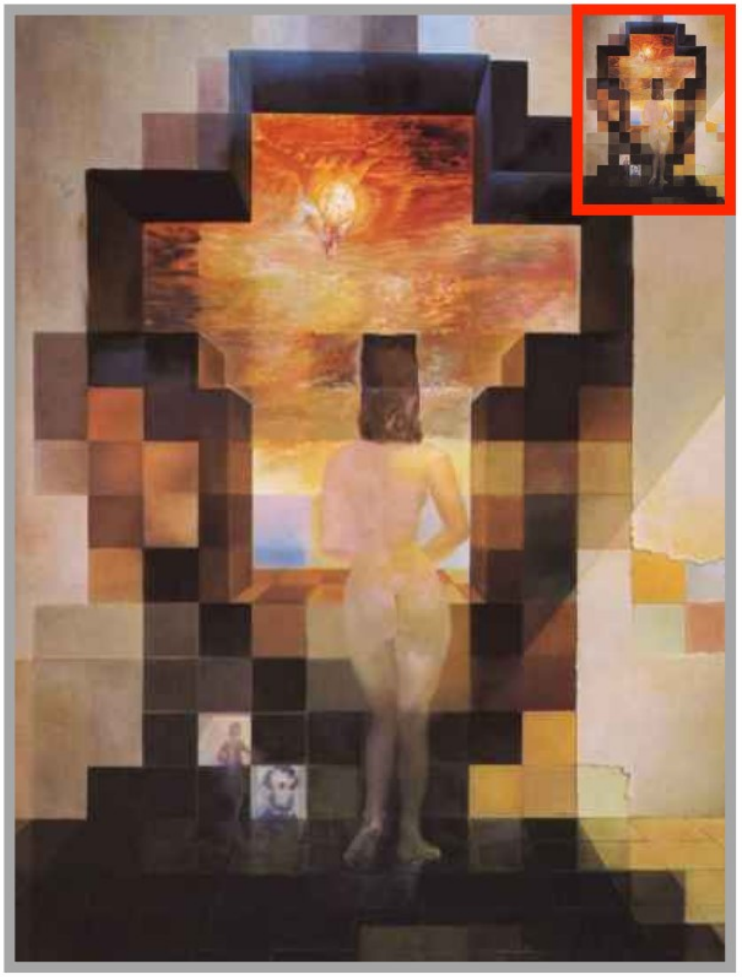
Fit of 2D Gabor wavelet is indistinguishable from noise.

figure from Daugman, 1990
data from Jones and Palmer, 1987

Visual Perception

For computer vision systems to "understand art" other properties of human visual perception must be taken into account

Salvador Dali



Changes of Meaning with Scale

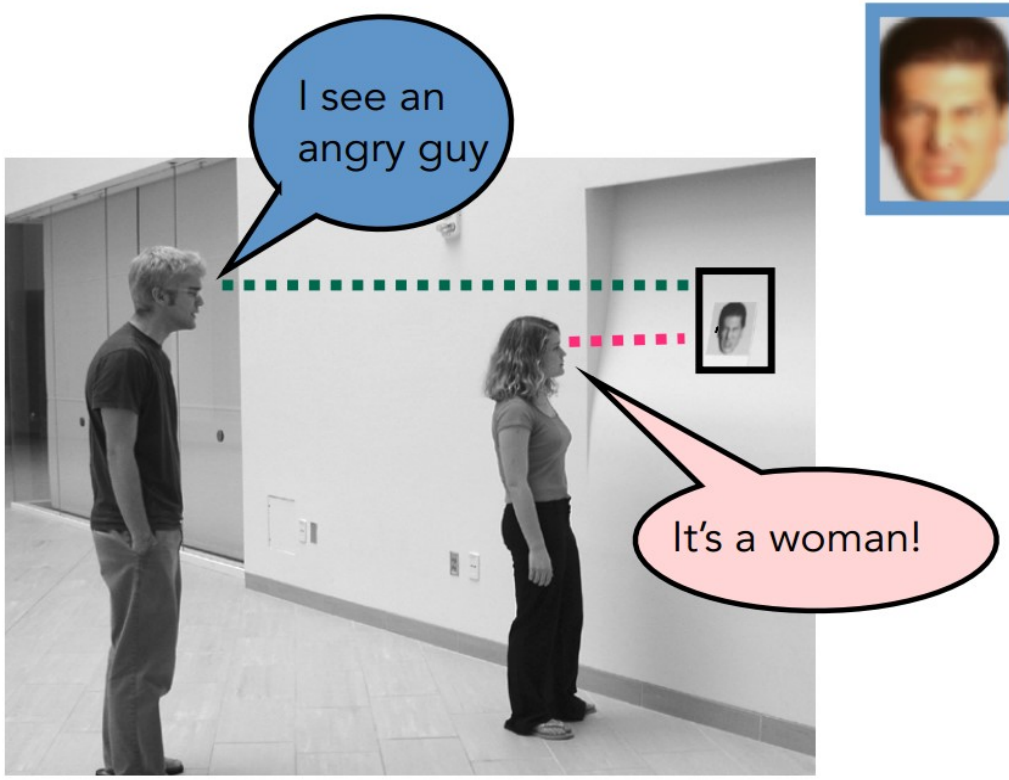
Strong percept of a woman

Strong percept of a skull

Looking at the image one way, a skull appears, comprised of the woman's form and her reflection in a mirror (the outline of the skull itself).

"All Is Vanity", by C. Allan Gilbert 1873-1929.

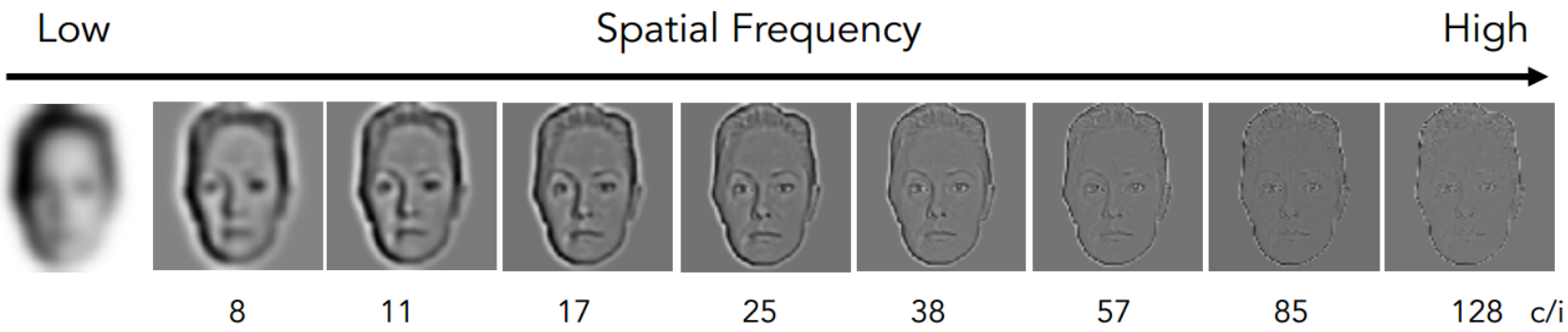
Hybrid Images

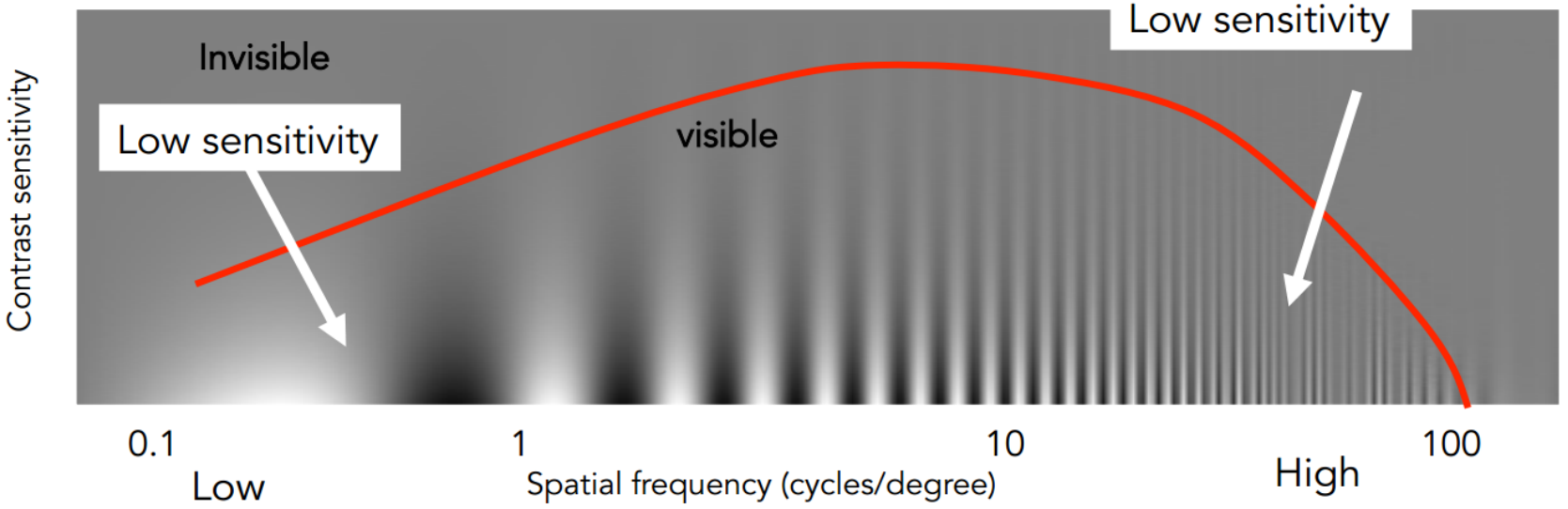


From Far Away

Up Close

Multiscale subband decomposition + Human Contrast Sensitivity Function





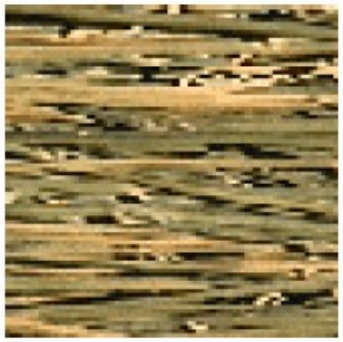
Today's class

- Template matching
- Gaussian pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian pyramids
  - Hybrid images
  - Application for image blending
- Steerable pyramids
  - Steerable filters
  - Orientation analysis
- Human vision system
  - Visual perception
- **Filter Banks and Texture Analysis**

Application: Representing Texture

Texture

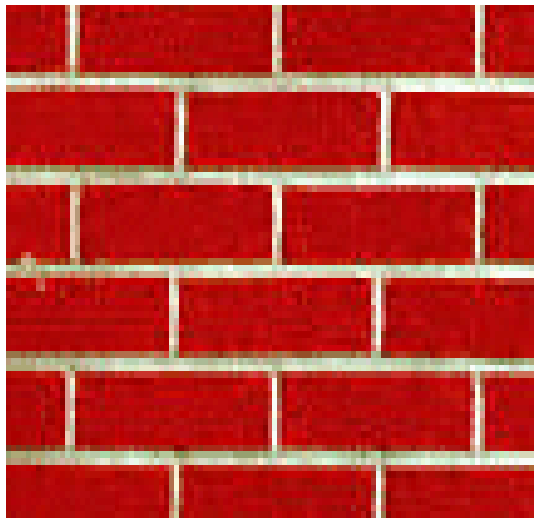


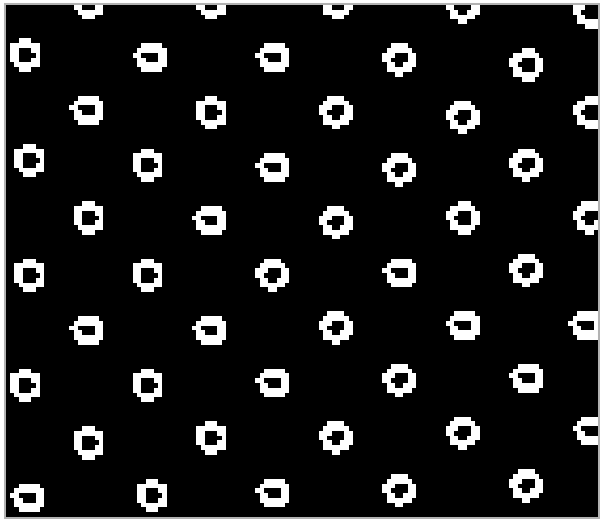


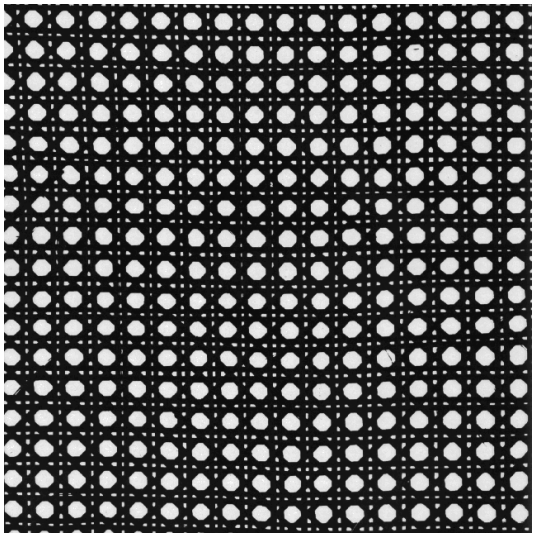



What defines a texture?

Includes: more regular patterns

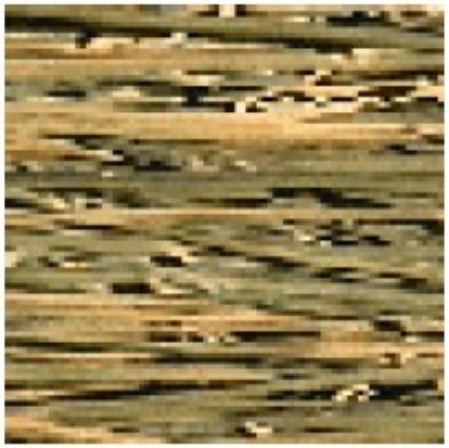
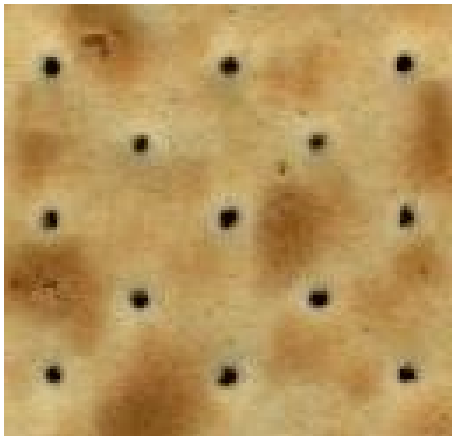




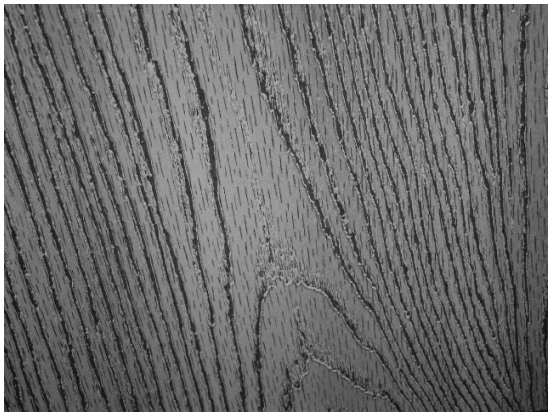


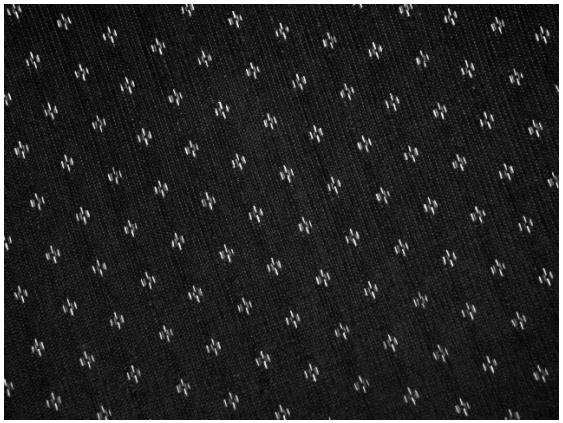
Includes: more random patterns

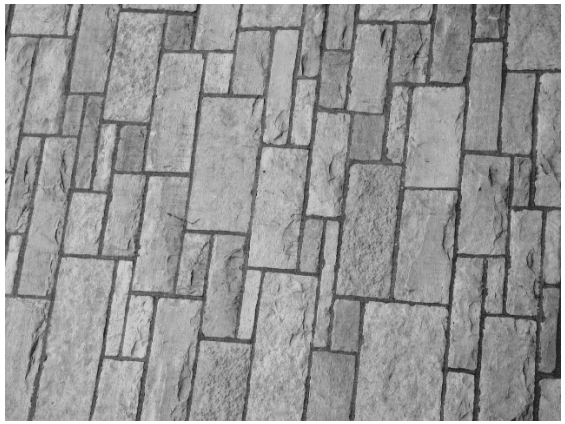


Texture and Material

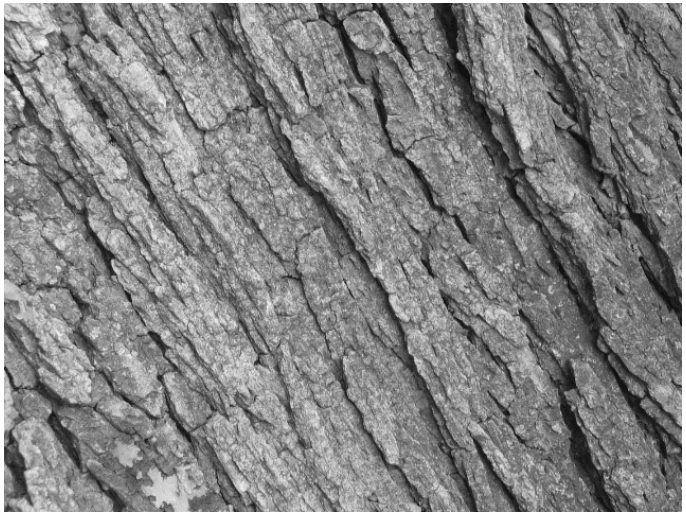


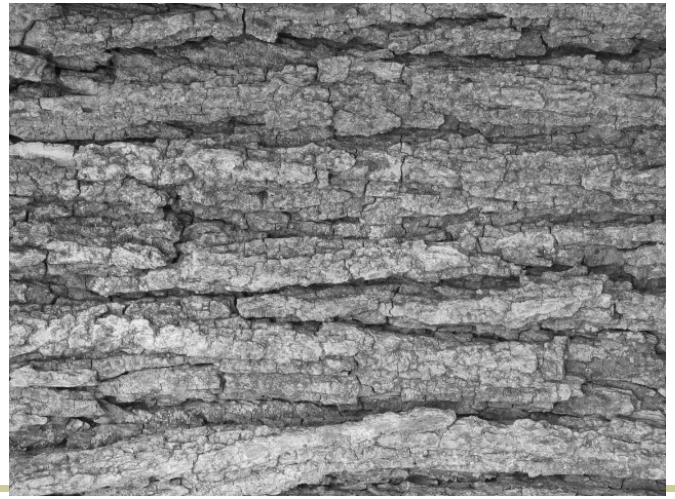




http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Orientation





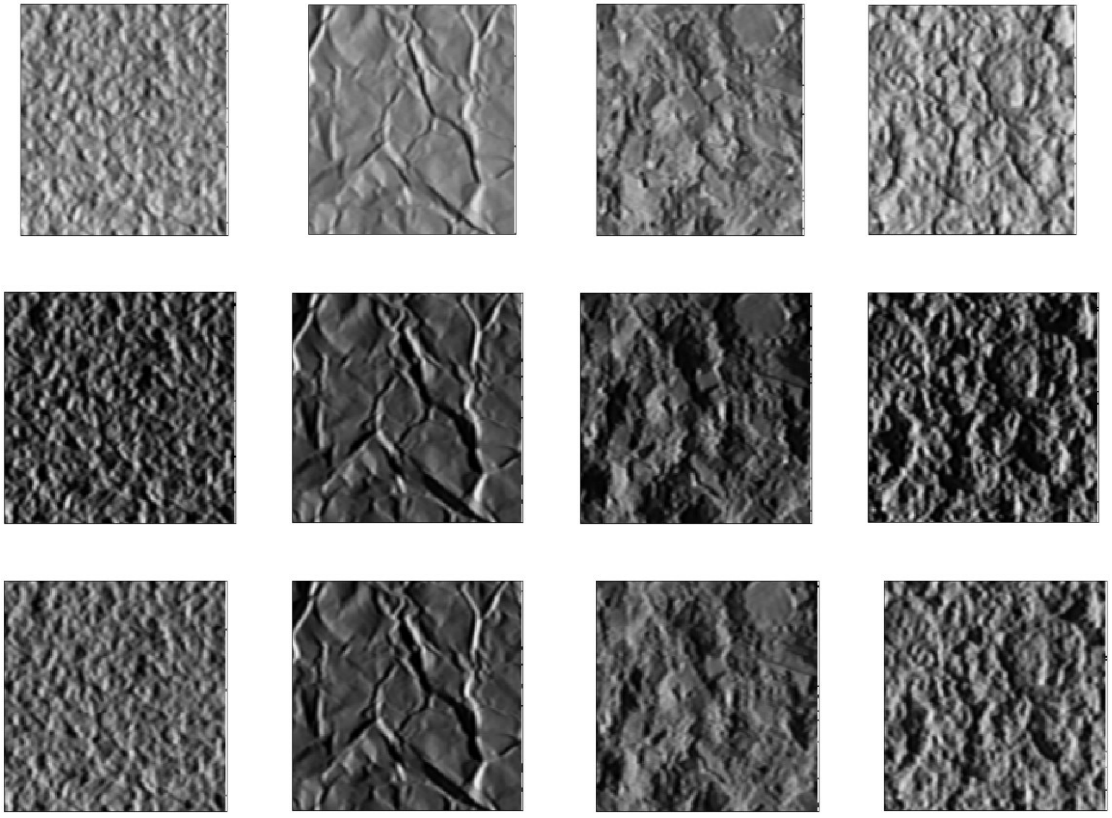
http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Materials under different illumination and viewing directions

Different illumination and viewing directions

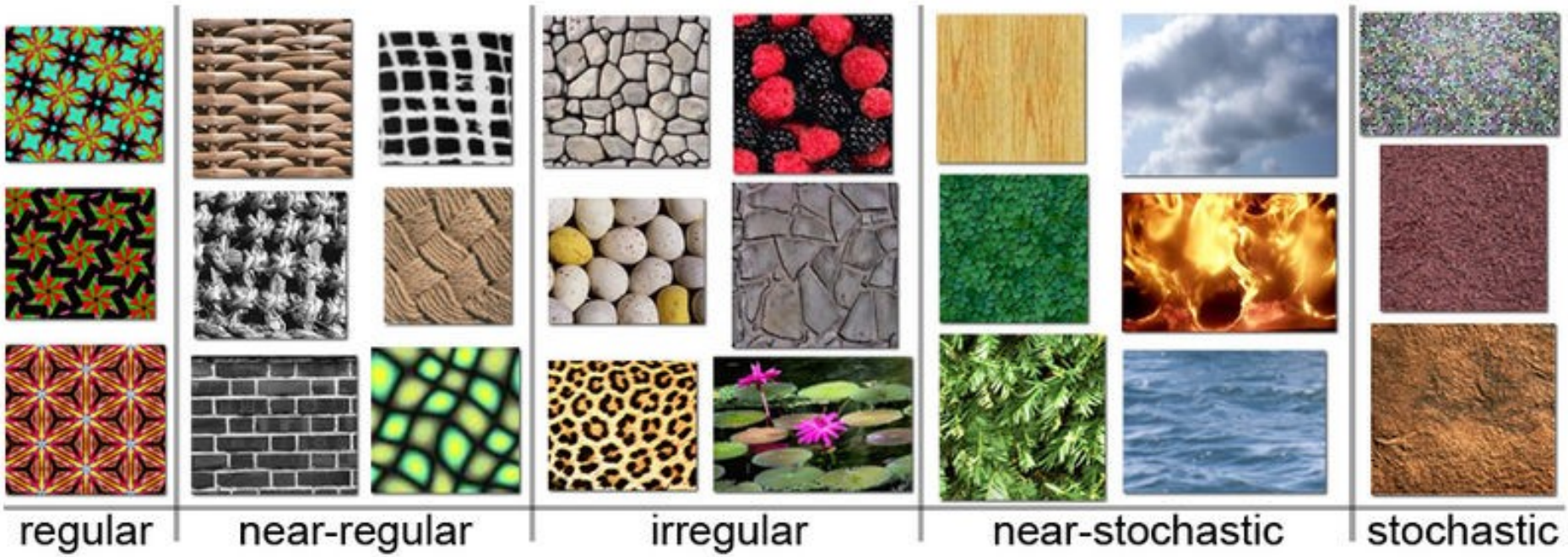


Plaster-a | Crumpled Paper | Concrete | Plaster-b (zoomed)

What is texture?

- **Texture is a phenomenon that is widespread, easy to recognize, and hard to define.**
- Views of large numbers of small objects
- Regular or stochastic patterns caused by bumps, grooves, and/or markings
- Textures tend to show repetition: the same local patch appears again and again.

Texture overview

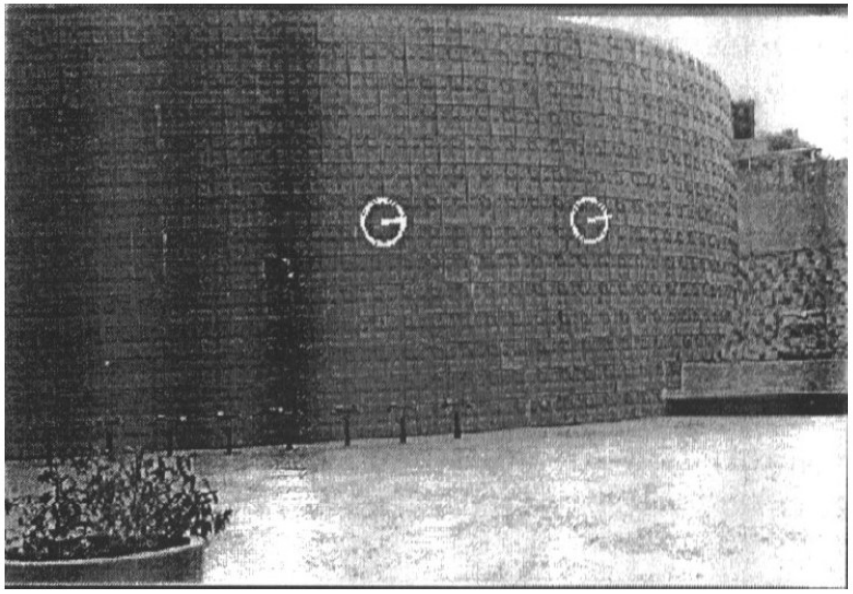


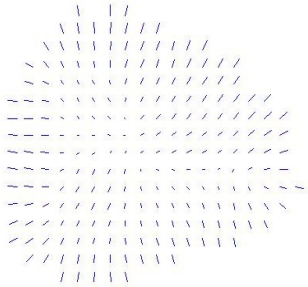
Texture-related tasks

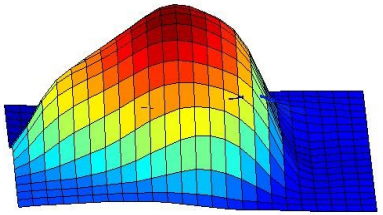
- **Shape from texture**
  - Estimate surface orientation or shape from image texture
- **Segmentation/classification** from texture cues
  - Analyze, represent texture
  - Group image regions with consistent texture
- **Synthesis**
  - Generate new texture patches/images given some examples

Shape from texture

- Use deformation of texture from point to point to estimate surface shape

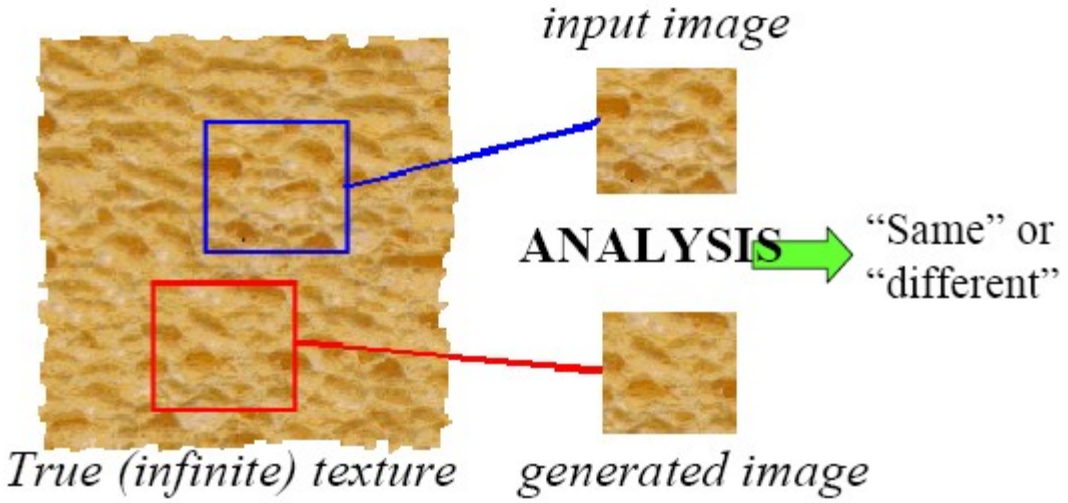




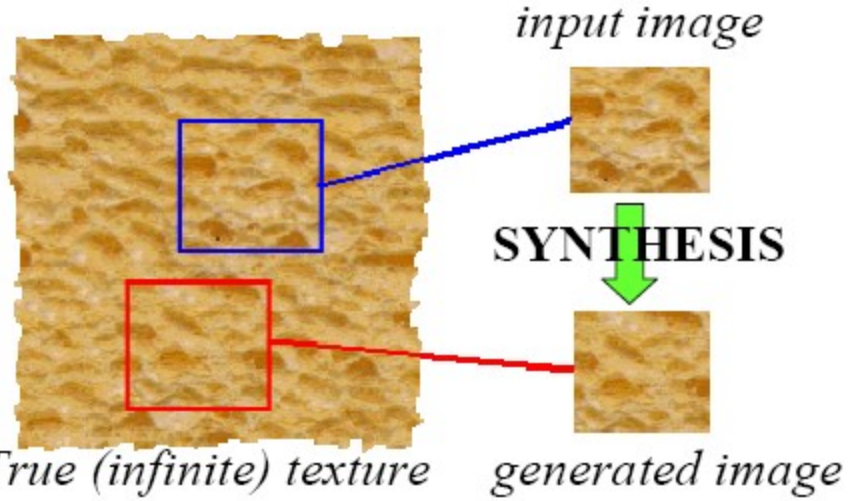


Pics from A. Loh: http://www.csse.uwa.edu.au/~angie/phdpics1.html

Analysis vs. Synthesis

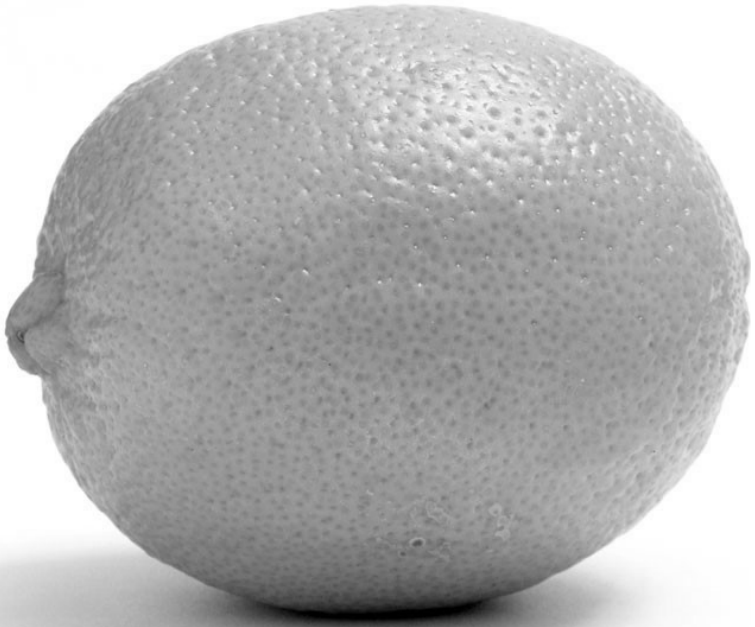


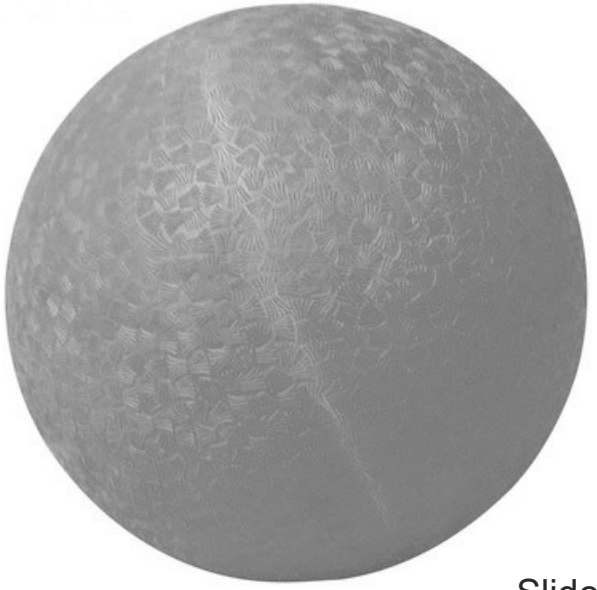
Why analyze texture?



Images:Bill Freeman, A. Efros

Slide credit:
Kristen Grauman





Slide credit:
Kristen Grauman

Slide credit: Kristen Grauman

http://animals.nationalgeographic.com/

Why analyze texture?

Importance to perception:

- Often indicative of a material's properties
- Can be important appearance cue, especially if shape is similar across objects
- Aim to distinguish between shape, boundaries, and texture

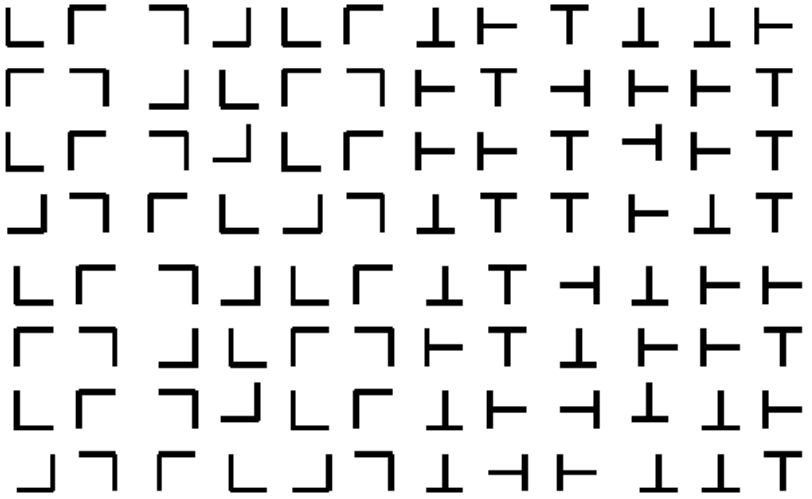
Technically:

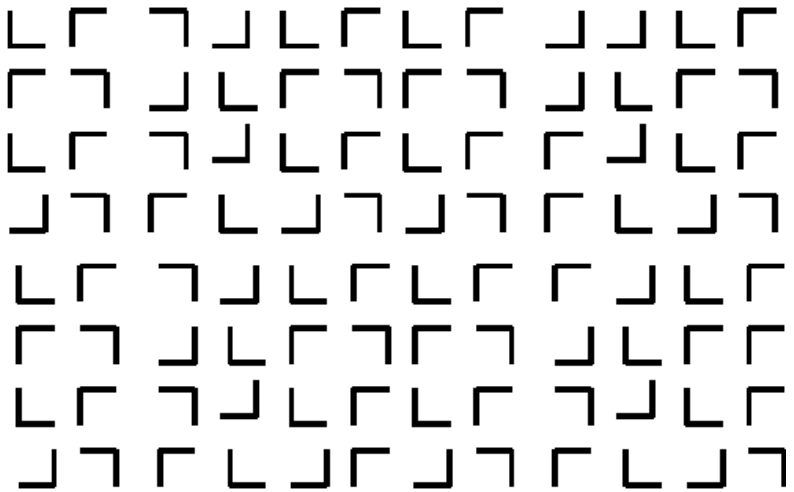
- Representation-wise, we want a feature one step above "building blocks" of filters, edges.

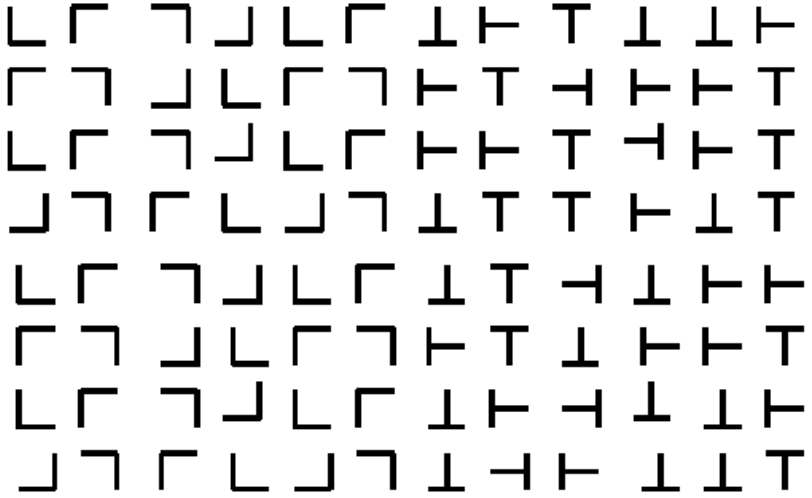
Psychophysics of texture

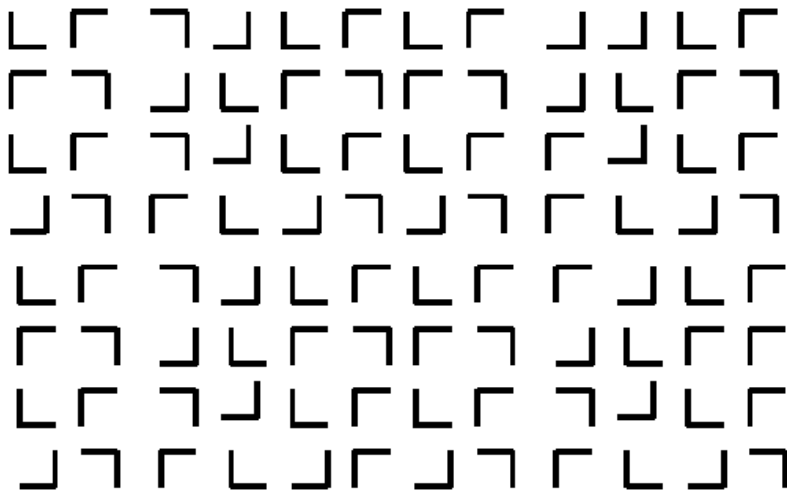
- Some textures distinguishable with *preattentive* perception– without scrutiny, eye movements [Julesz 1975]

Same or different?

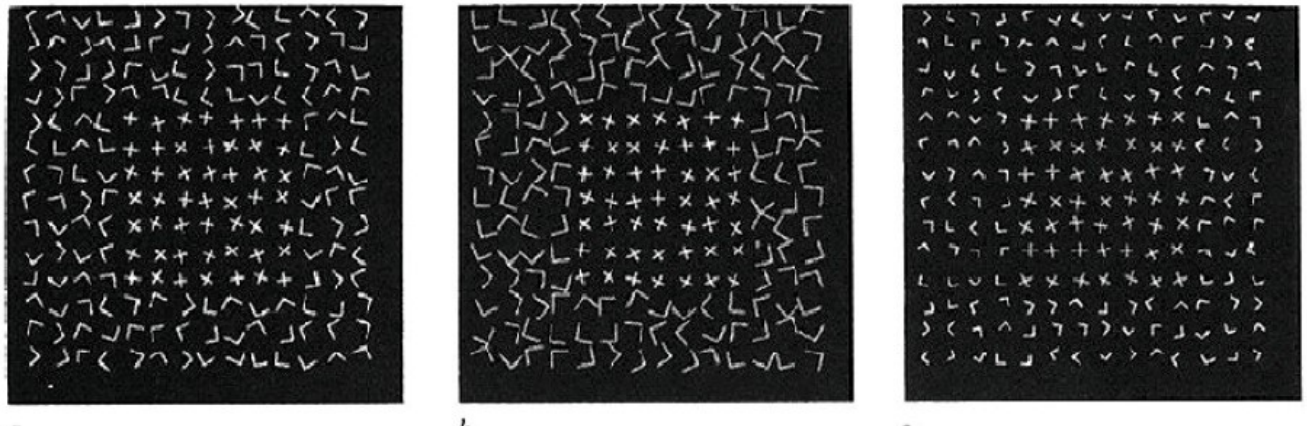








Capturing the local patterns with image measurements



[Bergen & Adelson, *Nature* 1988]

Scale of patterns influences discriminability

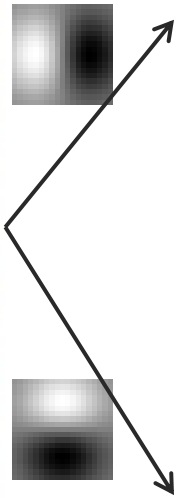
Size-tuned linear filters

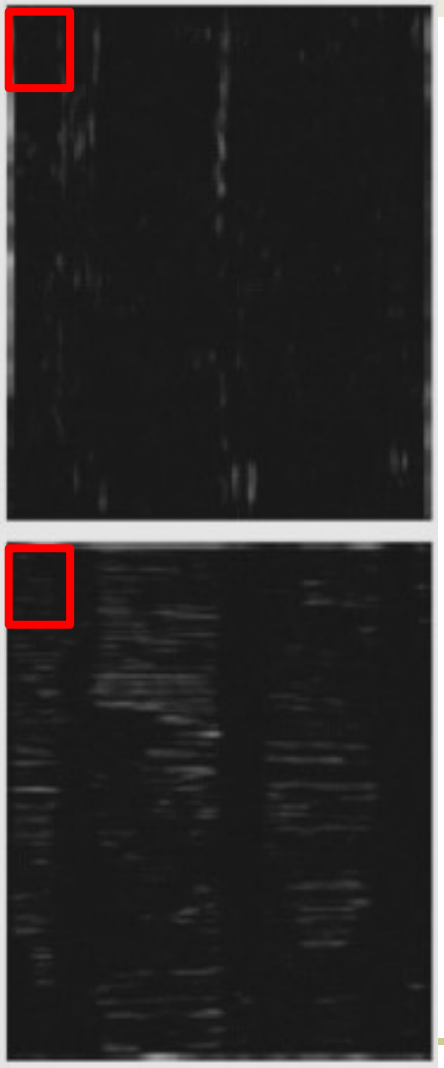
Texture representation

- Textures are made up of repeated local patterns, so:
  - Find the patterns
    - Use filters that look like patterns (spots, bars, raw patches…)
    - Consider magnitude of response
  - Describe their statistics within each local window, e.g.,
    - Mean, standard deviation
    - Histogram
    - Histogram of “prototypical” feature occurrences

Texture representation: example

original image





derivative filter responses, squared

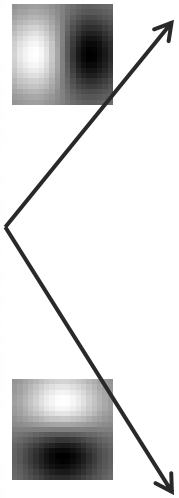
	mean d/dx value	mean d/dy value
Win. #1	4	10

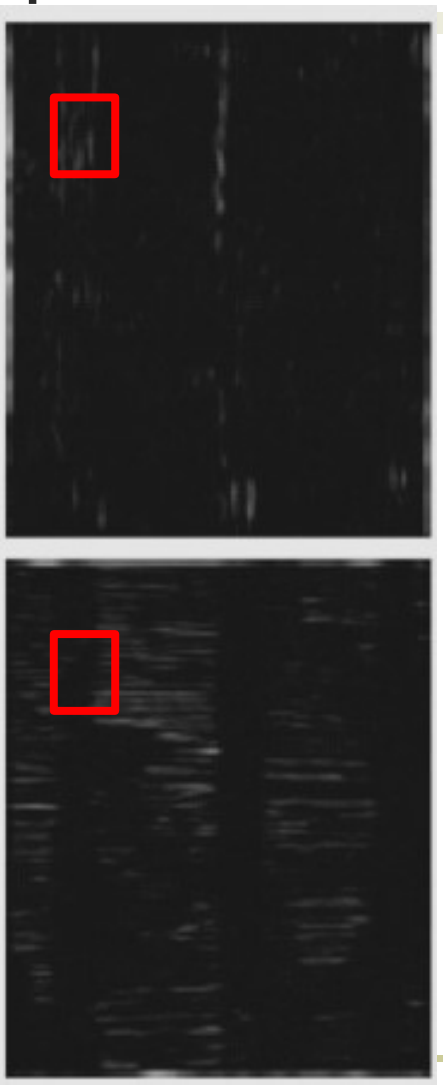
statistics to summarize patterns in small windows

Slide credit:
Kristen Grauman

Texture representation: example

original image





derivative filter responses, squared

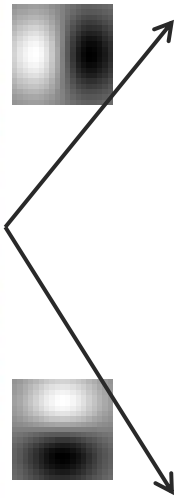
	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7	

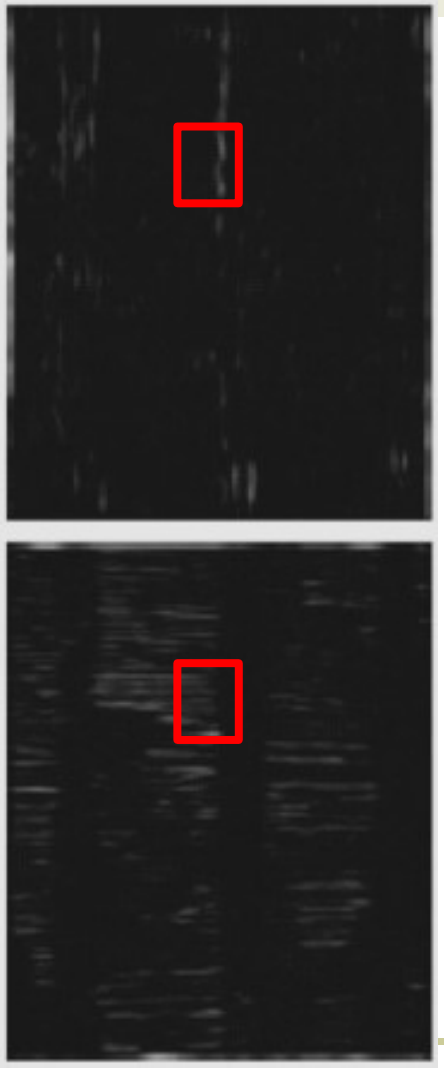
statistics to summarize patterns in small windows

Slide credit:
Kristen Grauman

Texture representation: example

original image





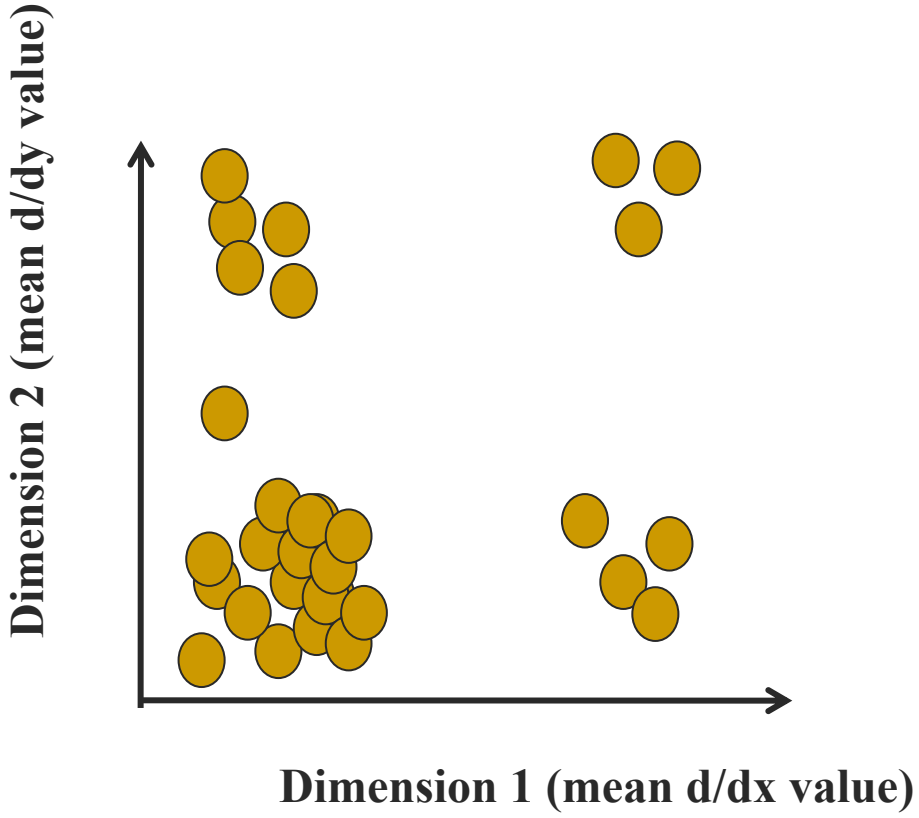
derivative filter responses, squared

	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
Win.#9	20	20

statistics to summarize patterns in small windows

Slide credit:
Kristen Grauman

Texture representation: example

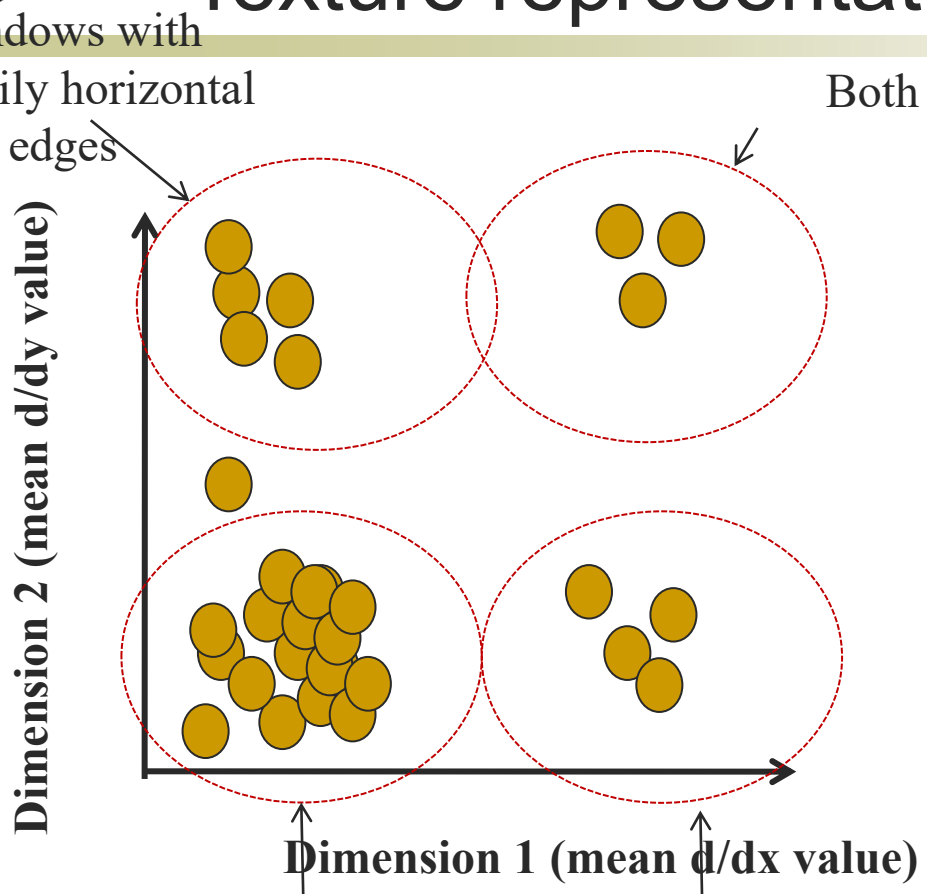


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
Win.#9	20	20	

statistics to summarize patterns in small windows

Slide credit:
Kristen Grauman

Texture representation: example



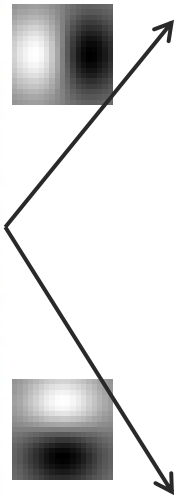
	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
Win.#9	20	20	

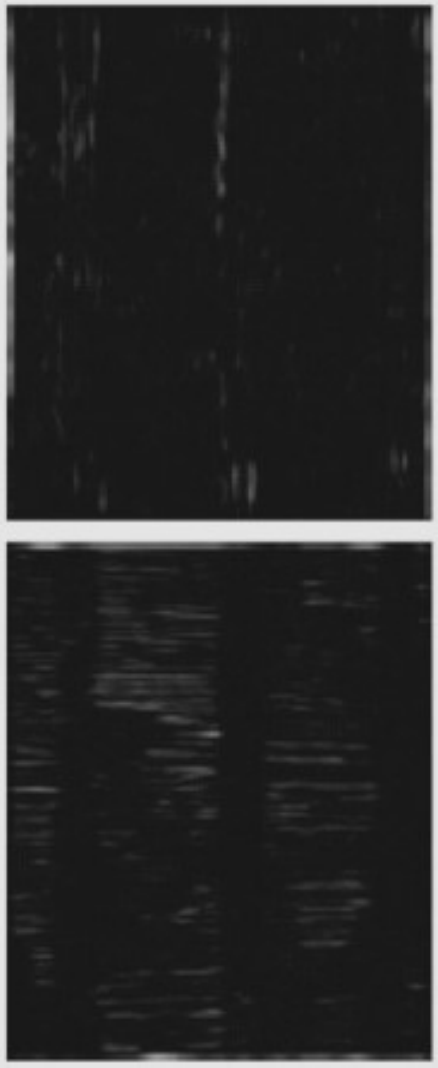
statistics to summarize patterns in small windows

Slide credit: Kristen Grauman

Texture representation: example

original image



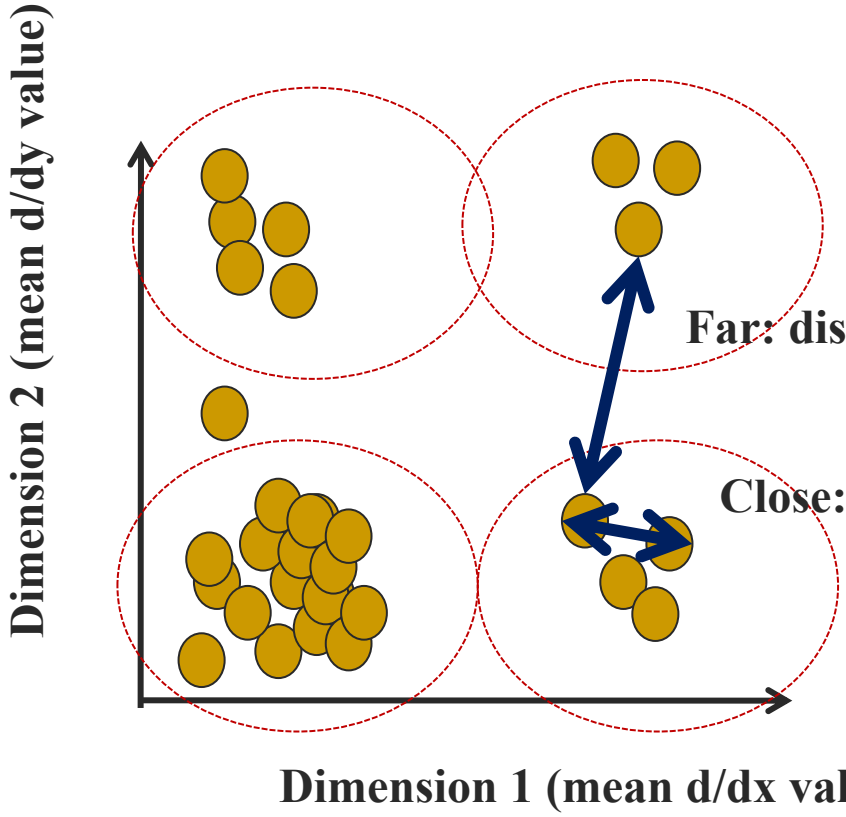


derivative filter responses, squared

visualization of the assignment to texture "types"

Slide credit:
Kristen Grauman

Texture representation: example

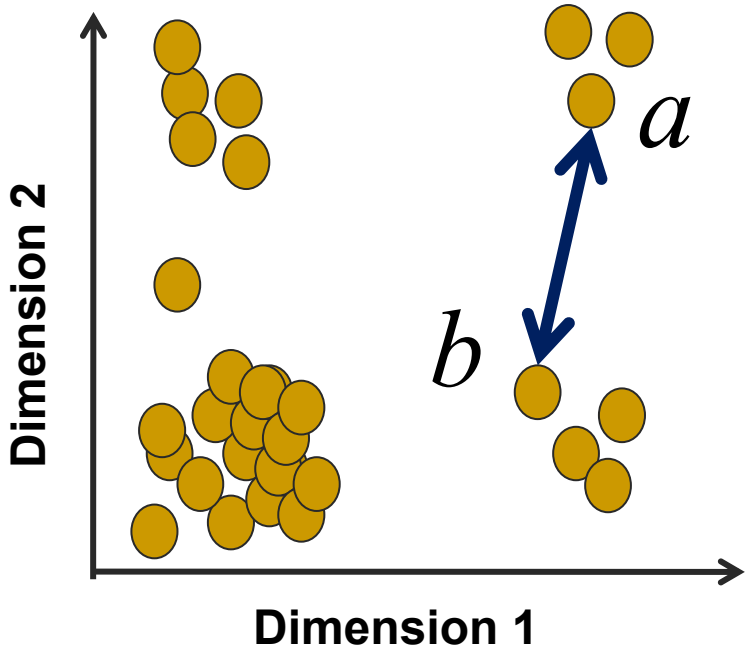


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
Win.#9	20	20	

statistics to summarize patterns in small windows

Slide credit:
Kristen Grauman

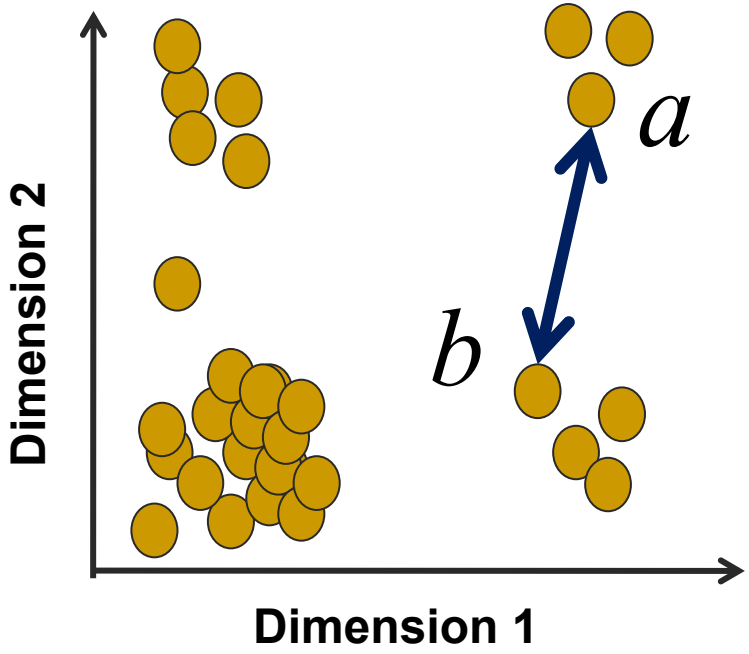
Texture representation: example

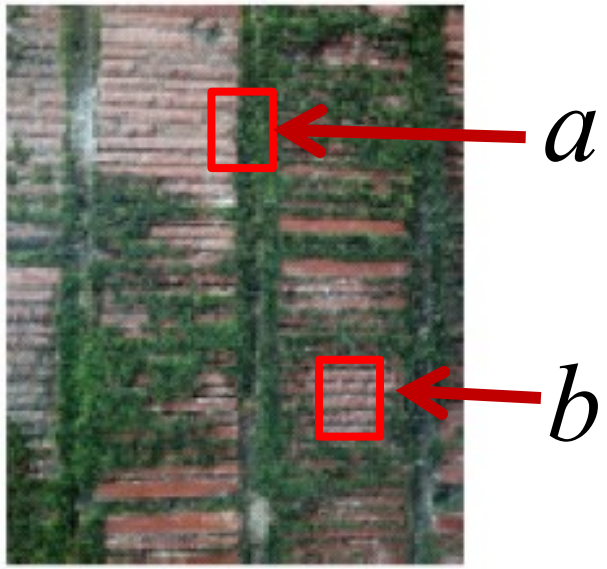


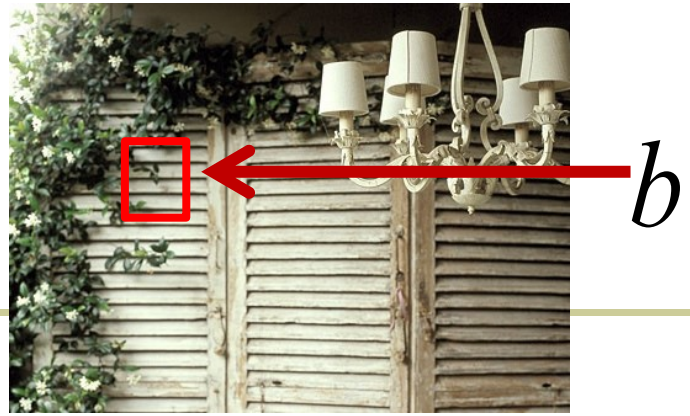
$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Slide credit:
Kristen Grauman

Texture representation: example





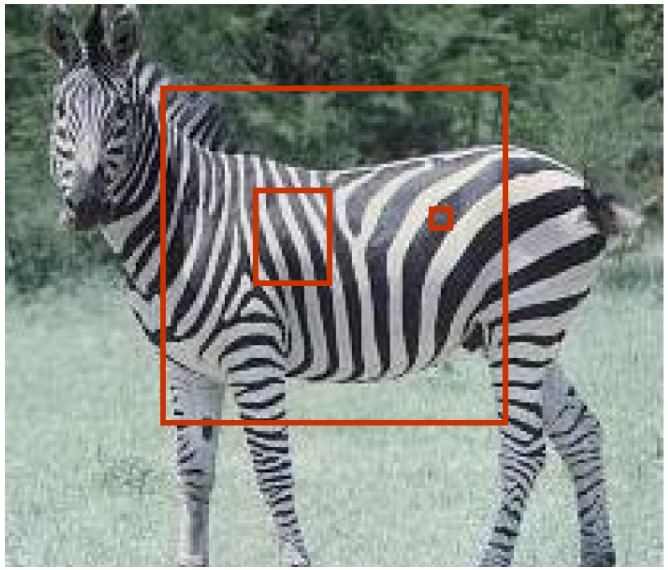


Distance reveals how dissimilar texture from window a is from texture in window b.

Slide credit:
Kristen Grauman

Texture representation: window scale

- We're assuming we know the relevant window size for which we collect these statistics.



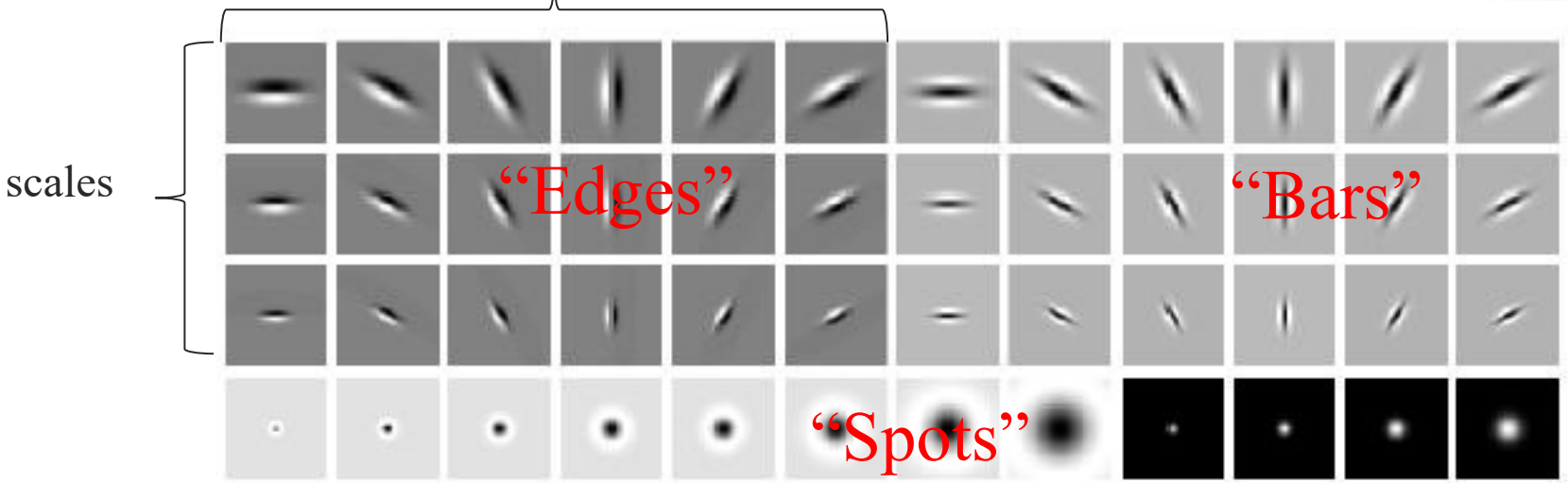
Possible to perform **scale selection** by looking for window scale where texture description not changing.

Slide credit:
Kristen Grauman

Filter banks

- Our previous example used two filters, and resulted in a 2-dimensional feature vector to describe texture in a window.
  - x and y derivatives revealed something about local structure.
- We can generalize to apply a collection of multiple (*d*) filters: a “filter bank”
- Then our feature vectors will be *d*-dimensional.
  - still can think of nearness, farness in feature space

Filter banks

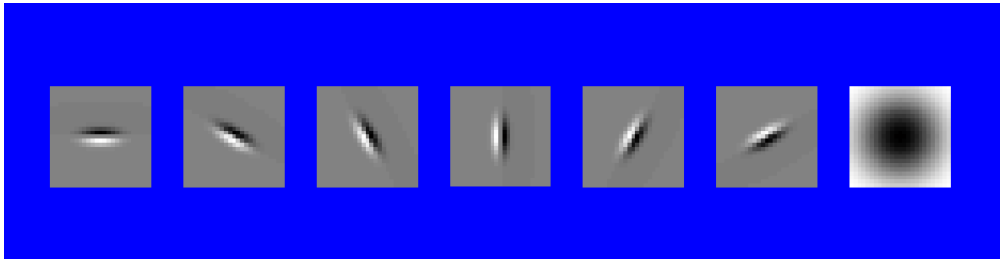


- What filters to put in the bank?
  - Typically we want a combination of scales and orientations, different types of patterns.

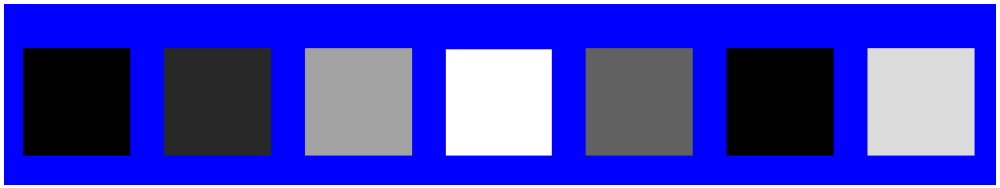
Matlab code available for these examples:
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

You try: Can you match the texture to the response?

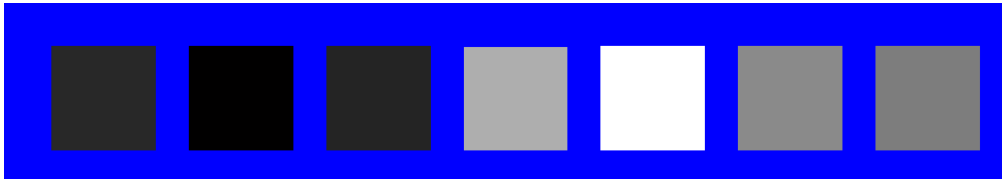
Filters



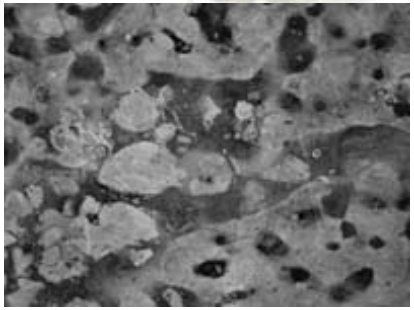
1



2

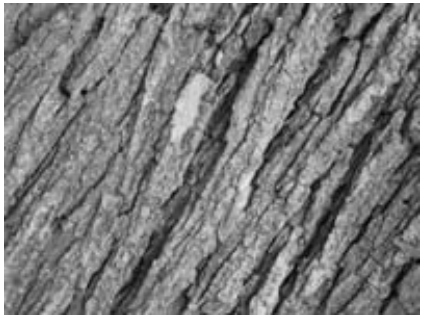


3



A

B

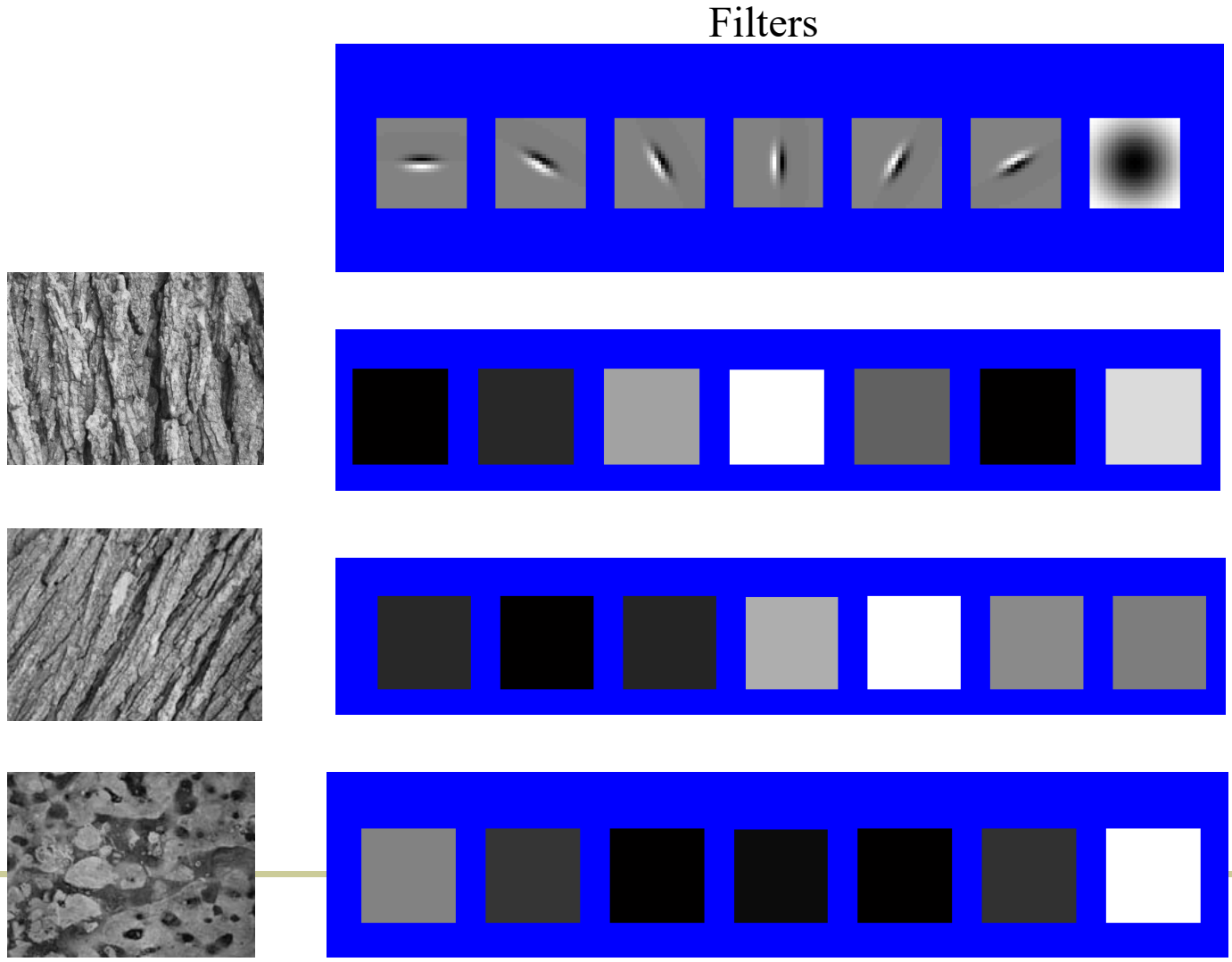


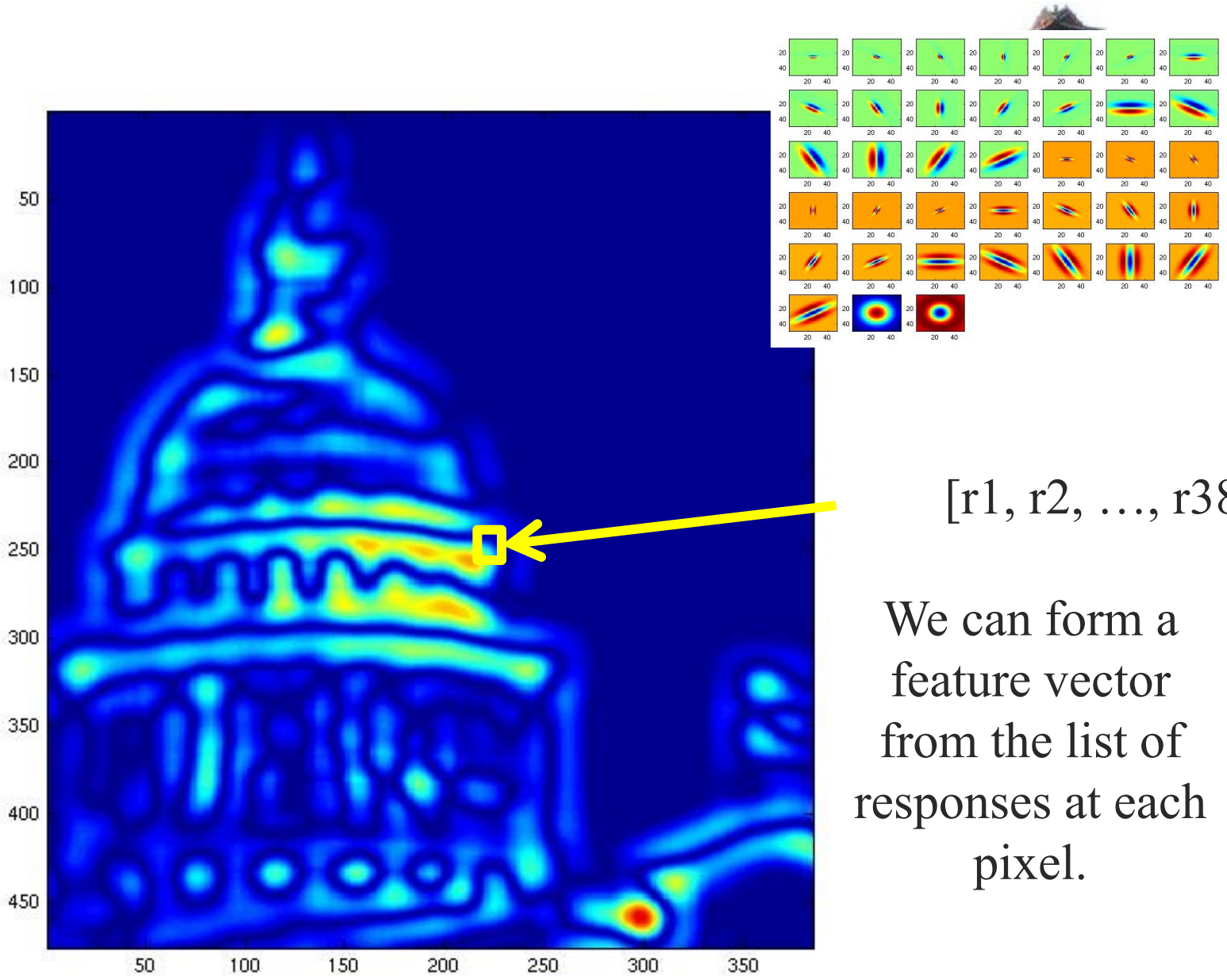
C

Mean abs responses

Derek Hoiem

Representing texture by mean abs response





[r1, r2, …, r38

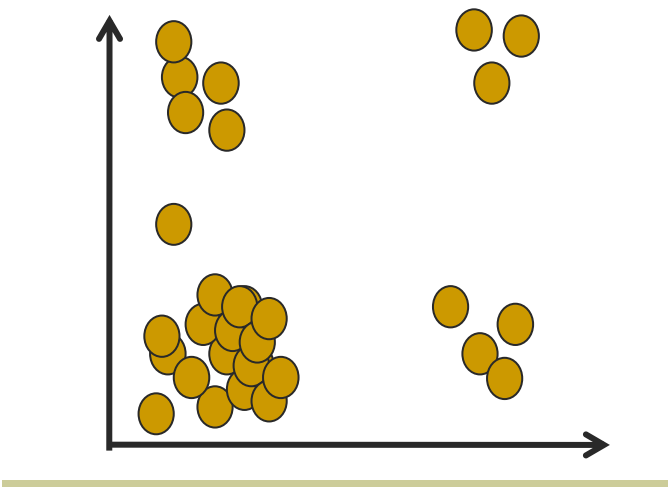
We can form a feature vector from the list of responses at each pixel.

Slide credit:
Kristen Grauman

d-dimensional features

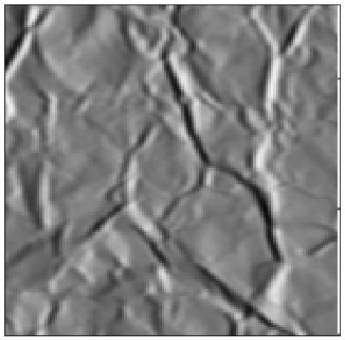
$$D(a,b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

Euclidean distance (L_2)

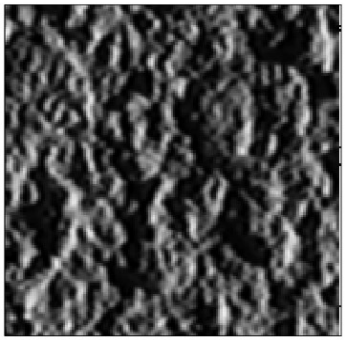


Slide credit:
Kristen Grauman

Texture Recognition: Local to Global



?



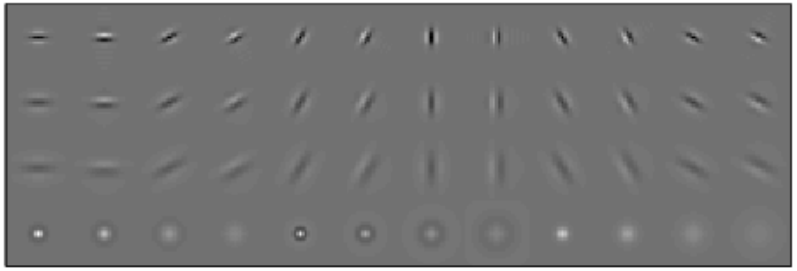
?

Felt?
Polyester?
Terrycloth?
Rough Plaster?
Leather?
Plaster?
Concrete?
Crumpled Paper?
Sponge?
Limestone?
Brick?

2D Textons

- Goal: find canonical local features in a texture;

  1) Filter image with linear filters:

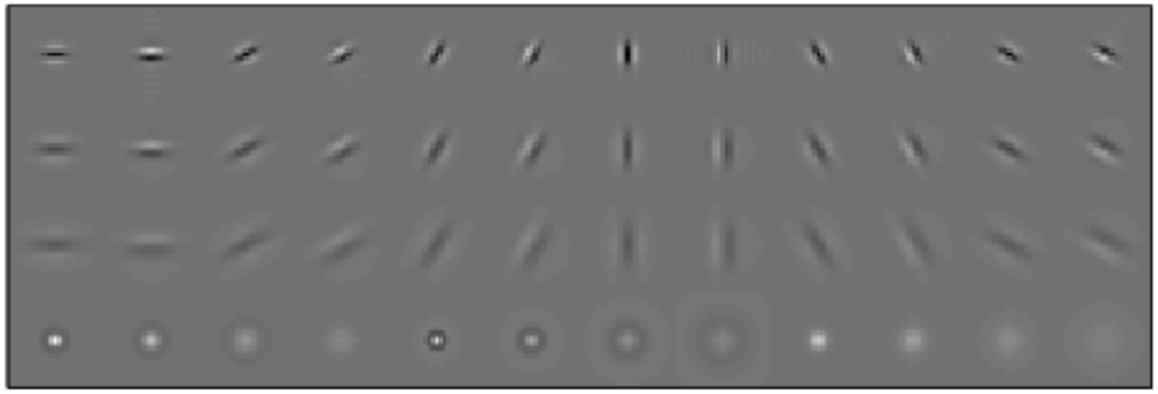


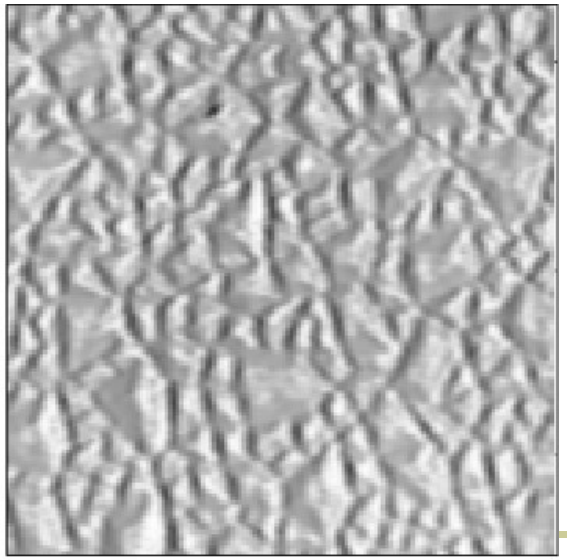
  2) Vector quantization (k-means) on filter outputs;

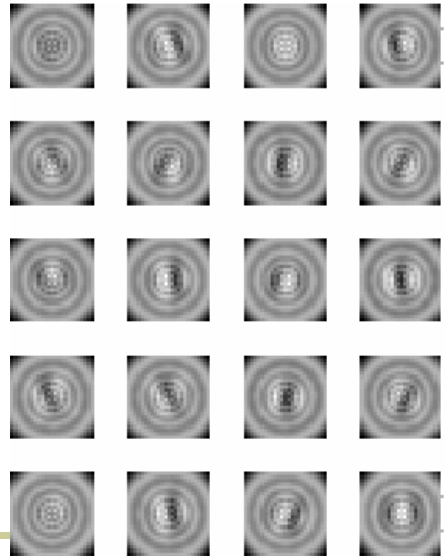
  3) Quantization centers are the textons.

- Spatial distribution of textons defines the texture;

2D Textons (cont’d)

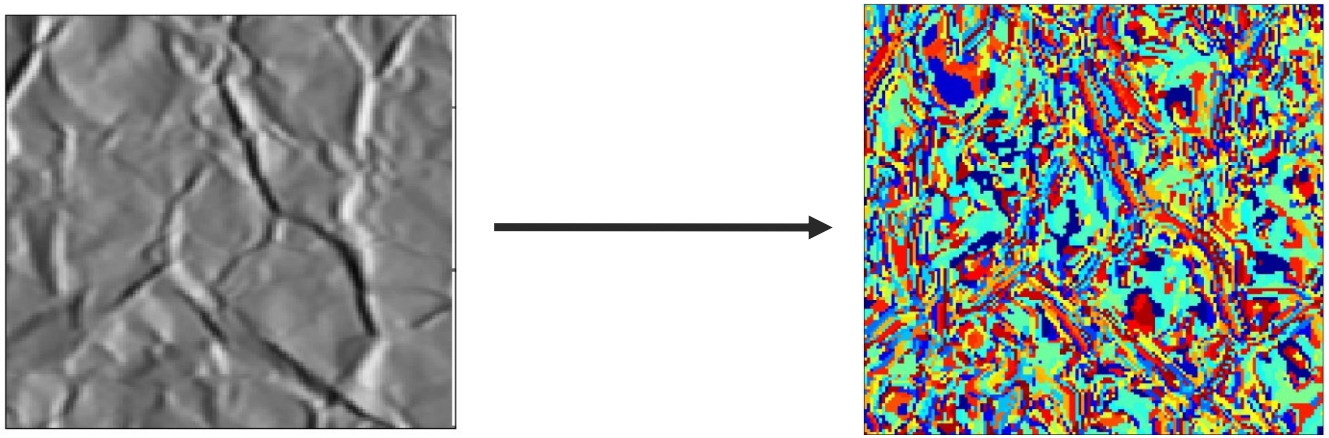






Texton Labeling

- Each pixel labeled to texton *i* (1 to K) which is *most similar in appearance*;
- Similarity measured by the Euclidean distance between the filter responses;

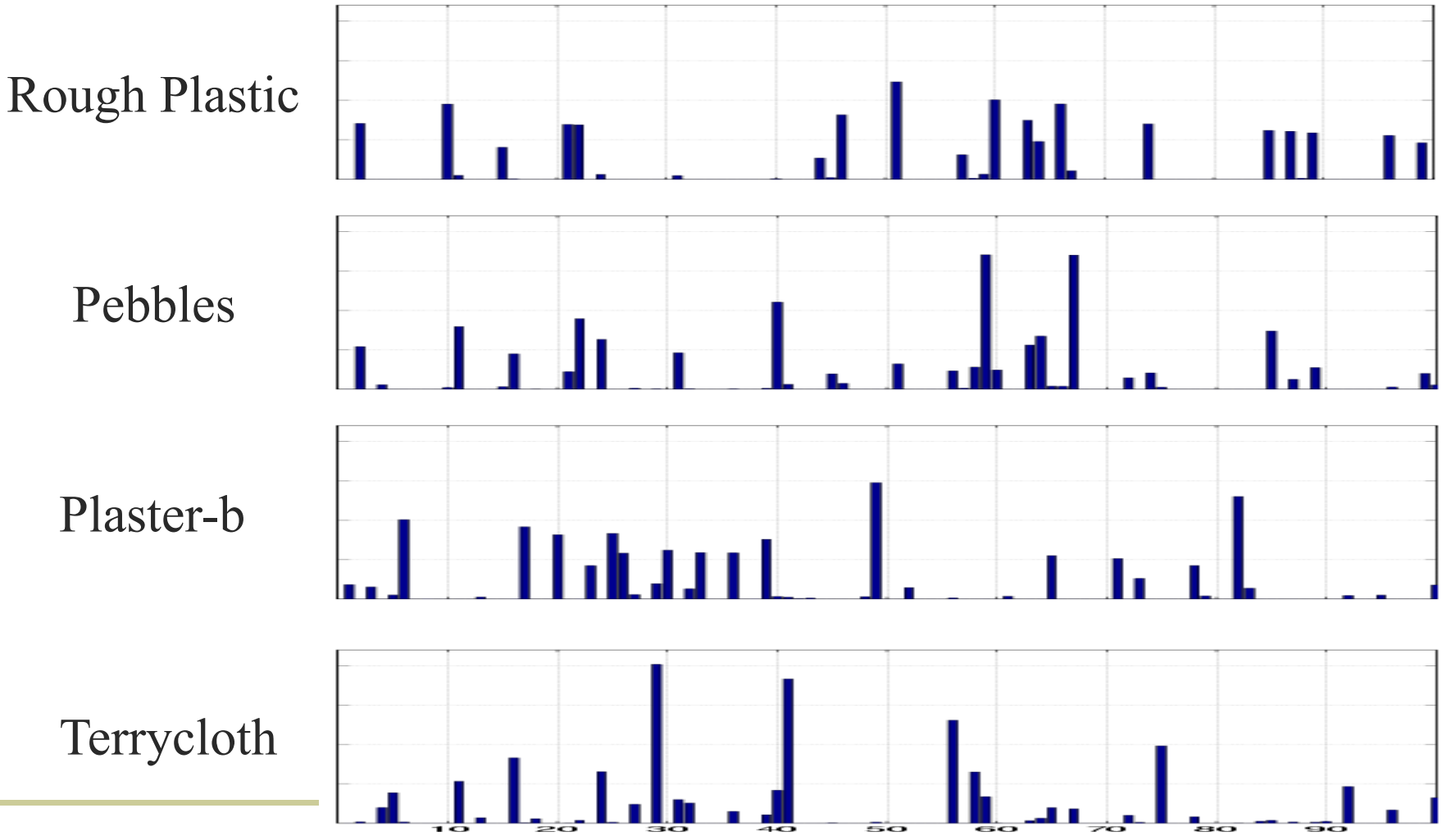


Material Representation

- Each material is now represented as a spatial arrangement of symbols from the texton vocabulary;

- Recognition：ignore spatial arrangement, use histogram (K=100);

Histogram Models for Recognition (Leung & Malik, 1999)



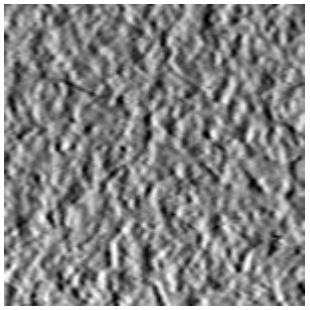
Similarity of materials

- Similarity between histograms measured using chi-square difference:

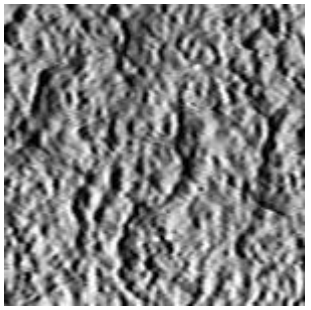
$$\chi^2(h_1, h_2) = \sum_{n=1}^{N} \frac{(h_1(n) - h_2(n))^2}{h_1(n) + h_2(n)}$$

Similarity Matrix

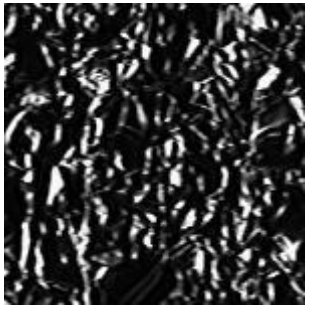
	Felt	Terrycloth	Rough Plastic	Leather	Sandpaper	Pebbles	Plastera	Plasterb	Rough Paper	Artificial Grass	Rough Shingle	Aluminum Foil	Cork	Rough Tile
Felt	**0.6**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Terrycloth	0.0	**1.0**	0.0	0.0	0.3	0.0	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0
Rough Plastic	0.0	0.0	**0.9**	0.0	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0
Leather	0.2	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Sandpaper	0.0	0.1	0.0	0.0	**1.0**	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pebbles	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
Plastera	0.0	0.1	0.2	0.0	0.1	0.0	**1.0**	0.7	0.0	0.0	0.0	0.0	0.0	0.0
Plasterb	0.0	0.2	0.1	0.0	0.0	0.0	0.8	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0
Rough Paper	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	**0.5**	0.0	0.0	0.0	0.0	0.0
Artificial Grass	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.1	0.1	0.0	0.0
Roof Shingle	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	**1.0**	0.1	0.0	0.0
Aluminum Foil	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	**1.0**	0.0	0.0
Cork	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.2
Rough Tile	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.9**



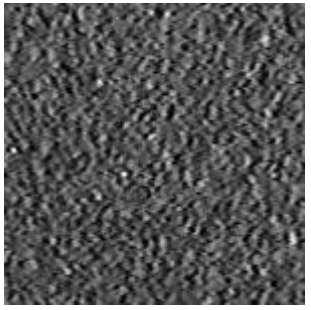
Plaster-a



Plaster-b



Aluminum Foil



Cork

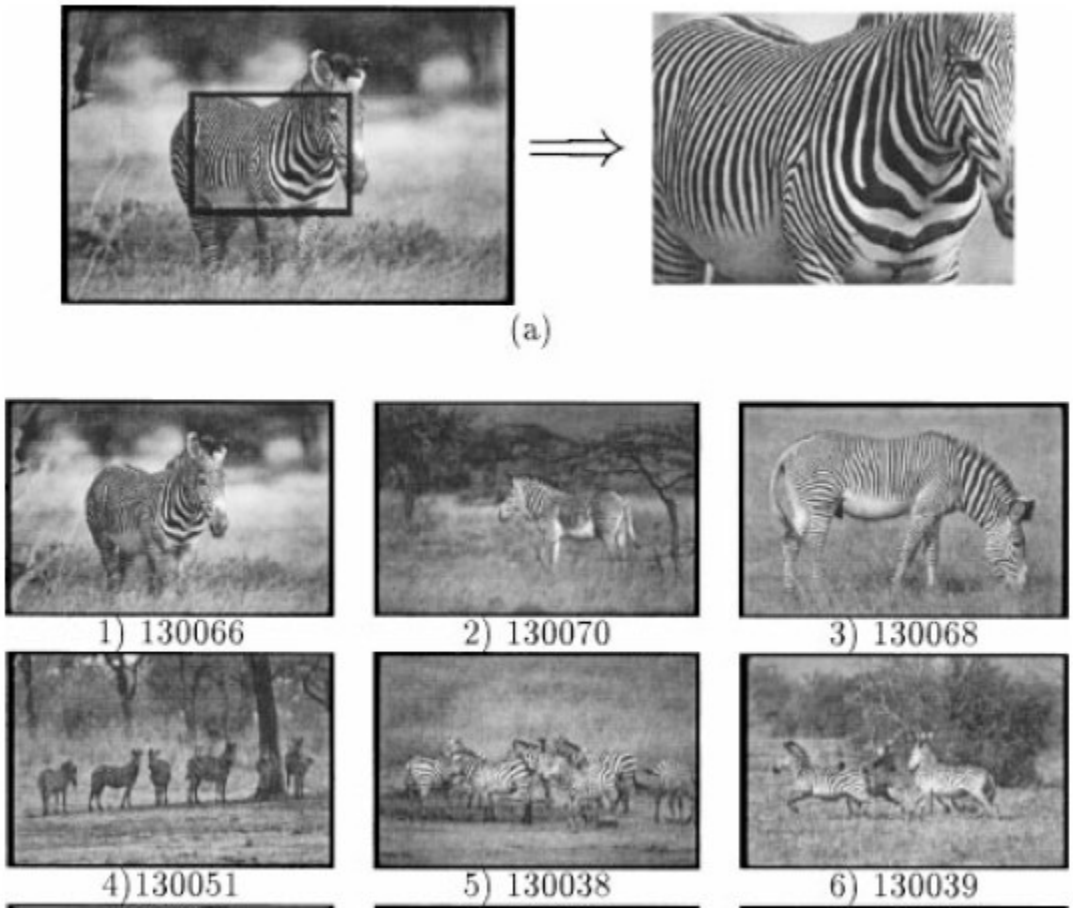
$$e_{ij} = \text{Similarity}(\text{material} = \text{i}, \text{sample} = \text{j})$$

Example uses of texture in vision: analysis

Classifying materials, “stuff”

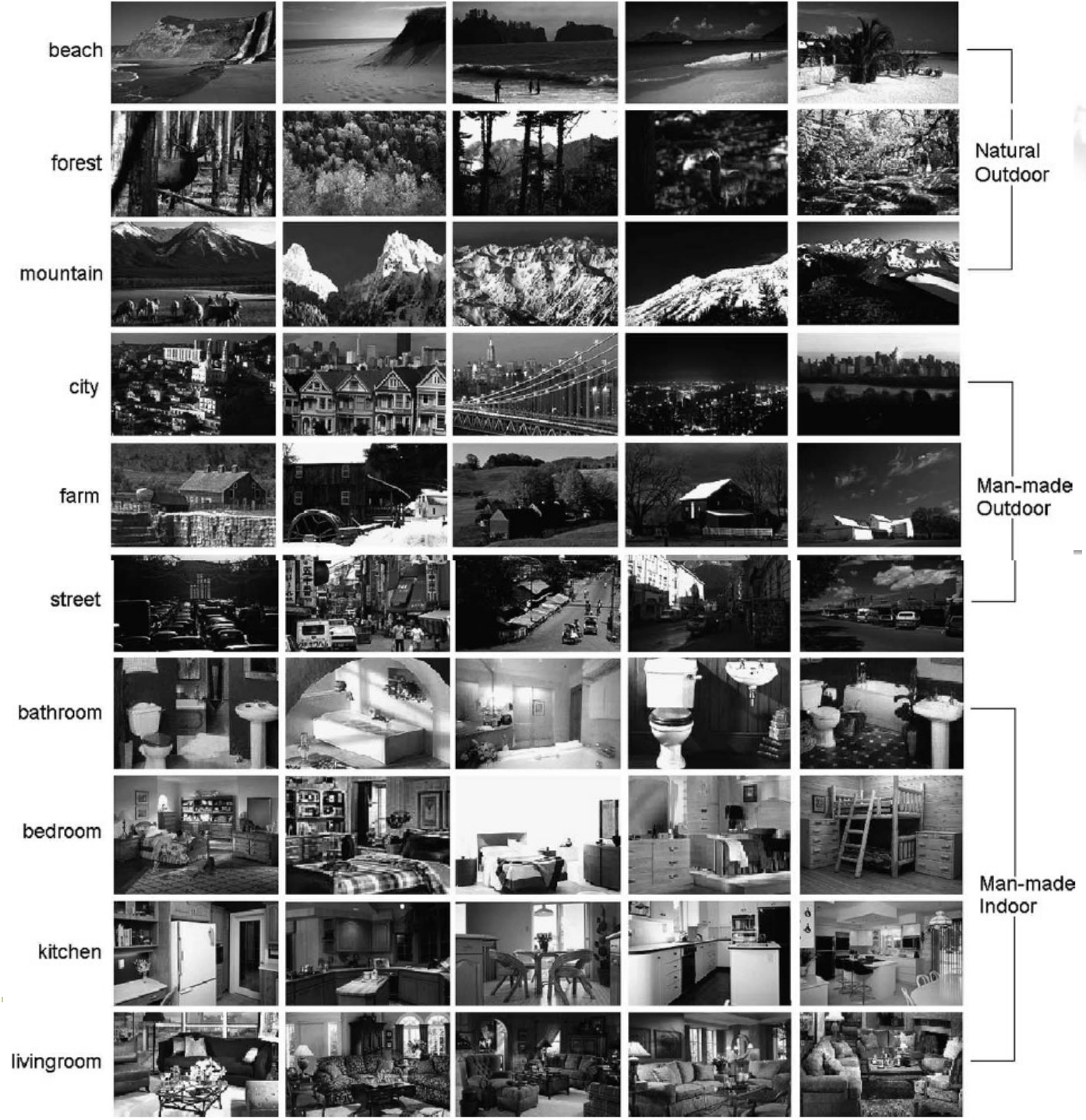
Figure by Varma & Zisserman

Texture features for image retrieval



Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99-121, November 2000,

Characterizing scene categories by texture



L. W. Renninger and J. Malik. When is scene identification just texture recognition? Vision Research 44 (2004) 2301–2311

Segmenting aerial imagery by textures

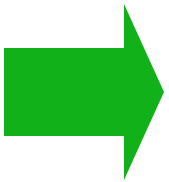
http://www.airventure.org/2004/gallery/images/073104_satellite.jpg

Texture-related tasks

- **Shape from texture**
  - Estimate surface orientation or shape from image texture
- **Segmentation/classification** from texture cues
  - Analyze, represent texture
  - Group image regions with consistent texture
- **Synthesis**
  - Generate new texture patches/images given some examples

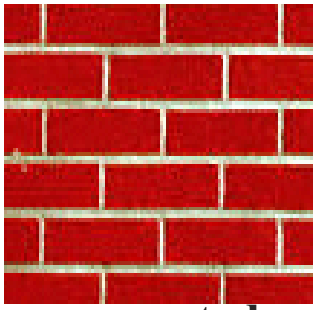
Texture synthesis

- Goal: create new samples of a given texture
- Many applications: virtual environments, hole-filling, texturing surfaces

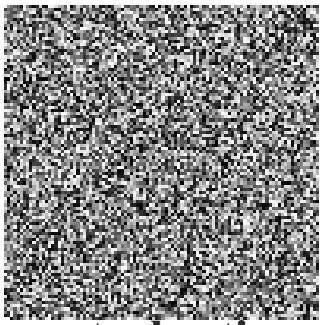


The Challenge

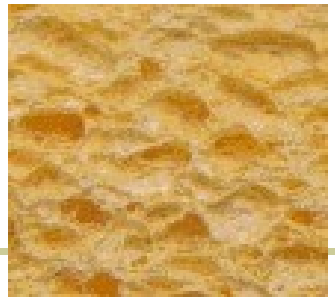
- Need to model the whole spectrum: from repeated to stochastic texture



repeated



stochastic

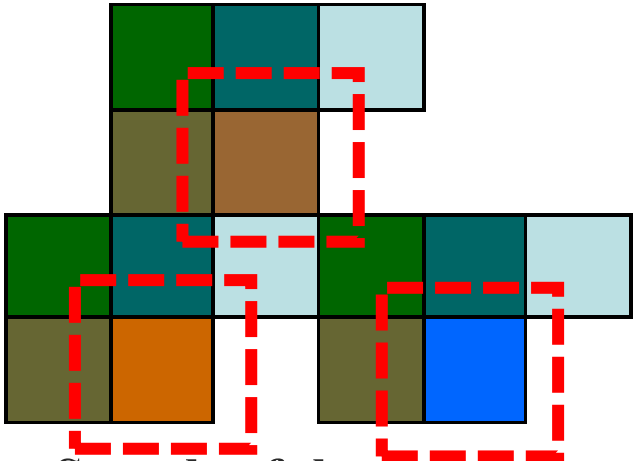


Both?

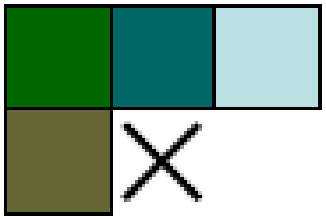
Alexei A. Efros and Thomas K. Leung, "Texture Synthesis by Non-parametric Sampling," Proc. International Conference on Computer Vision (ICCV), 1999.

Texture synthesis: intuition

Now we want to insert **pixel intensities** based on existing nearby pixel values.



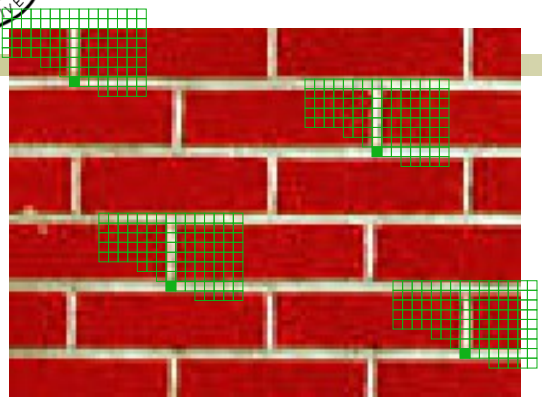
Sample of the texture ("corpus")



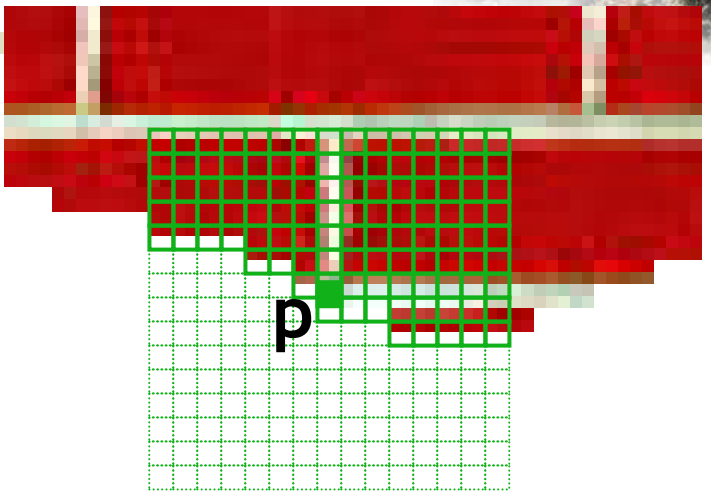
Place we want to insert next

Distribution of a value of a pixel is conditioned on its neighbors alone.

Synthesizing One Pixel



input image

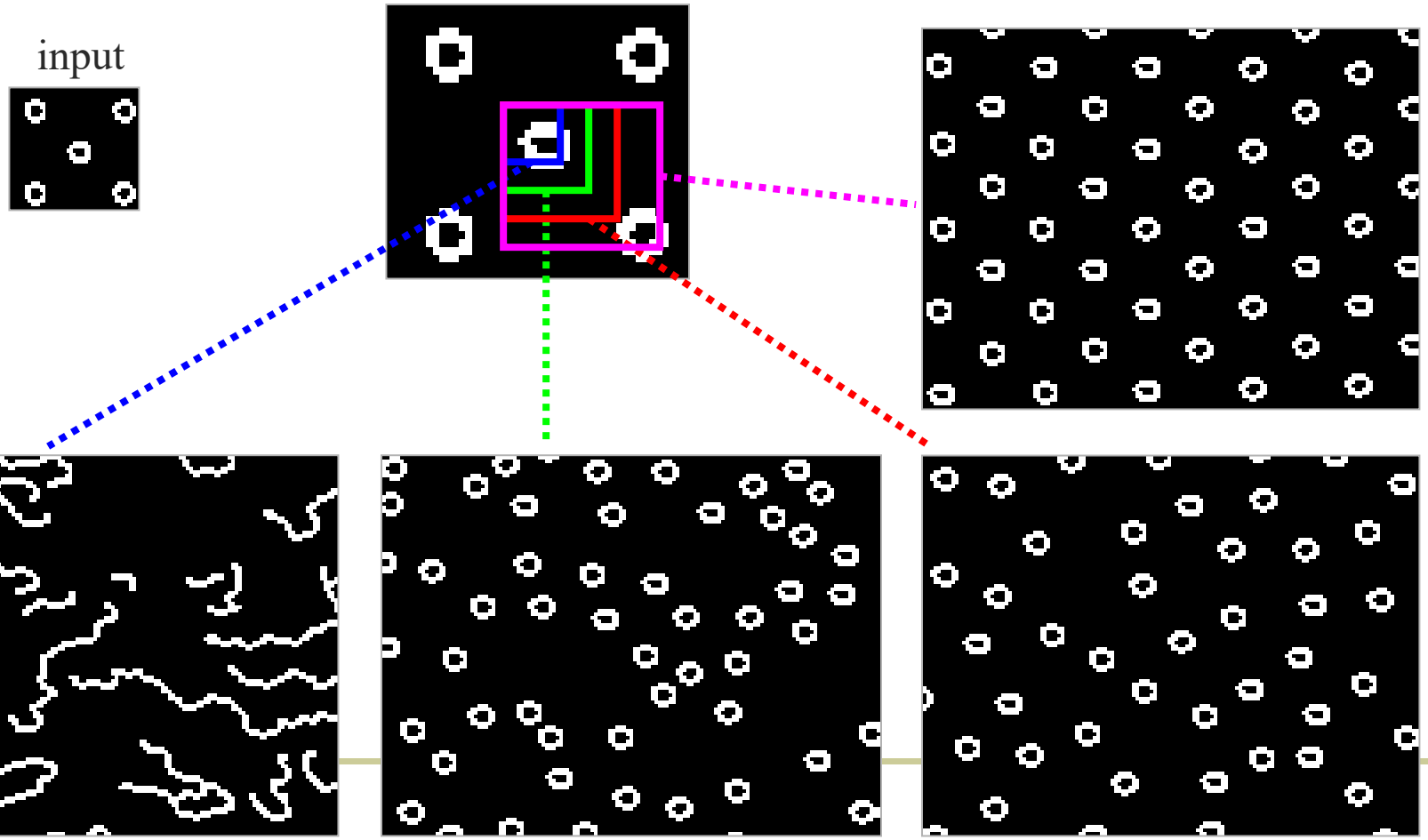


synthesized image

- What is $P(\mathbf{x}|\text{neighborhood of pixels around x})$?
- Find all the windows in the image that match the neighborhood
- To synthesize **x**
  - pick one matching window at random
  - assign **x** to be the center pixel of that window

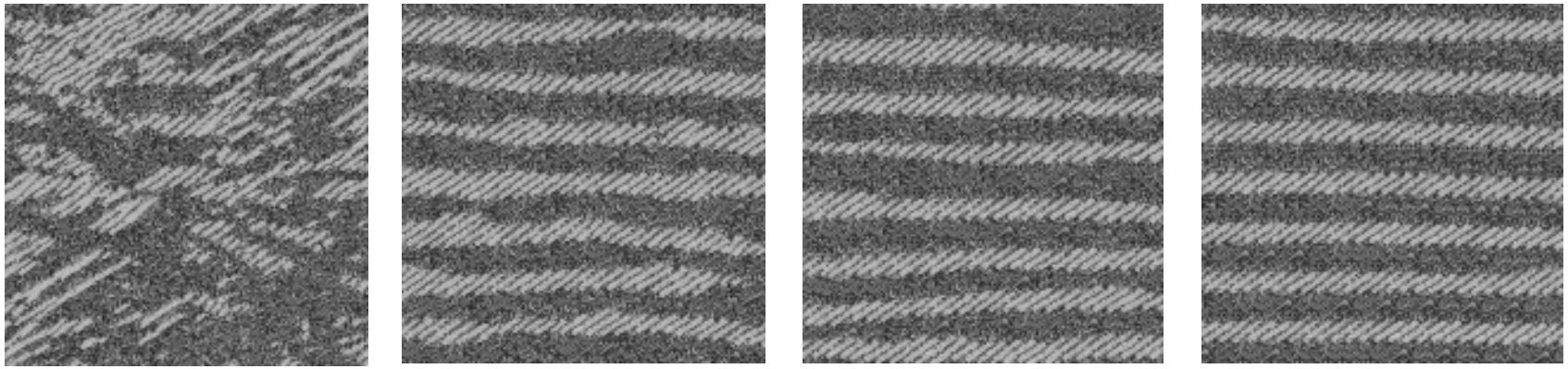
Slide from Alyosha Efros, ICCV 1999

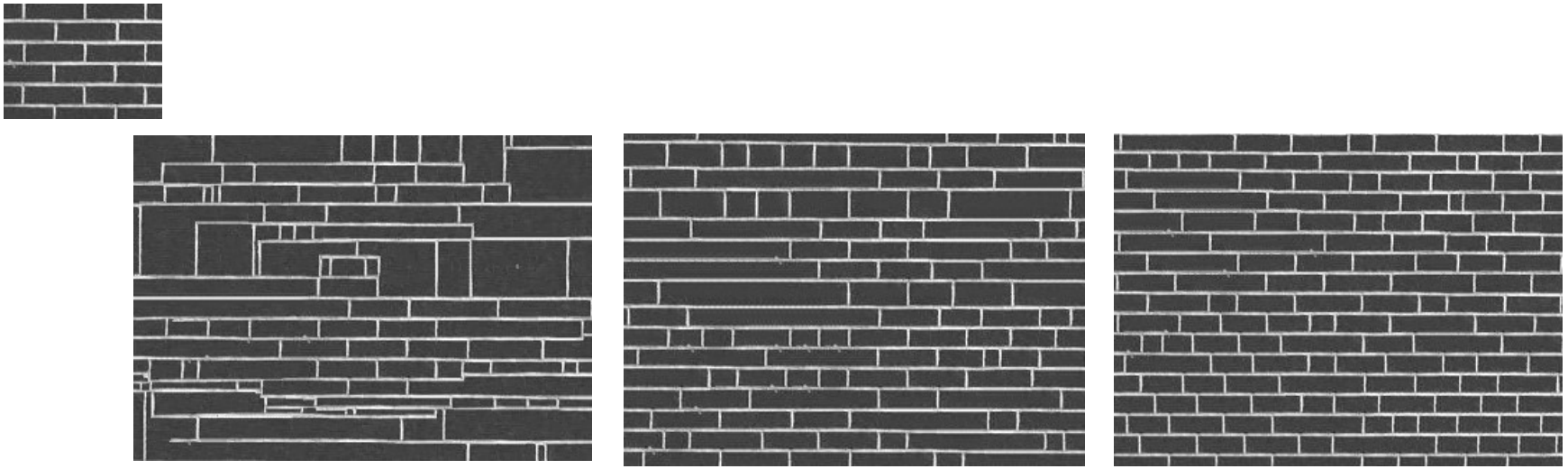
Neighborhood Window



Slide from Alyosha Efros, ICCV 1999

Varying Window Size



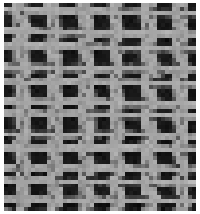


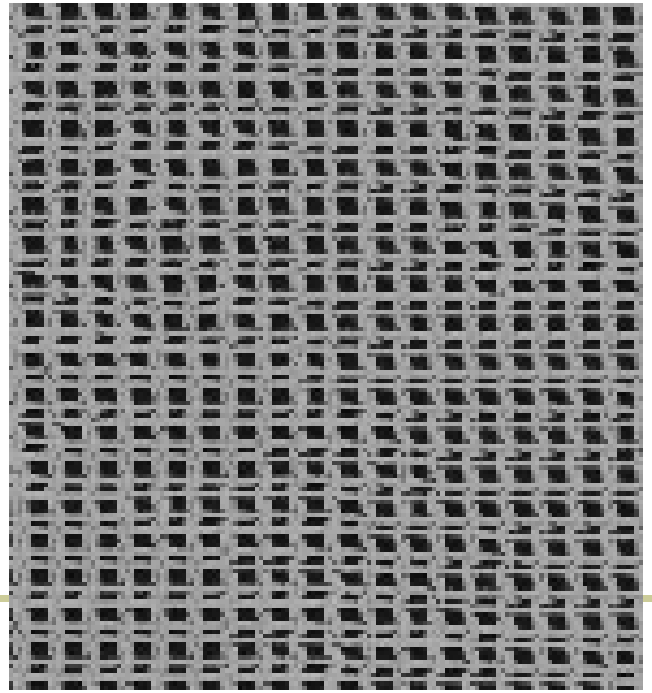
Increasing window size

Slide from Alyosha Efros, ICCV 1999

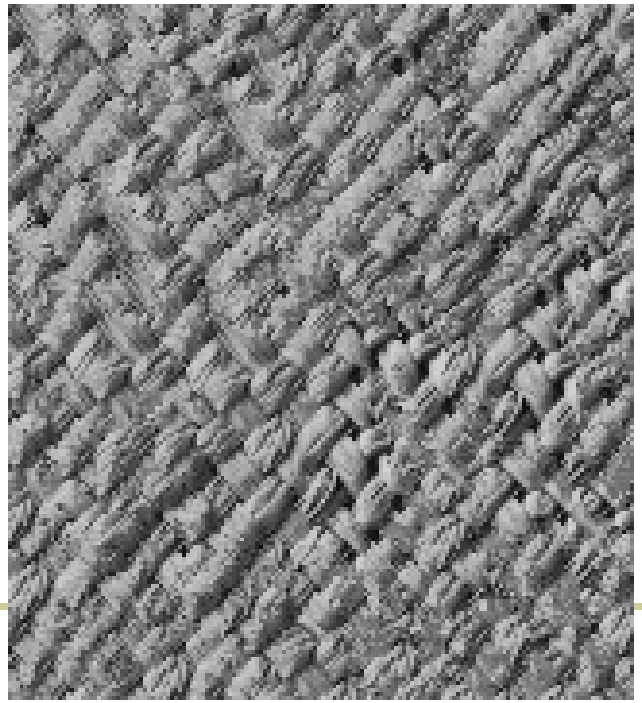
Synthesis results

french canvas





rafia weave

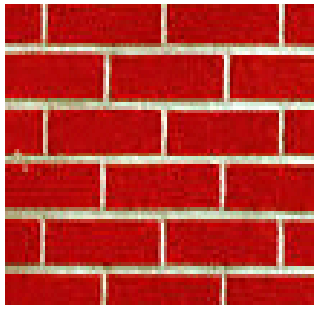


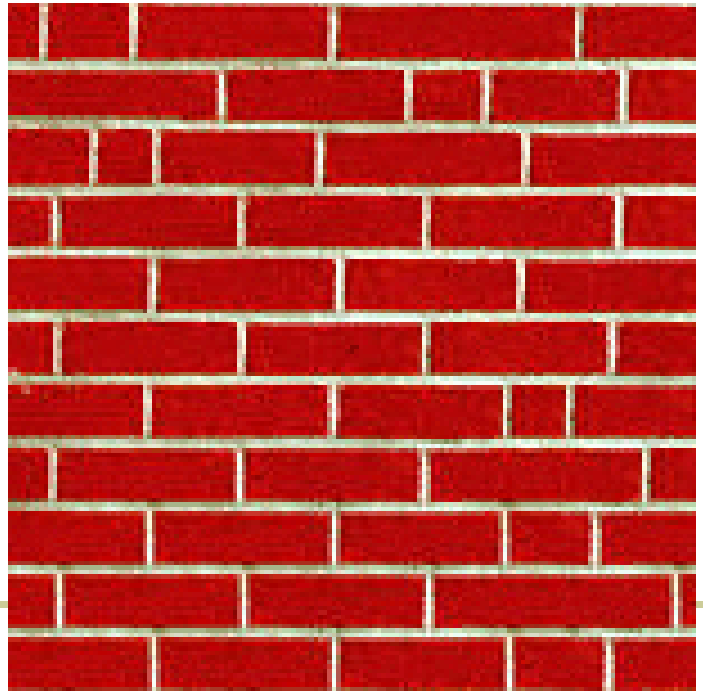
Slide from Alyosha Efros, ICCV 1999

Synthesis results

white bread

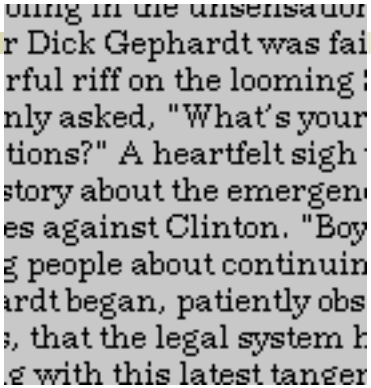
brick wall

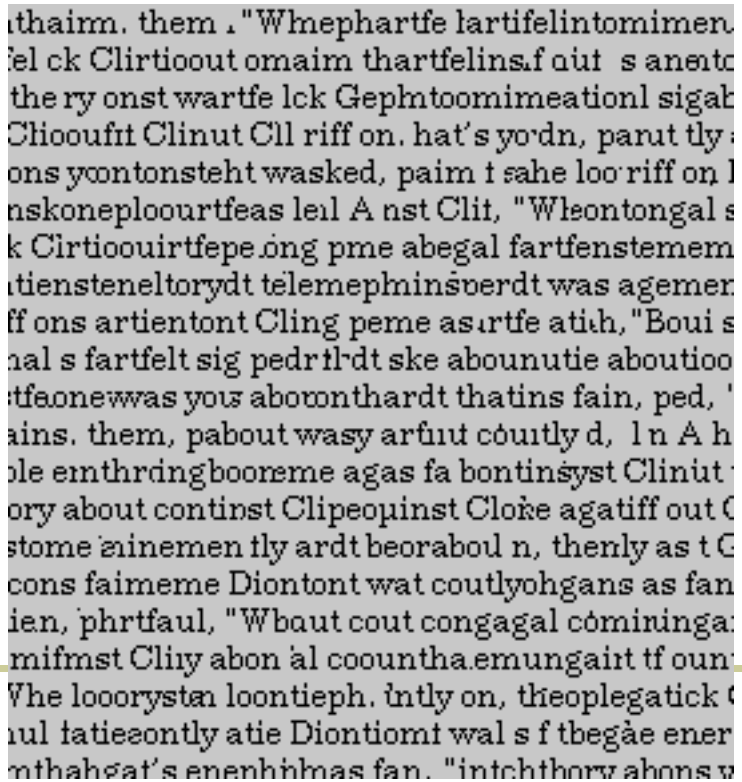




Slide from Alyosha Efros, ICCV 1999

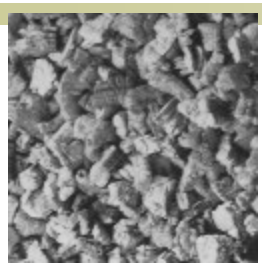
Synthesis results

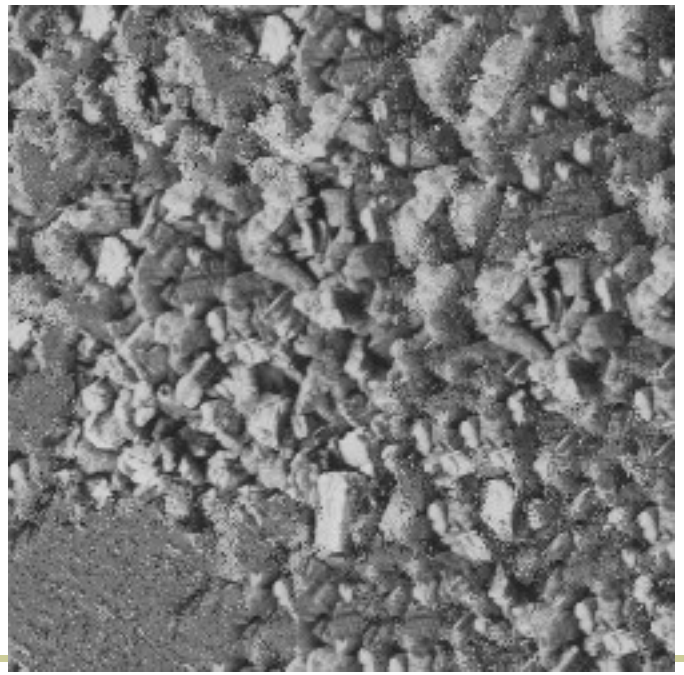




Slide from Alyosha Efros, ICCV 1999

Failure Cases



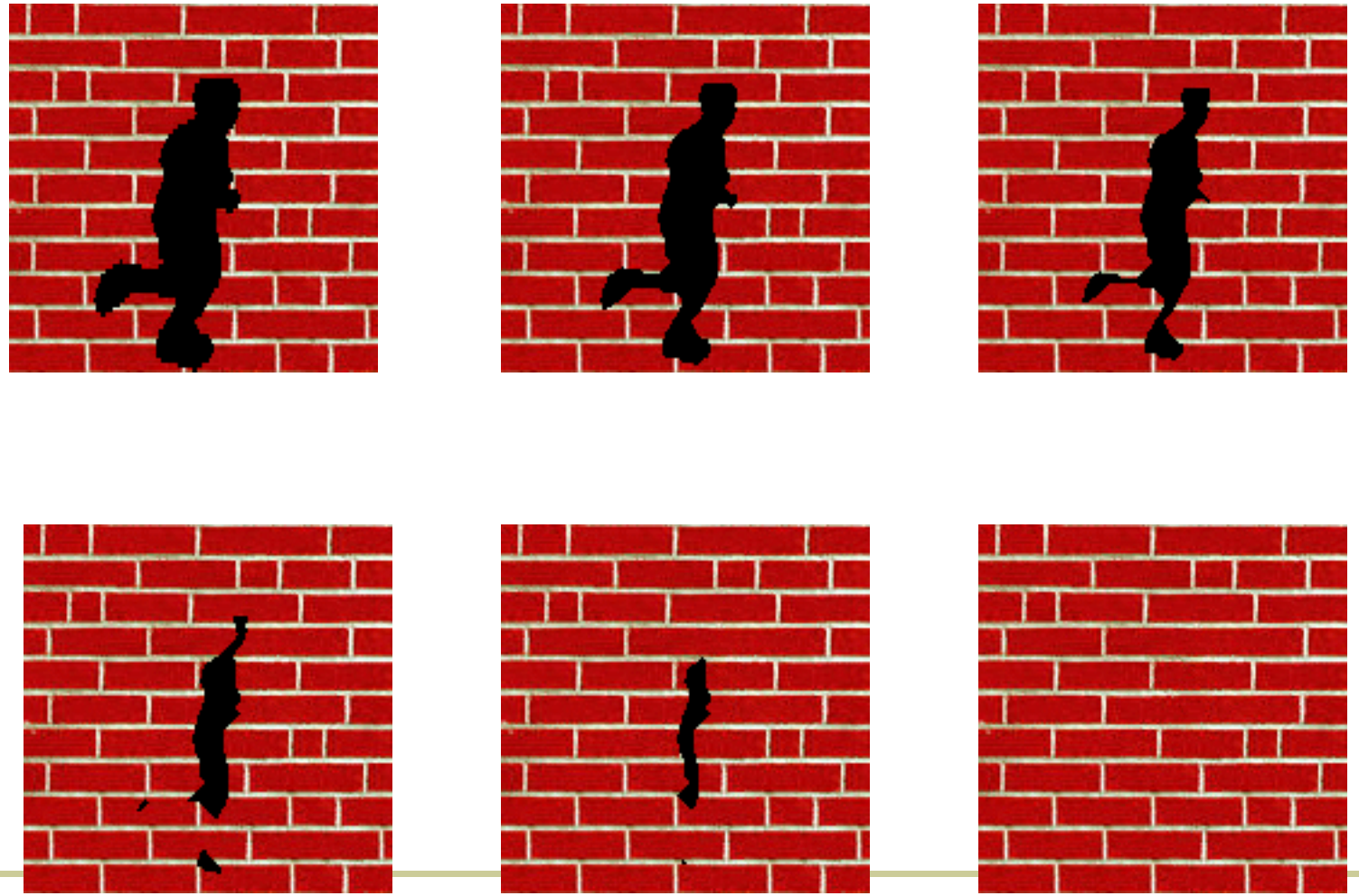


Growing garbage

Verbatim copying

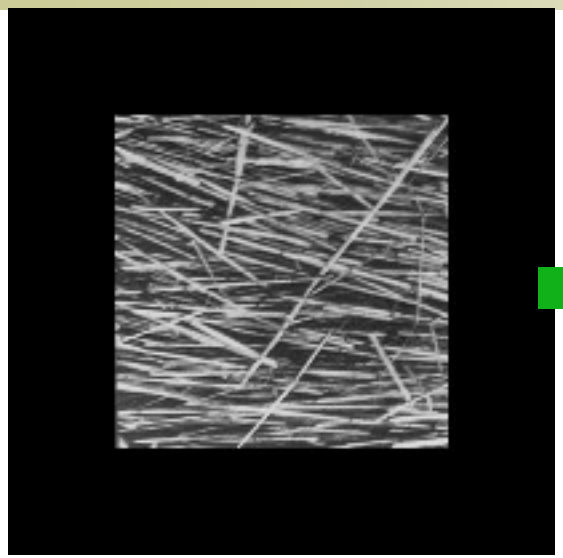
Slide from Alyosha Efros, ICCV 1999

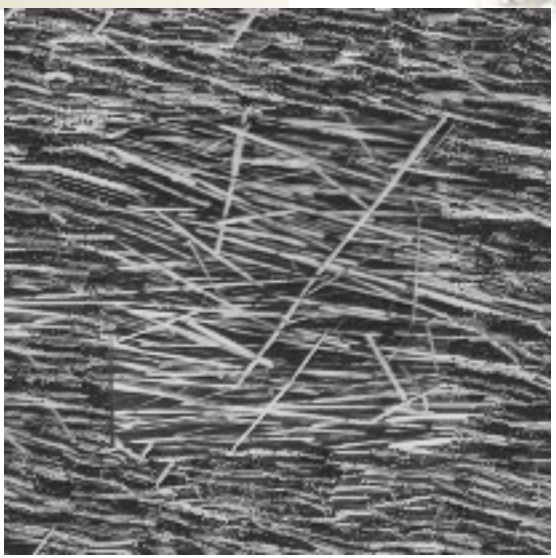
Hole Filling

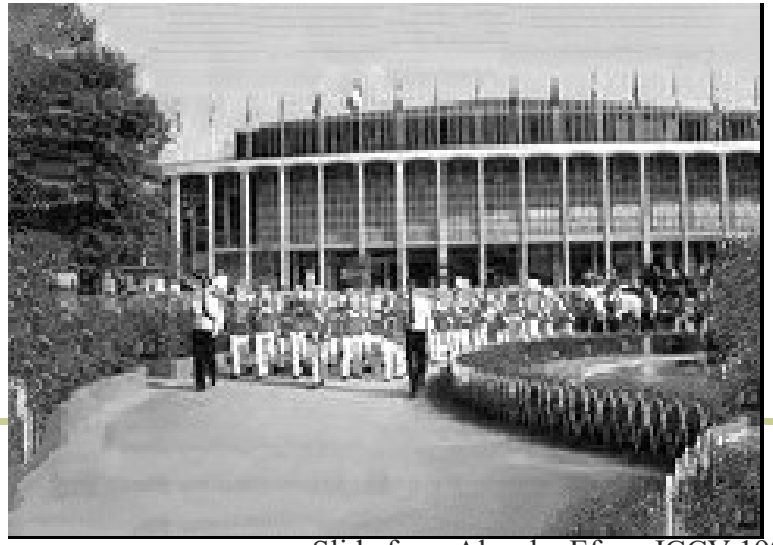


Slide from Alyosha Efros, ICCV 1999

Extrapolation



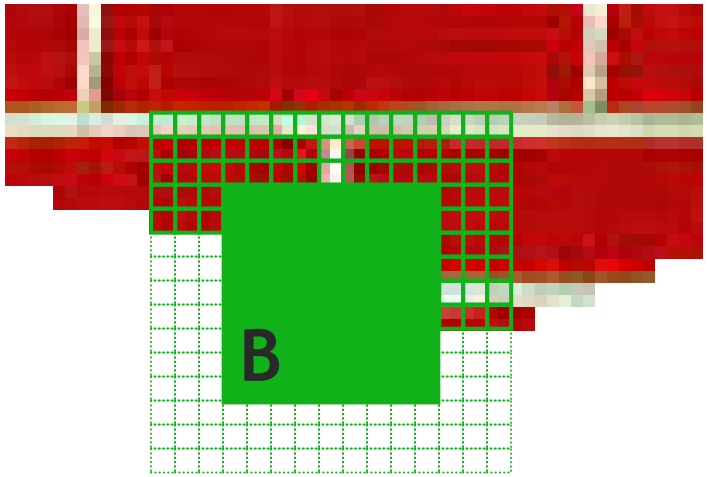




Slide from Alyosha Efros, ICCV 1999

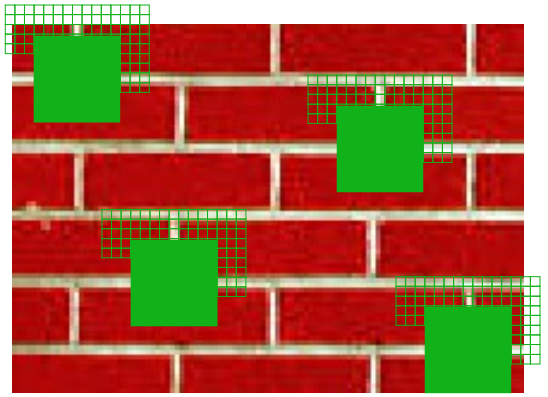
- The Efros & Leung algorithm
  - Simple
  - Surprisingly good results
  - Synthesis is easier than analysis!
  - …but very slow

Image Quilting [Efros & Freeman 2001]



Synthesizing a block

non-parametric sampling



Input image

- Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once

Slide from Alyosha Efros, ICCV 1999

Summary

- Template, Pyramid, and Texture
  - Template matching (SSD or Normxcorr2)
    - SSD can be done with linear filters, is sensitive to overall intensity
  - Gaussian pyramid
    - Coarse-to-fine search, multi-scale detection
  - Laplacian pyramid
    - More compact image representation
    - Can be used for compositing in graphics
  - Steerable pyramid
    - Filter banks for representing texture