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/

Assignment 1

Format requirements

(i) Project materials including report template: `proj1.zip`

(ii) Please use the provided **proj1_report.pptx** template to write your report, and convert the slide deck into a PDF for your submission. The report should contain your name, student ID, and e-mail address;

(iii) You should choose python to write your code, and provide a README file to describe how to execute the code;

(iv) Pack your **proj1_report.pdf**, **proj1_code** and **README** into a zip file, named with your student ID, like MG1933001.zip. If you have an improved version, add an extra '_' with a number, like MG1933001_1.zip. We will take the final submitted version as your results.

(v) Do **not** install any additional packages inside the conda environment. The TAs will use the same environment as defined in the config files we provide you, so anything that's not in there by default will probably cause your code to break during grading. Do not use absolute paths in your code or your code will break. Use relative paths like the starter code already does. Failure to follow any of these instructions will lead to point deductions.

Assignment 1

Submission Way

(i) Please submit your results to email cvcourse.nju@gmail.com , the email subject is "Assignment 1";

(ii) The deadline is 23:59 on April 10, 2020. No submission after this deadline is acceptable.

About Plagiarize

DO NOT PLAGIARIZE! We have no tolerance for plagiarizing and will penalize it with giving zero score. You may refer to some others' materials, please make citations such that one can tell which part is actually yours.

Evaluation Criterion

We mainly evaluate your submission according to your code and report. Efficient implementation, elegant code style, concise and logical report are all important factors towards a high score.

Image filtering

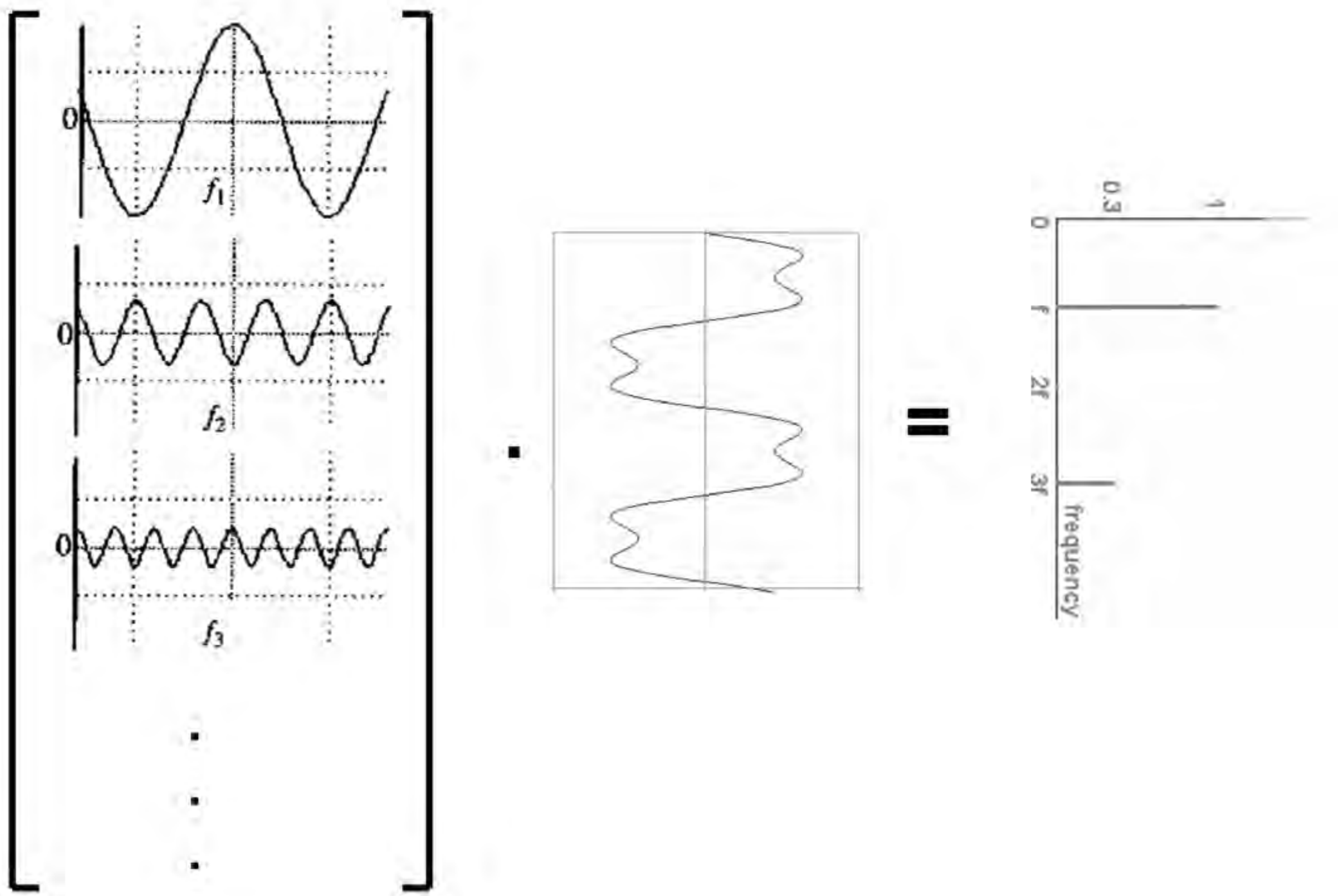
- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Three views of filtering

- Image filters in spatial domain
  - Filter is a mathematical operation on values of each patch
  - Smoothing, sharpening, measuring texture
- **Image filters in the frequency domain**
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression
- Templates and Image Pyramids
  - Filtering is a way to match a template to the image
  - Detection, coarse-to-fine registration

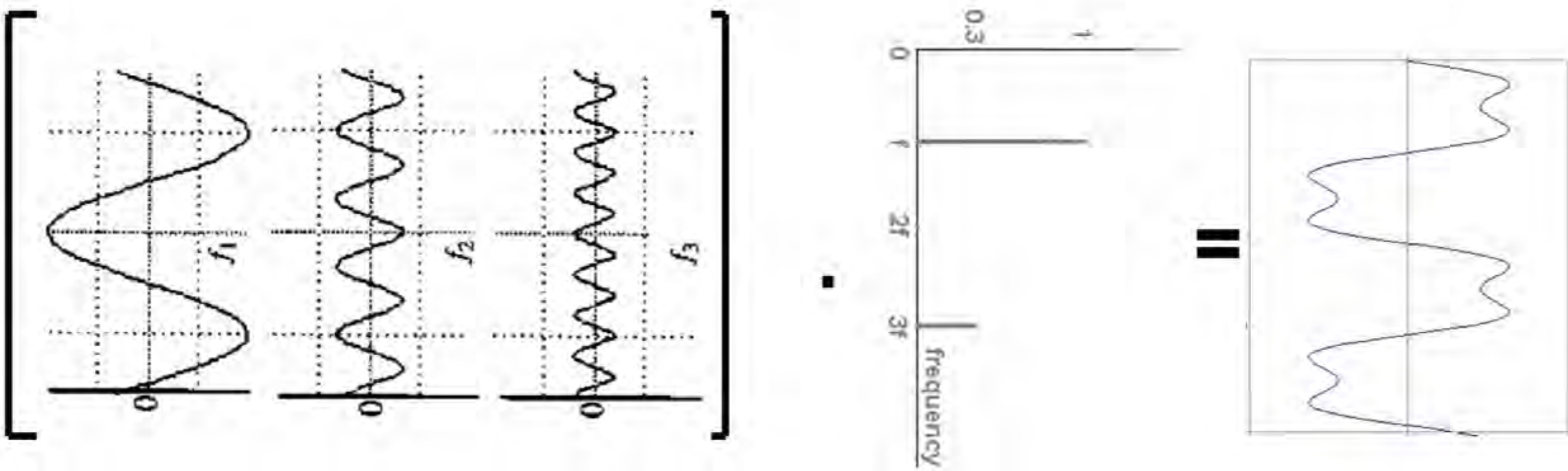
Fourier series: just a change of basis

$$M\ f(x) = F(\omega)$$



Inverse FT: Just a change of basis

$$M^{-1}\, F(\omega) = f(x)$$



-
-
-

The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal *f* [*n*] into *F* [*u*] as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

The inverse of the DFT is:

$$f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp\left(2\pi j \frac{un}{N}\right)$$

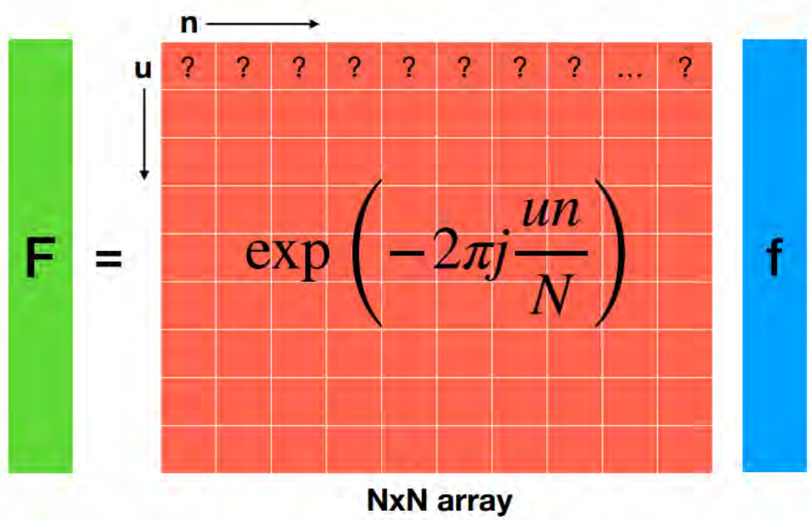
The signal *f* [*n*] is a weighted linear combination of complex exponentials with weights *F* [*u*]

The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal *f* [*n*] into *F* [*u*] as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

Discrete Fourier Transform (DFT) is a linear operator. Therefore, we can write:



Lets visualize the transform coefficients

69

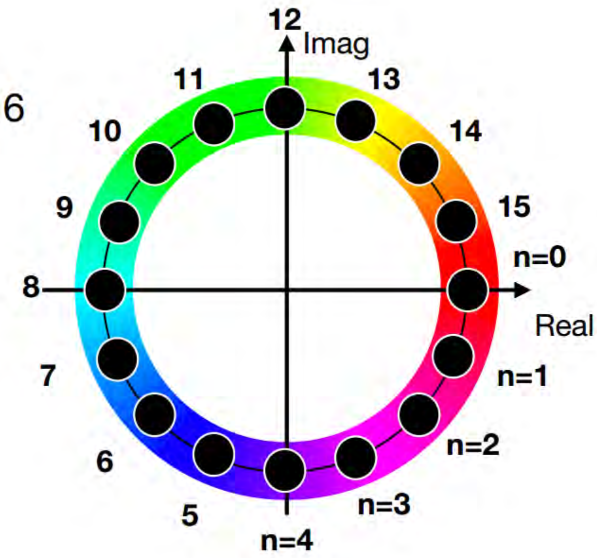
Visualizing the Fourier transform

$$F[u] = \sum_{n=0}^{N-1} f[n] \boxed{\exp\left(-2\pi j \frac{un}{N}\right)}$$

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

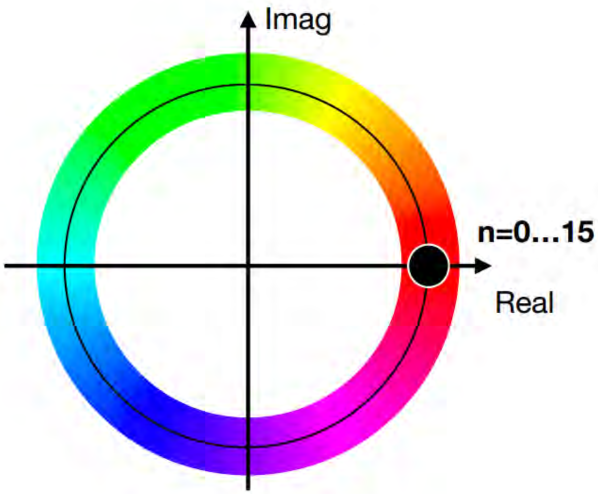
$$\cos\left(2\pi\frac{un}{N}\right) - j\,\sin\left(2\pi\frac{un}{N}\right)$$

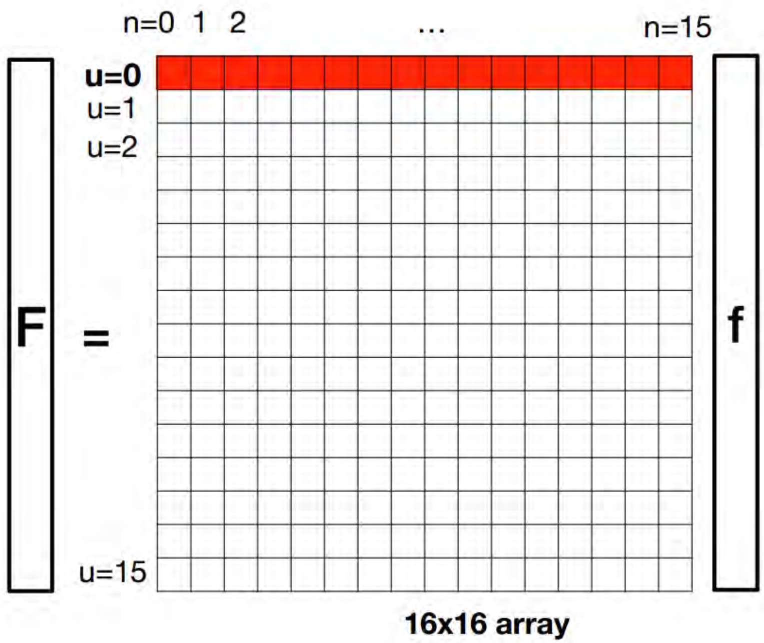
For:
u=1
N=16



Visualizing the transform coefficients

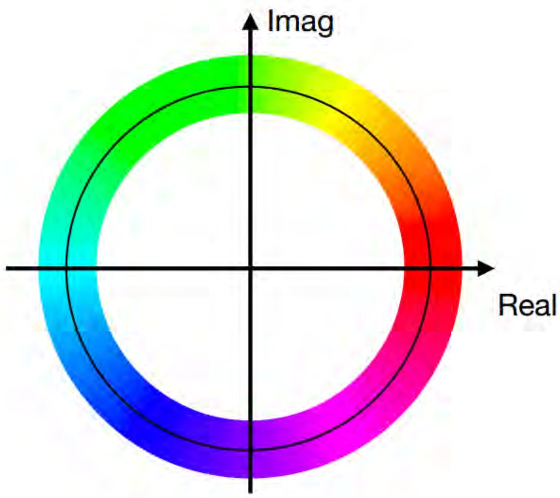
$$\exp\left(-2\pi j \frac{un}{N}\right)$$ For N=16

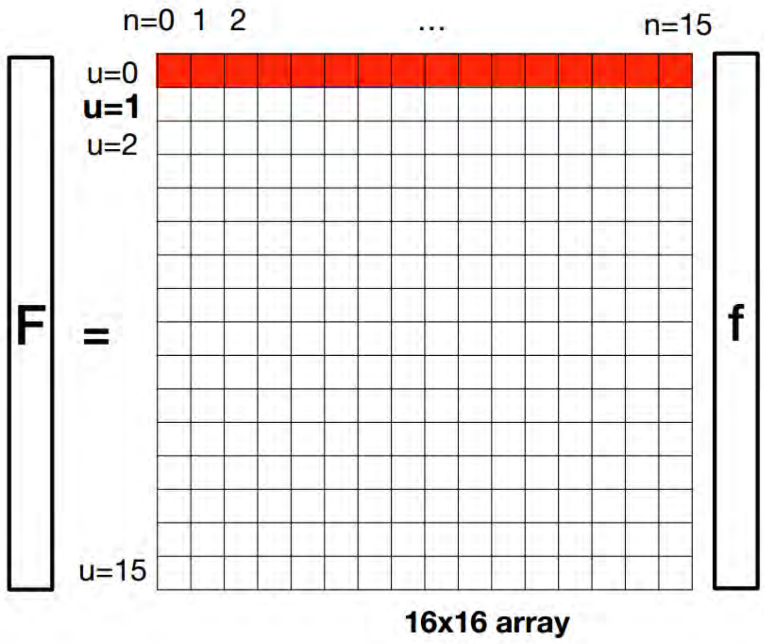




Visualizing the transform coefficients

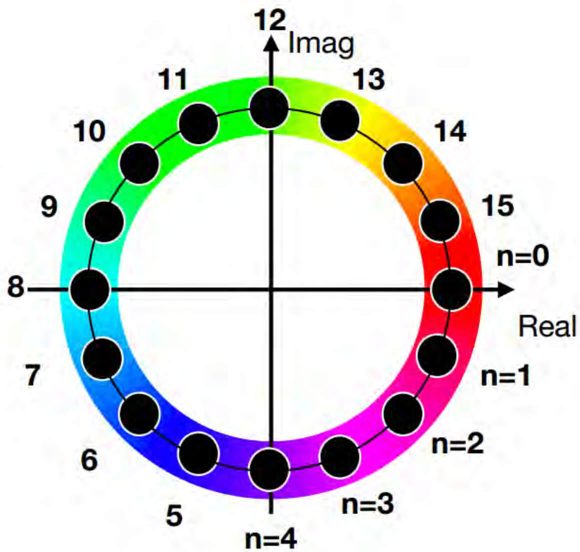
$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

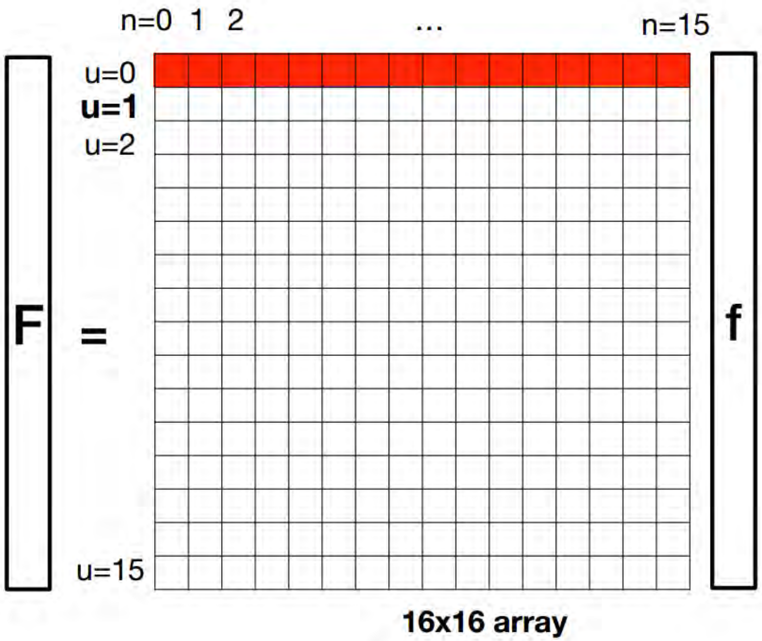




Visualizing the transform coefficients

$$\exp\left(-2\pi j\frac{un}{N}\right)$$ For N=16

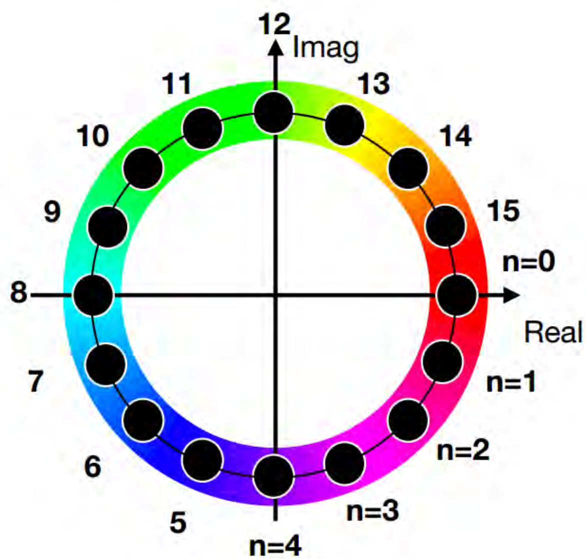


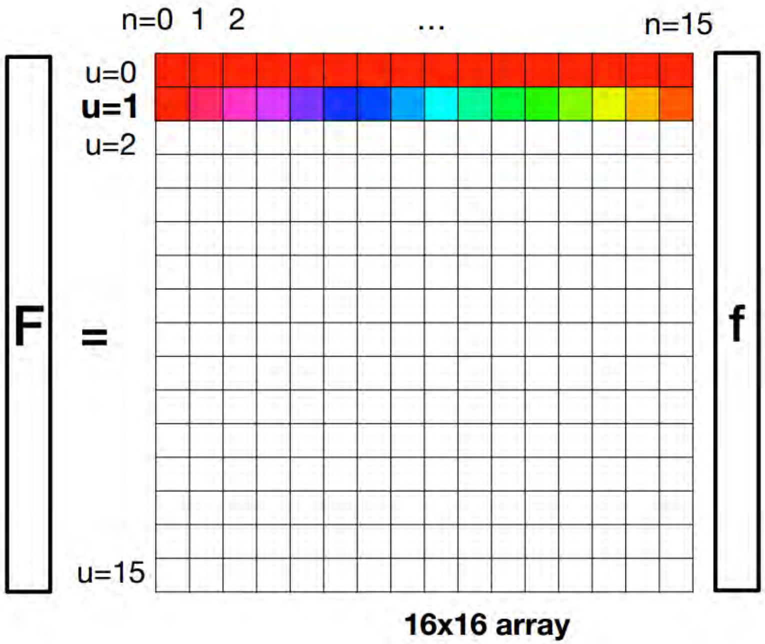


Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$

For N=16

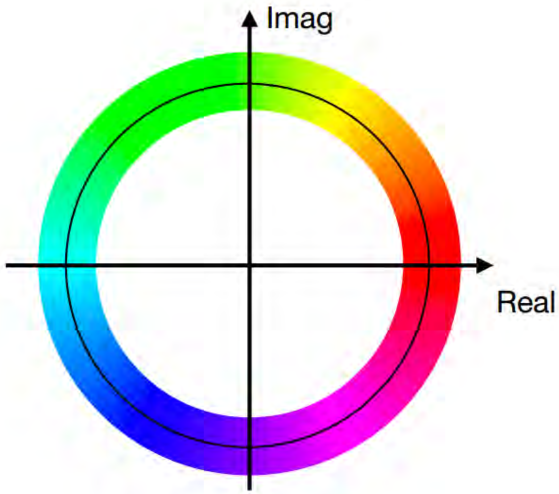


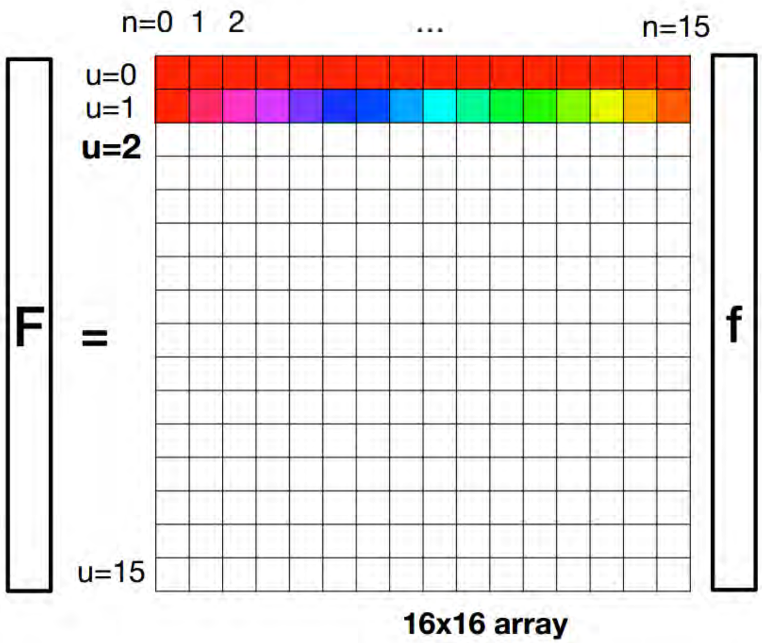


Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$

For N=16

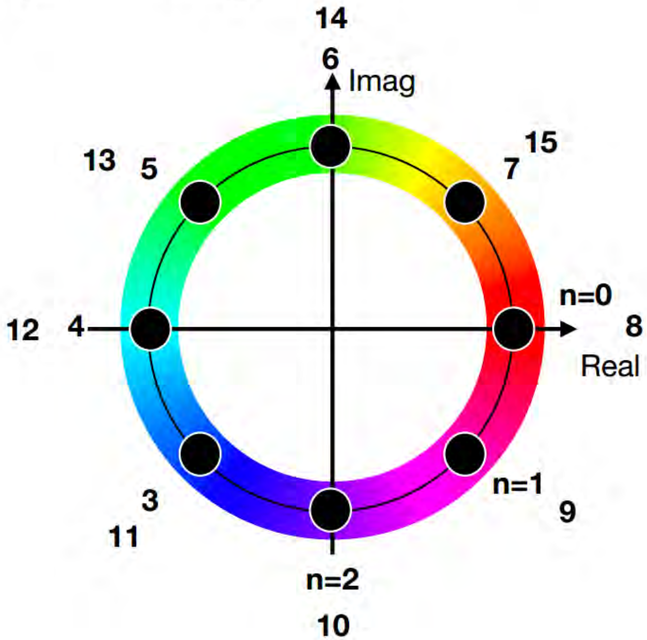


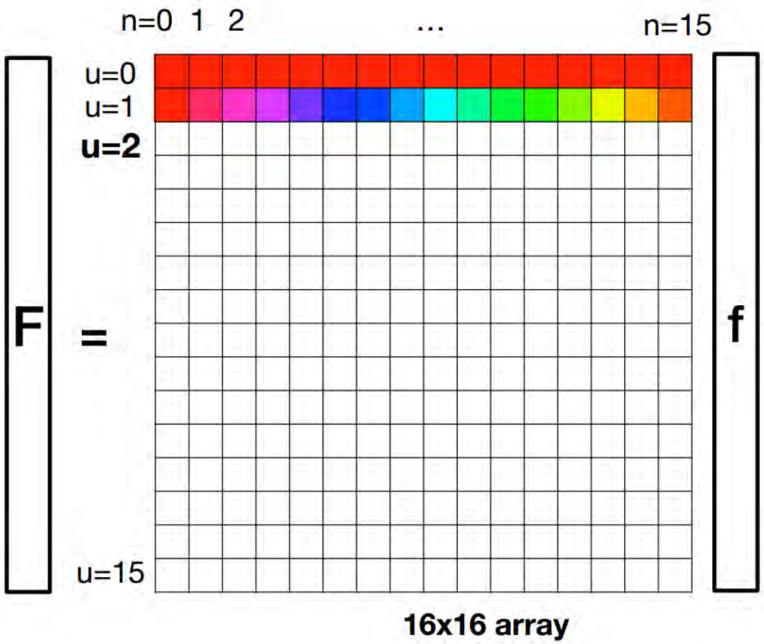


Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$

For N=16

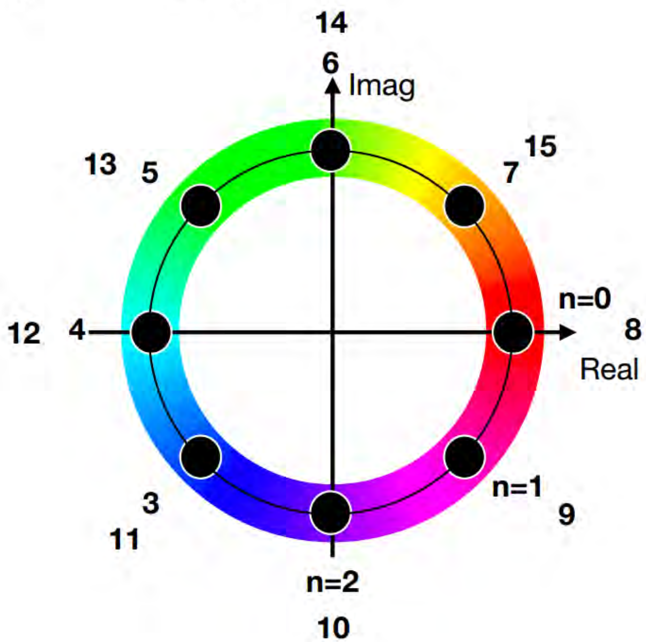


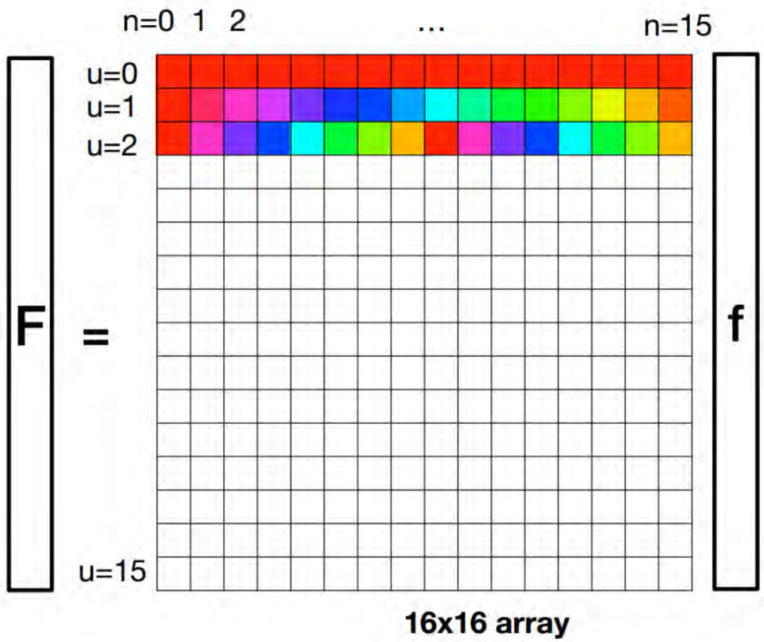


Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$

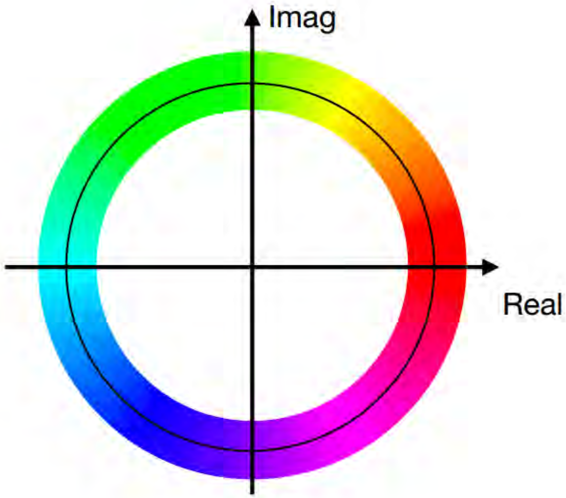
For N=16

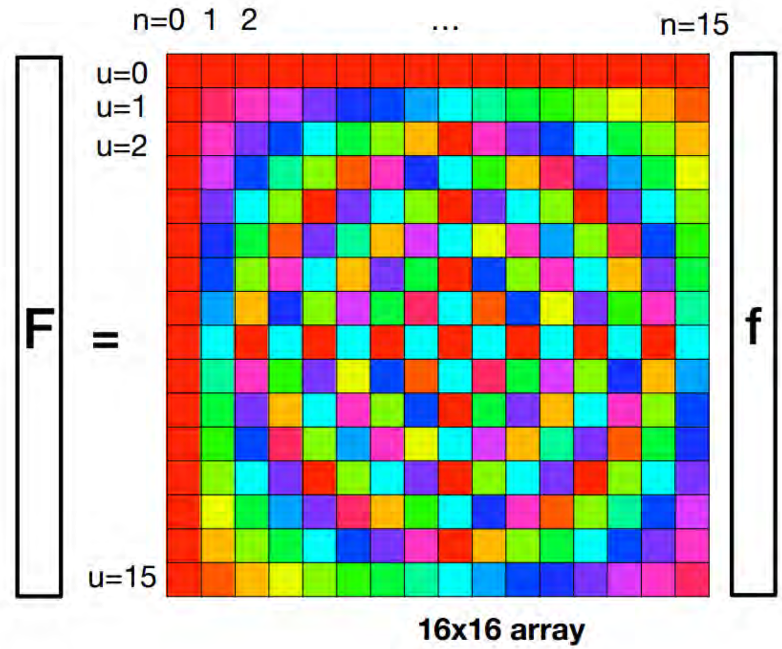




Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$ For N=16

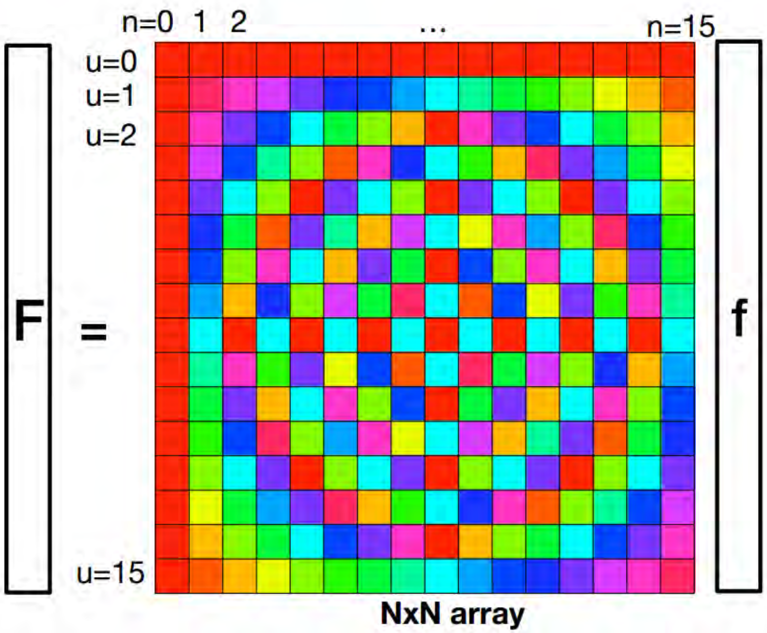




The inverse of the Discrete Fourier transform

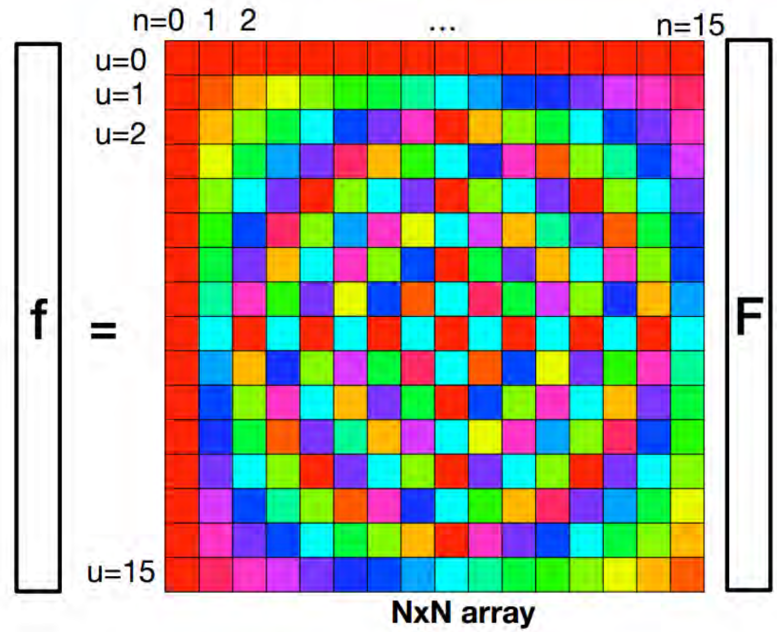
Discrete Fourier Transform (DFT):

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$



Its inverse:

$$f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp\left(2\pi j \frac{un}{N}\right)$$

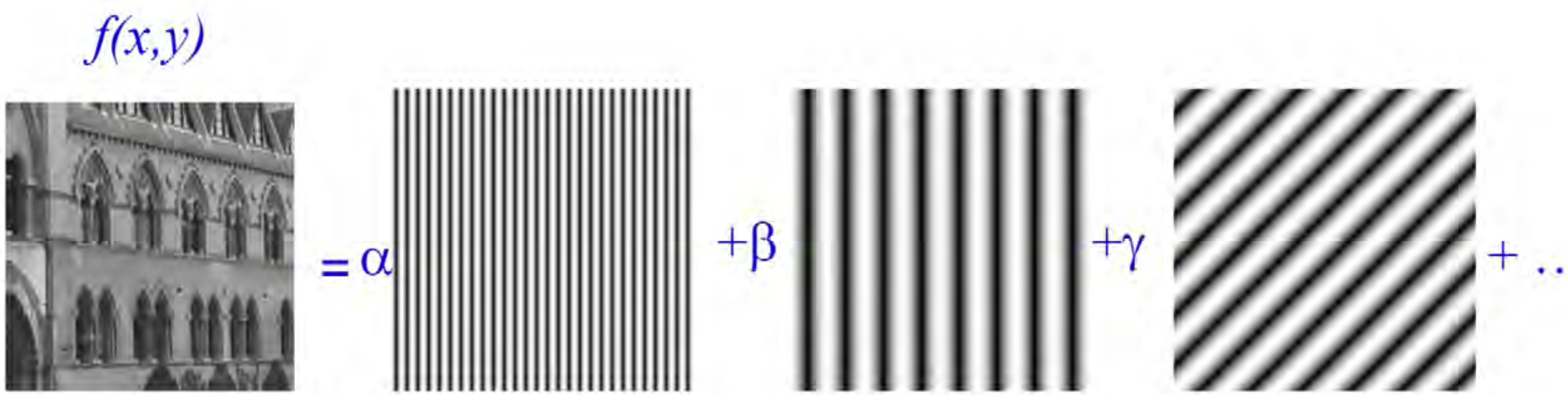


Summary

The spatial function $f(x, y)$

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

is decomposed into a weighted sum of 2D orthogonal basis functions in a similar manner to decomposing a vector onto a basis using scalar products.



2-D DFT

变换对公式：1D—>2D推广

$$F(u,v)=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\exp[-j2\pi(\frac{ux}{M}+\frac{vy}{N})]$$

$$f(x,y)=\frac{1}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)\exp[j2\pi(\frac{ux}{M}+\frac{vy}{N})]$$

频谱（幅度） $|F(u,v)|=\left[R^2(u,v)+I^2(u,v)\right]^{1/2}$

相位角 $\phi(u,v)=\arctan[I(u,v)/R(u,v)]$

功率谱 $P(u,v)=|F(u,v)|^2=R^2(u,v)+I^2(u,v)$

原图像

幅度谱

相位谱

由幅度谱重建

相位谱为0

由相位谱重建

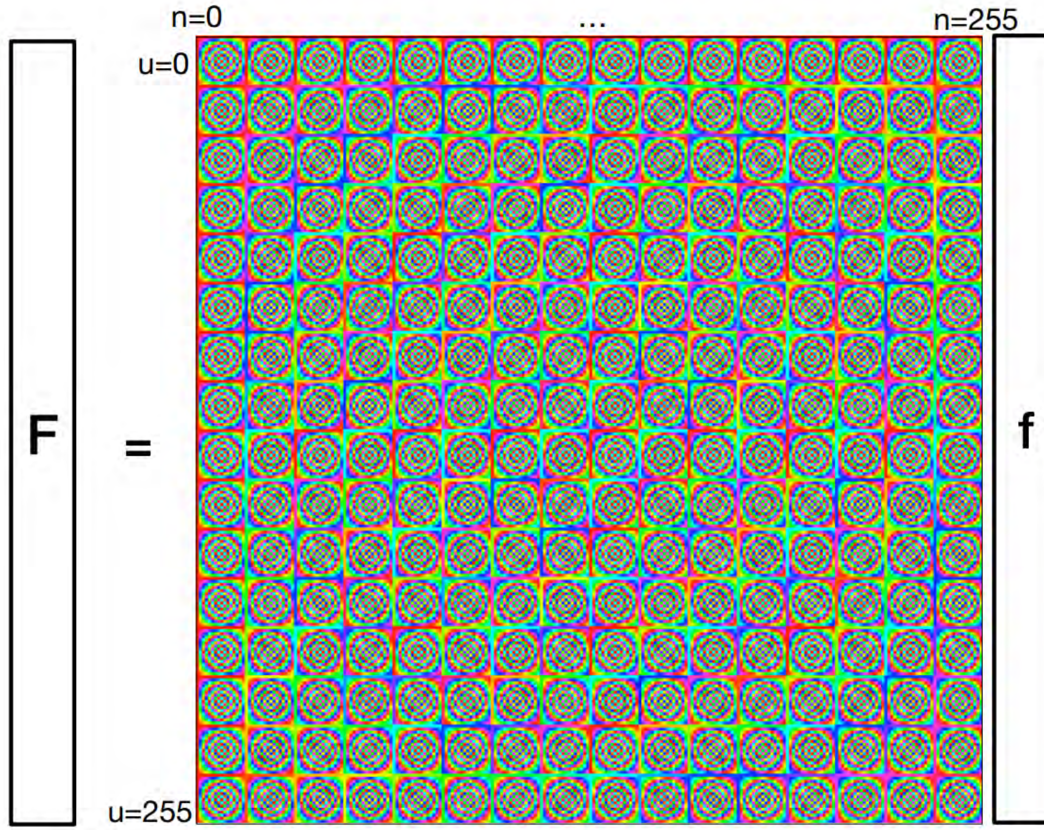
幅度谱为常数

结论：相位谱可能具有更重要的应用

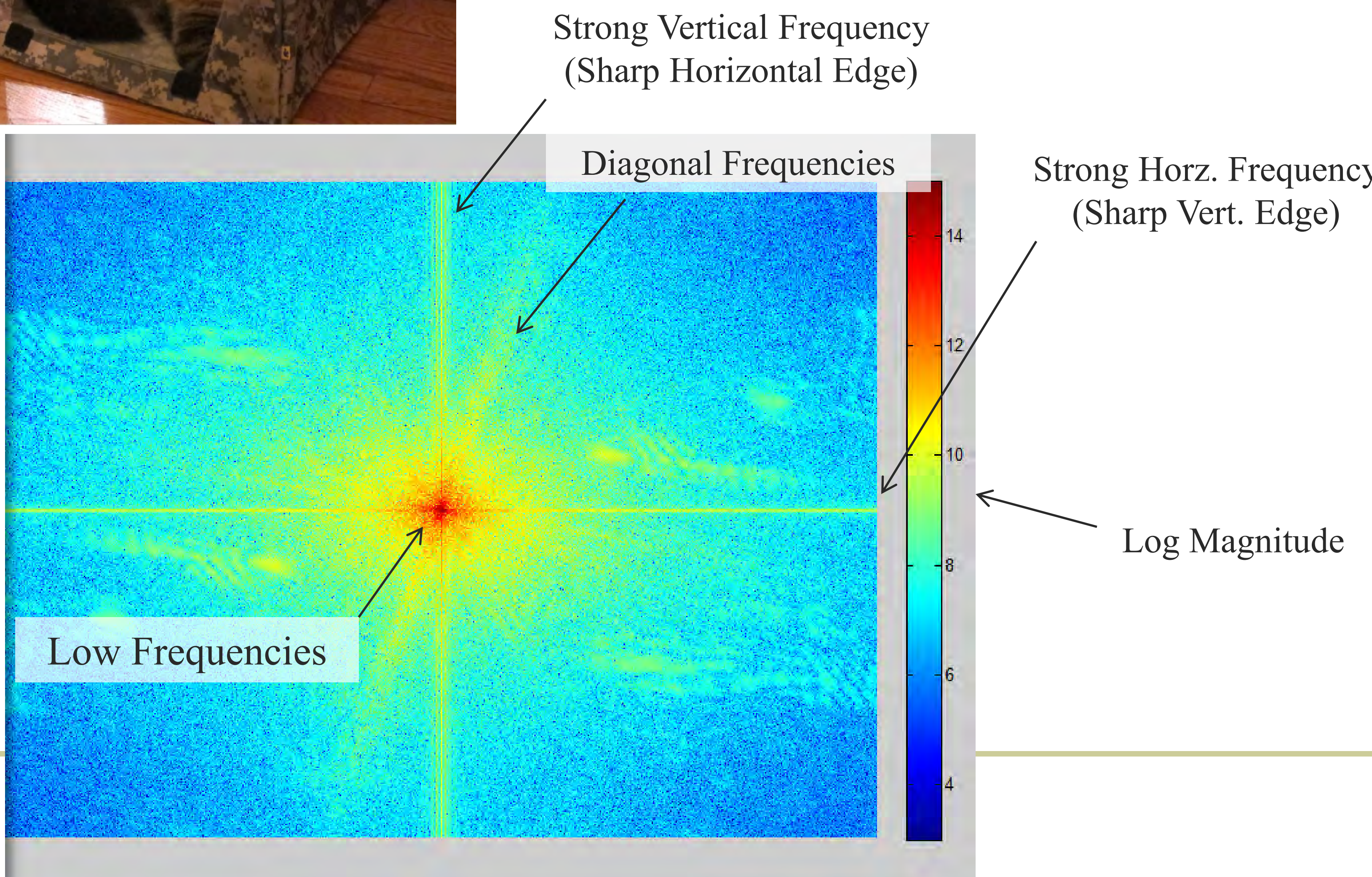
Visualizing the 2D DFT coefficients

$$\underbrace{F[u,v]}_{} = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

For N=M=16



256x256 array



Strong Vertical Frequency
(Sharp Horizontal Edge)
Diagonal Frequencies
Strong Horz. Frequency
(Sharp Vert. Edge)
Log Magnitude
Low Frequencies
14
12
10
8
6
4

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\,\mathrm{F}[h]$$

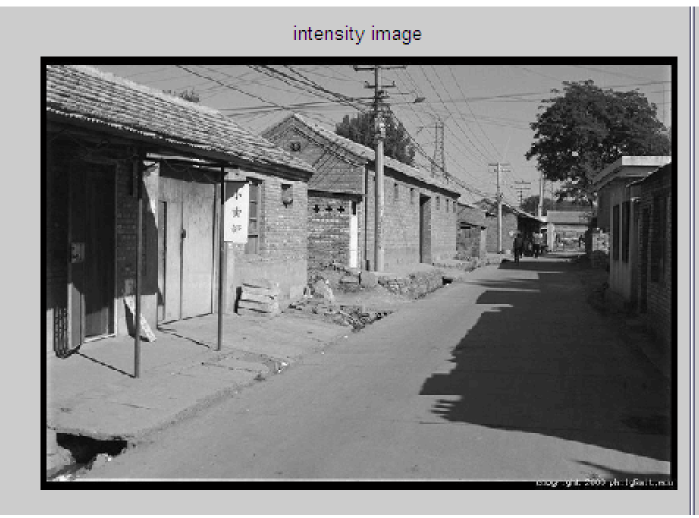
- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$\mathrm{F}^{-1}[gh] = \mathrm{F}^{-1}[g] * \mathrm{F}^{-1}[h]$$

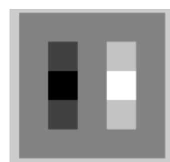
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

Filtering in spatial domain

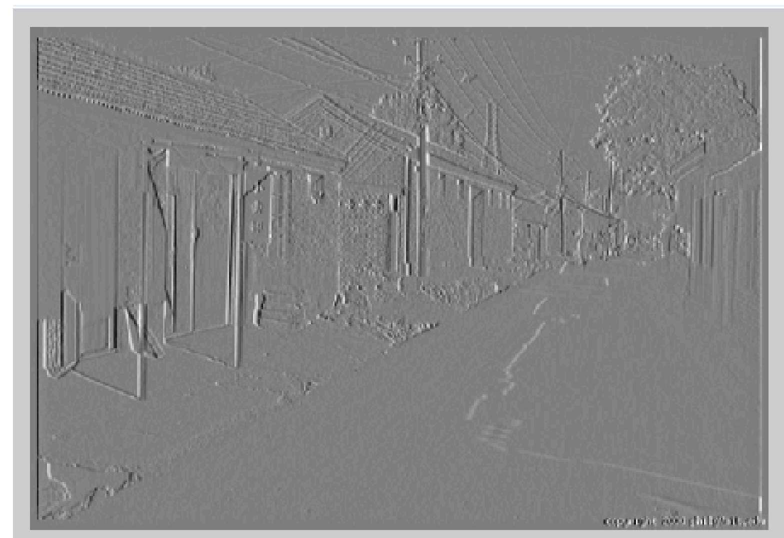
1	0	-1
2	0	-2
1	0	-1



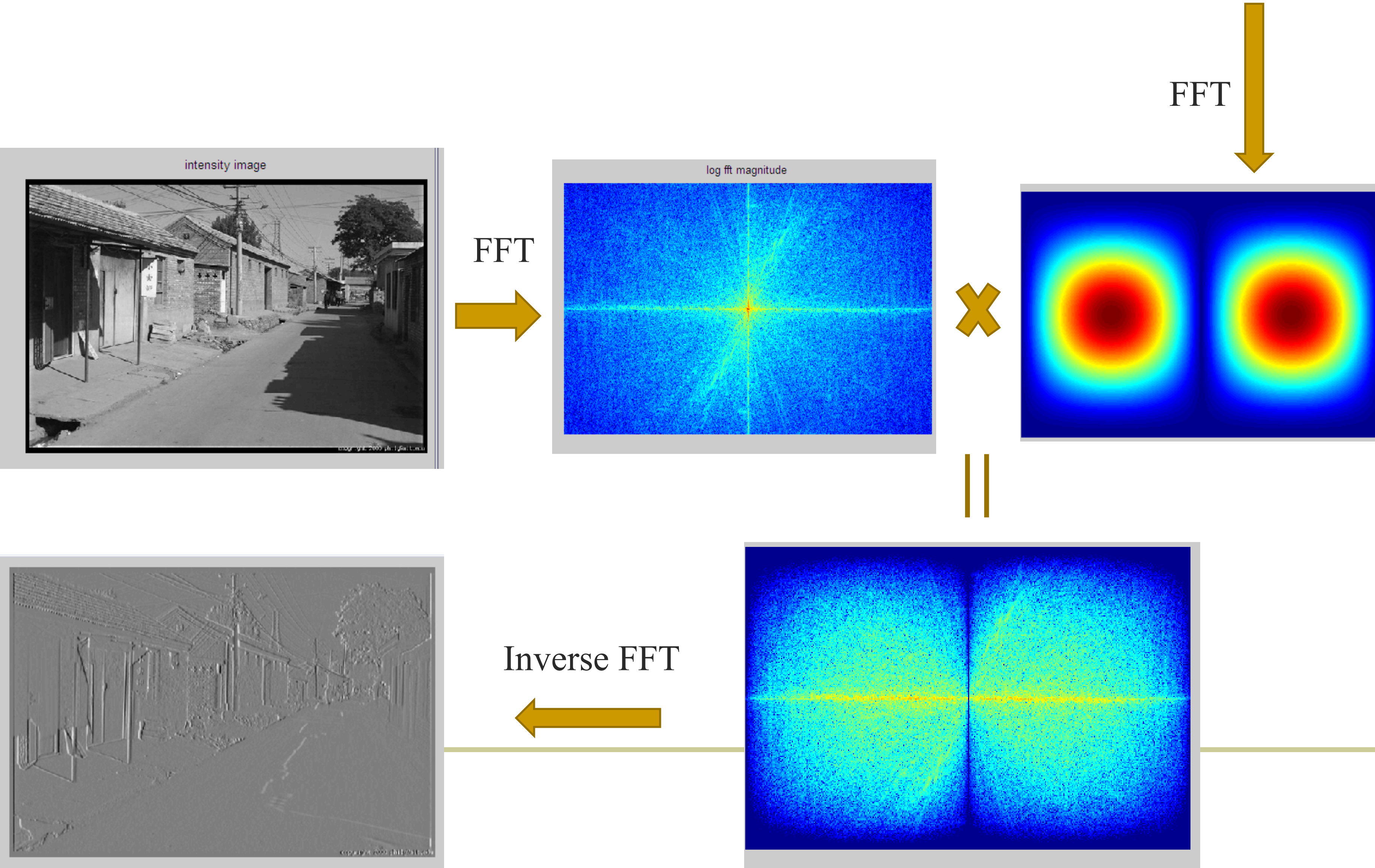
*



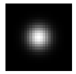
=

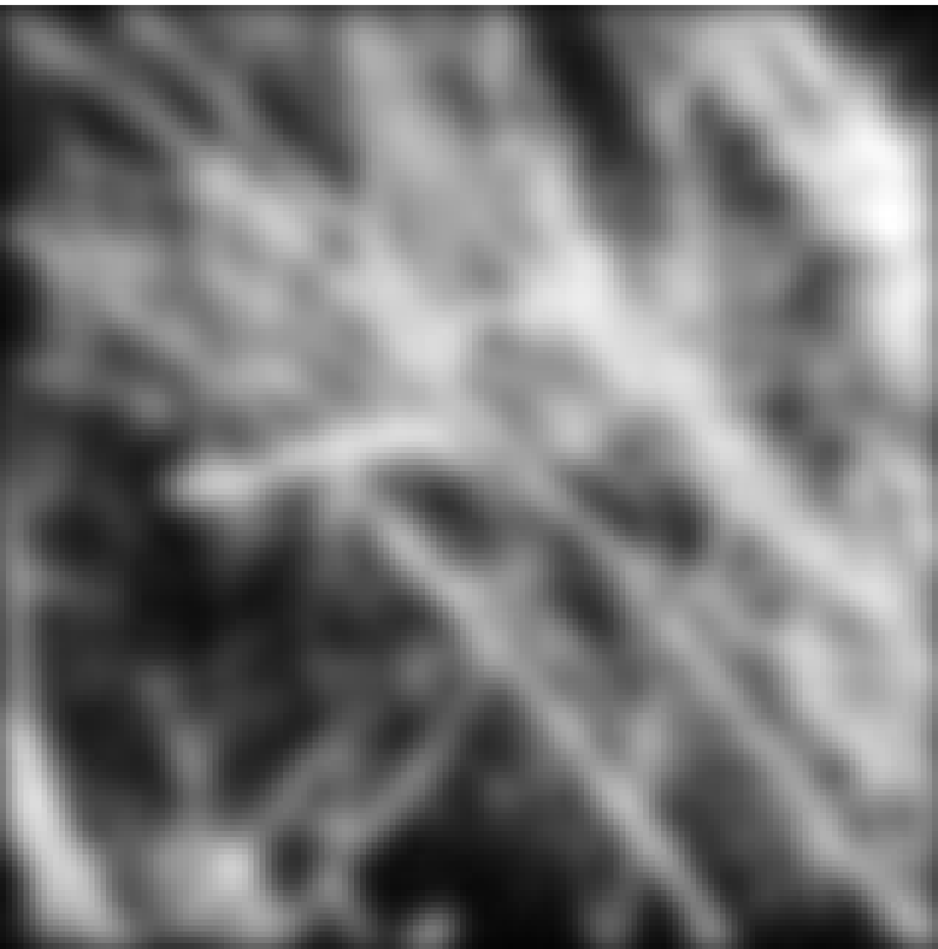


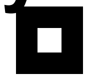
Filtering in frequency domain

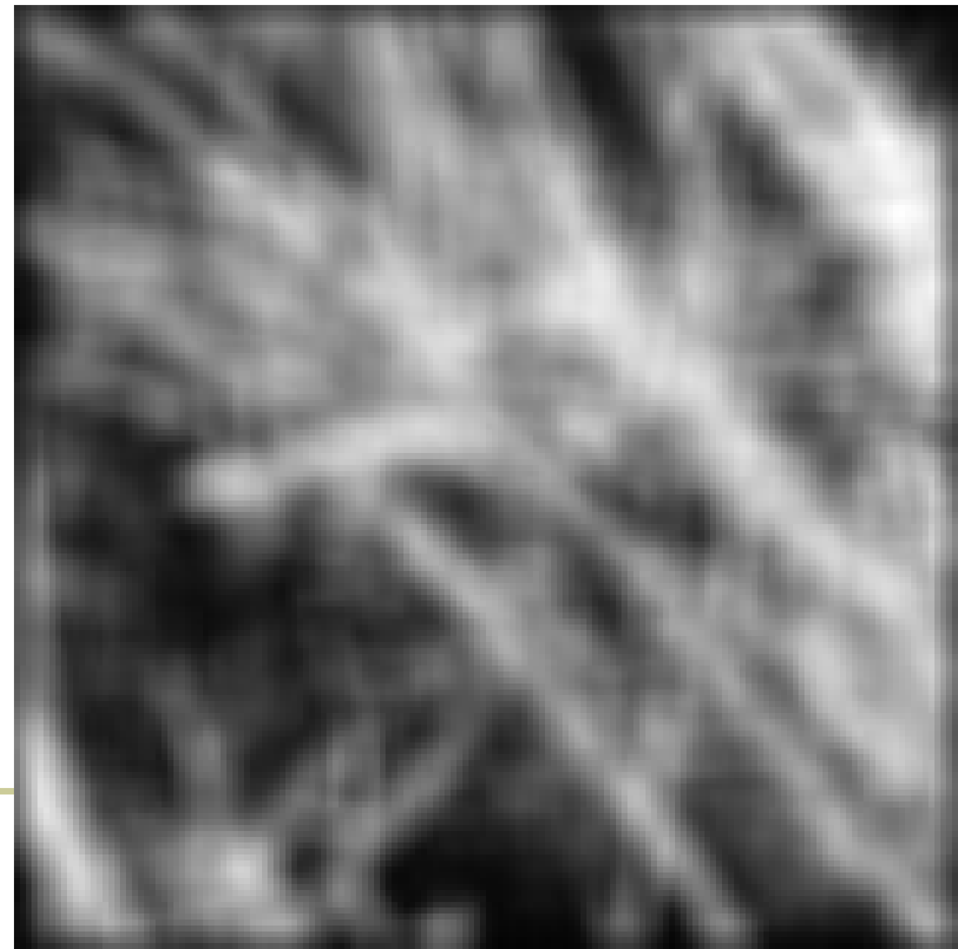


Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

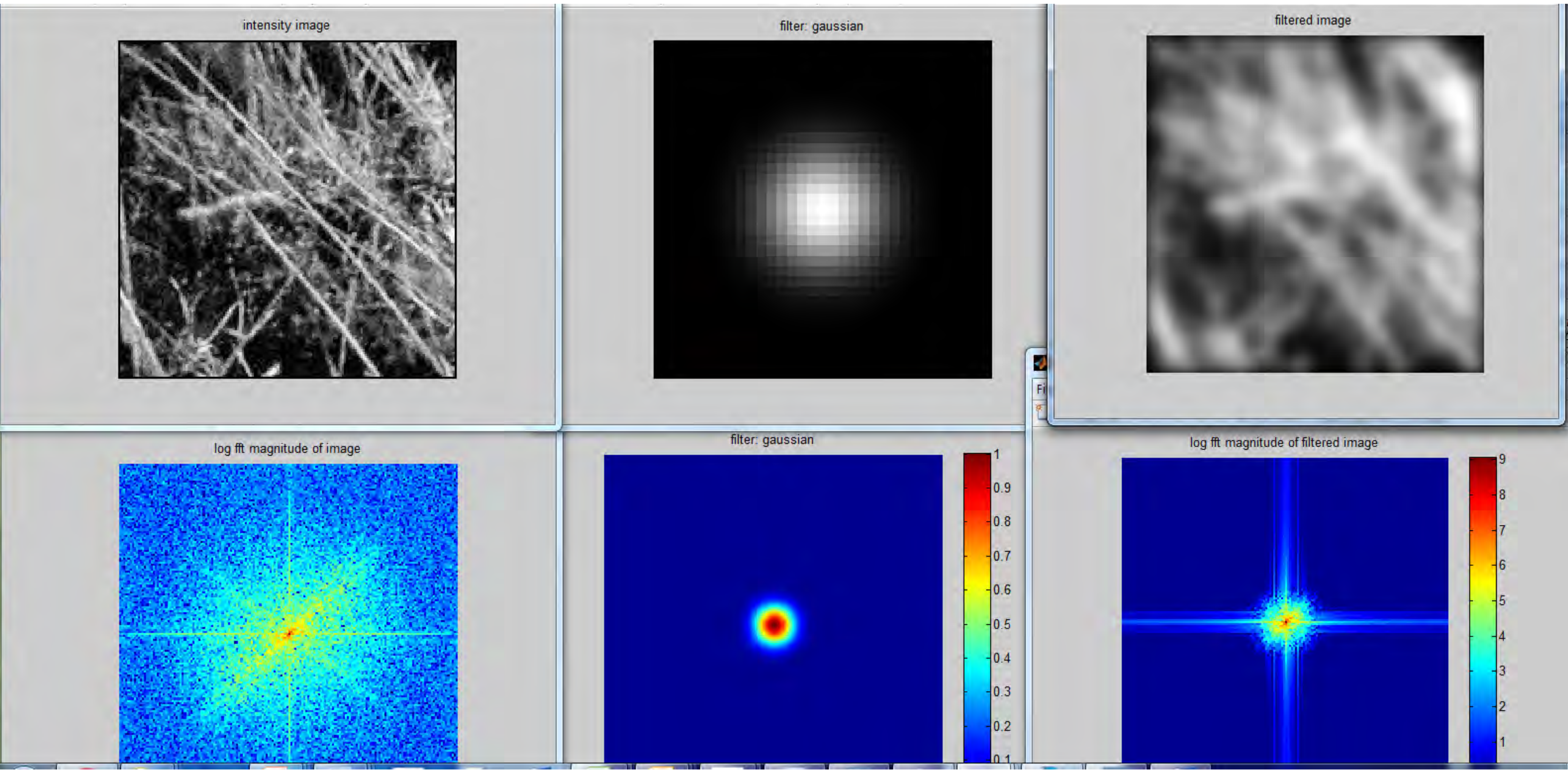
Gaussian 



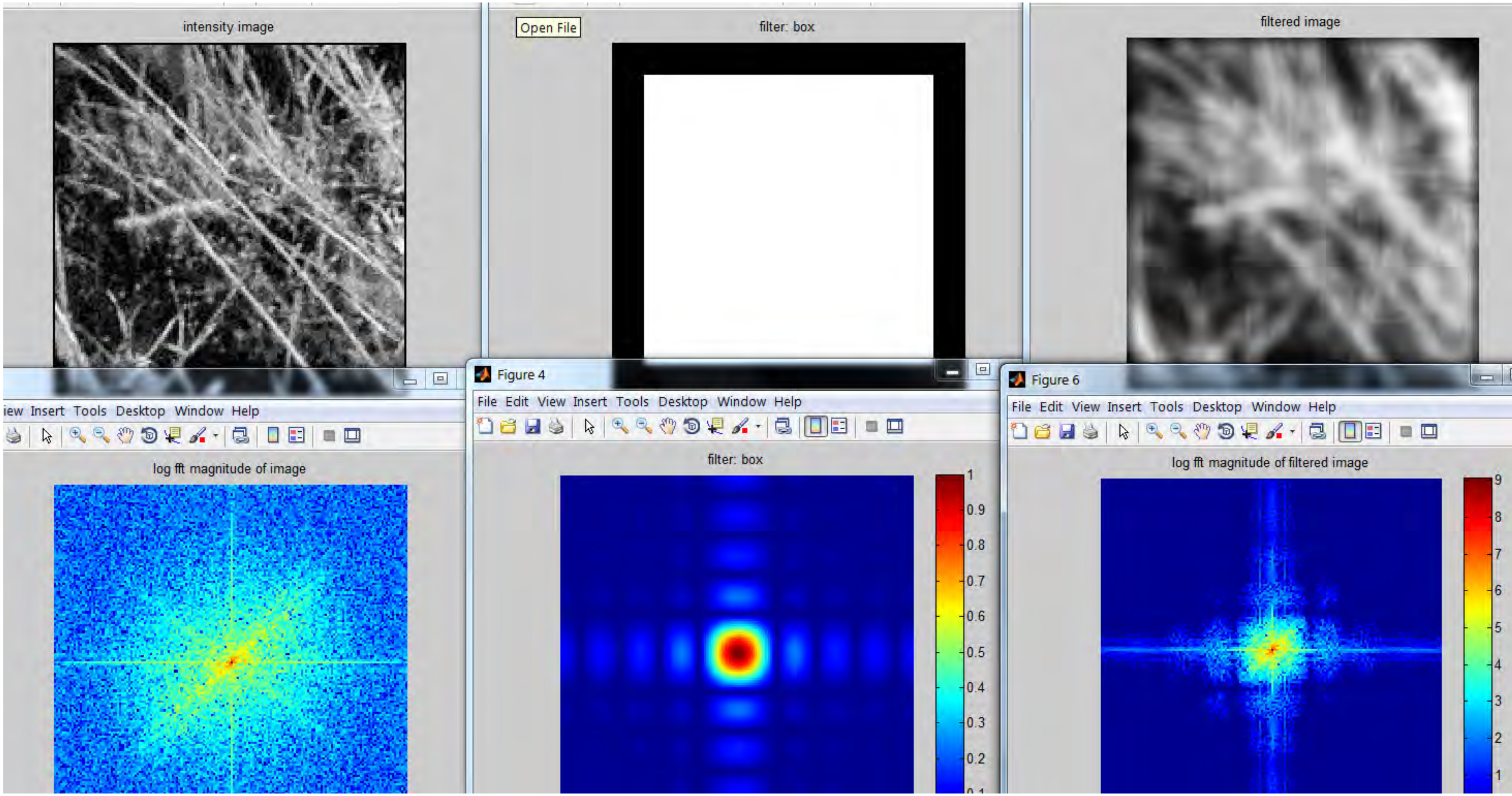
Box filter 



Gaussian Filter



Box Filter


intensity image
Open File
filter: box
filtered image
iew Insert Tools Desktop Window Help
log fft magnitude of image
Figure 4
File Edit View Insert Tools Desktop Window Help
filter: box
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of filtered image
9
8
7
6
5
4
3
2
1

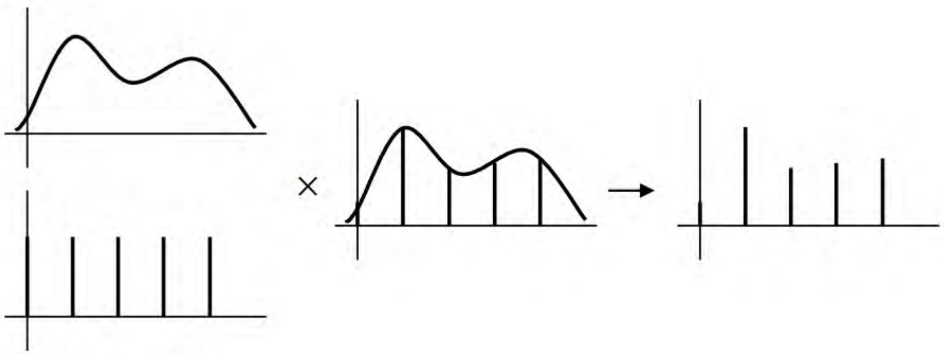
Sampling

Why does a lower resolution image still make sense to us? What do we lose?

Image: http://www.flickr.com/photos/igorms/136916757/

Fourier Interpretation: Sampling

- Sampling in the spatial domain is like multiplying with a spike function.

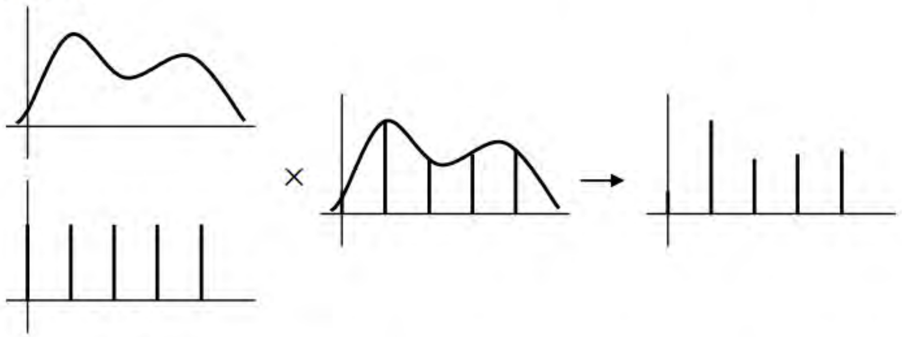


- Sampling in the frequency domain is like...

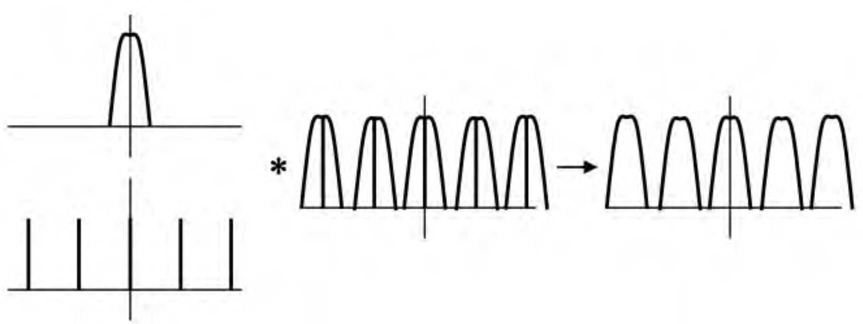
?

Fourier Interpretation: Sampling

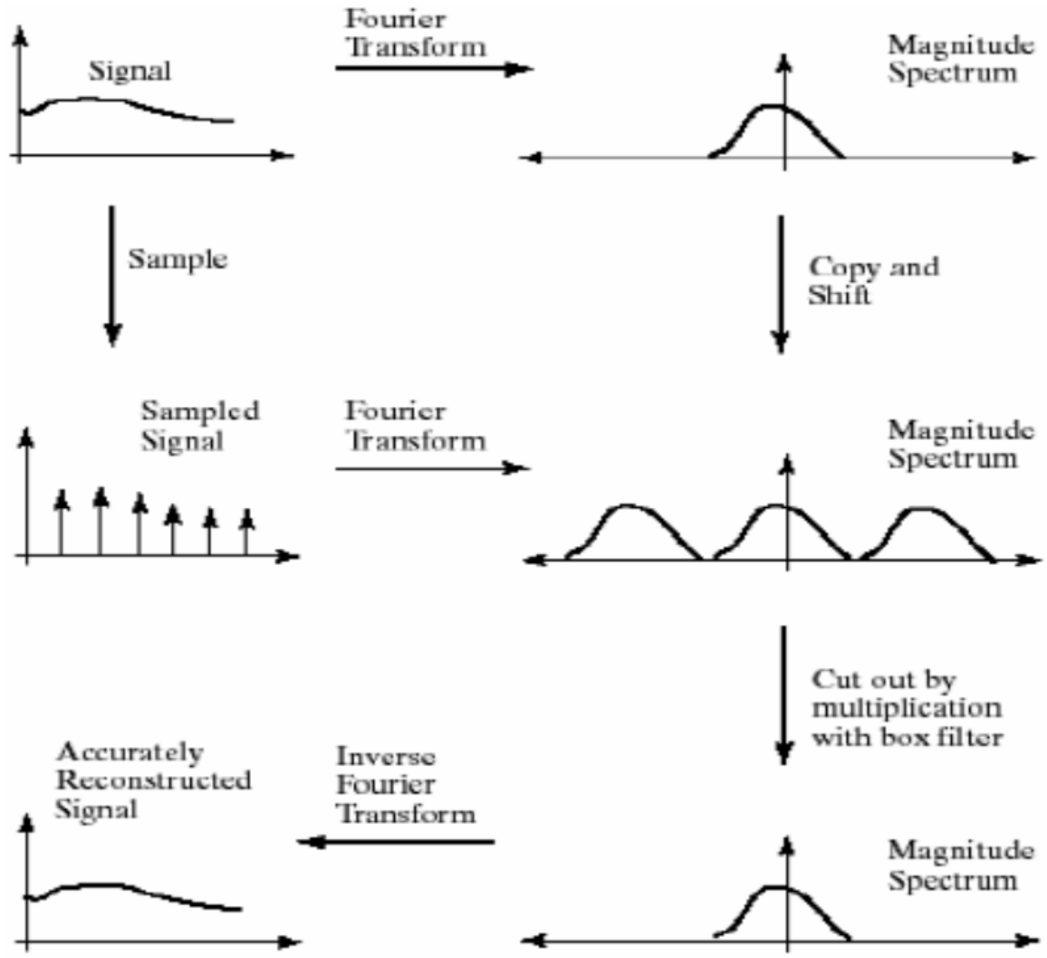
- Sampling in the spatial domain is like multiplying with a spike function.



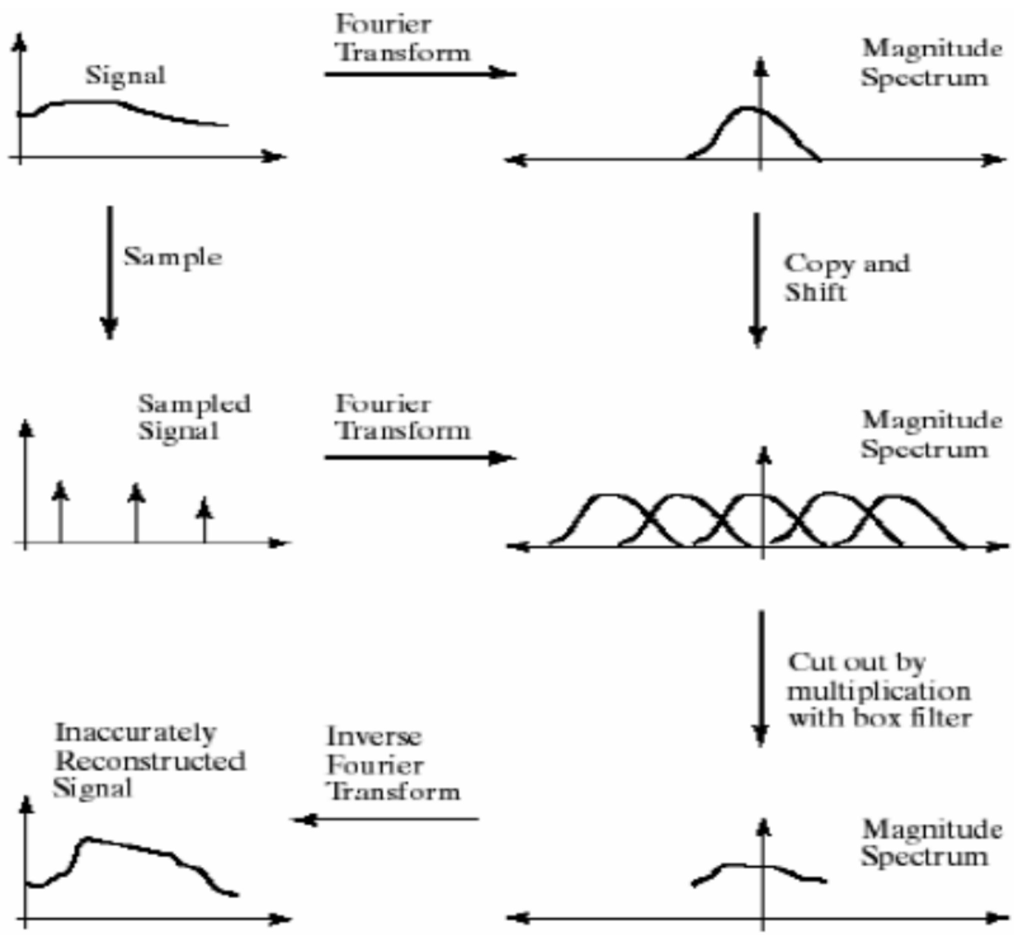
- Sampling in the frequency domain is like convolving with a spike function.



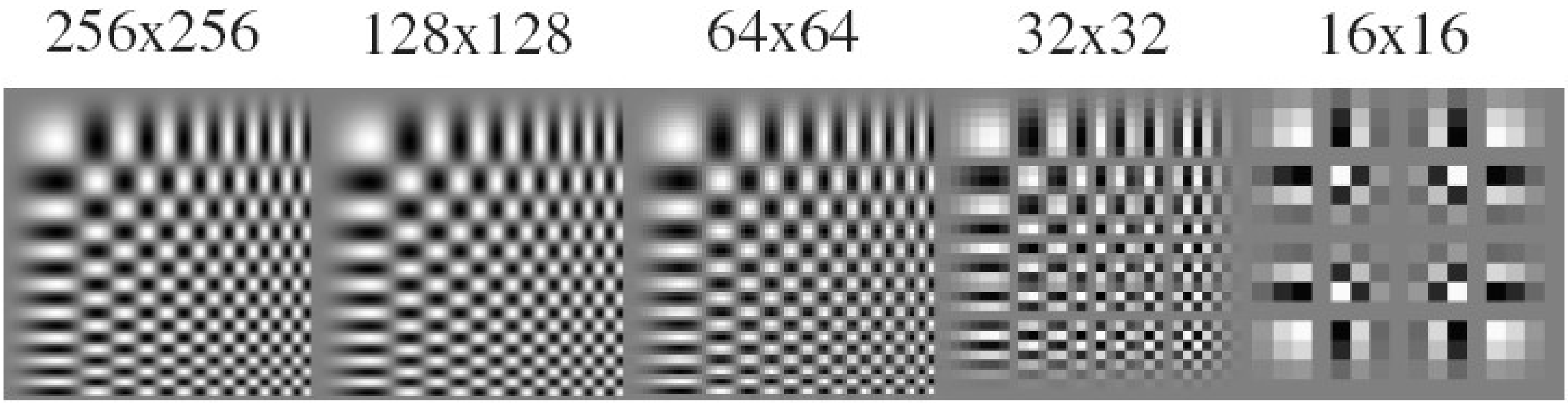
Fourier Interpretation: Sampling



Fourier Interpretation: Sampling



Sampling and aliasing



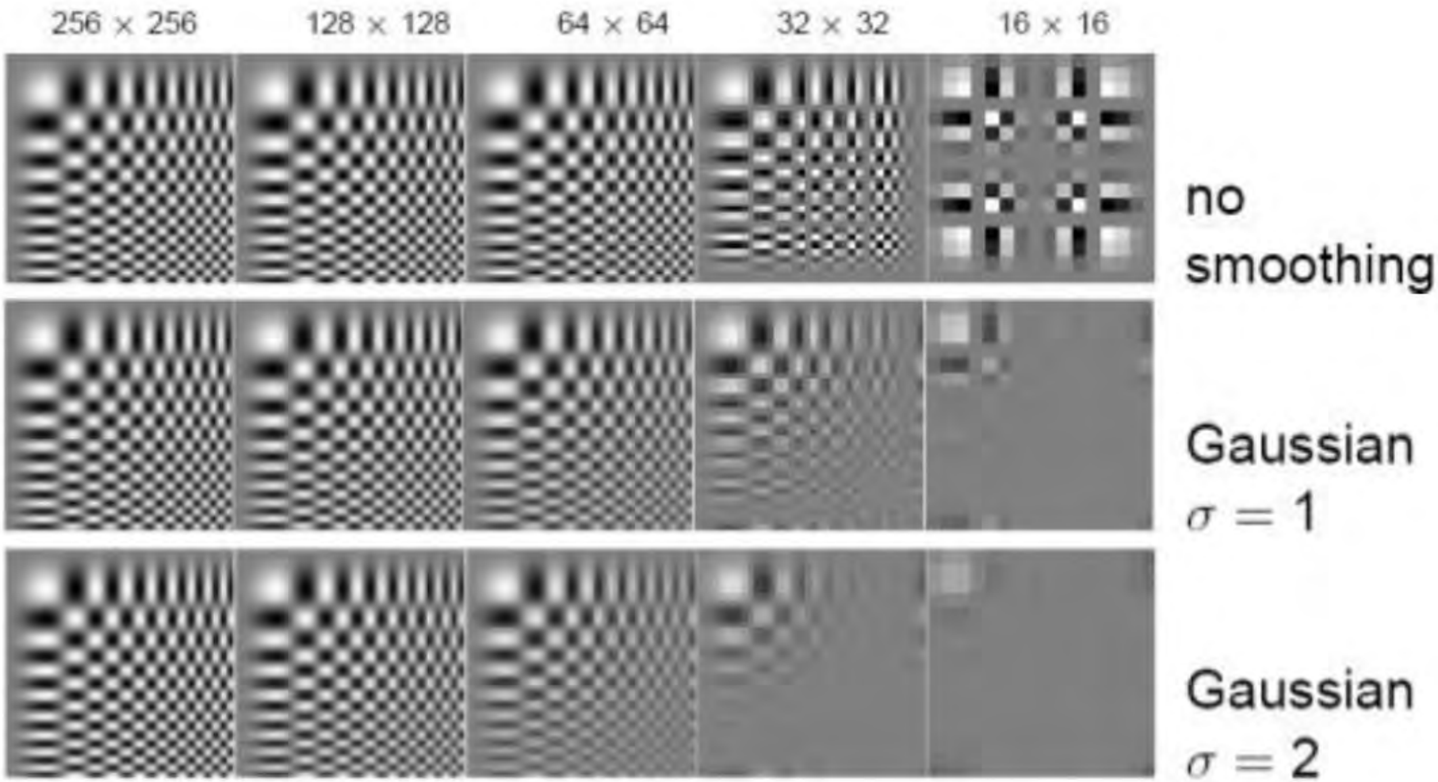
Anti-aliasing

Solutions:

- Sample more often

- Get rid of all frequencies that are greater than half the new sampling frequency
  - Will lose information
  - But it's better than aliasing
  - Apply a smoothing filter

Anti-aliasing



Note: We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling.

Three views of filtering

- Image filters in spatial domain
  - Filter is a mathematical operation on values of each patch
  - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression
- **Templates and Image Pyramids**
  - Filtering is a way to match a template to the image
  - Detection, coarse-to-fine registration

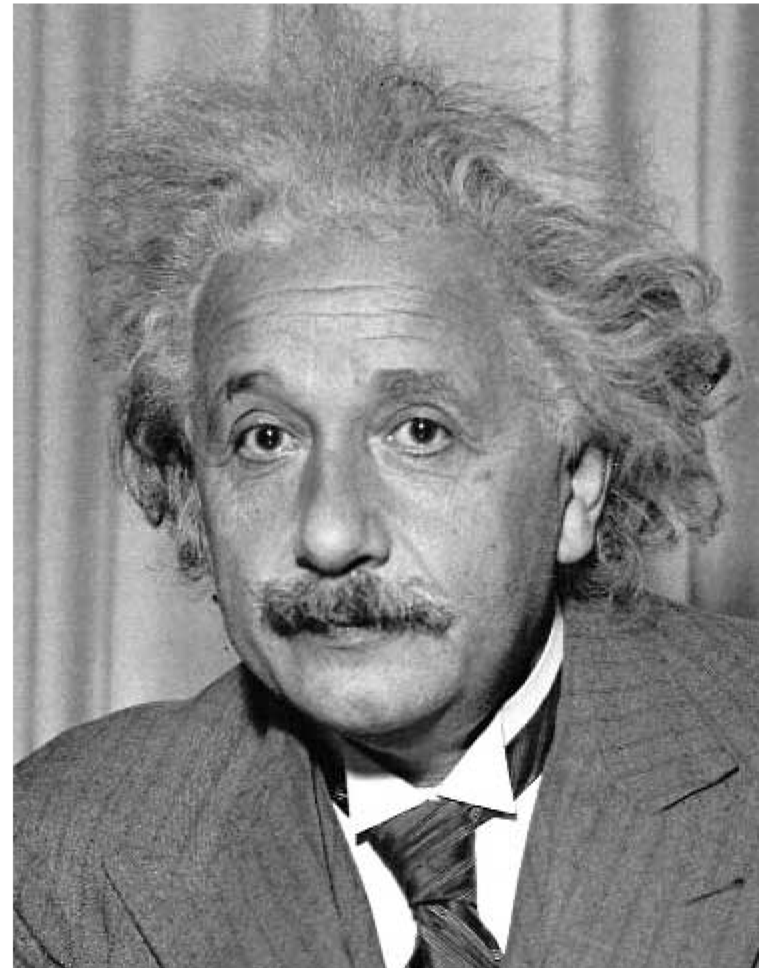
Today’s class

- **Template matching**
- Gaussian Pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian Pyramids
  - Application for image blending
  - Hybrid images
- Steerable pyramids：
  - 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

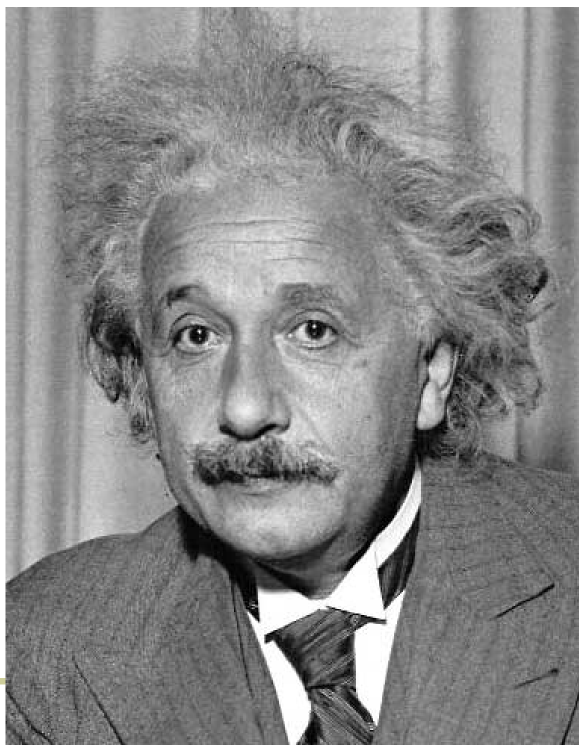


Matching with filters

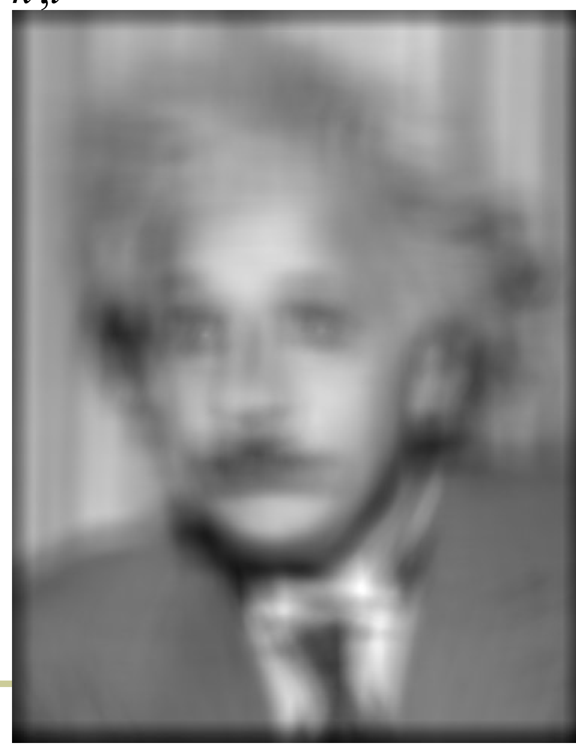
- Goal: find  in image
- Method 0: filter the image with eye patch

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

f = image
g = filter



Input



Filtered Image

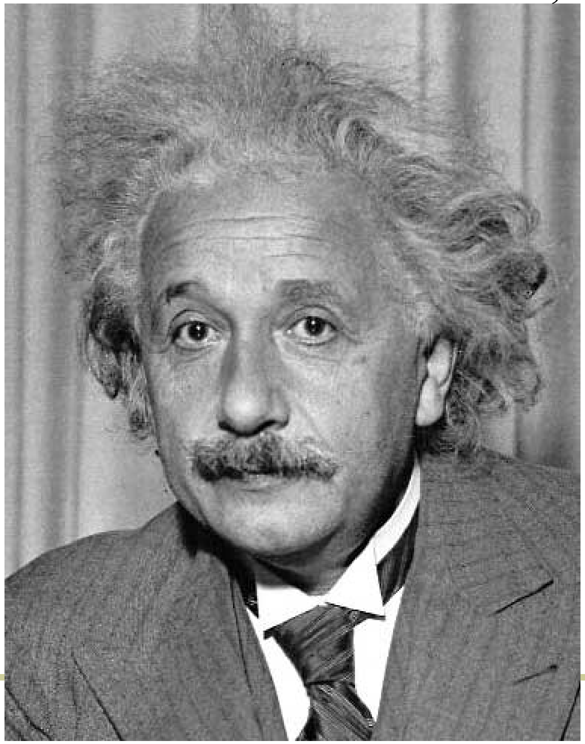
What went wrong?

Matching with filters

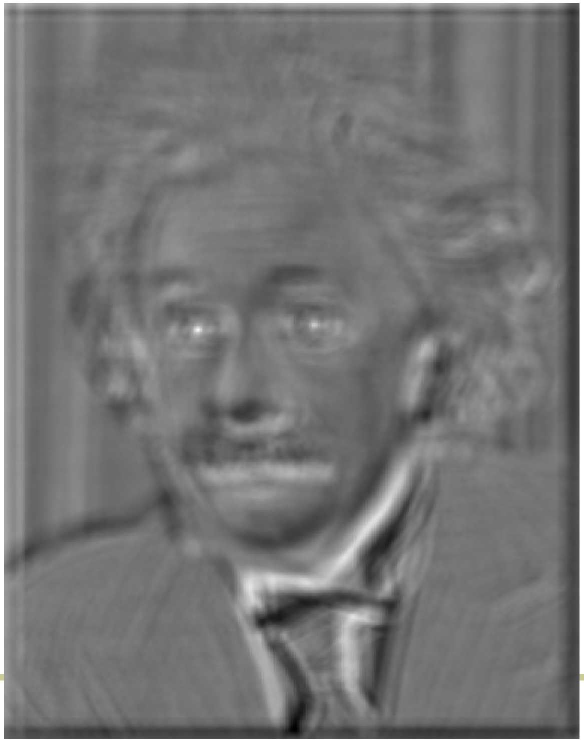
- Goal: find  in image
- Method 1: filter the image with zero-mean eye

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

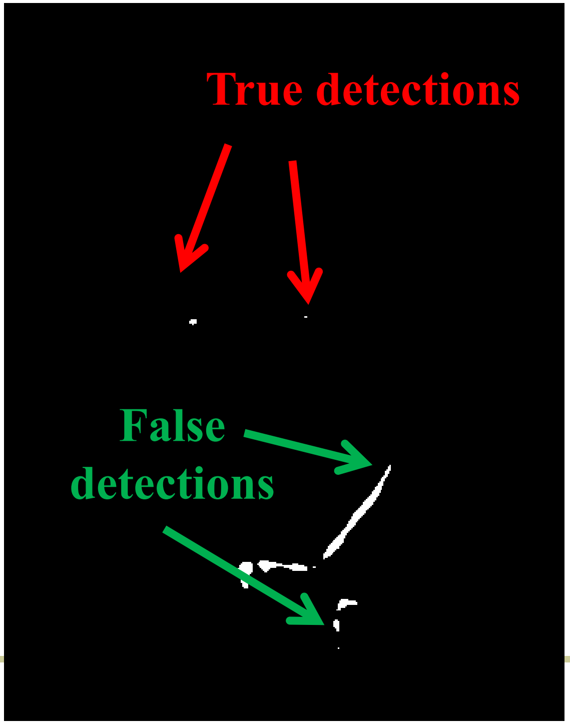
mean of template g



Input

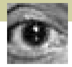


Filtered Image (scaled)

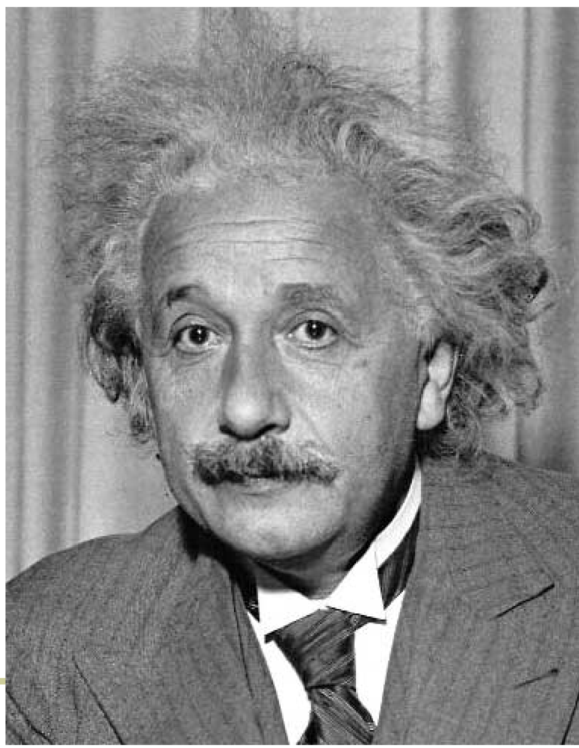


Thresholded Image

Matching with filters

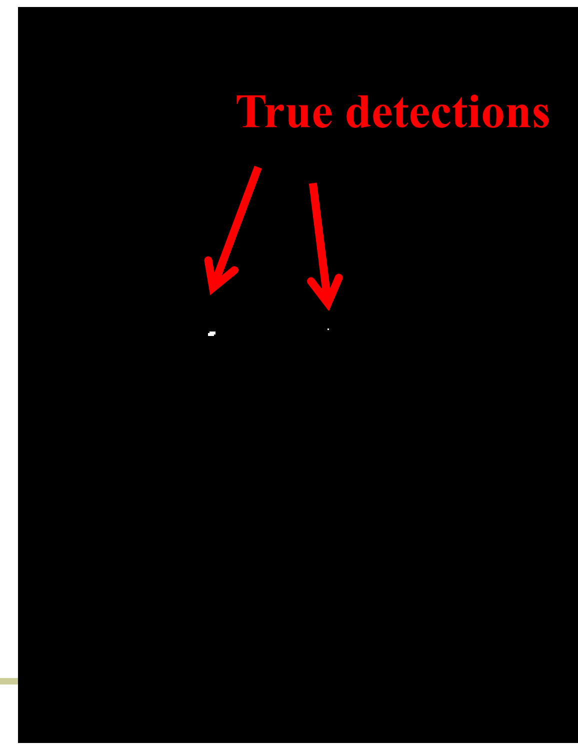
- Goal: find  in image
- Method 2: SSD

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



Input

1- sqrt(SSD)



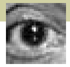
Thresholded Image

Matching with filters

Can SSD be implemented with linear filters?

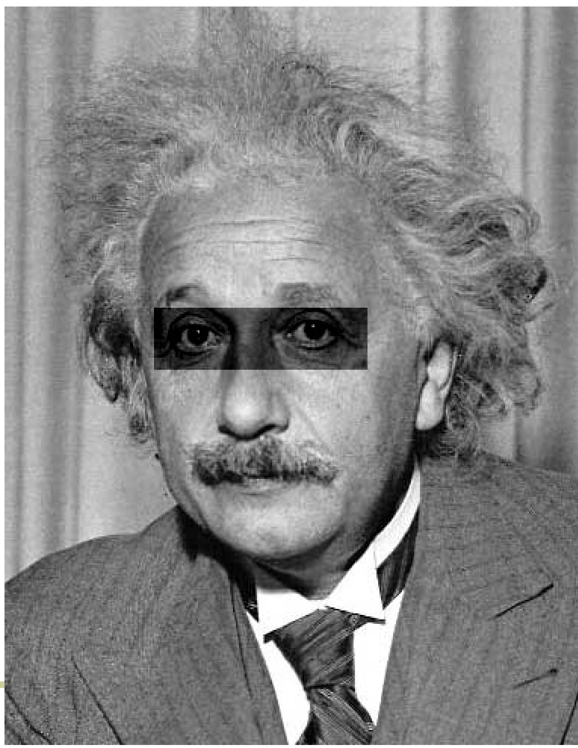
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

Matching with filters

- Goal: find  in image
- Method 2: SSD

What's the potential downside of SSD?

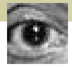
$$h[m,n]=\sum_{k,l}(g[k,l]-f[m+k,n+l])^2$$



Input

1- sqrt(SSD)

Matching with filters

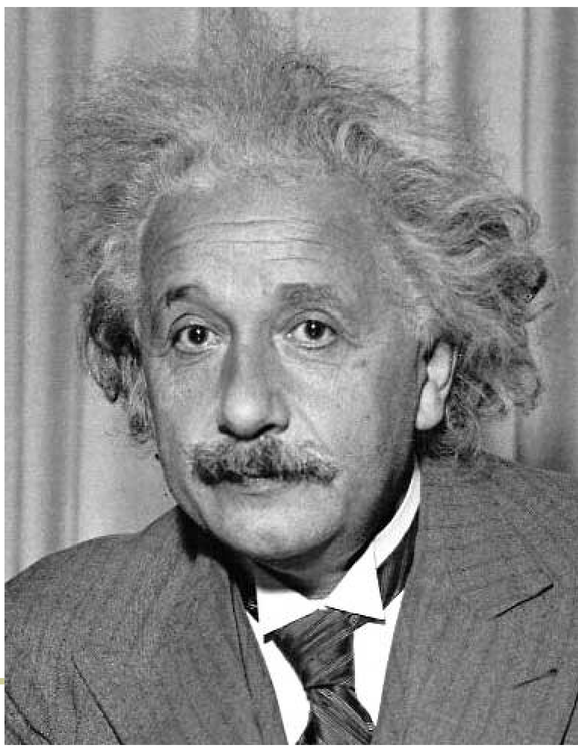
- Goal: find  in image
- Method 3: Normalized cross-correlation

$$h[m,n]=\frac{\sum_{k,l}(g[k,l]-\overset{\substack{\text{mean template}\\ \downarrow}}{\bar{g}})(f[m+k,n+l]-\overset{\substack{\text{mean image patch}\\ \downarrow}}{\bar{f}_{m,n}})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

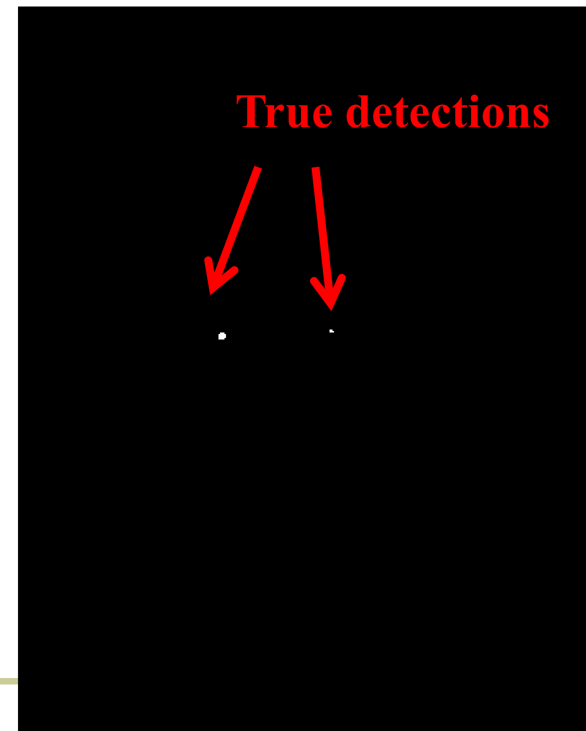
Matching with filters

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



Input

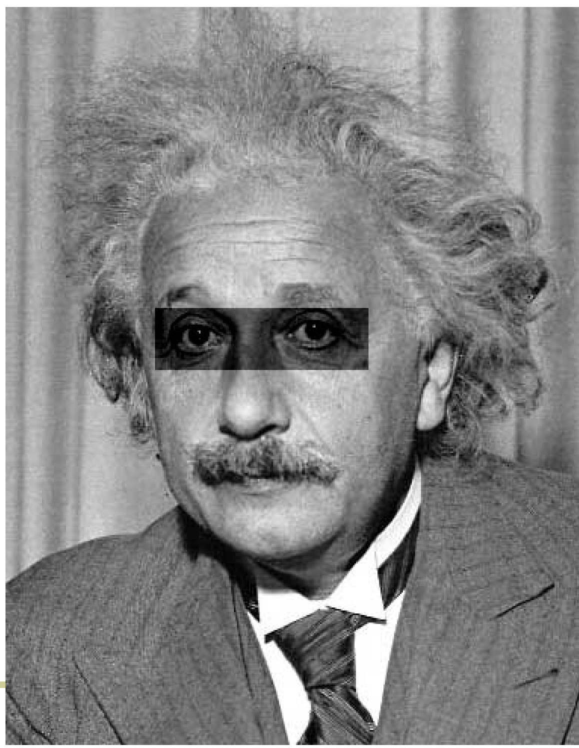
Normalized X-Correlation



Thresholded Image

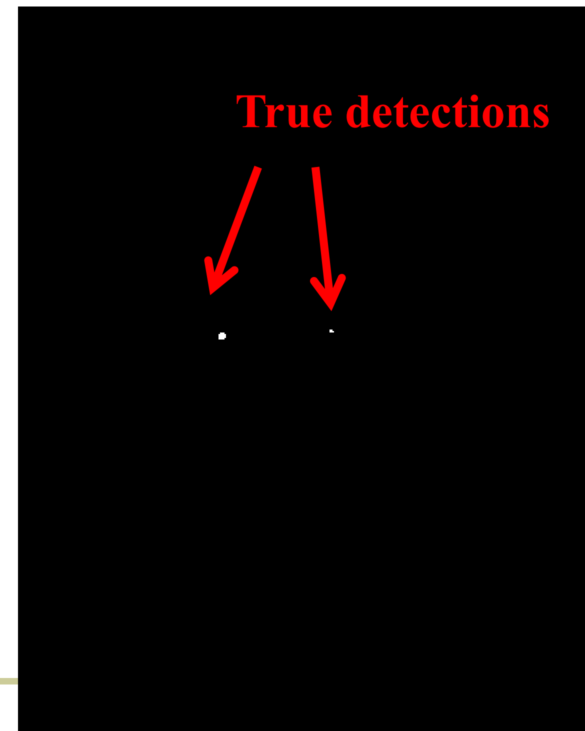
Matching with filters

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



Input

Normalized X-Correlation



Thresholded Image

Q: What is the best method to use?

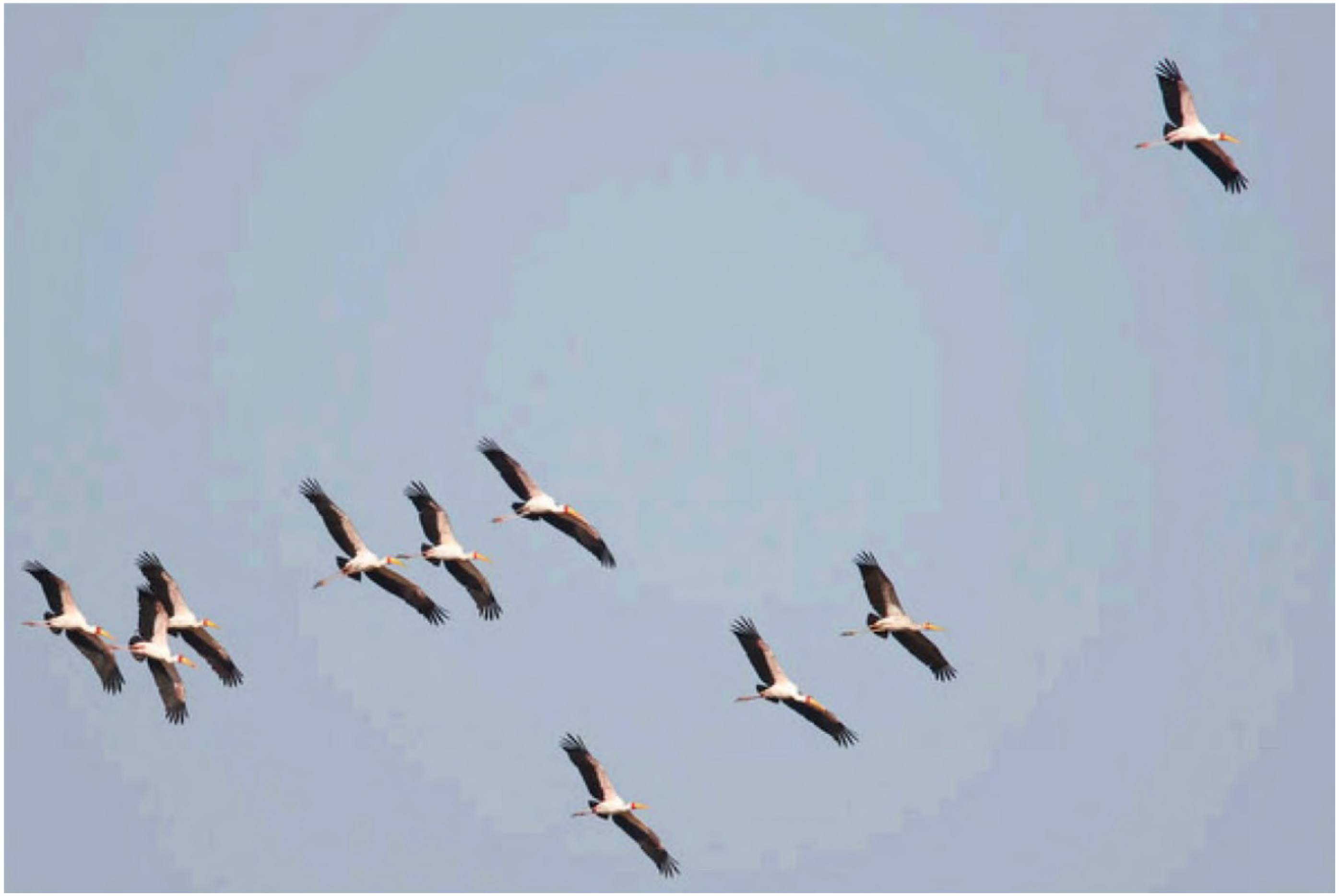
A: Depends

- 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

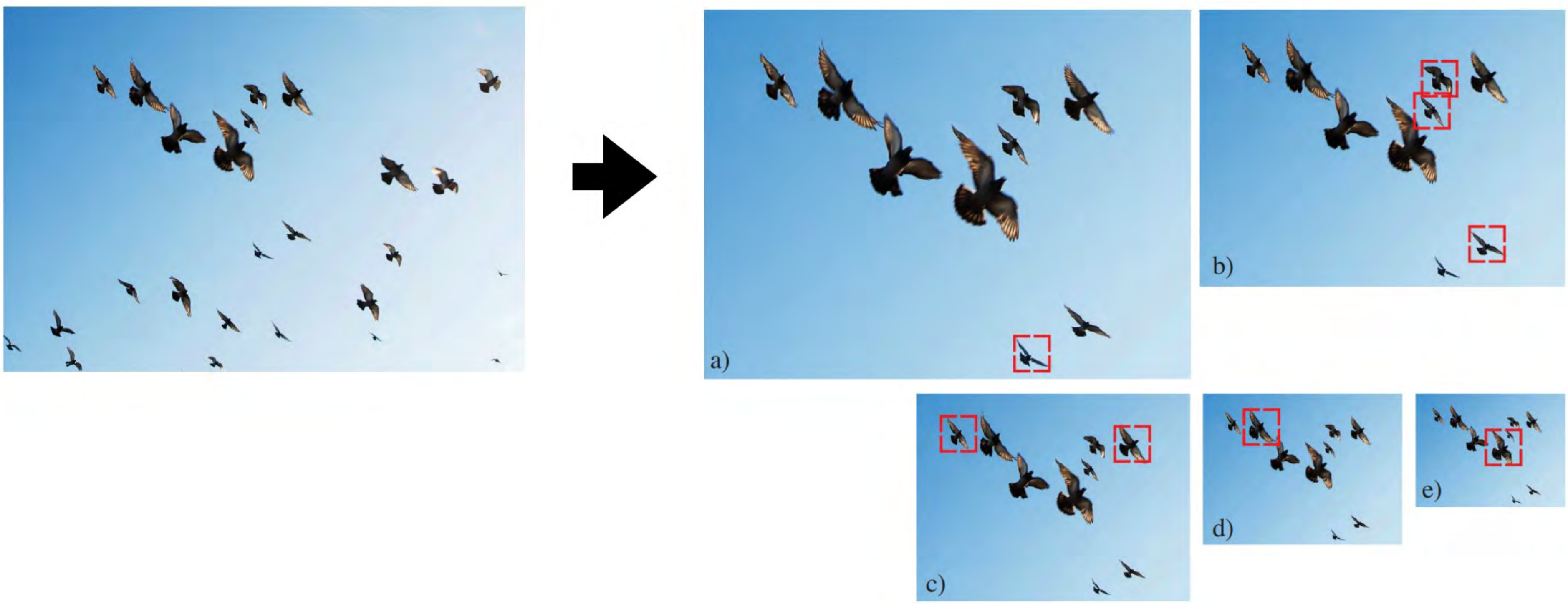
- Template matching
- **Gaussian Pyramids**
  - Application for recognition
  - Pyramids representation in deep learning
- **Laplacian Pyramids**
  - Application for image blending
  - Hybrid images
- Steerable pyramids：
  - filter banks and texture analysis

Translation invariance

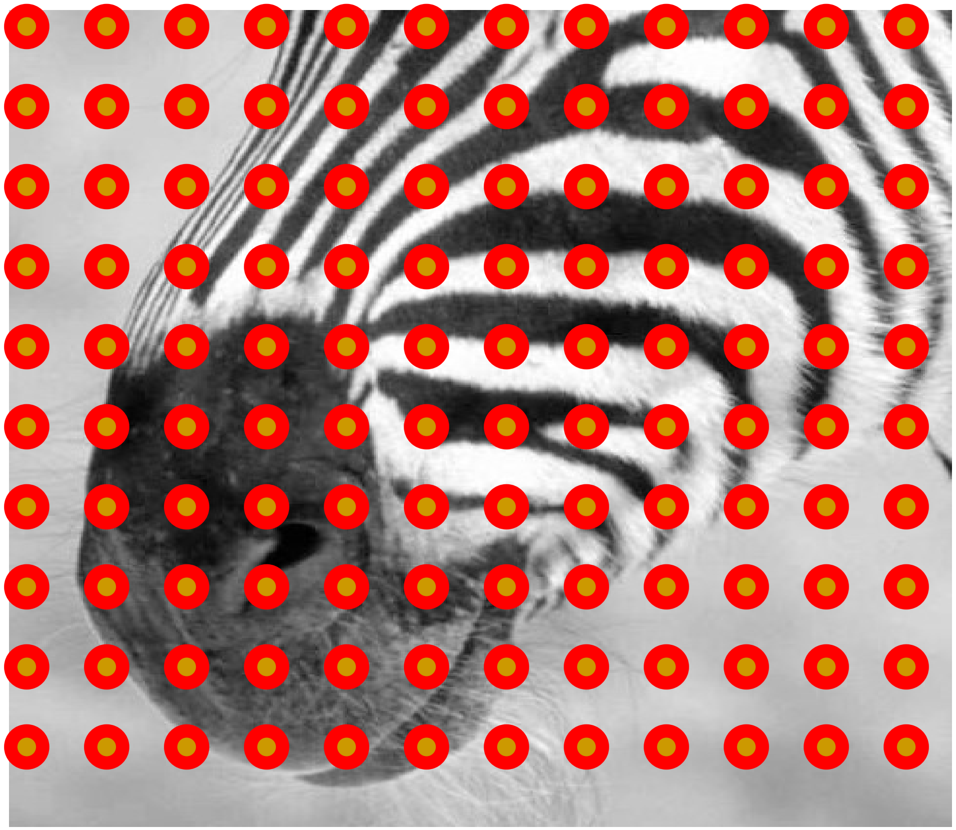


We need translation and scale invariance

Image pyramids



Subsampling by a factor of 2



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

Recall: sampling

Image $\xrightarrow{\text{Gaussian Filter}}$ Low-Pass Filtered Image $\xrightarrow{\text{Sample}}$ Low-Res Image

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

Image pyramids

- **Gaussian pyramid**
  - Application for recognition
- Laplacian pyramid
  - Application for image blending

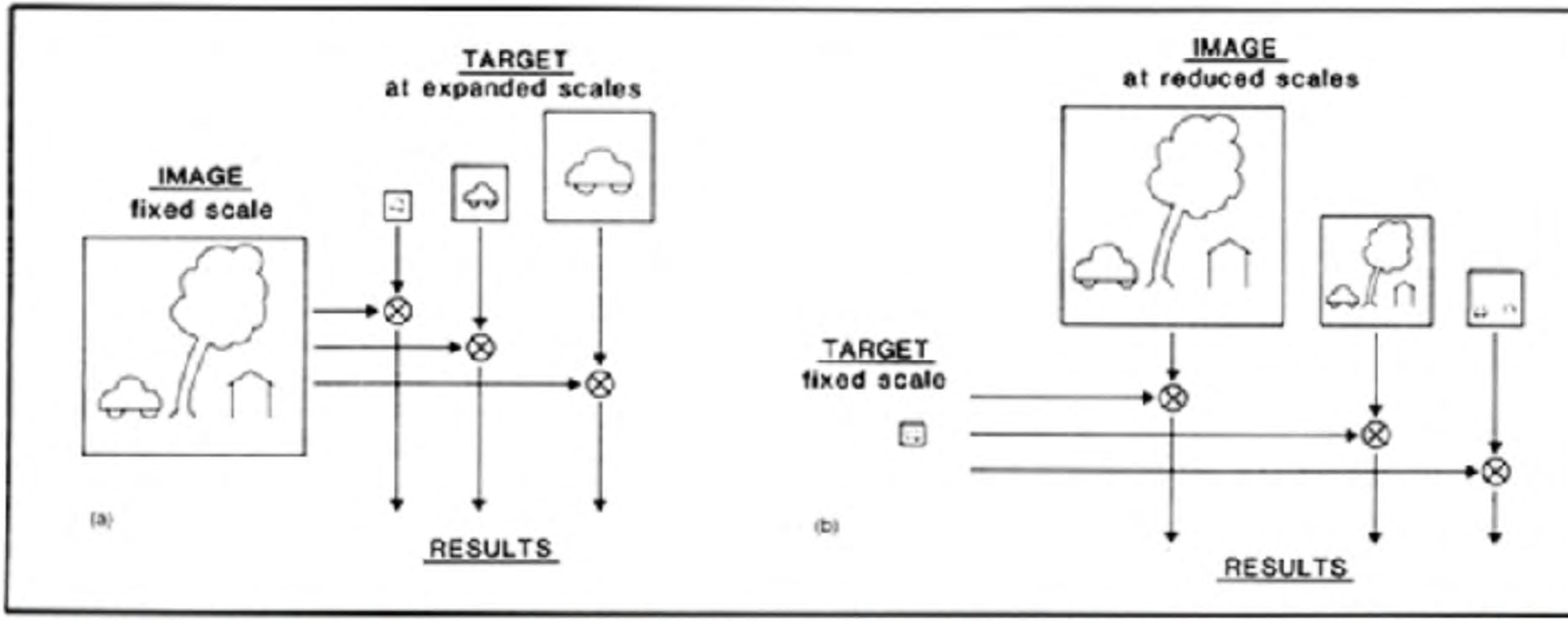
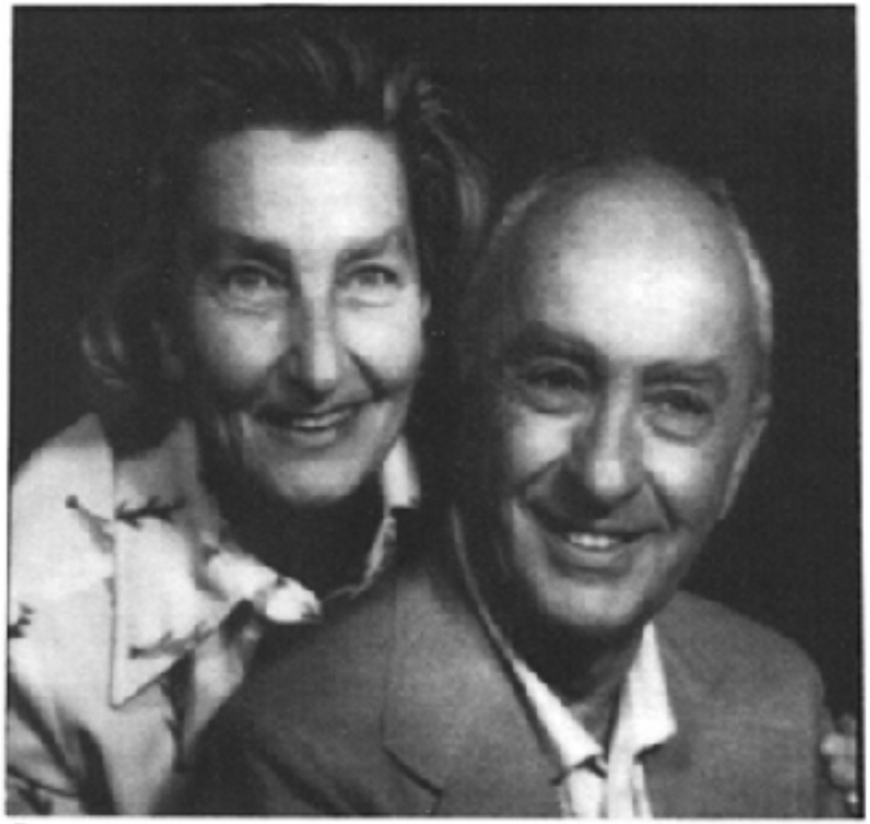


Fig. 1. Two methods of searching for a target pattern over many scales. In the first approach, (a), copies of the target pattern are constructed at several expanded scales, and each is convolved with the original image. In the second approach, (b), a single copy of the target is convolved with copies 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').

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.*

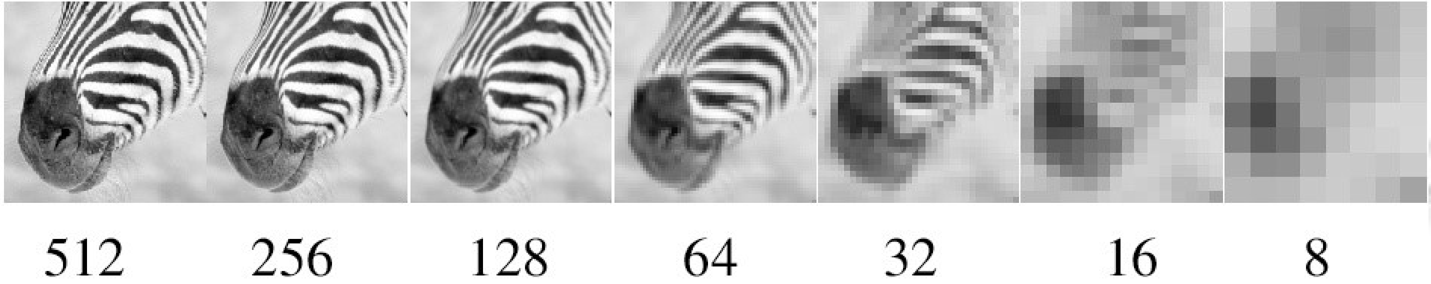


G_0

G_1 G_2 G_3 G_4

$G_{0,0}$ $G_{1,1}$ $G_{2,2}$

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

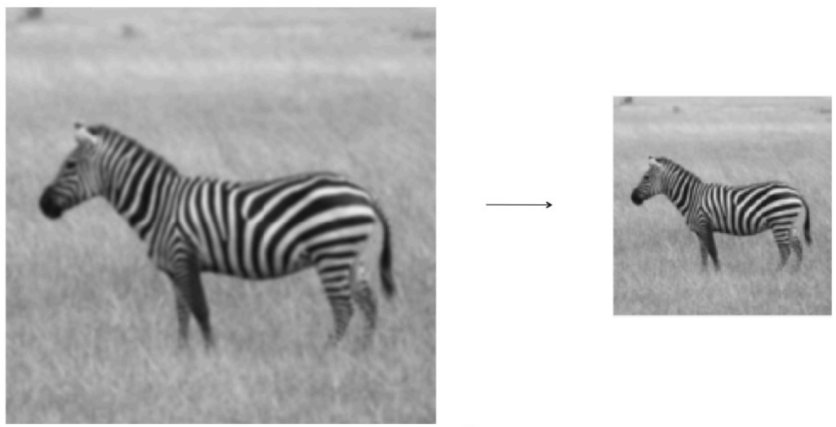


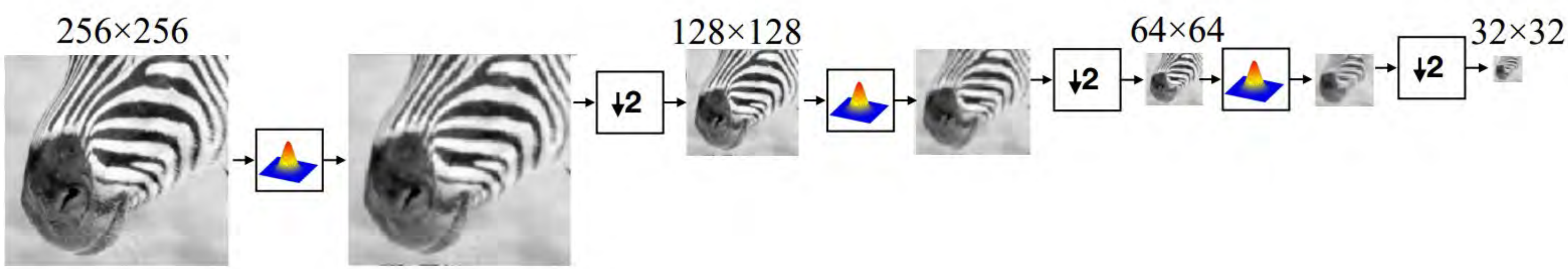
Gaussian Pyramid

Gaussian Pyramid

For each level

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





Downsampling & Upsampling

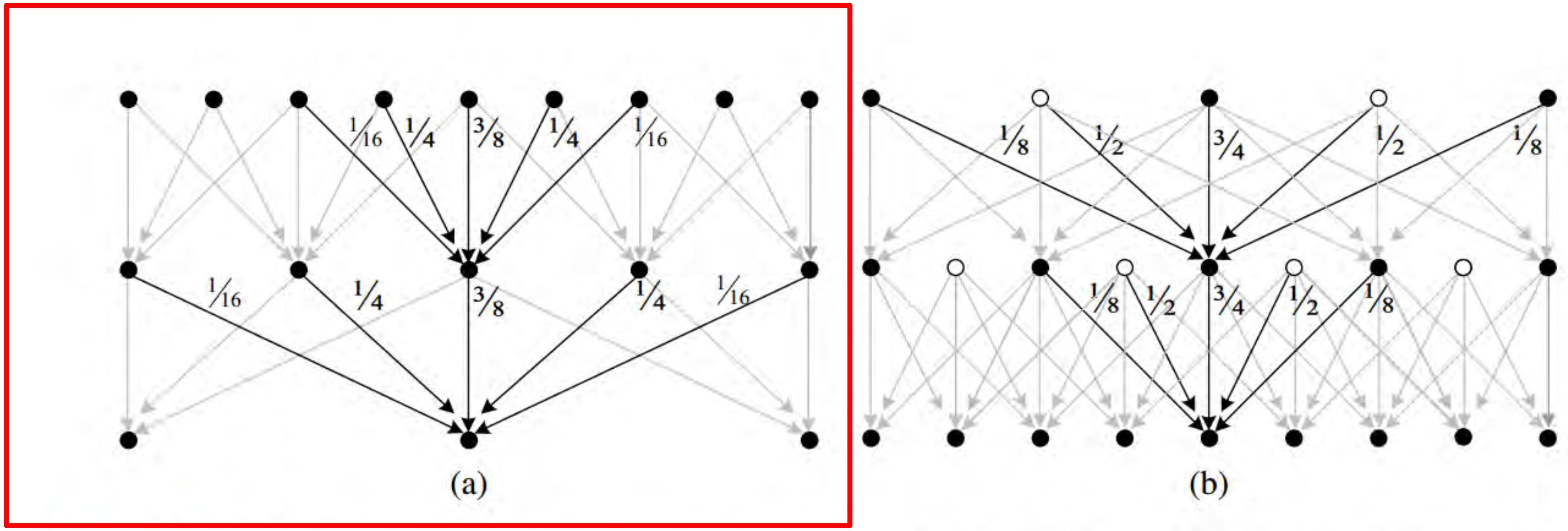
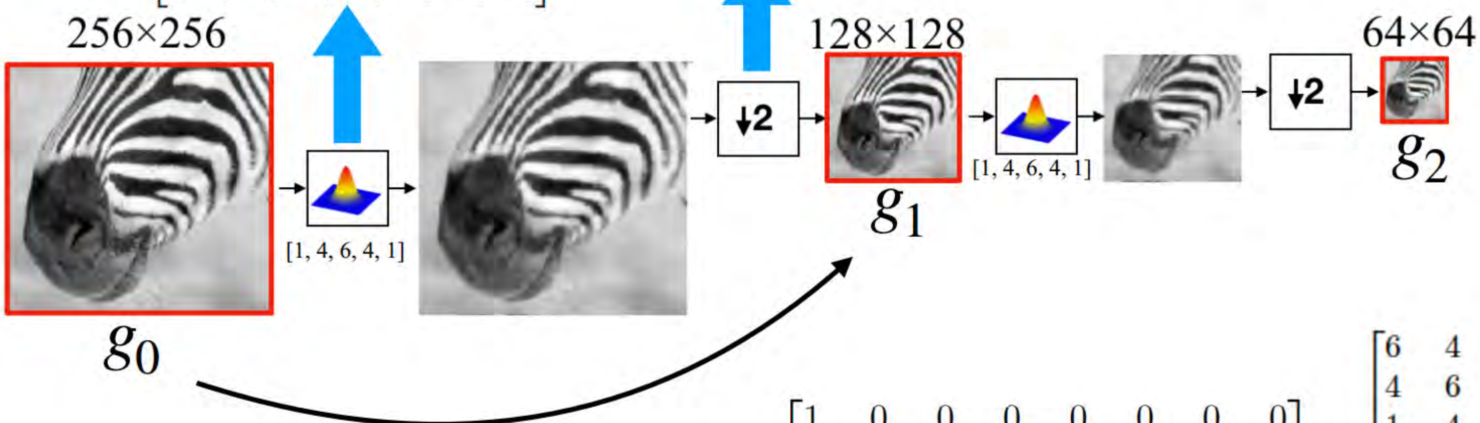


Figure 3.33 The Gaussian pyramid shown as a signal processing diagram: The (a) analysis and (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the ↑ 2 upsampling operation. Notice how the reconstruction filter coefficients are twice the 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

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



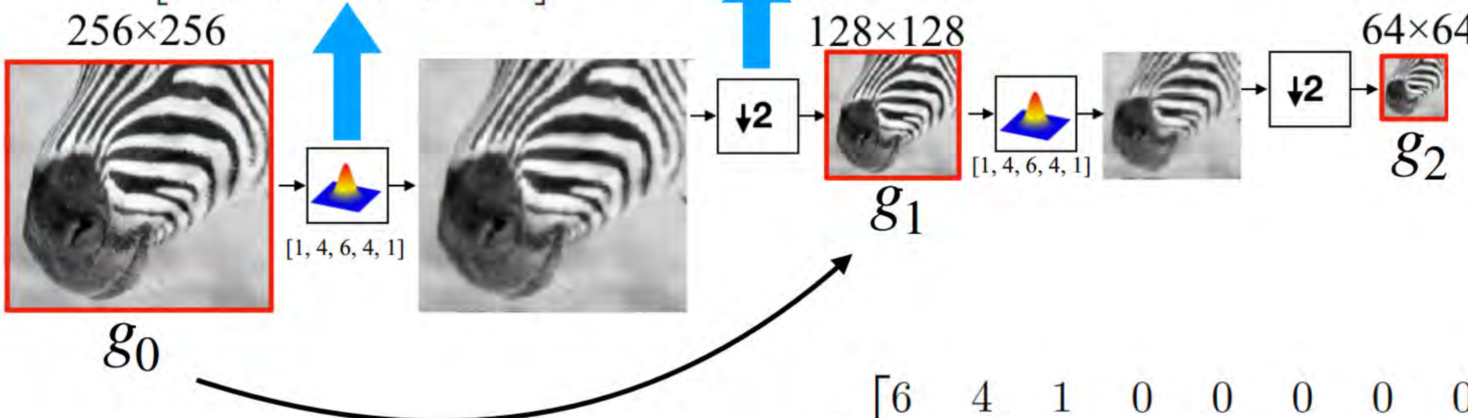
$$g_1 = G_0 g_0$$

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

The Gaussian pyramid

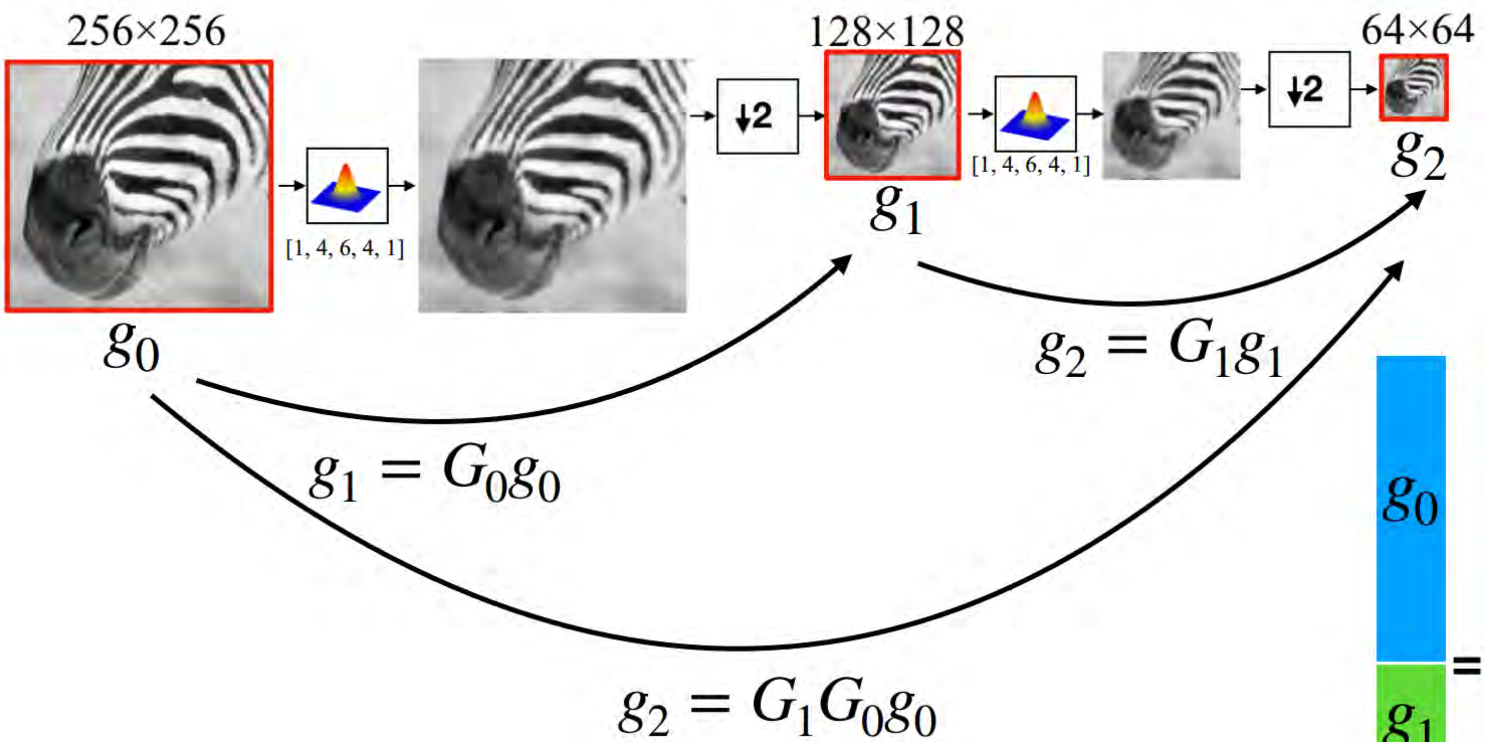
$$\frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

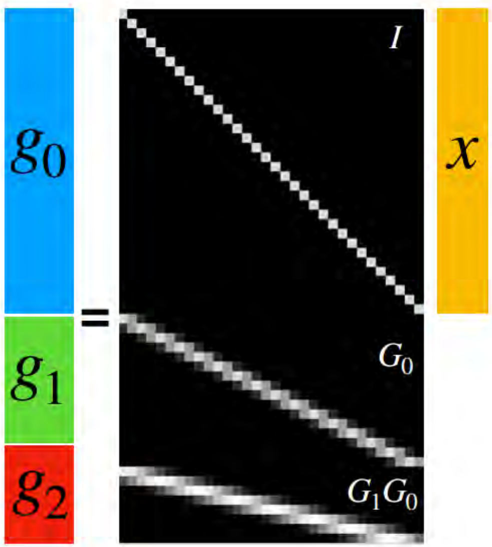
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



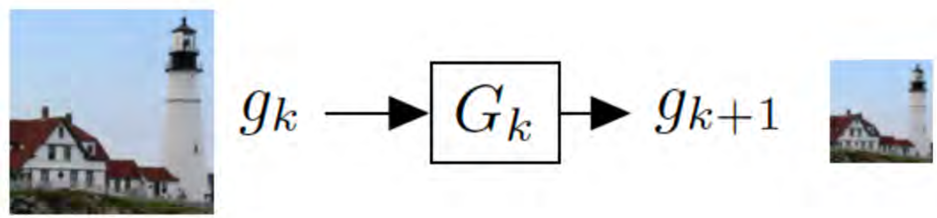
$$g_1 = G_0 g_0 \qquad G_0 = \frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \end{bmatrix}$$

The Gaussian pyramid





Gaussian Pyramid



For each level

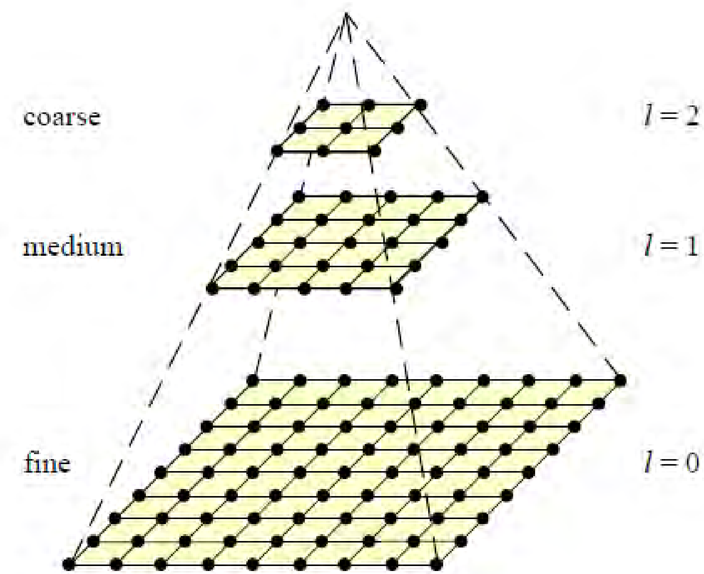
1. Blur input image with a Gaussian filter
2. Downsample image

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

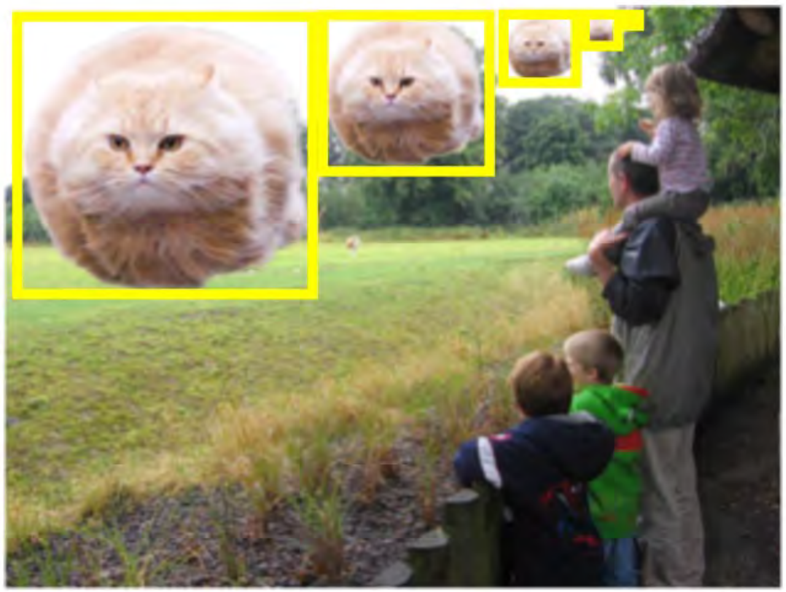
1. Match template at current scale

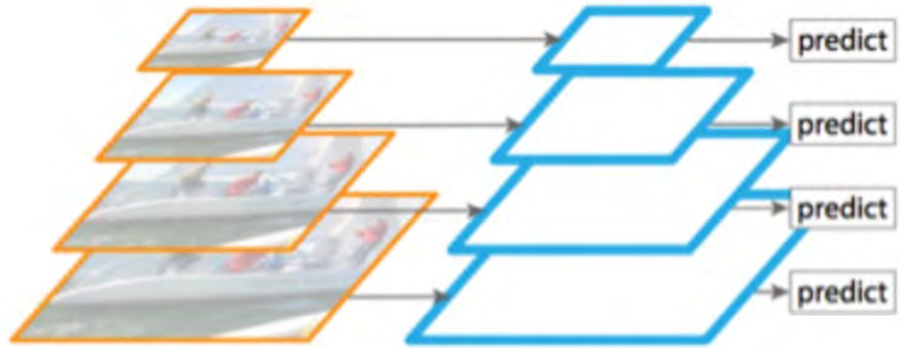
2. Downsample image
   - In practice, scale step of 1.1 to 1.2

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



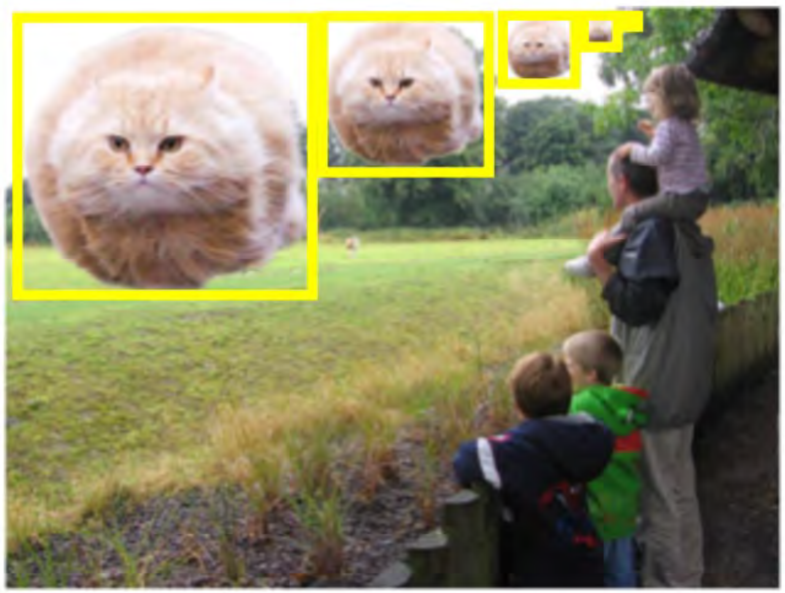


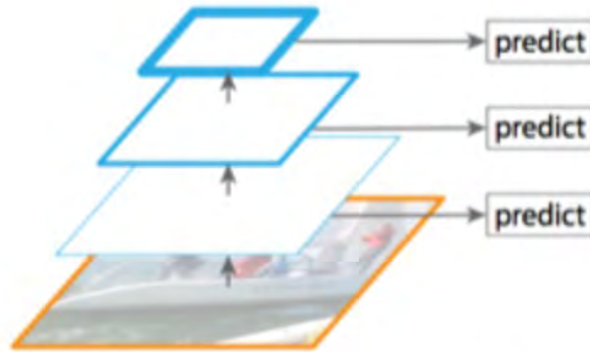
(a) Featurized image pyramid

Standard solution – *slow!*

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

Feature Pyramid Networks



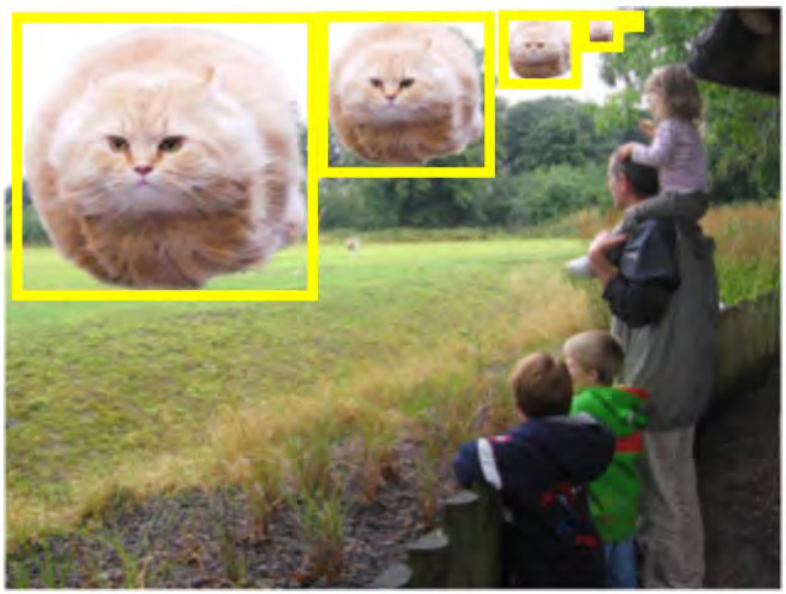


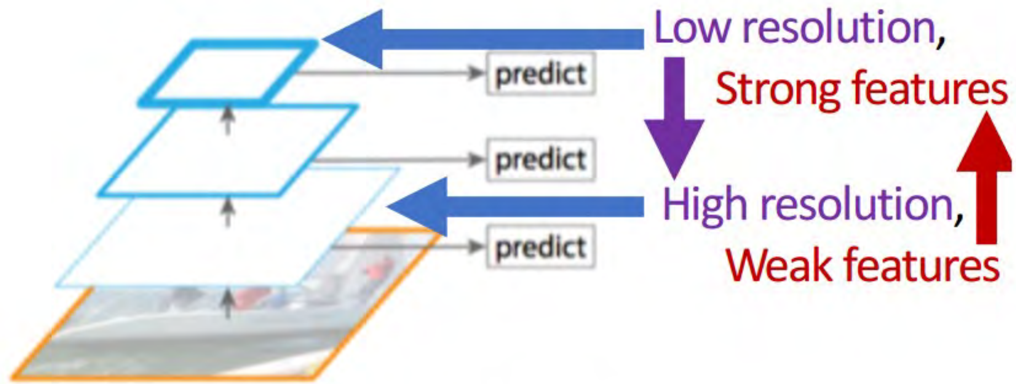
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, ...)

Feature Pyramid Networks



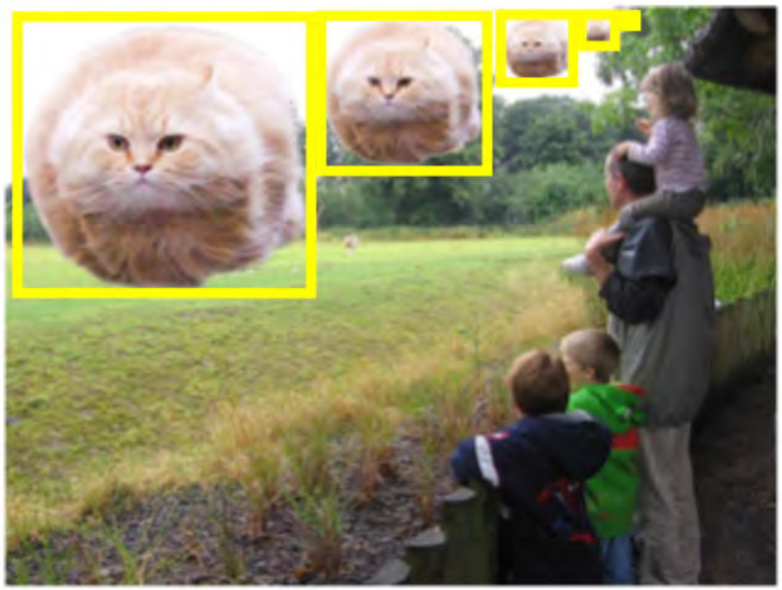


(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Feature Pyramid Networks

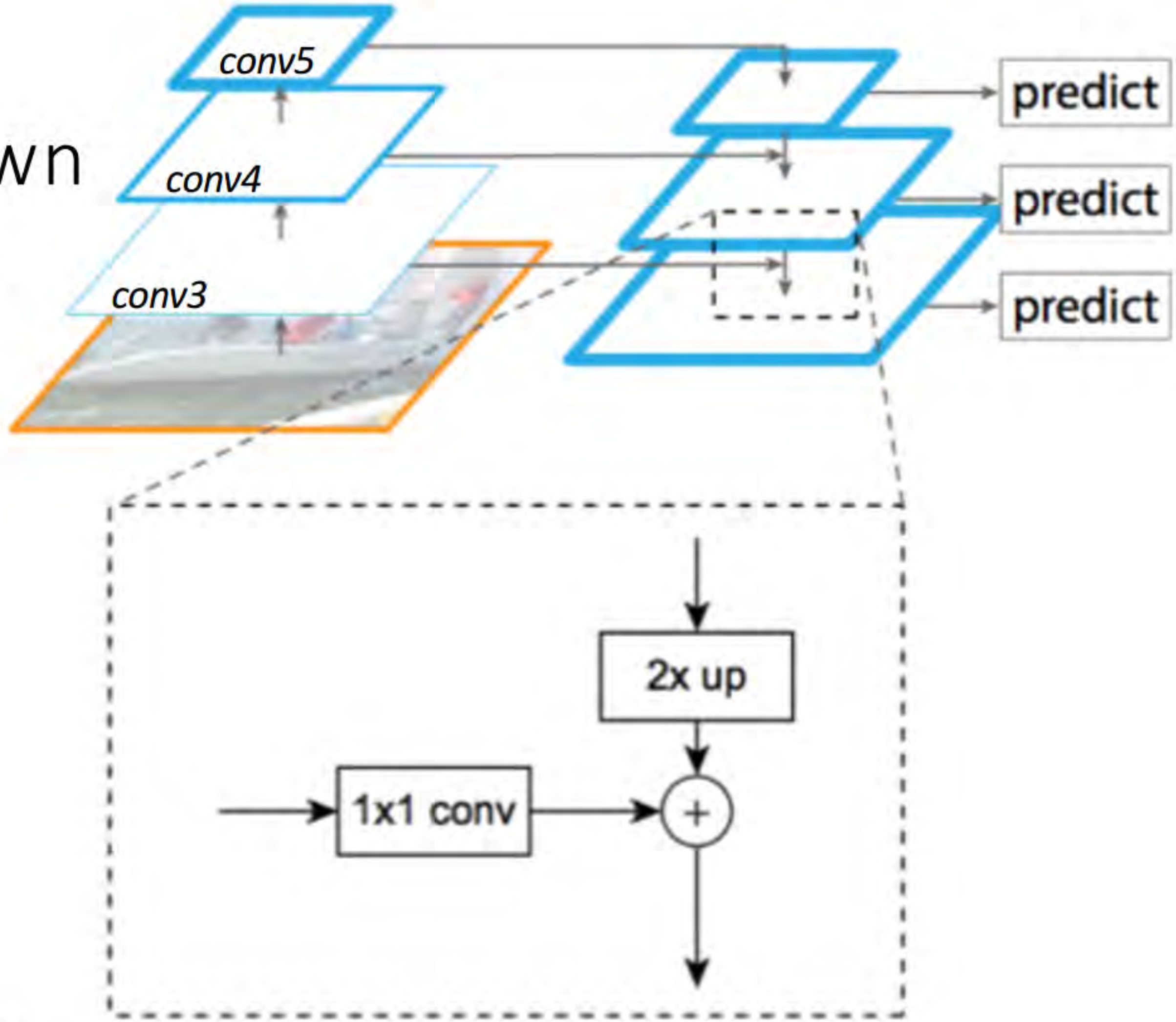


(d) Feature Pyramid Network

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

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

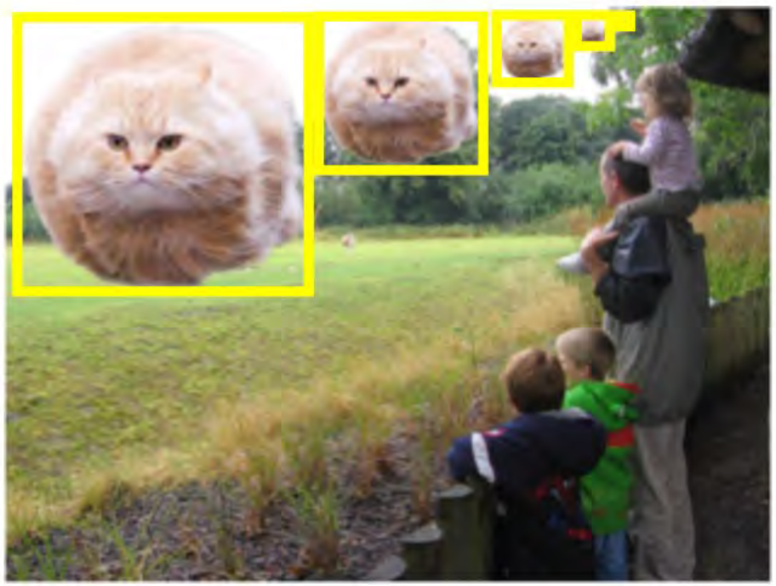
FPN Top-down Refinement Module

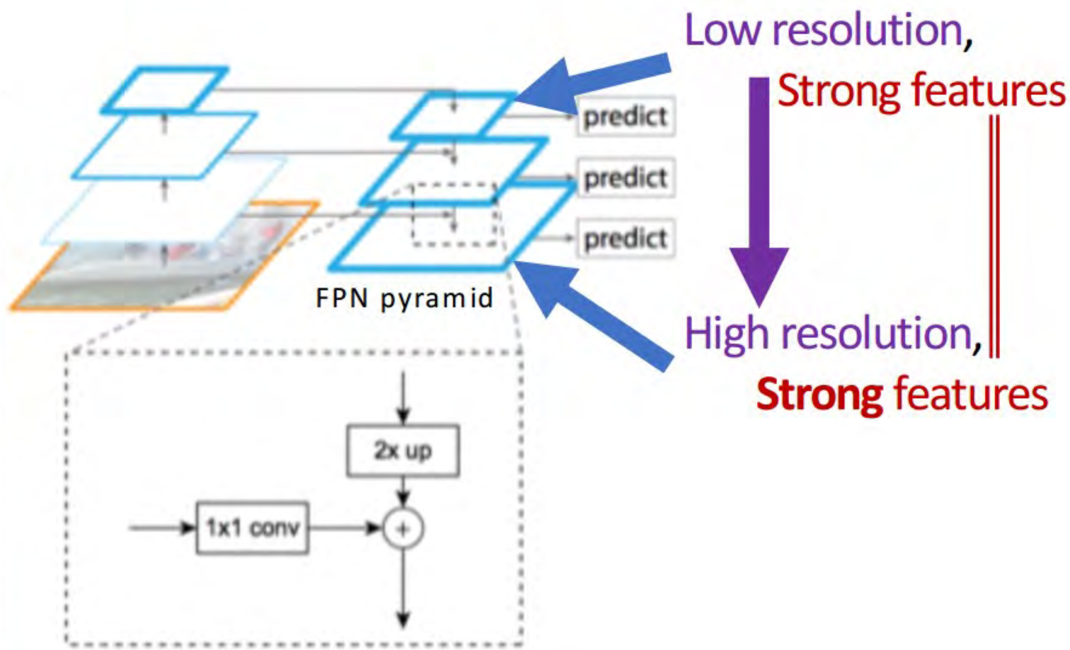


Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

Feature Pyramid Networks

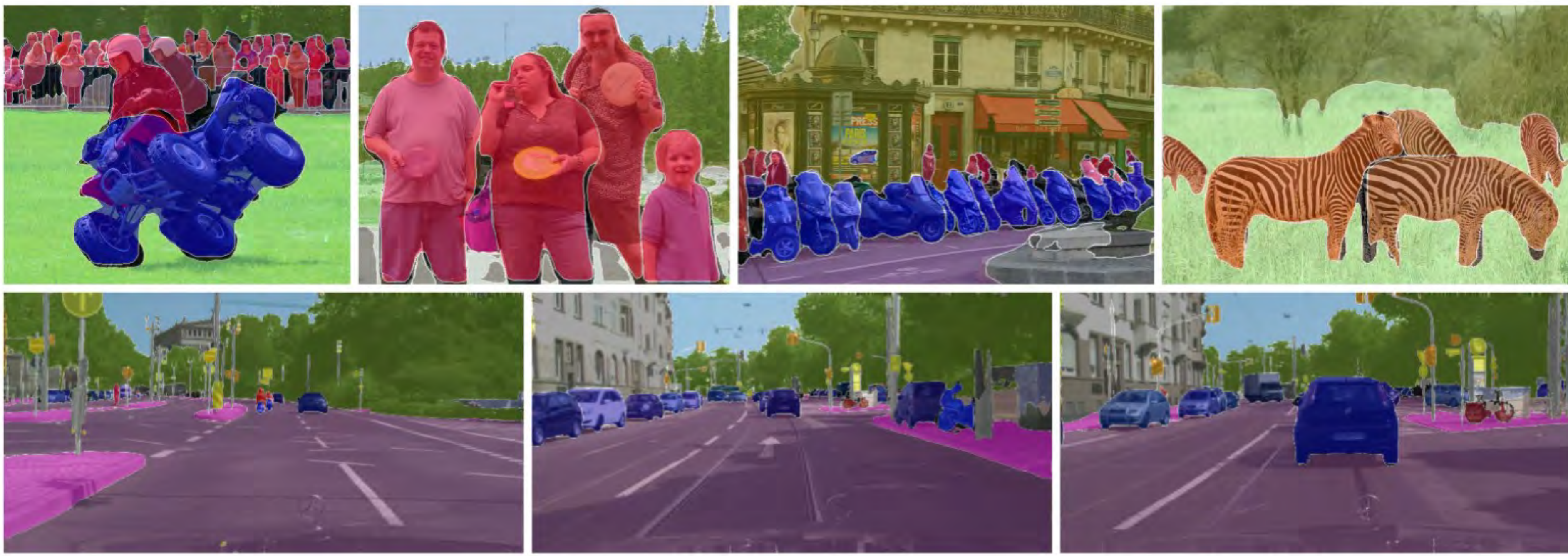
No Compromise on Feature Quality, still Fast



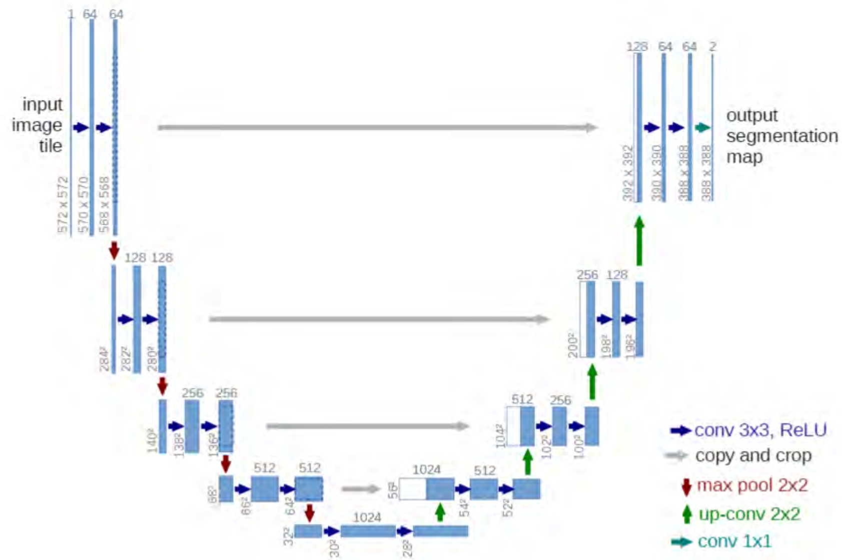


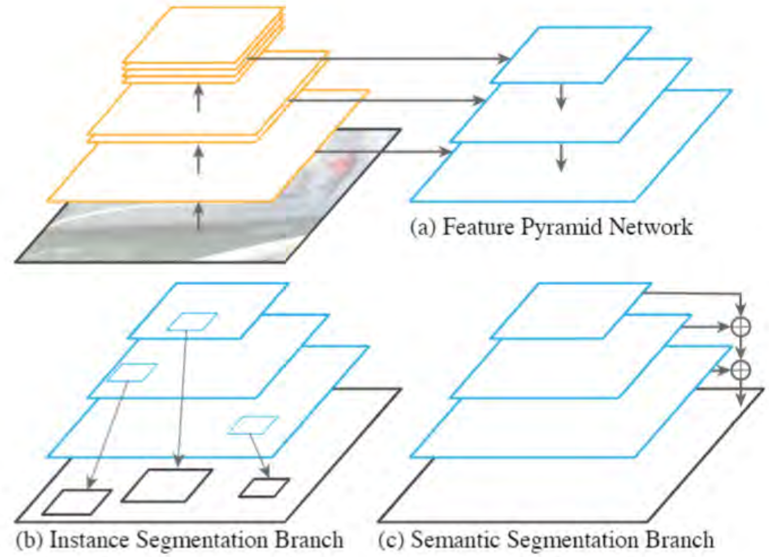
Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017. See also: Shrivastava's TDM.

Segmentation



Pyramid in segmentation

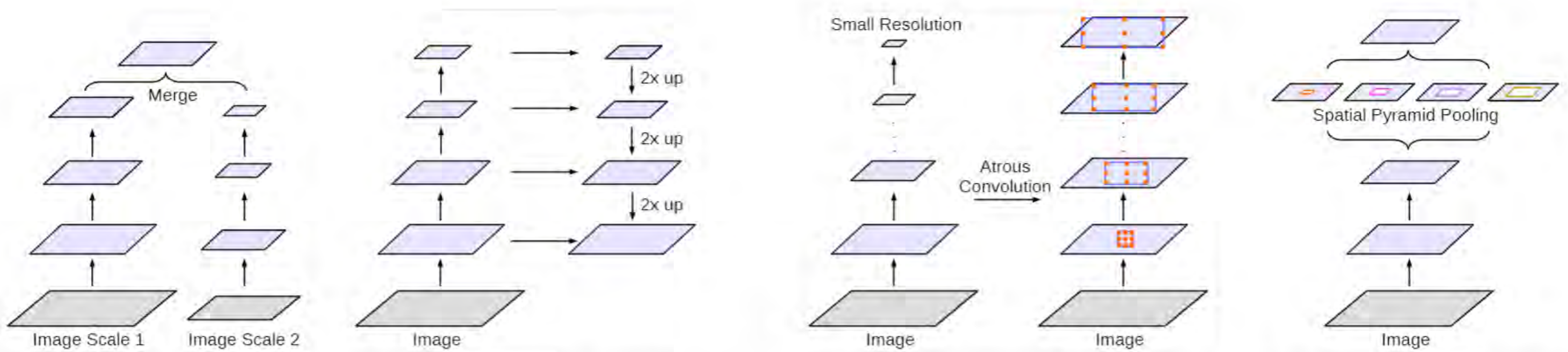




U-Net: Convolutional Networks for Biomedical Image Segmentation

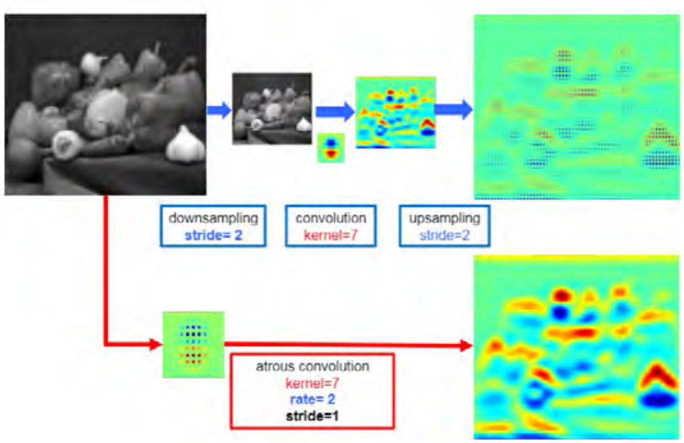
Panoptic Feature Pyramid Networks

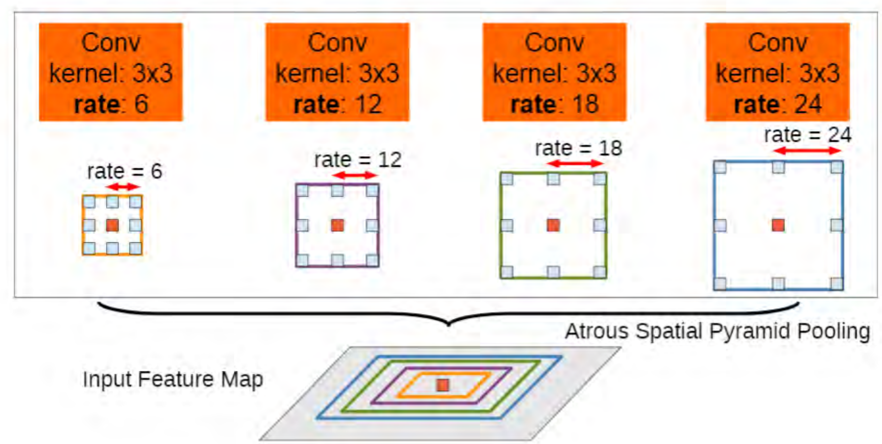
Atrous Convolution



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

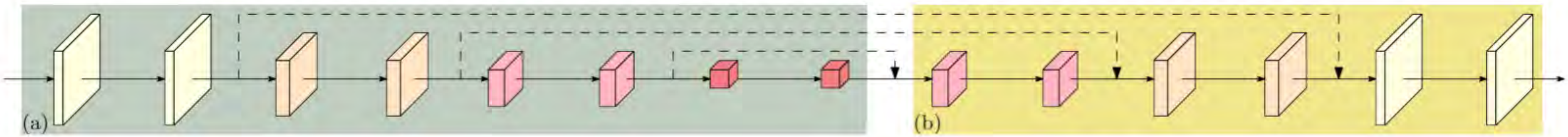
Figure 2. Alternative architectures to capture multi-scale context.

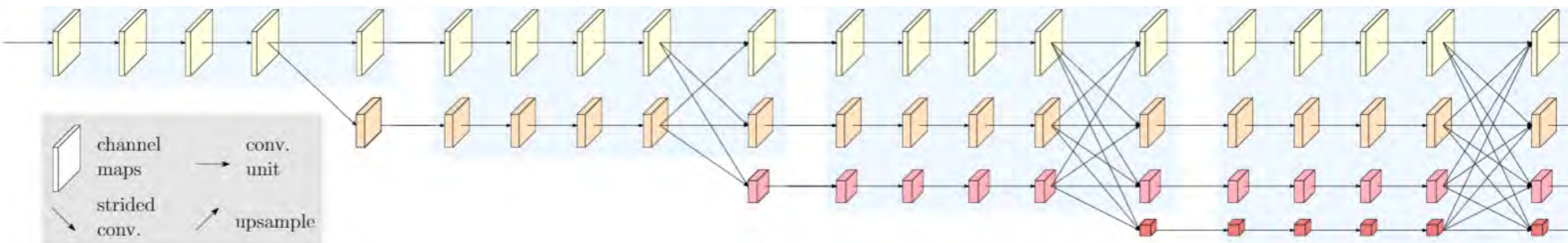




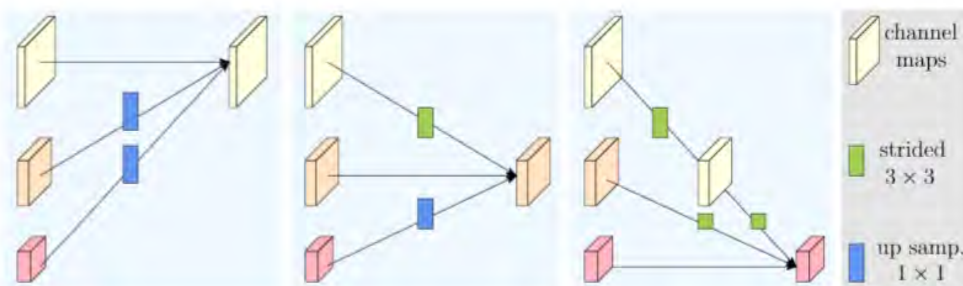
DeepLab: Semantic Image Segmentation withDeep Convolutional Nets, Atrous Convolution,and Fully Connected CRFs

HRNet



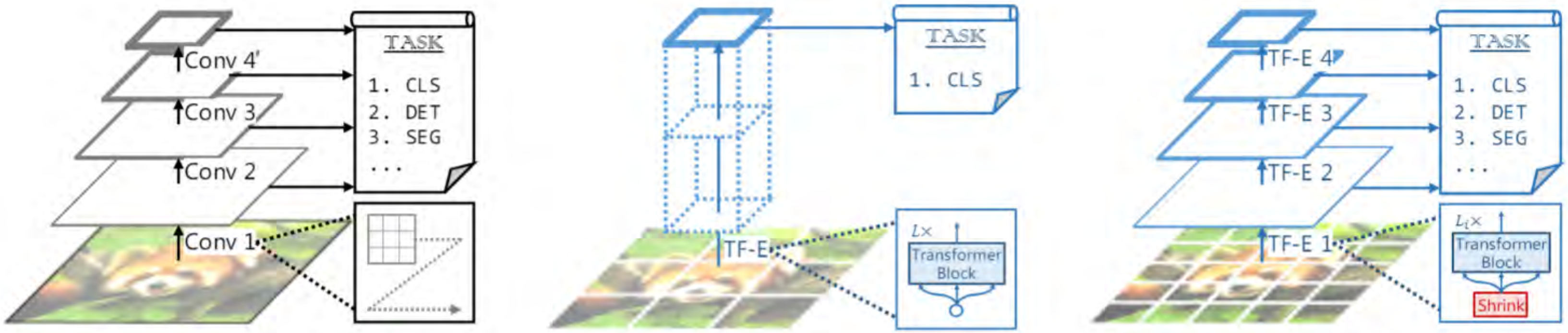


$$
\begin{array}{ccccccc}
\mathcal{N}_{11} & \rightarrow & \mathcal{N}_{21} & \rightarrow & \mathcal{N}_{31} & \rightarrow & \mathcal{N}_{41} \\
 & \searrow & \mathcal{N}_{22} & \rightarrow & \mathcal{N}_{32} & \rightarrow & \mathcal{N}_{42} \\
 & & & \searrow & \mathcal{N}_{33} & \rightarrow & \mathcal{N}_{43}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



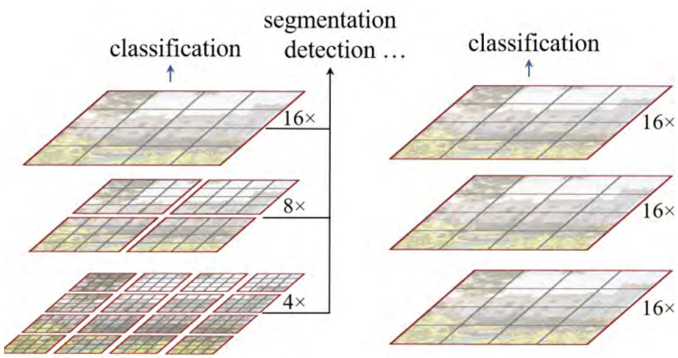
Deep High-Resolution Representation Learning for Visual Recognition

Pyramid in Transformer

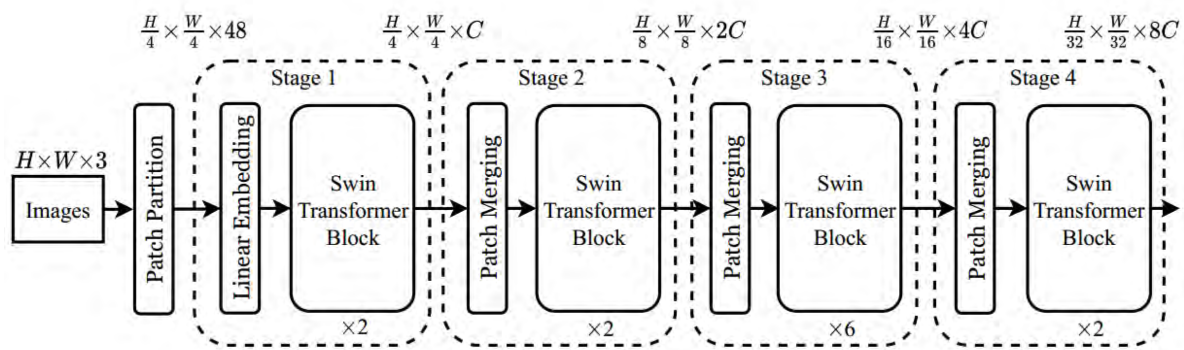


(a) CNNs: VGG [41], ResNet [15], *etc.* (b) Vision Transformer [10] (c) Pyramid Vision Transformer (ours)

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

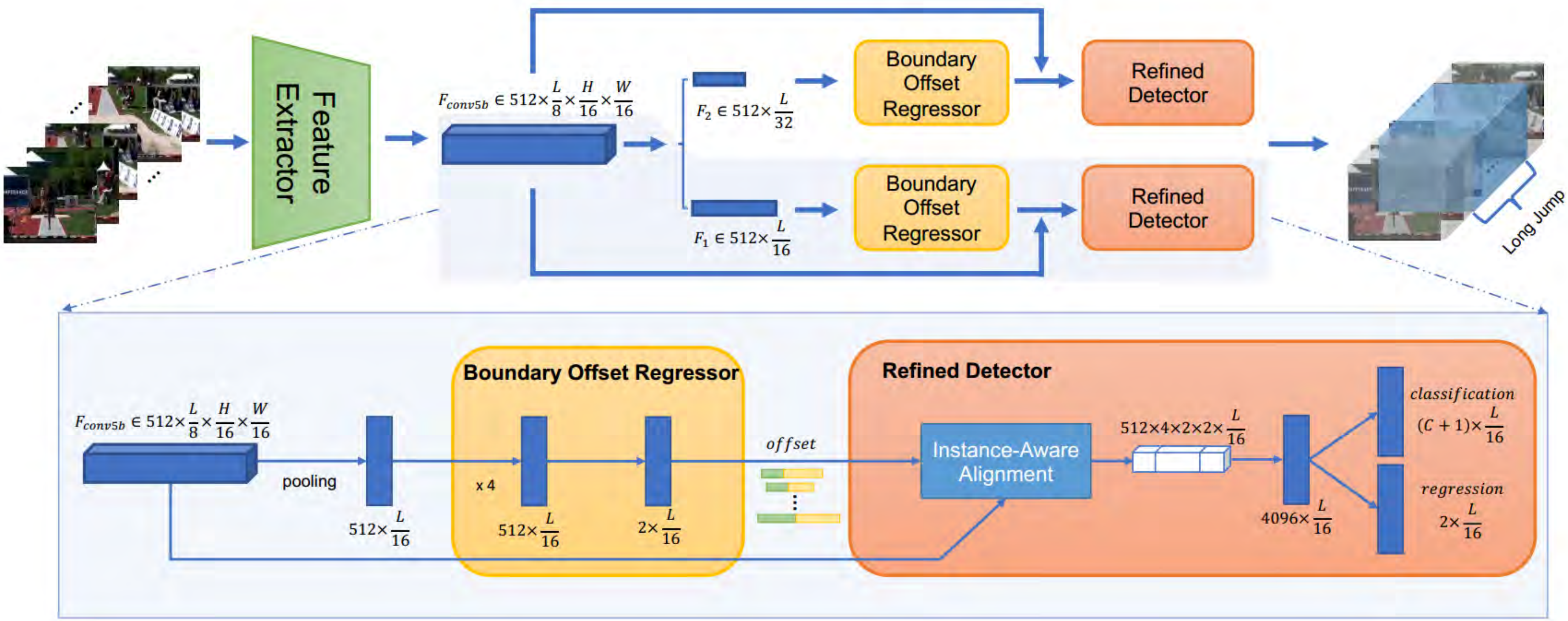


(a) Swin Transformer (ours) (b) ViT



Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Temporal Pyramid: Image to Video



Instance-Aware Alignment for Anchor Free Temporal Action Detection

Image pyramids

- Gaussian pyramid
  - Application for recognition
- **Laplacian pyramid**
  - Application for image blending

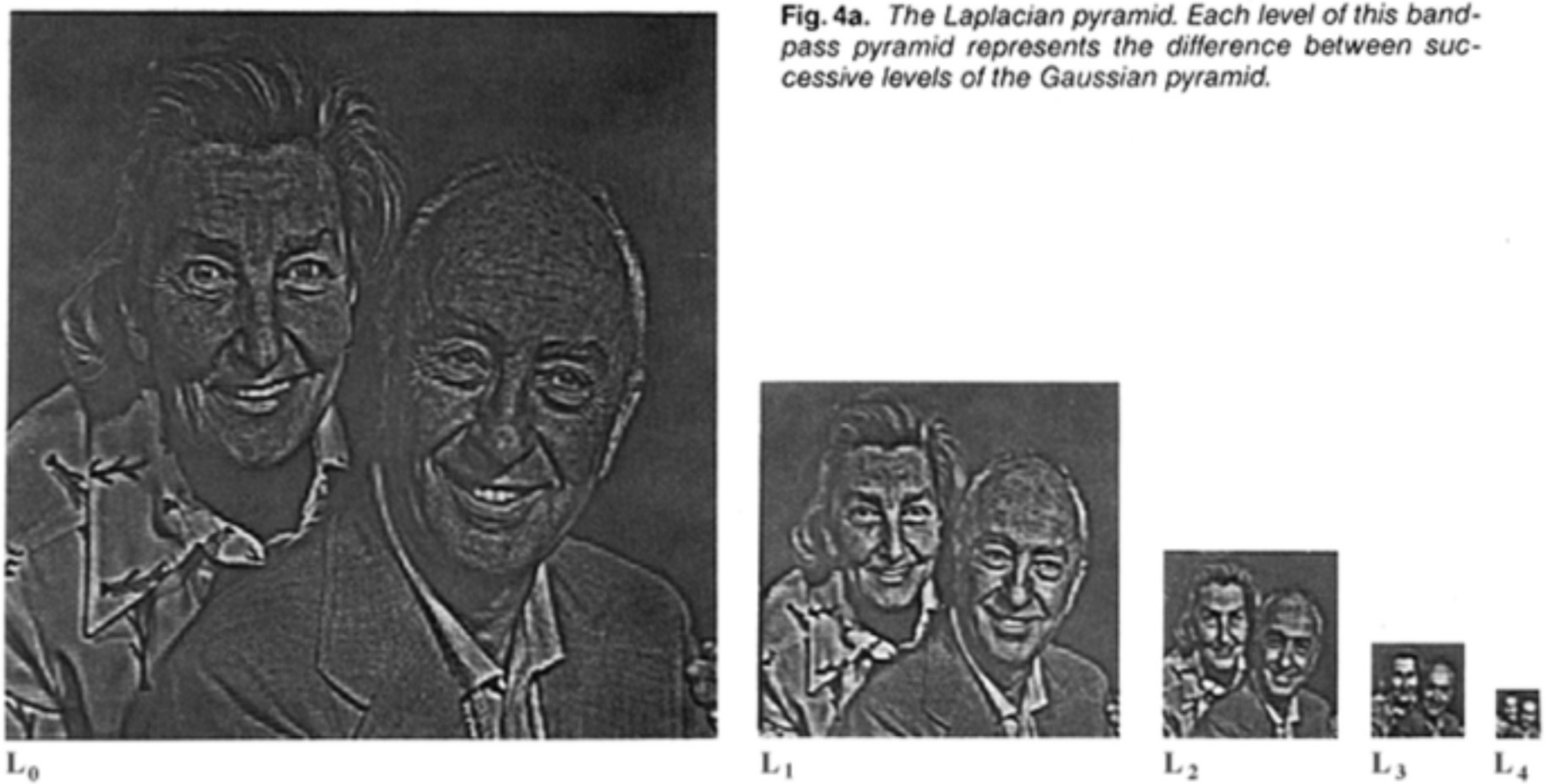
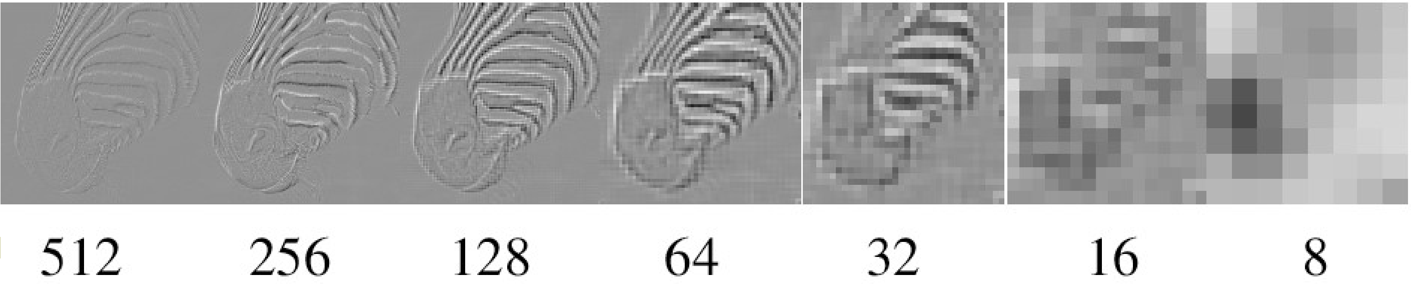
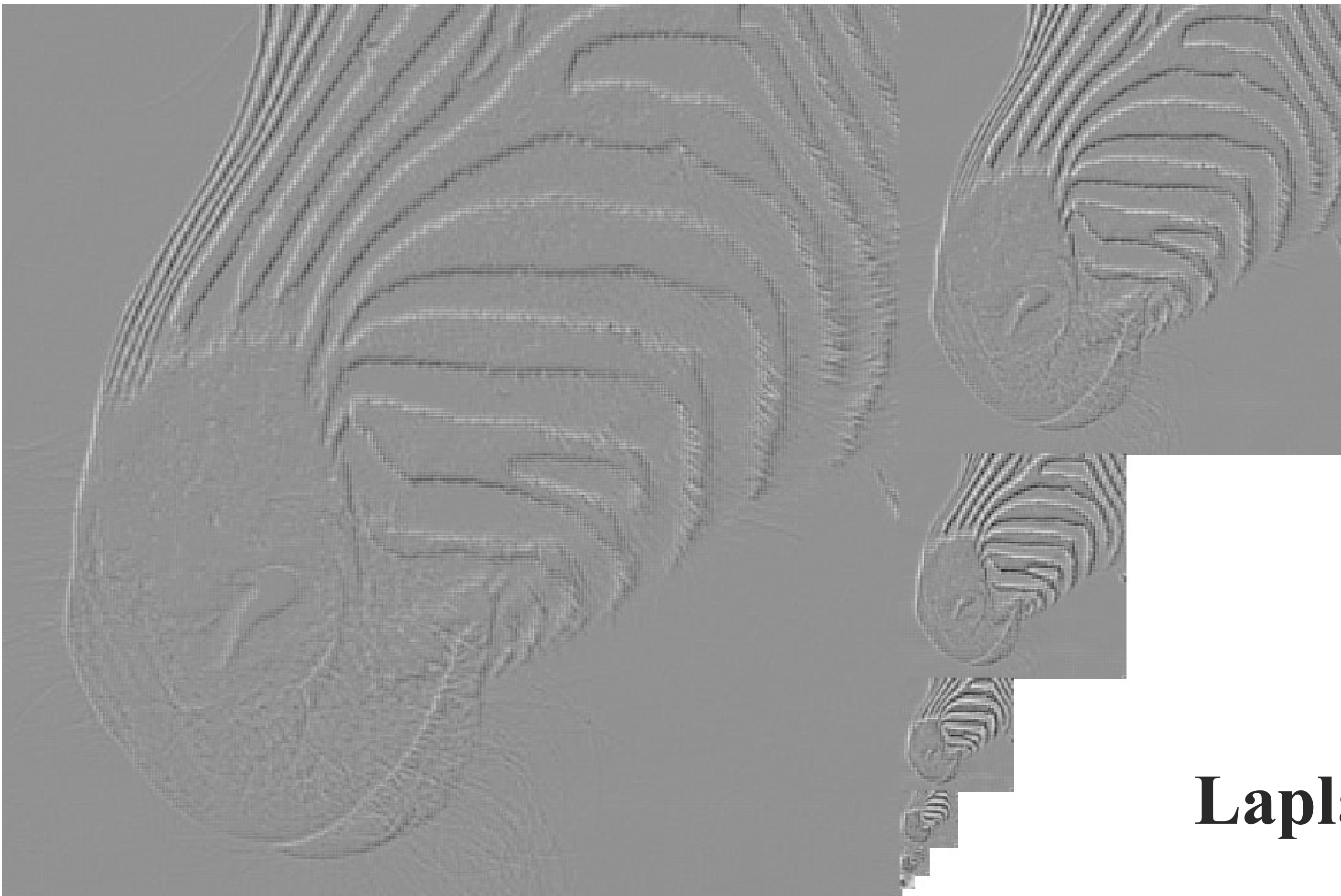


Fig. 4a. *The Laplacian pyramid. Each level of this band-pass pyramid represents the difference between successive levels of the Gaussian pyramid.*

$L_{0,0}$ $L_{1,0}$ $L_{2,2}$

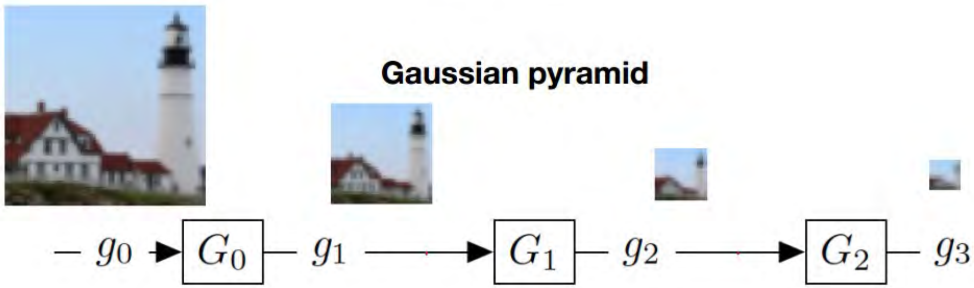
Fig. 4b. *Levels of the Laplacian pyramid expanded to the size of the original image. Note that edge and bar features are enhanced and segregated by size.*



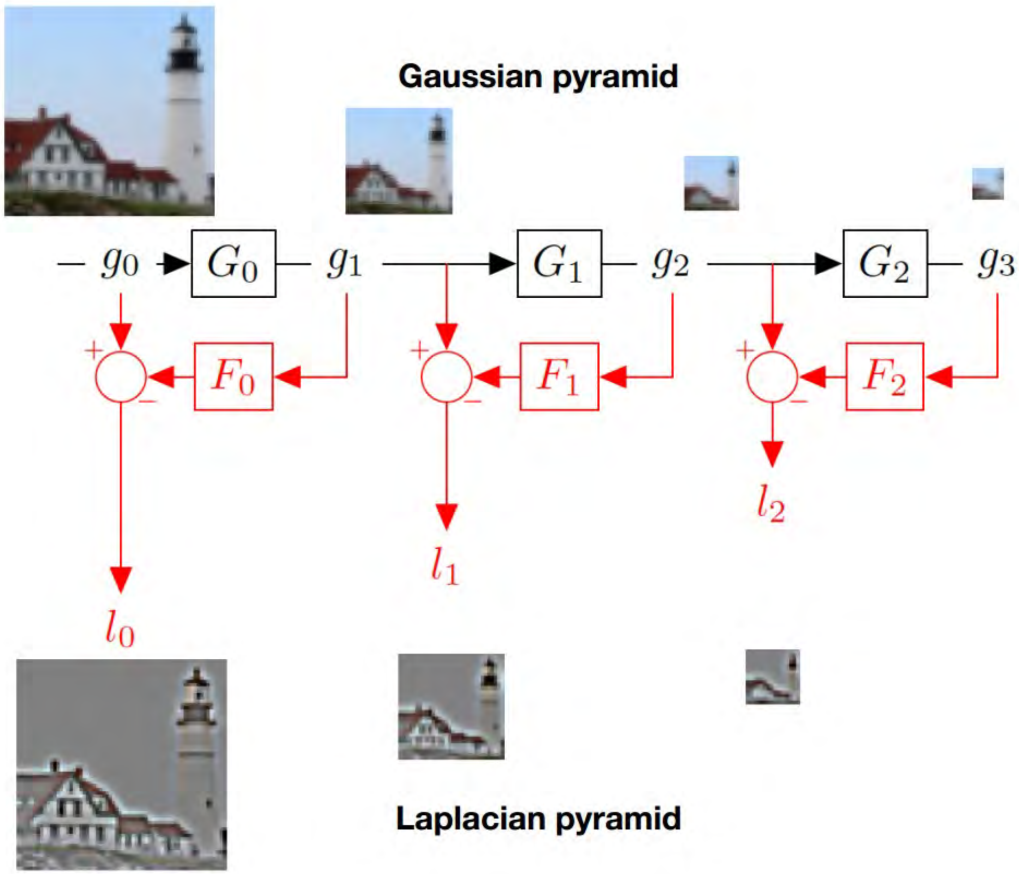


Laplacian Pyramid

The Laplacian Pyramid

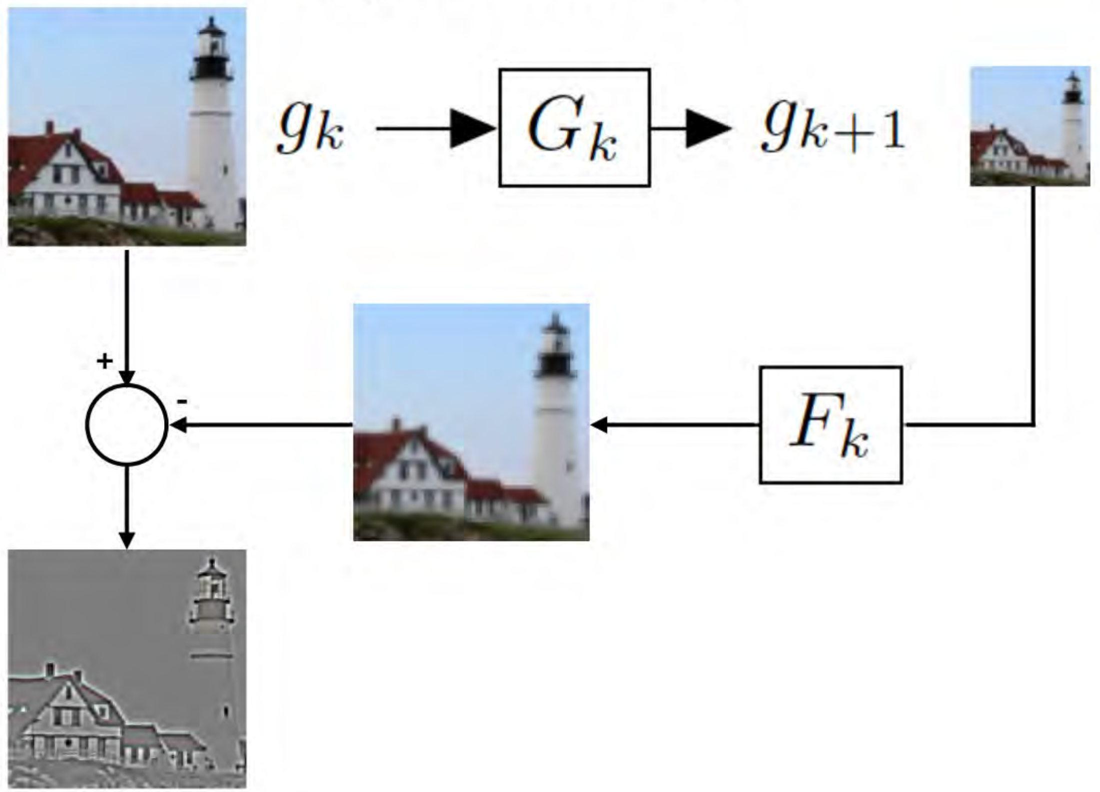


The Laplacian Pyramid

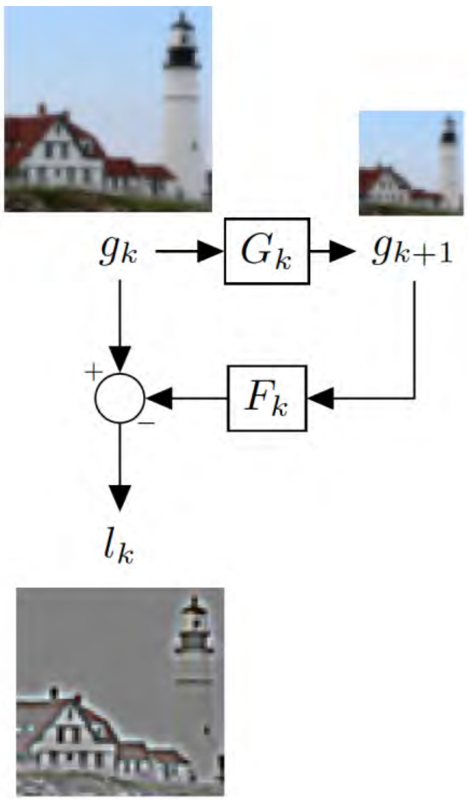


The Laplacian Pyramid

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



The Laplacian Pyramid



Blurring and downsampling:

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

(Downsampling by 2) (blur)

Upsampling and blurring:

$$F_0 =$$

Downsampling & Upsampling

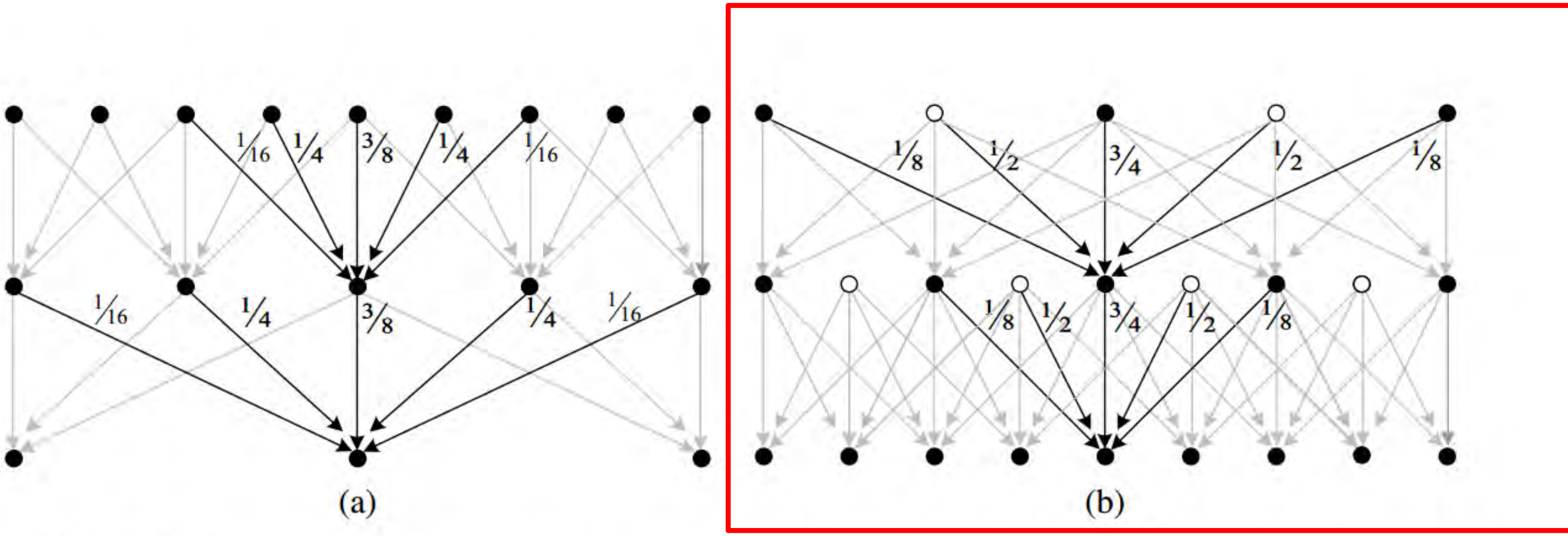


Figure 3.33 The Gaussian pyramid shown as a signal processing diagram: The (a) analysis and (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the $\uparrow 2$ upsampling operation. Notice how the reconstruction filter coefficients are 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

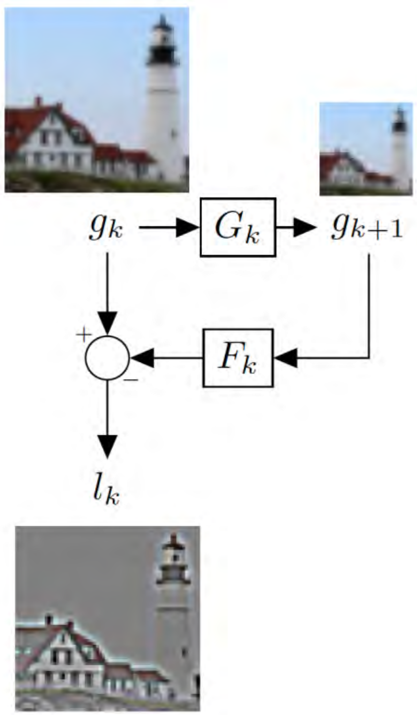
Insert zeros

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:

$$G_0 = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{(Downsampling by 2)}} \frac{1}{16} \underbrace{\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}}_{\text{(blur)}}$$

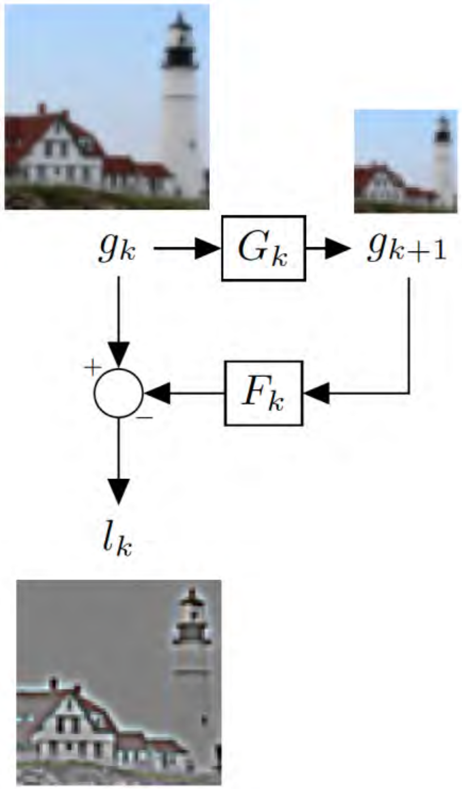
Upsampling and blurring:

$$F_0 = \frac{1}{8} \underbrace{\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}}_{\text{(blur)}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{(Upsampling by 2)}}$$

$$\boxed{l_0 = (I_0 - F_0 G_0) g_0}$$

22

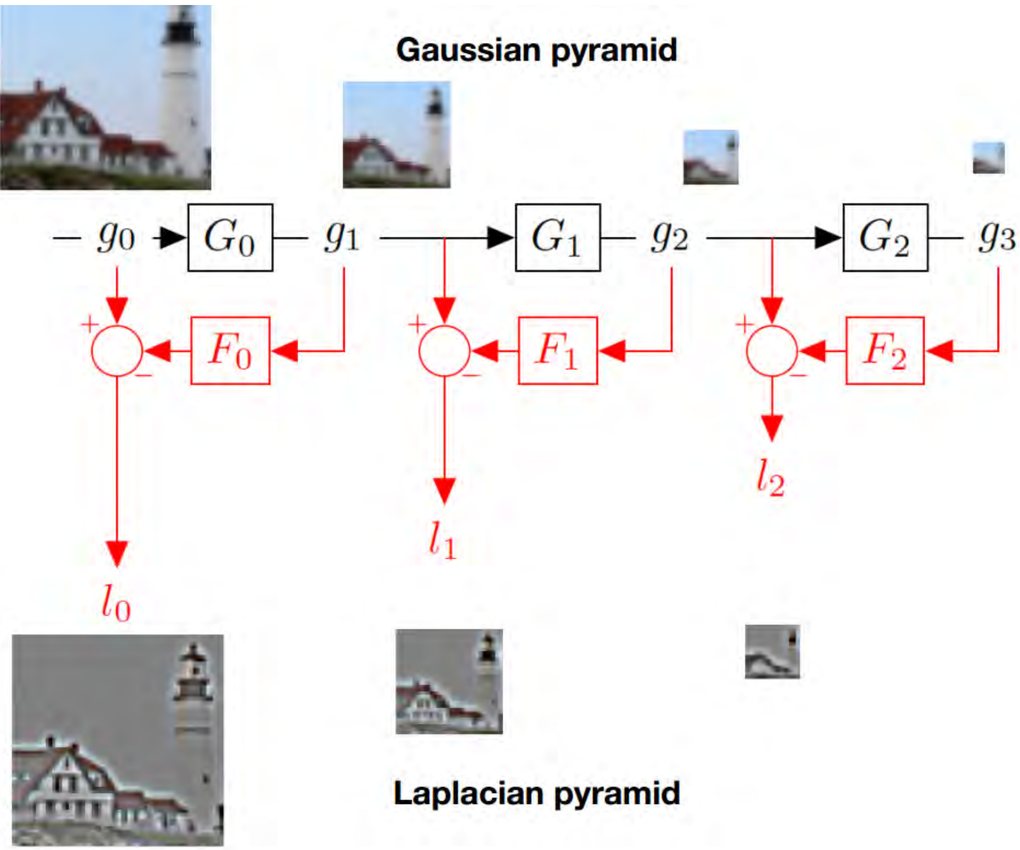
The Laplacian Pyramid

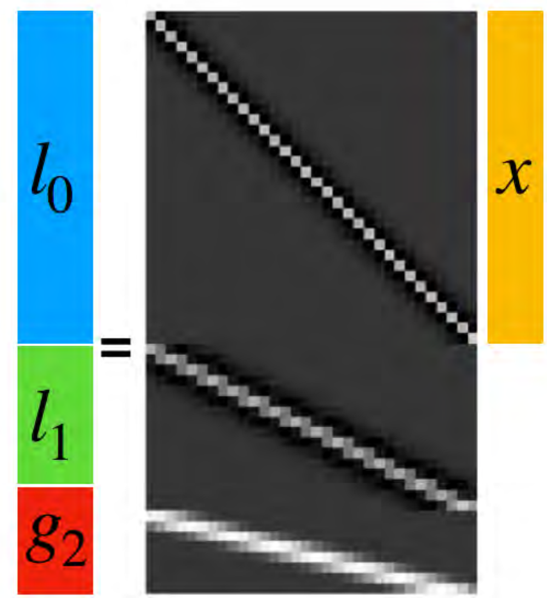


$$\boxed{l_0 = (I_0 - F_0 G_0) g_0}$$

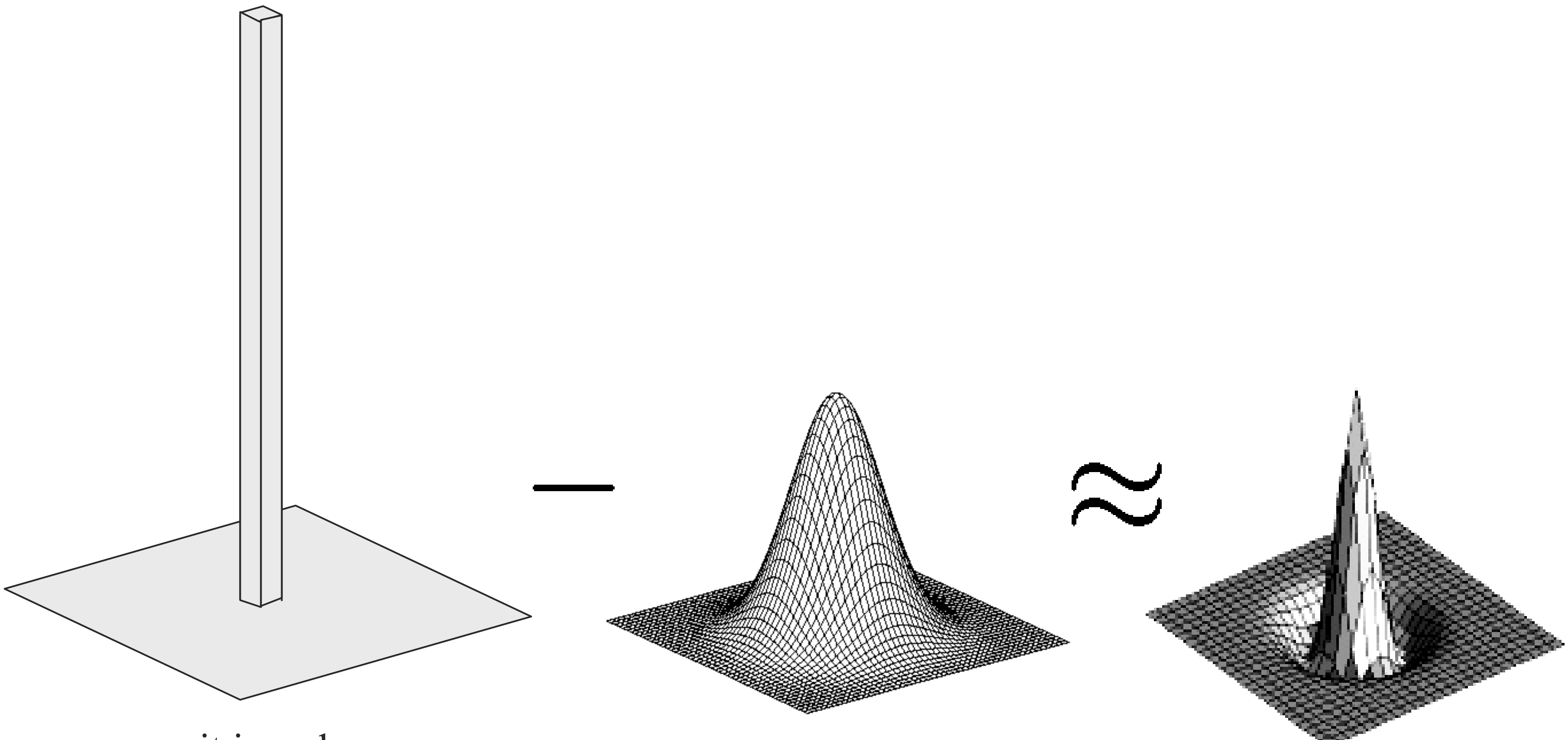
$$= \frac{1}{256}\begin{bmatrix} 182 & -56 & -24 & -8 & -2 & 0 & 0 & 0 \\ -56 & 192 & -56 & -32 & -8 & 0 & 0 & 0 \\ -24 & -56 & 180 & -56 & -24 & -8 & -2 & 0 \\ -8 & -32 & -56 & 192 & -56 & -32 & -8 & 0 \\ -2 & -8 & -24 & -56 & 180 & -56 & -24 & -8 \\ 0 & 0 & -8 & -32 & -56 & 192 & -56 & -32 \\ 0 & 0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

The Laplacian Pyramid





Laplacian filter



unit impulse

Gaussian

Laplacian of Gaussian

LoG vs. DoG

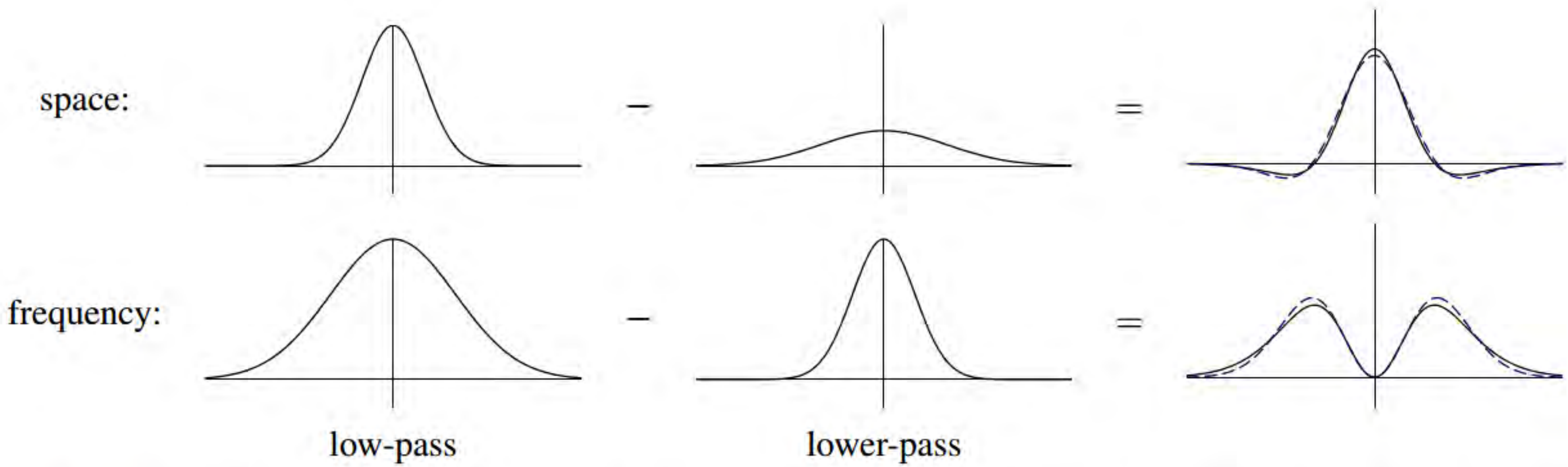


Figure 3.35 The difference of two low-pass filters results in a band-pass filter. The dashed blue lines show the close fit to a half-octave Laplacian of Gaussian.

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

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

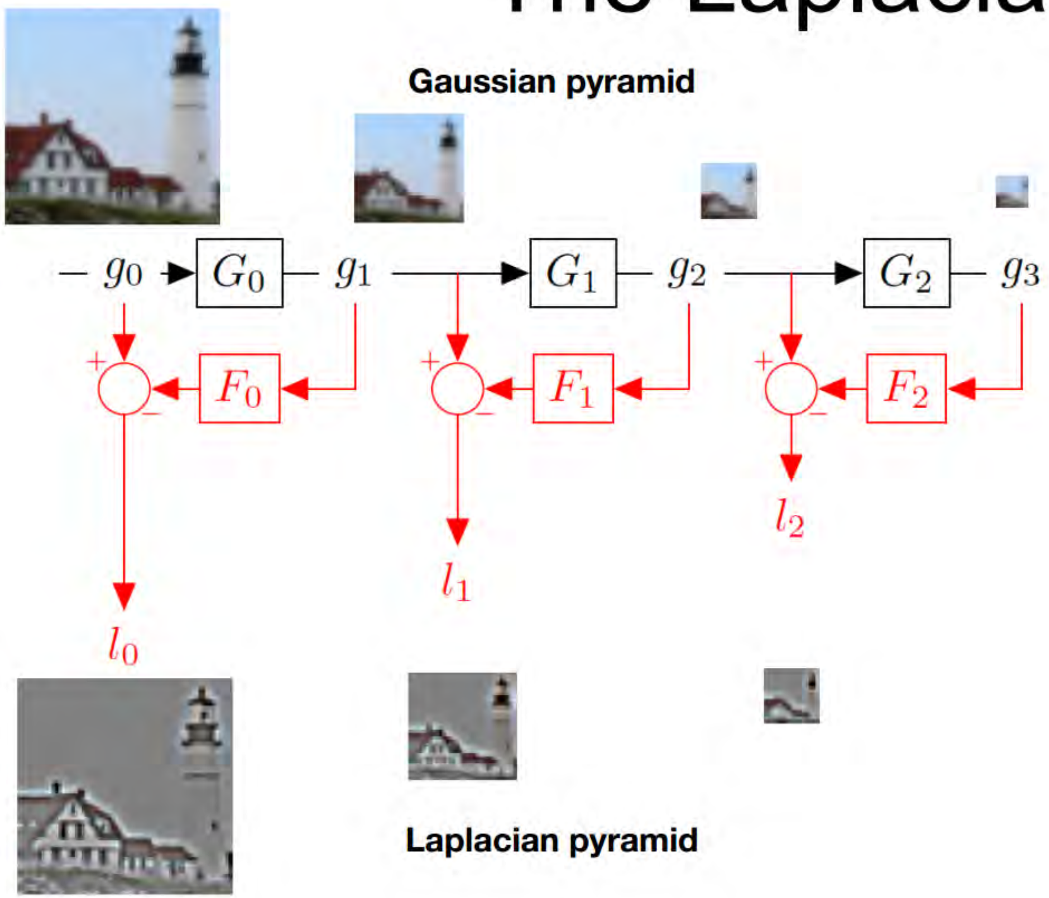
$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

The Laplacian Pyramid

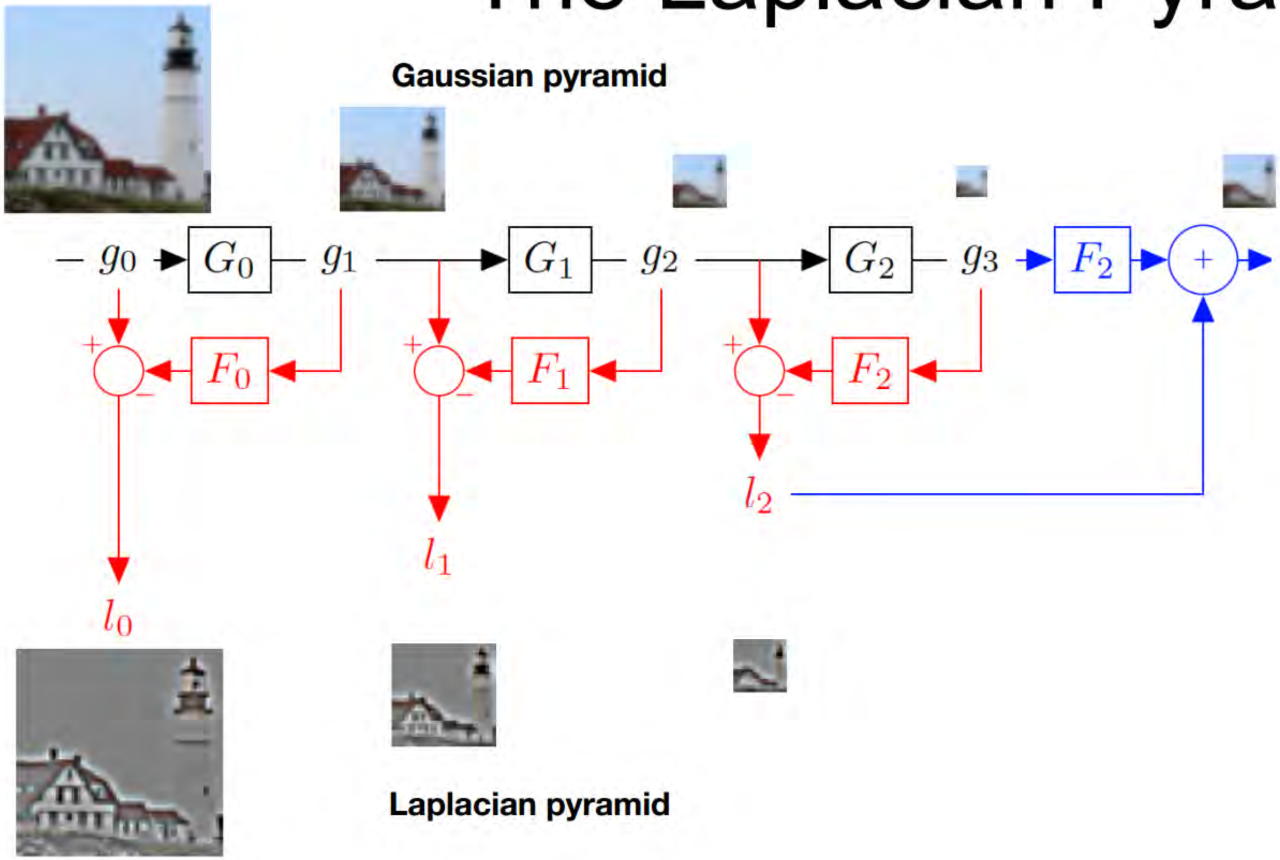
Laplacian pyramid

Can we invert the Laplacian Pyramid?

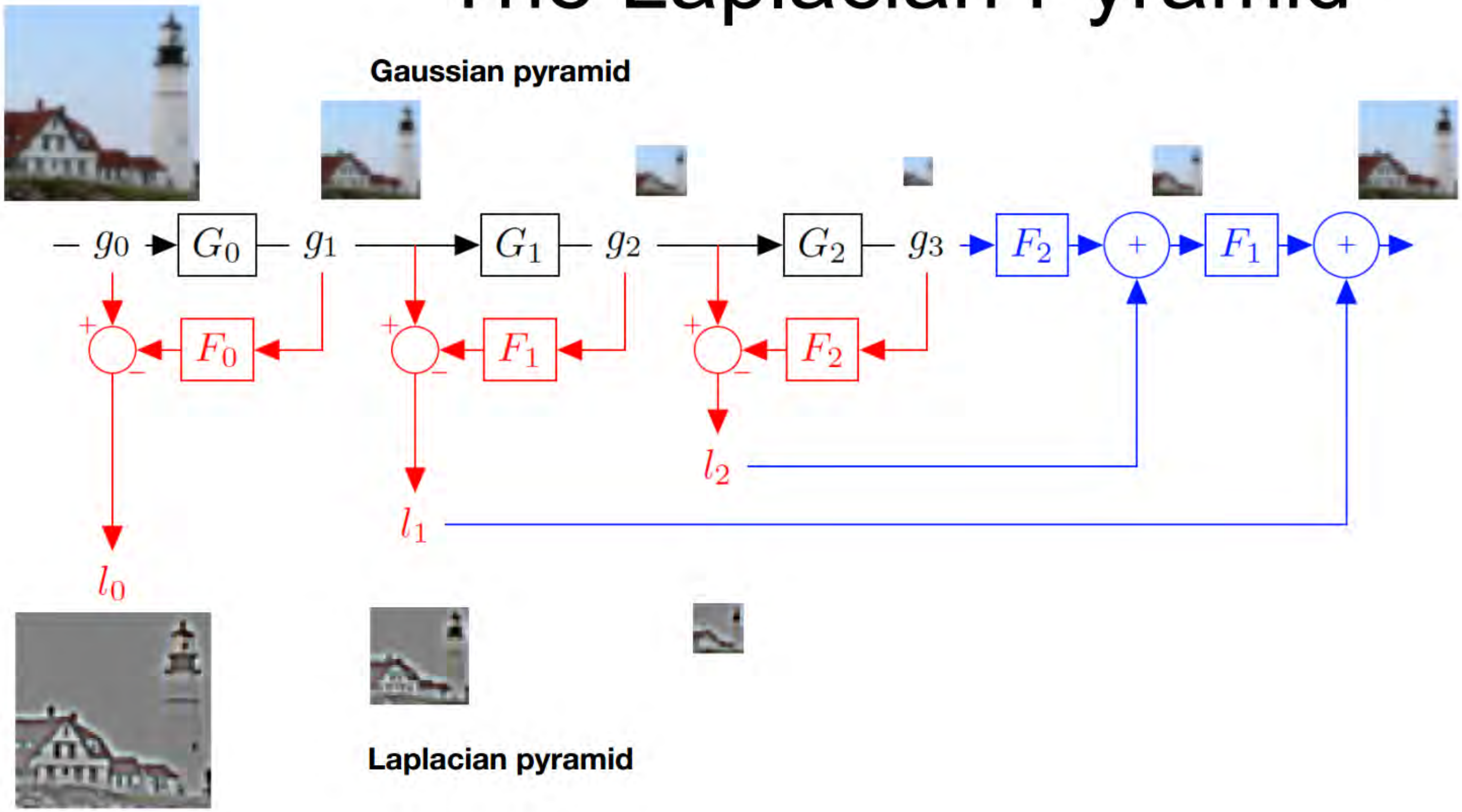
The Laplacian Pyramid



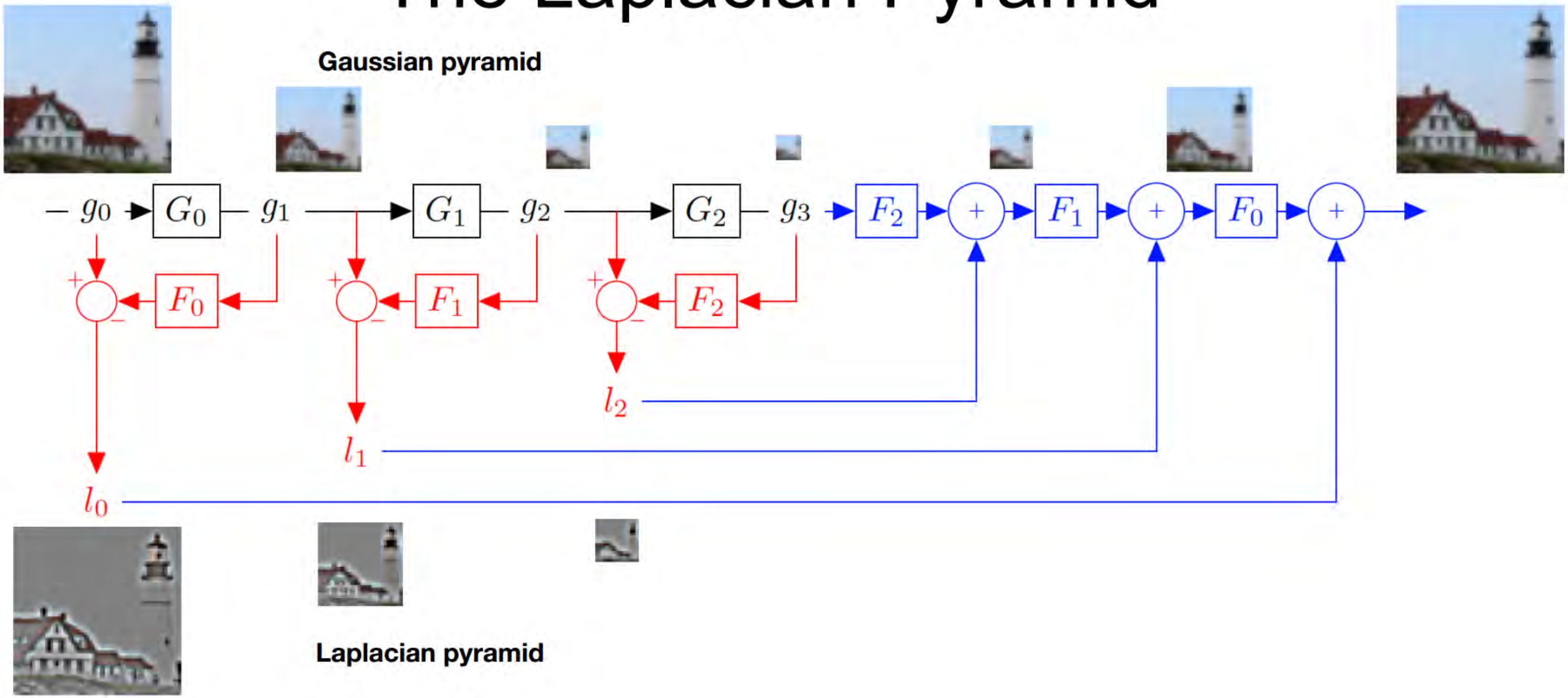
The Laplacian Pyramid



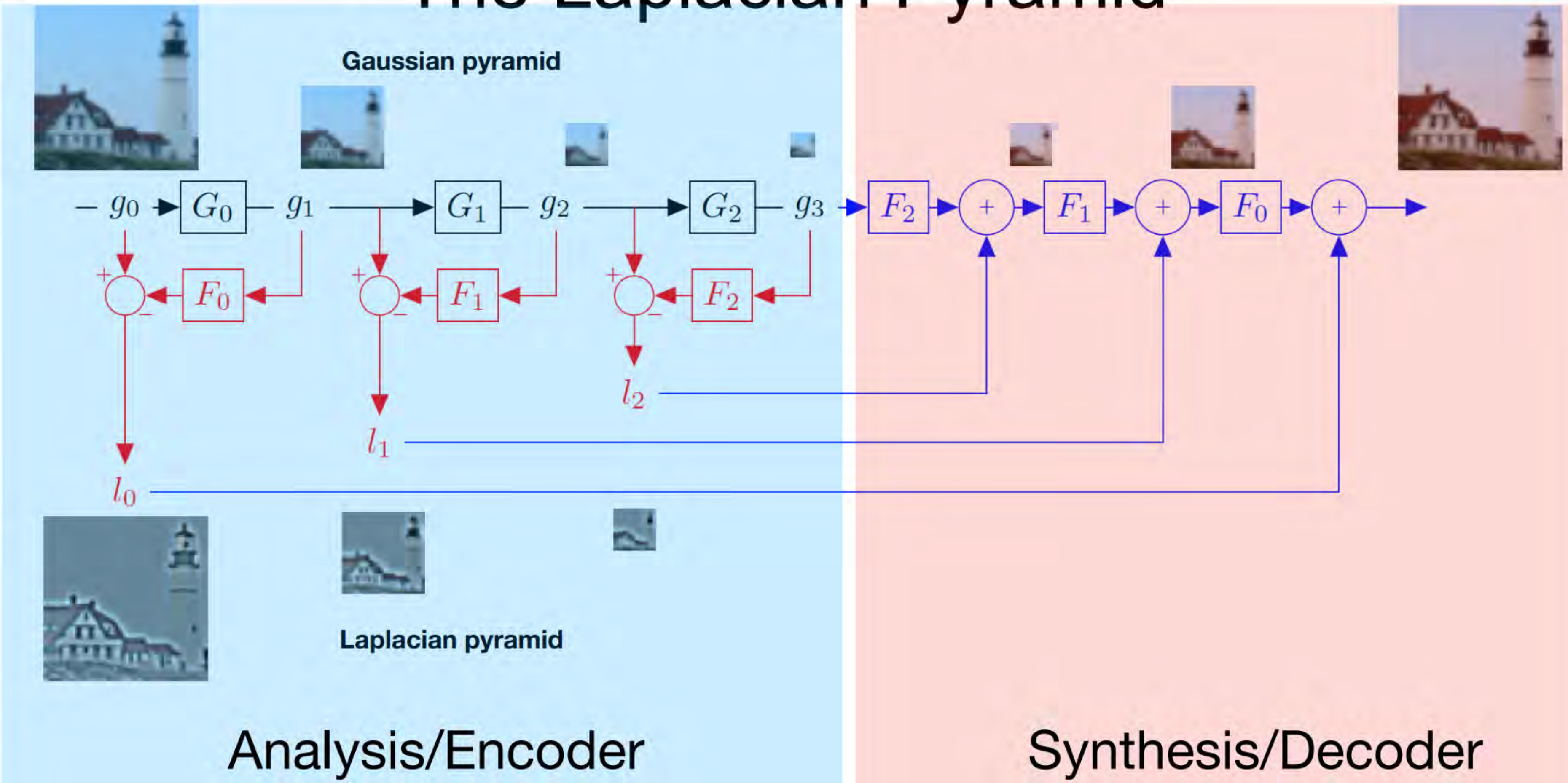
The Laplacian Pyramid



The Laplacian Pyramid



The Laplacian Pyramid



Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and x_4

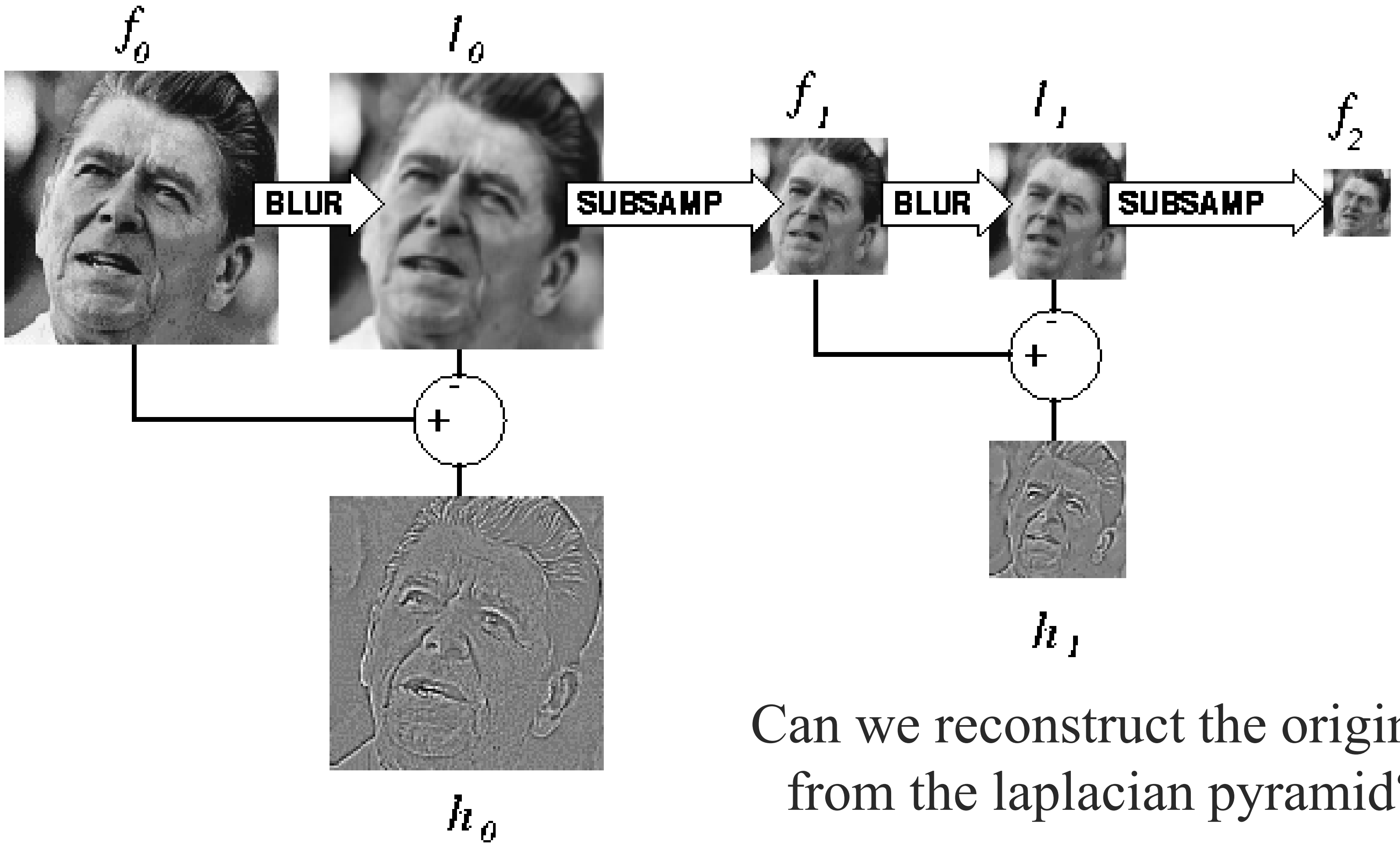
G# is the blur-and-downsample operator at pyramid level #
F# is the blur-and-upsample operator at pyramid level #

First, form Gaussian pyramid:
$x2 = G1\ x1$
$x3 = G2\ x2$
$x4 = G3\ x3$

Then the Laplacian pyramid elements are:
$L1 = (I - F1\ G1)\ x1$
$L2 = (I - F2\ G2)\ x2$
$L3 = (I - F3\ G3)\ x3$

Reconstruction of original image (x1) from Laplacian pyramid elements and the smallest level of the Gaussian pyramid, x4:
$x3 = L3 + F3\ x4$
$x2 = L2 + F2\ x3$
$x1 = L1 + F1\ x2$

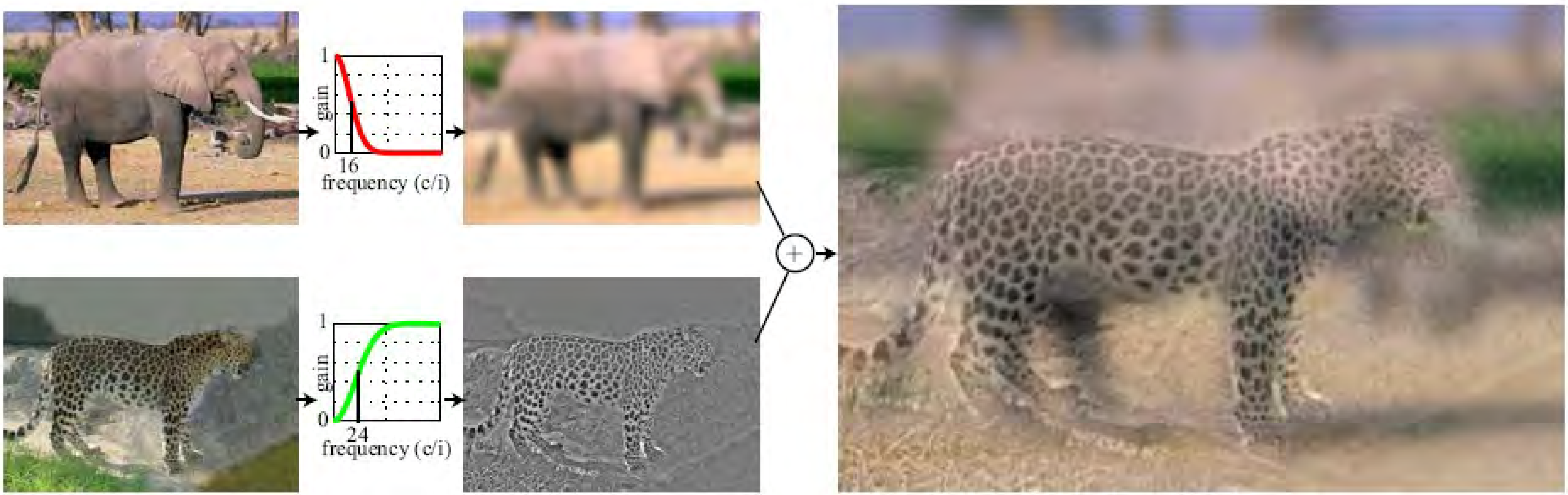
Computing Gaussian/Laplacian Pyramid



Can we reconstruct the original from the laplacian pyramid?

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

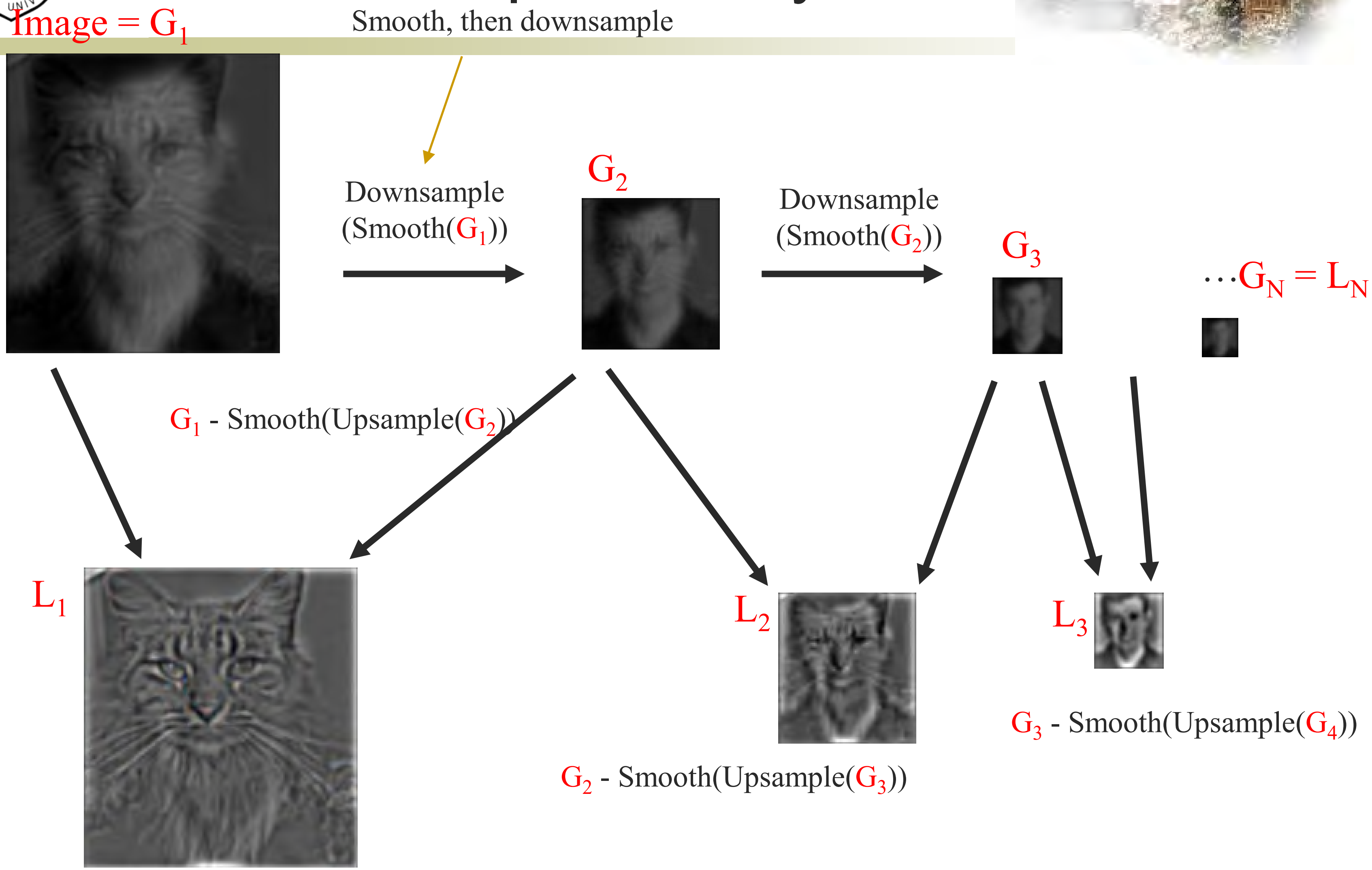
Hybrid Images



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

Hybrid Image

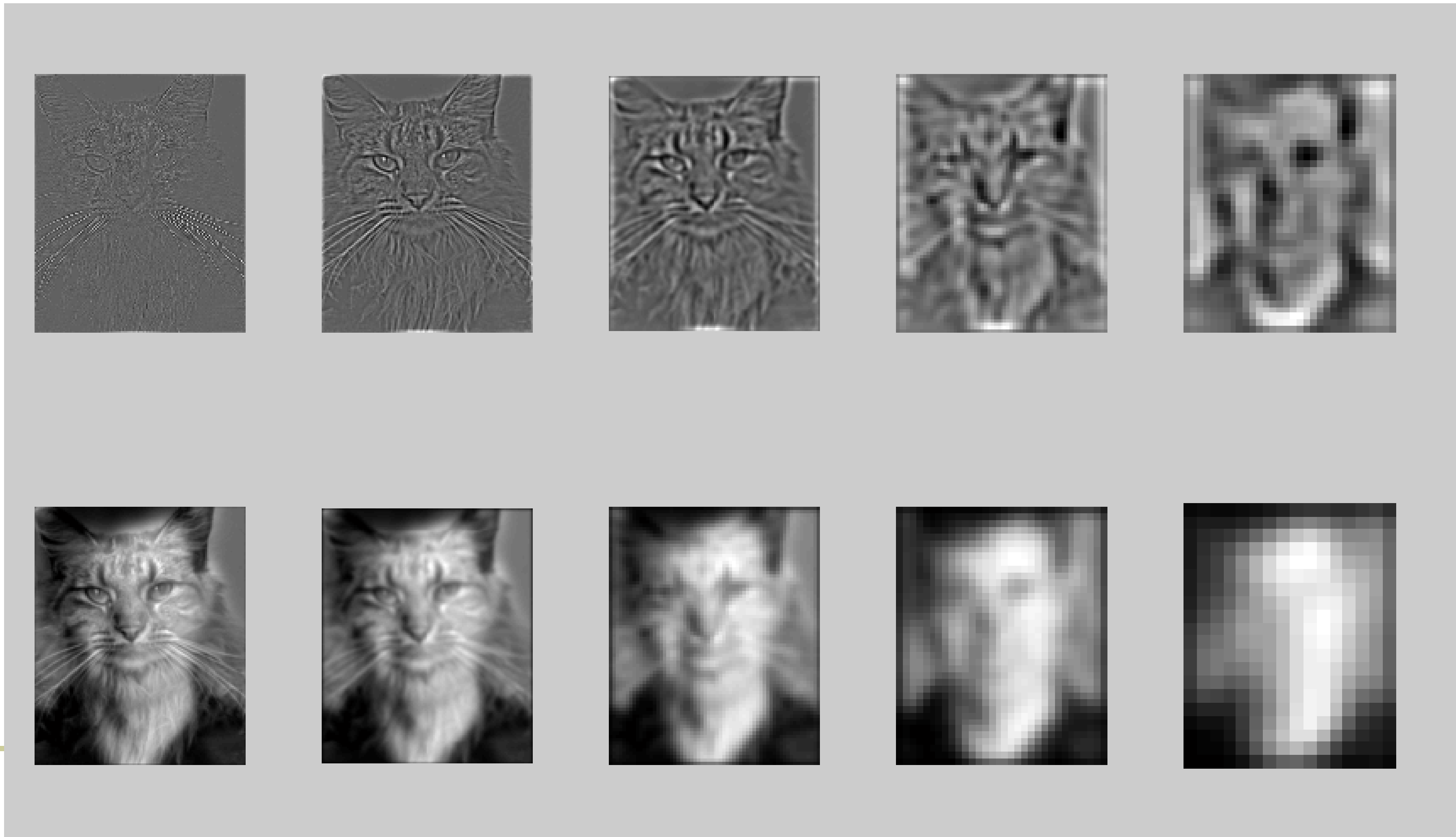
Creating the Gaussian/Laplacian Pyramid



- 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

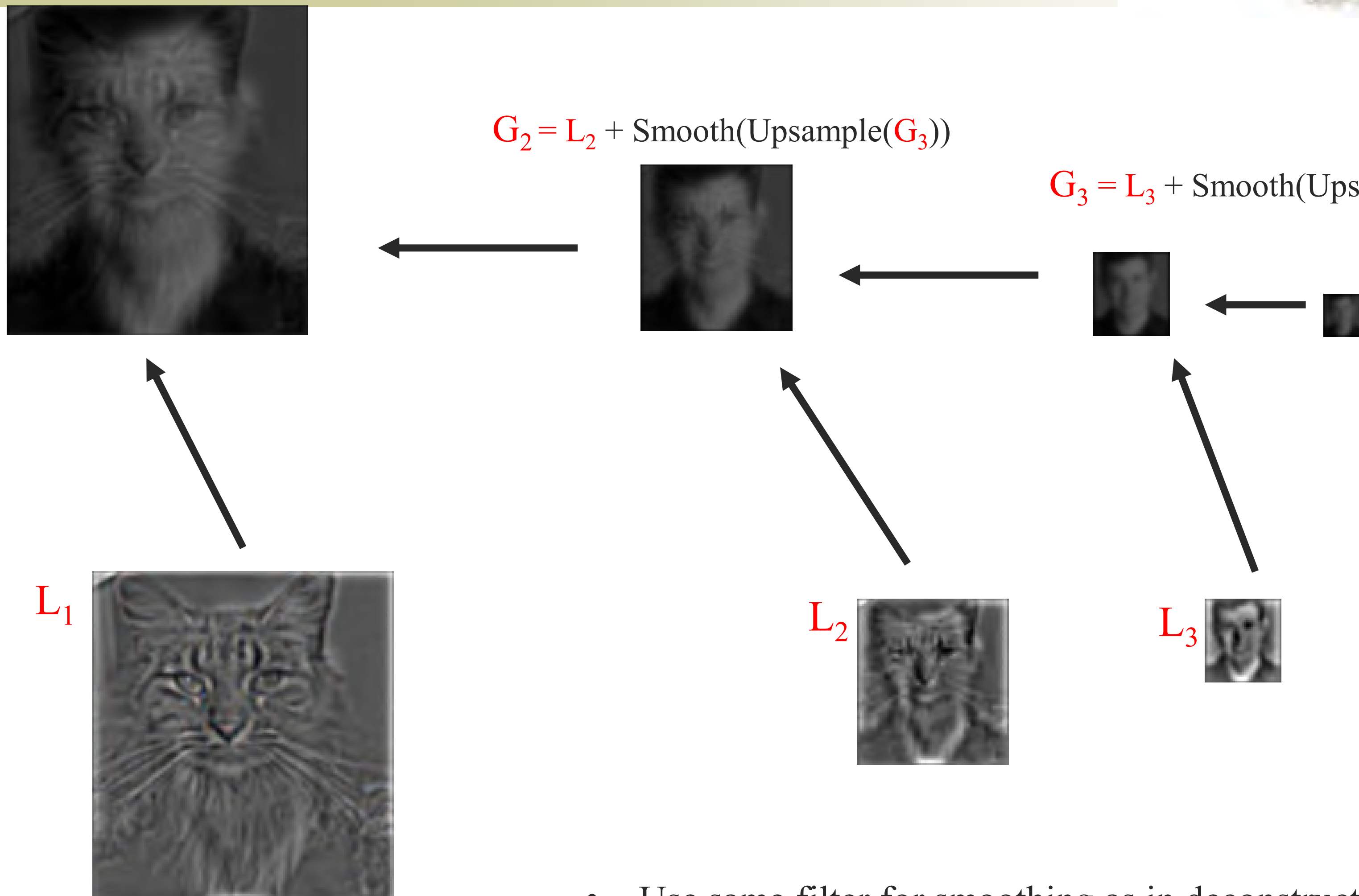


Reconstructing image from Laplacian pyramid

Image = L_1 + Smooth(Upsample(G_2))

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

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



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

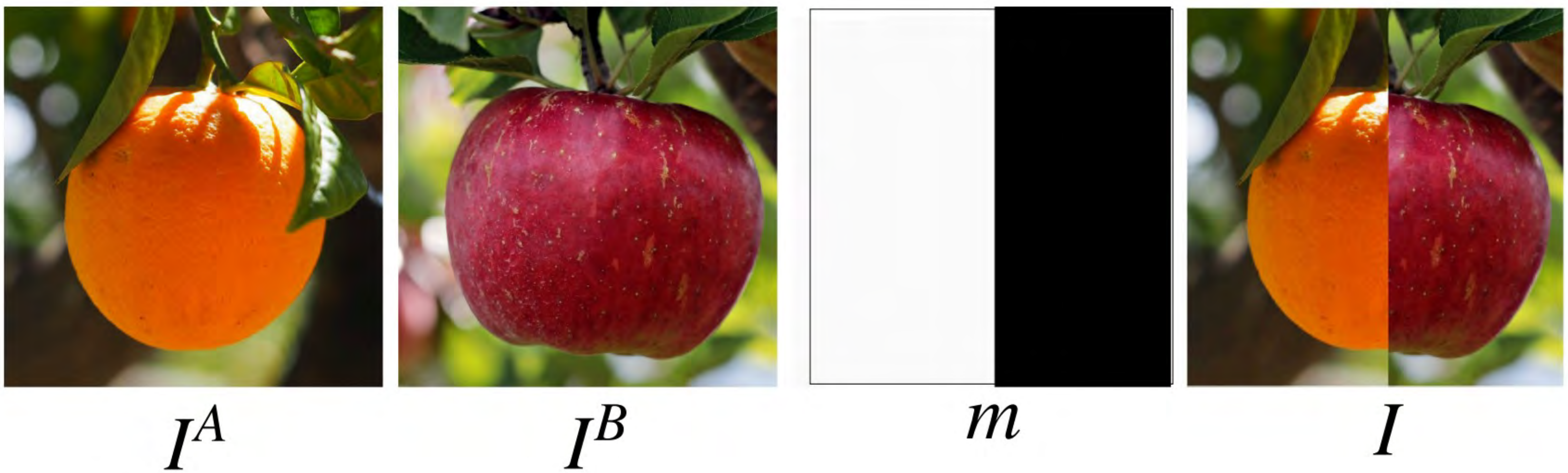
Laplacian pyramid applications

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

Image Blending

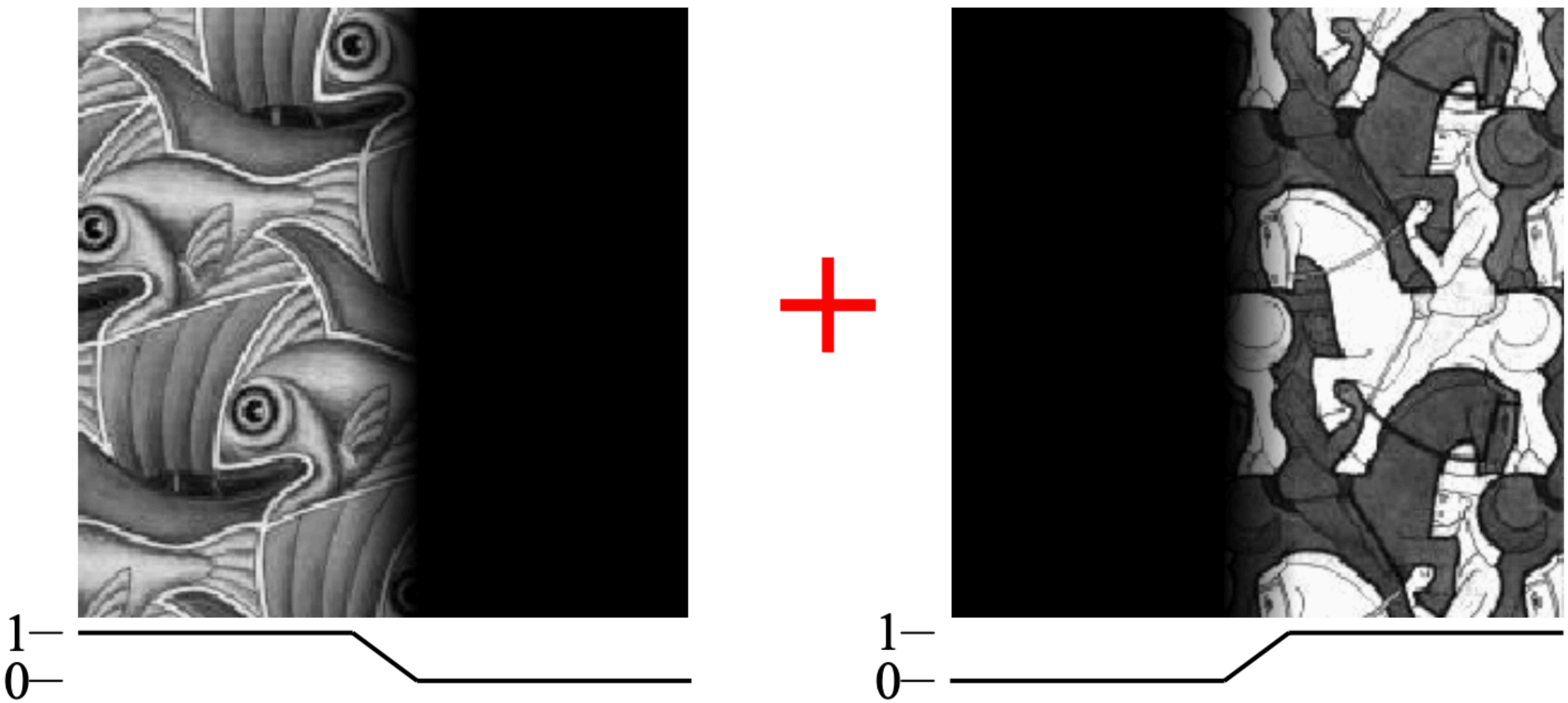
Image Blending

Image Blending

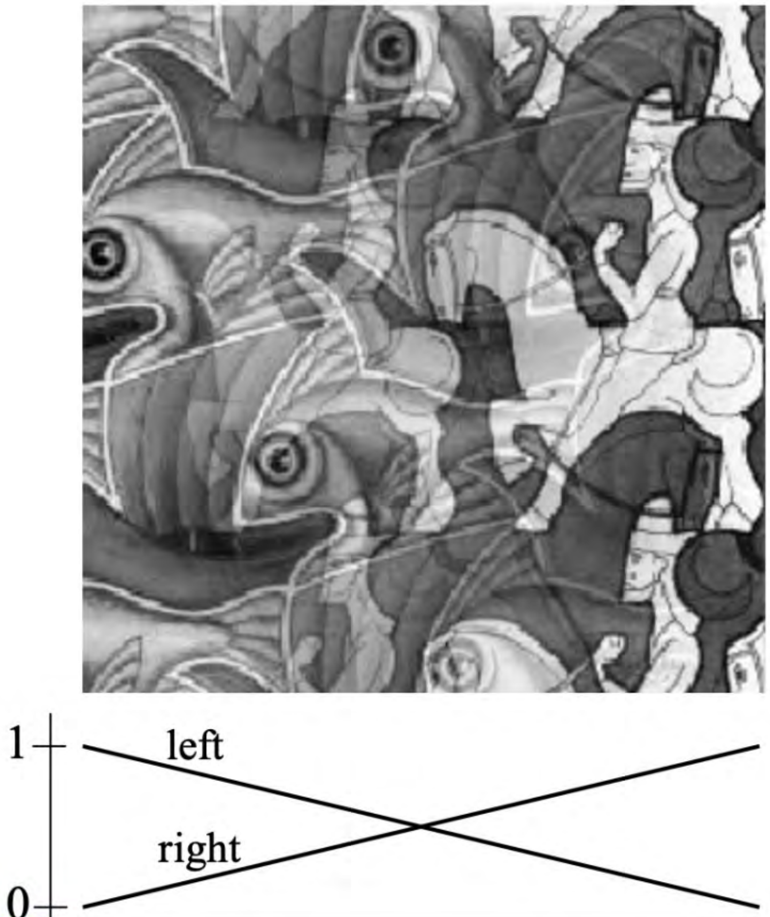


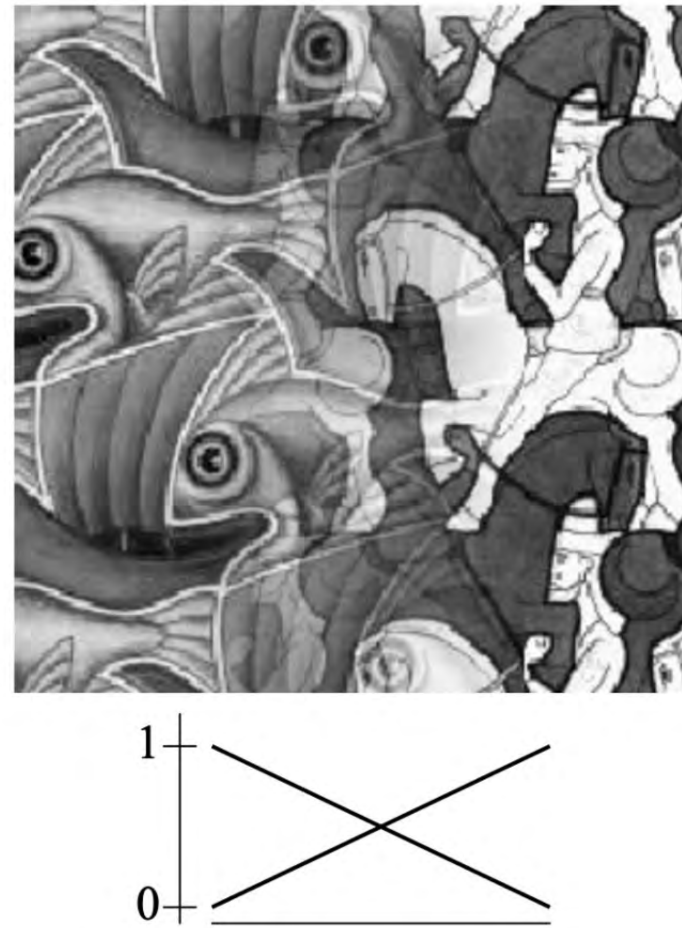
$$I = m * I^A + (1 - m) * I^B$$

Feathering



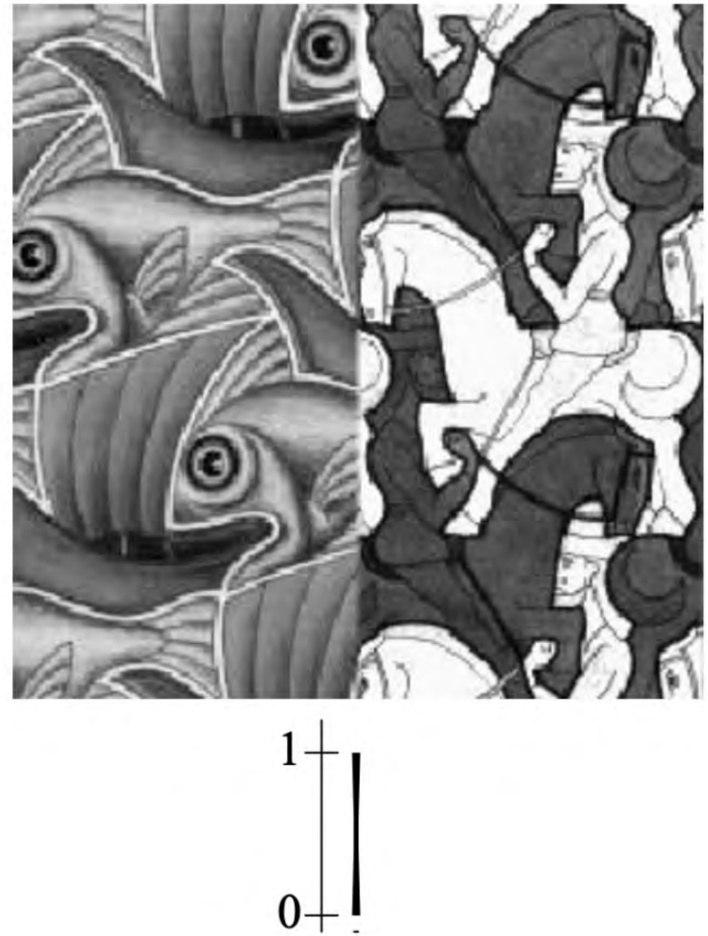
Affect of window size





Slide by A. Efros

Affect of window size

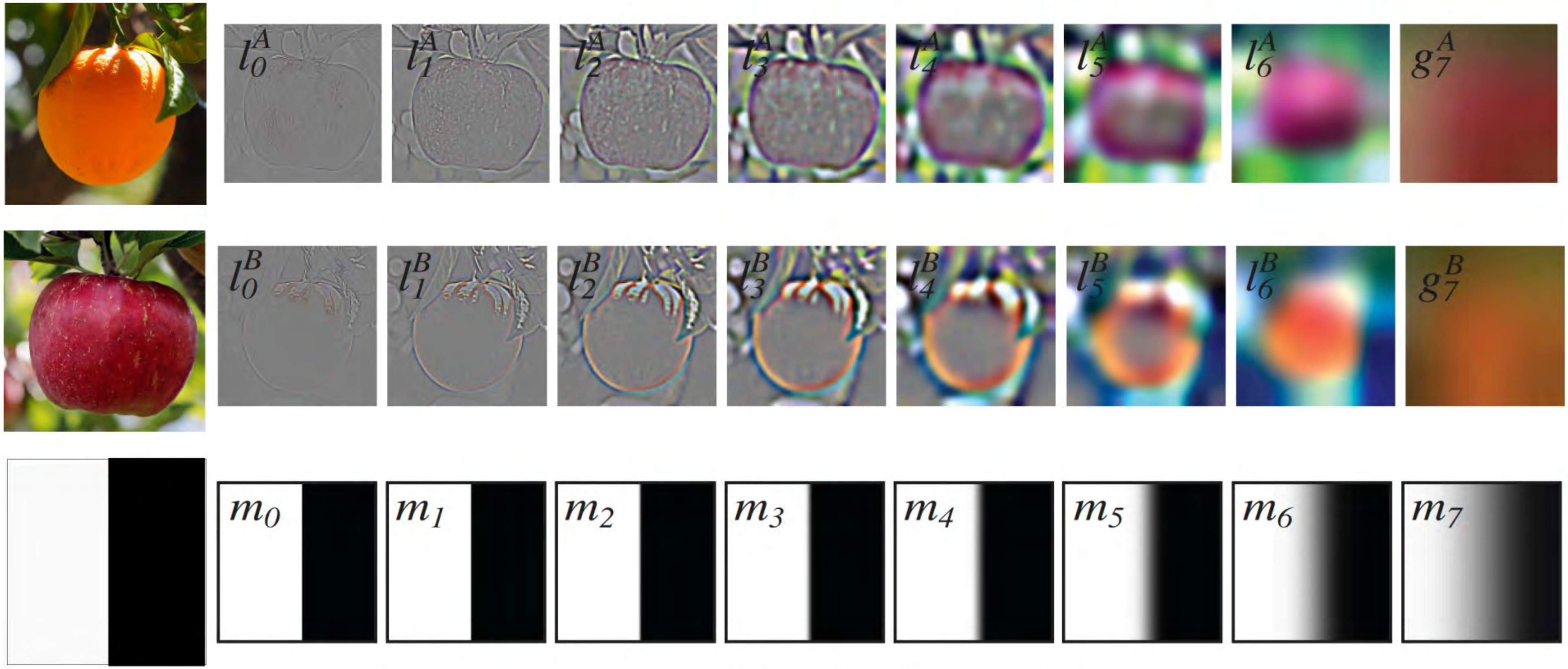


Slide by A. Efros

Good Window Size

"Optimal" Window: smooth but not ghosted

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

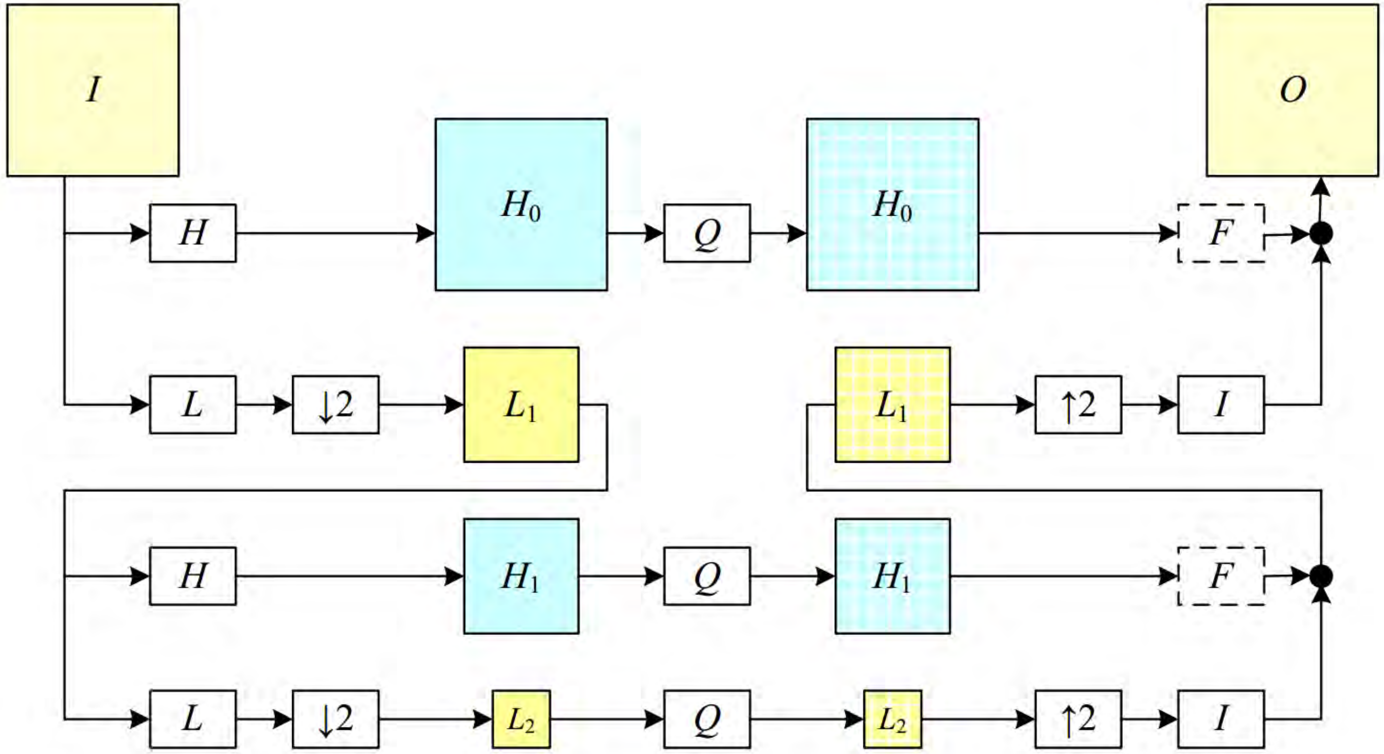
Image pyramids

- Gaussian

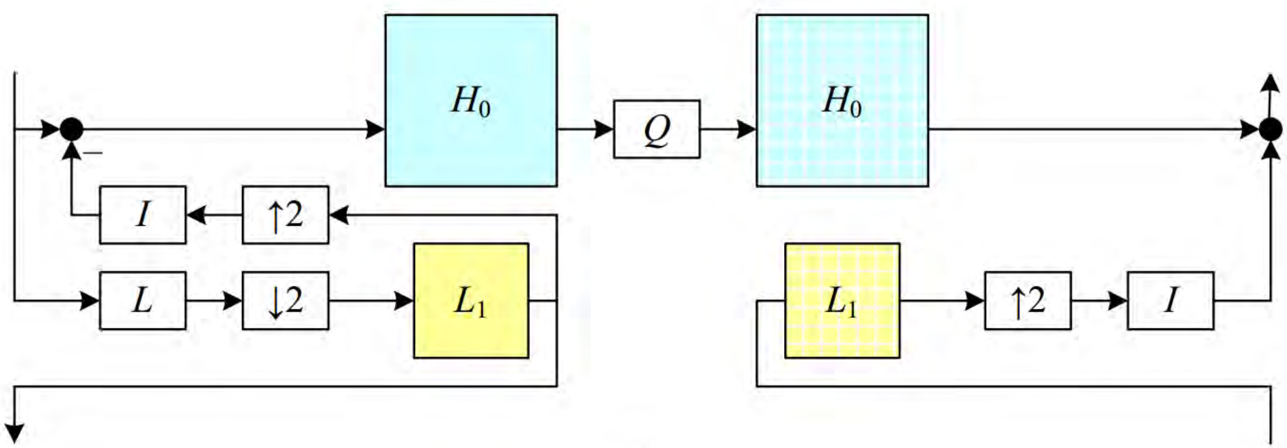
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian

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



(a)



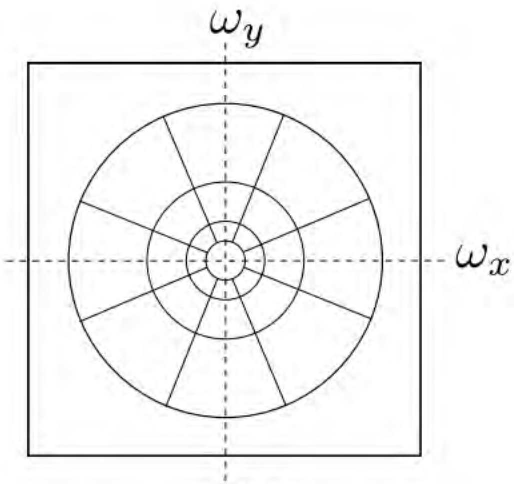
(b)

Today's class

- Template matching
- Gaussian Pyramids
  - Application for recognition
  - Pyramids representation in deep learning
- Laplacian Pyramids
  - Application for image blending
  - Hybrid images
- **Steerable Pyramids**：
  - filter banks and texture analysis

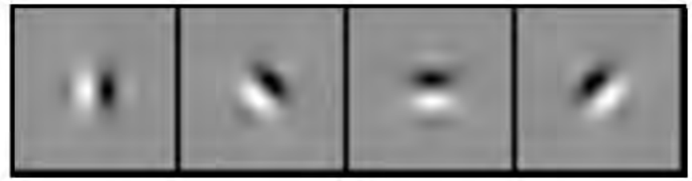
Orientations

Steerable Pyramid

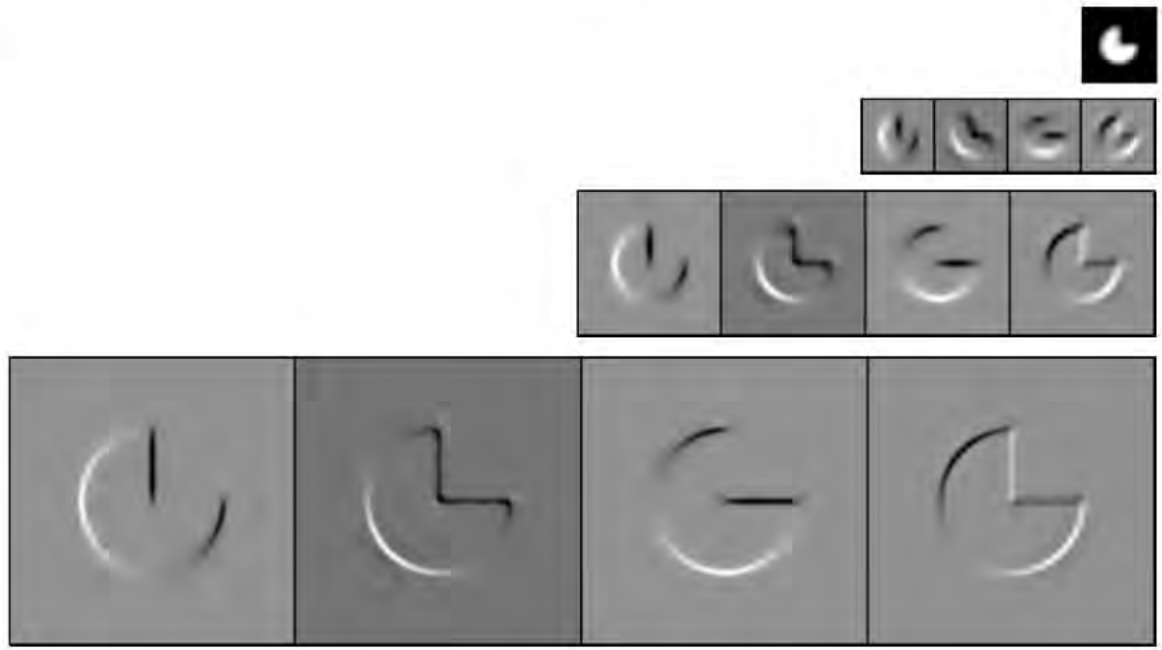


(a)

(b)



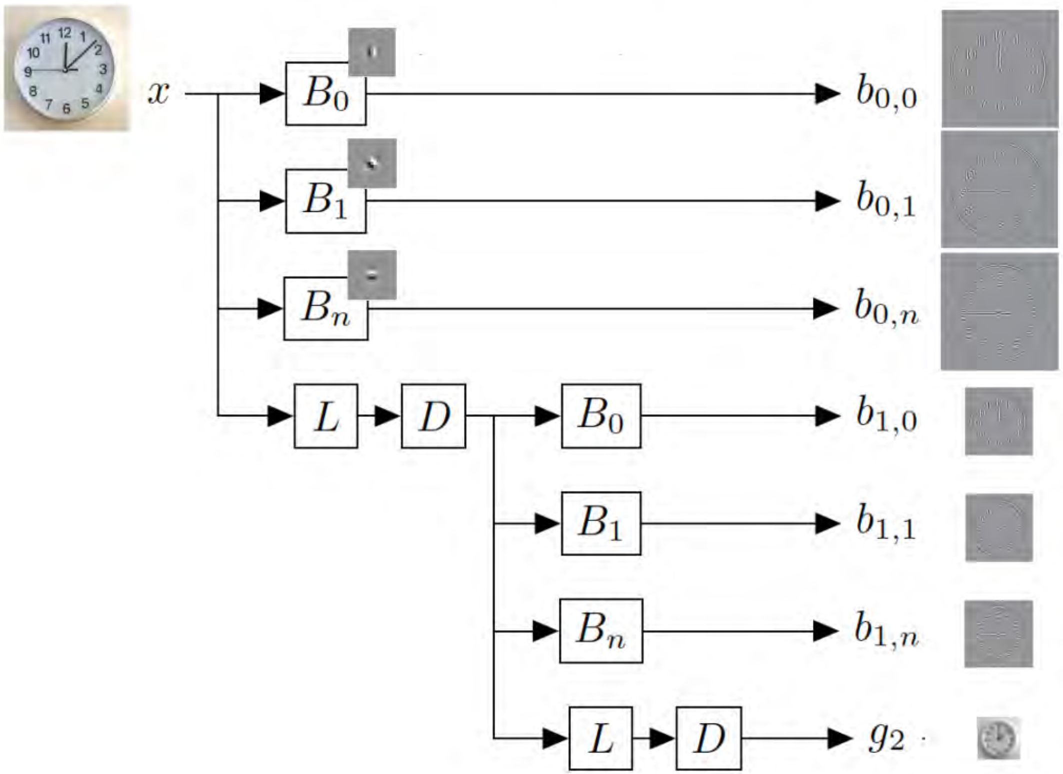
(c)

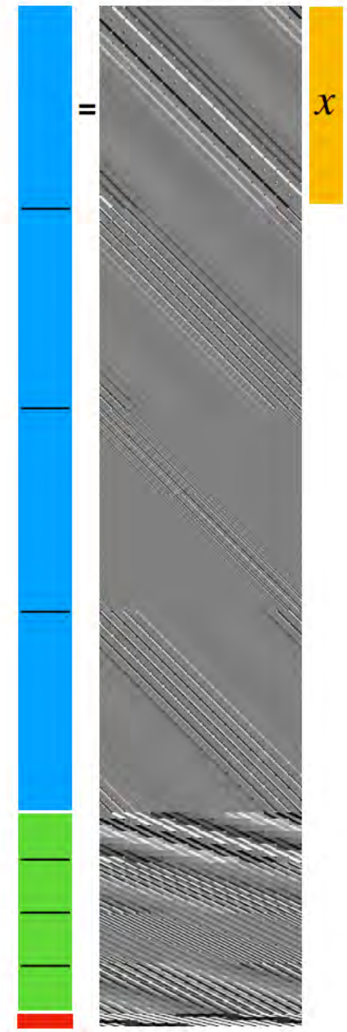


(d)

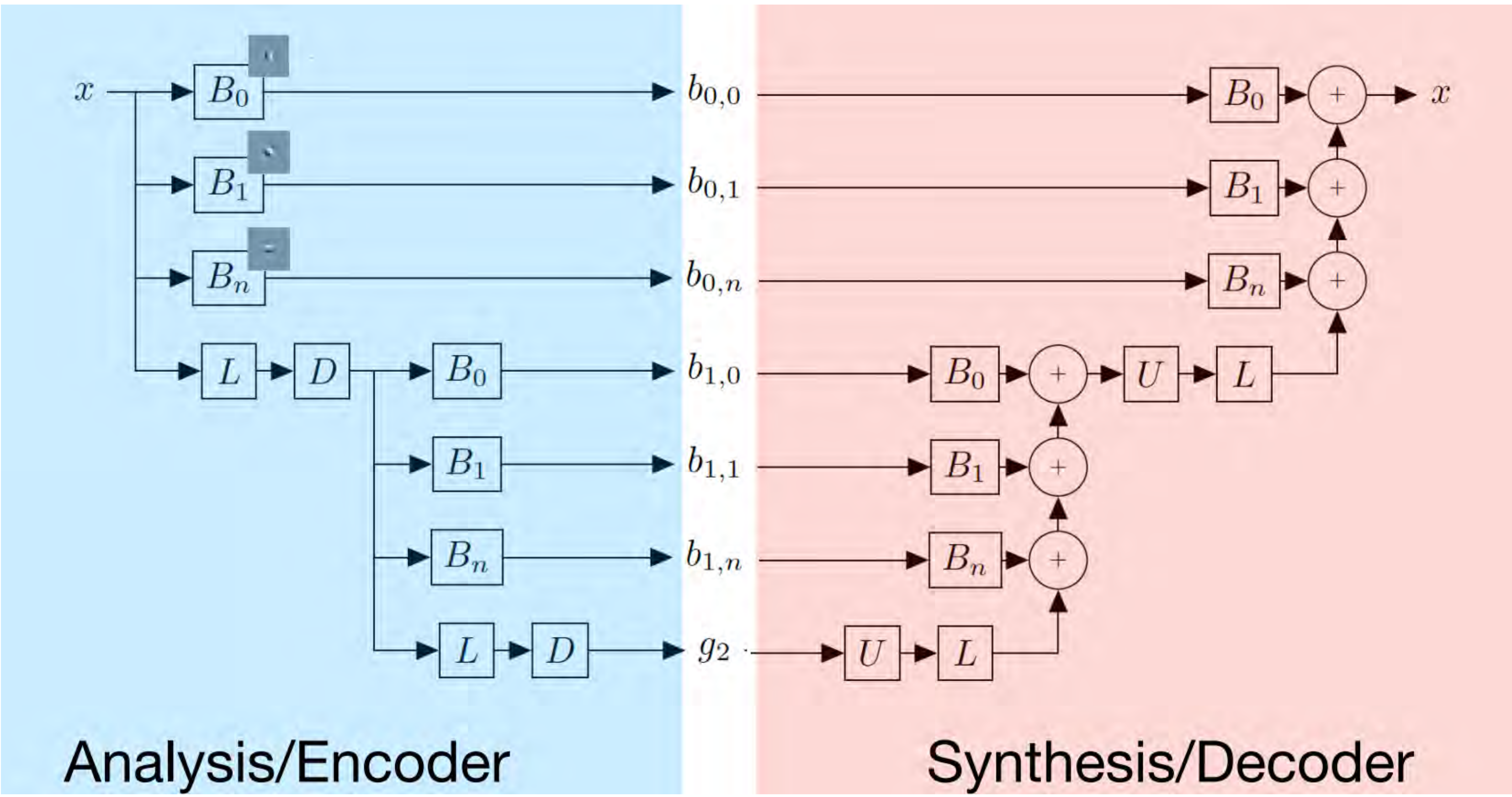
Figure 3.40 Steerable shiftable multiscale transforms (Simoncelli, Freeman, Adelson *et al.* 1992) © 1992 IEEE: (a) radial multi-scale frequency domain decomposition; (b) original image; (c) a set of four steerable filters; (d) the radial multi-scale wavelet decomposition.

Steerable Pyramid



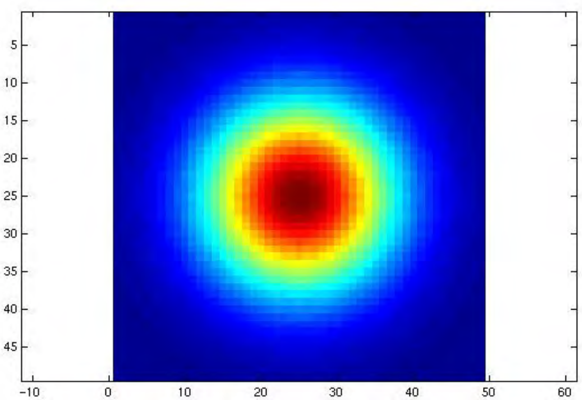


Steerable Pyramid

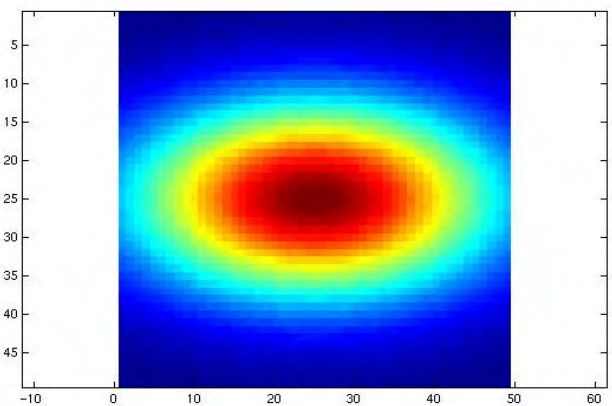


Multivariate Gaussian

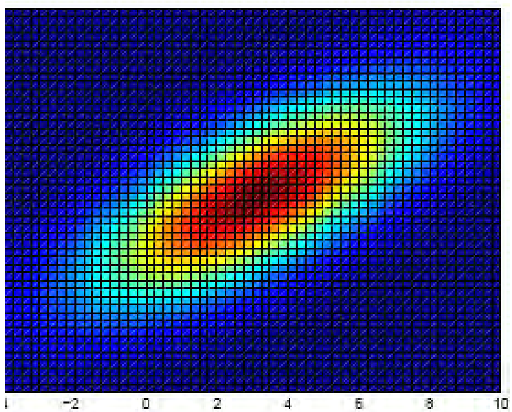
$$p(x;\mu,\Sigma)=\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$



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



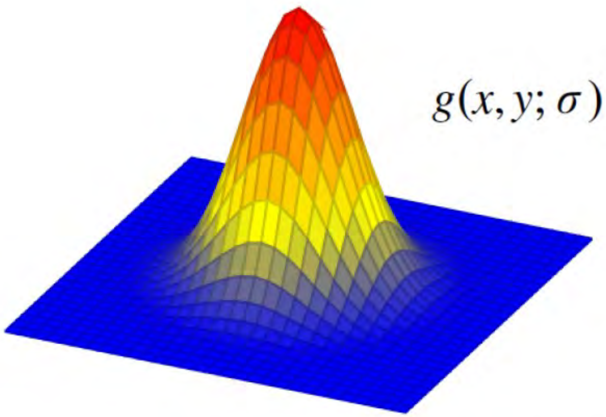
$$\Sigma=\begin{bmatrix}16 & 0\\ 0 & 9\end{bmatrix}$$



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

Slide credit:
Kristen Grauman

Gaussian Derivative



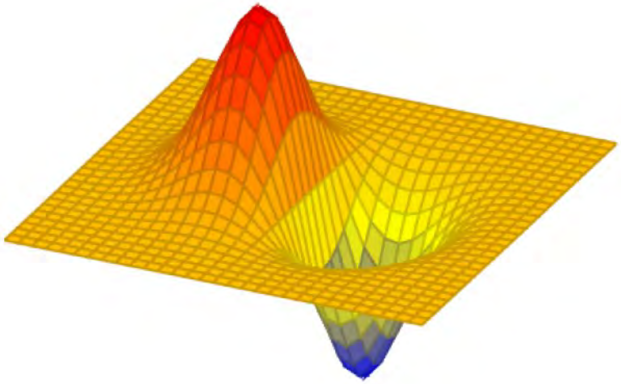
$$g(x,y;\sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2+y^2}{2\sigma^2}$$

The continuous derivative is:

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

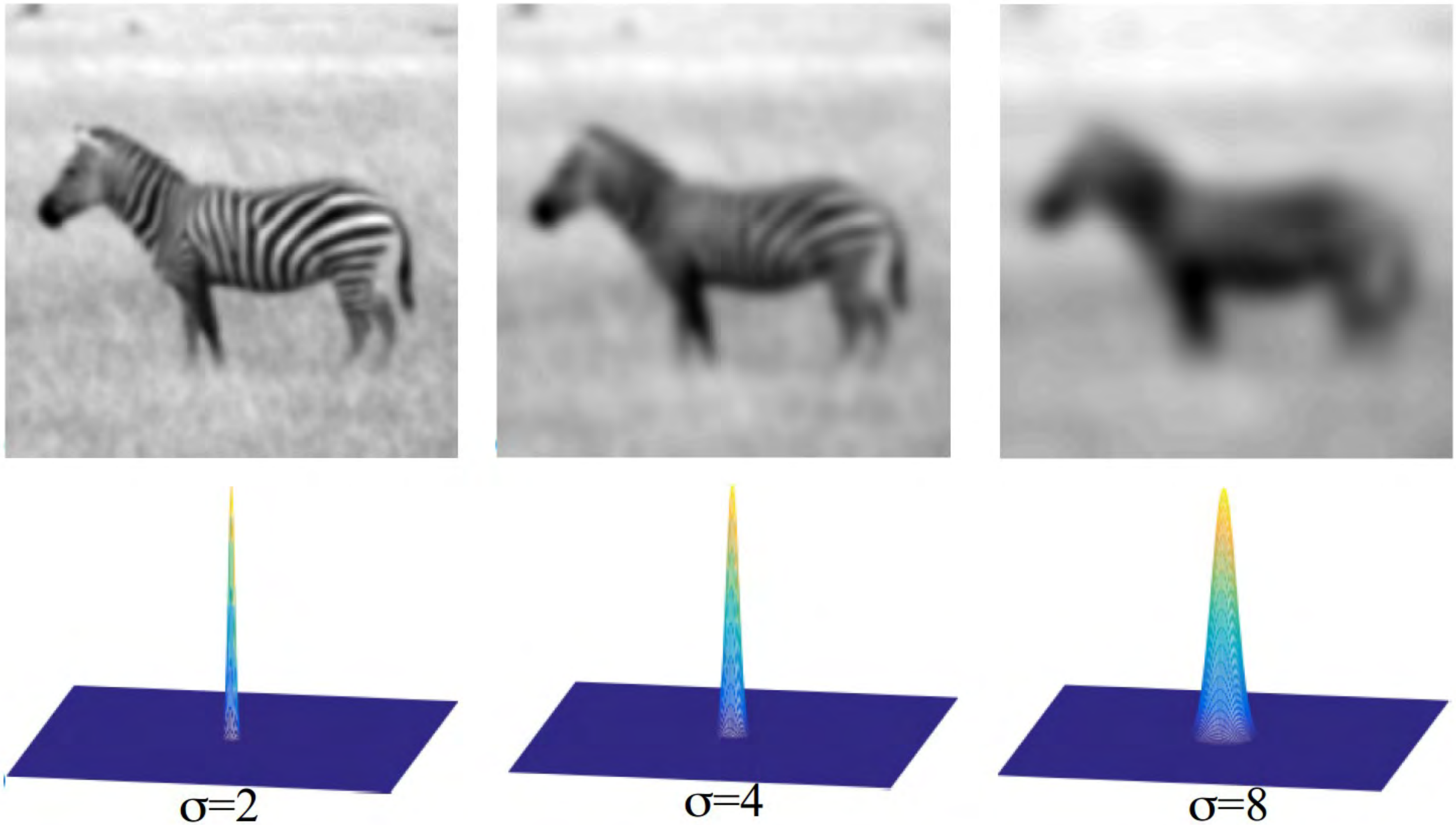
$$= \frac{-x}{2\pi\sigma^4} \exp -\frac{x^2+y^2}{2\sigma^2}$$

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

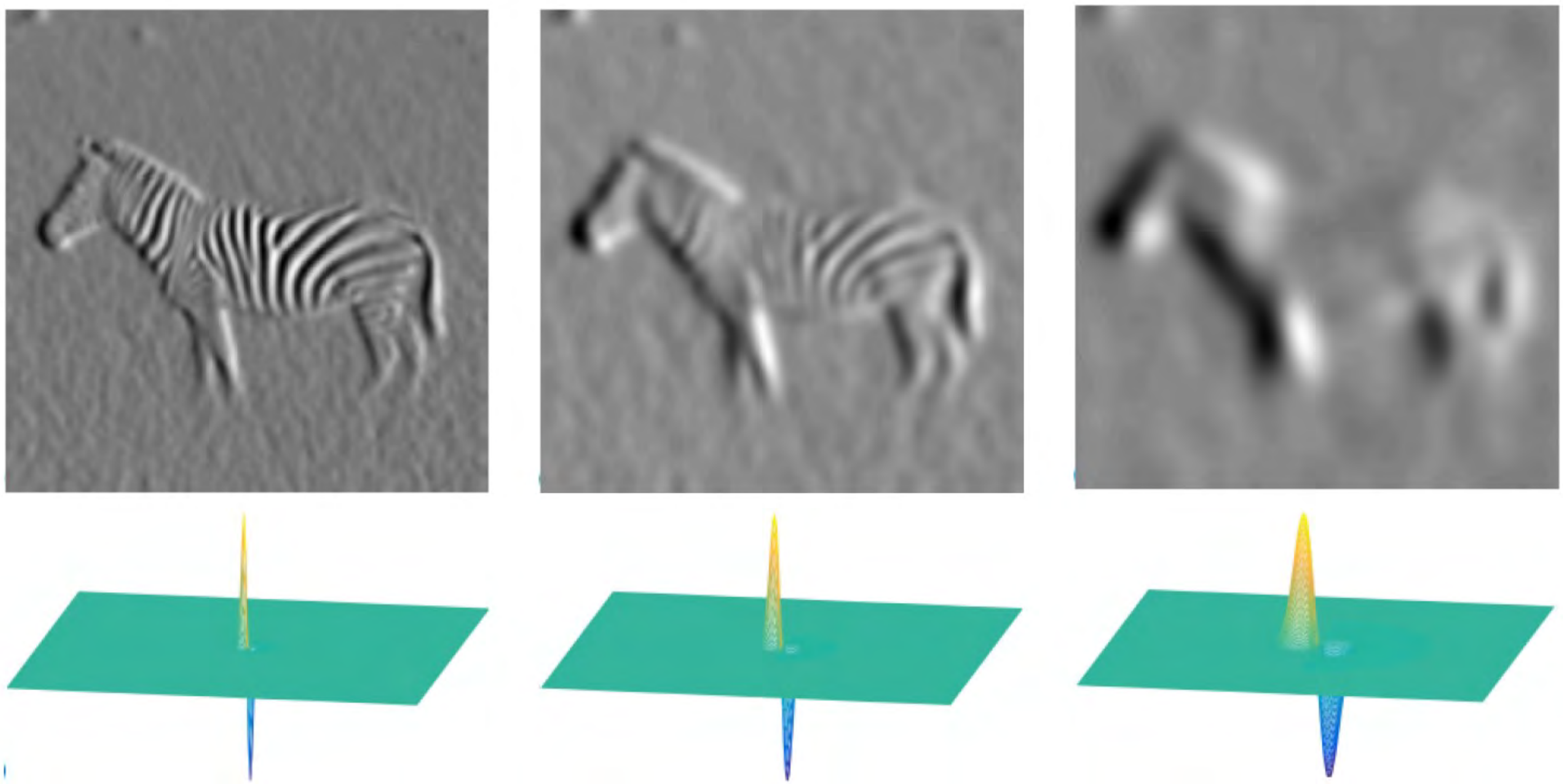


104

Gaussian Scale



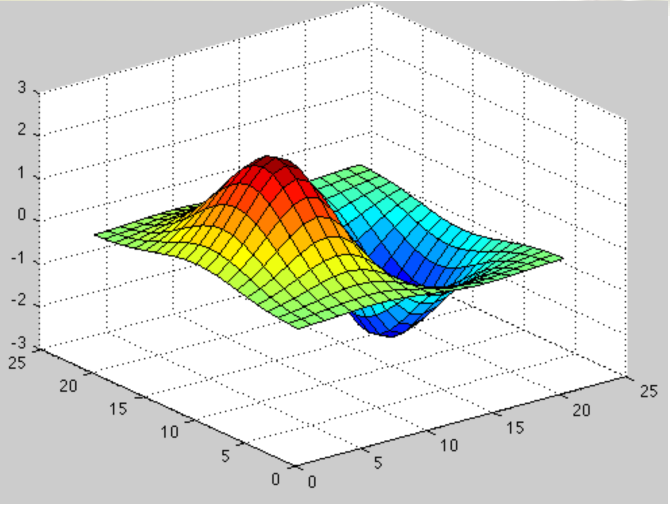
Derivatives of Gaussian: Scale



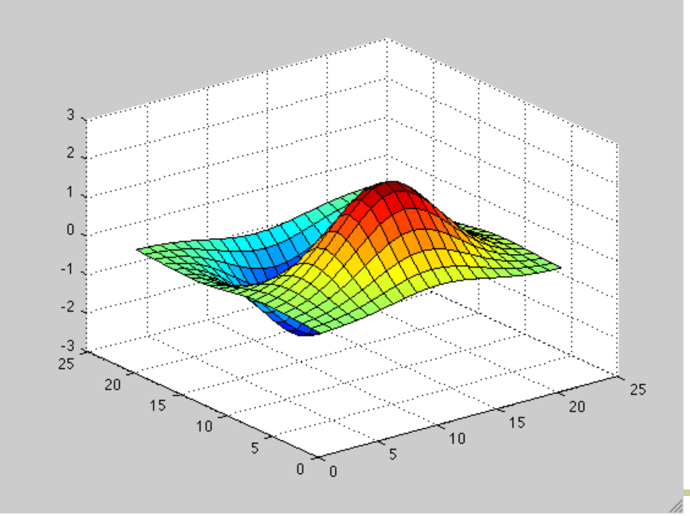
σ=2 σ=4 σ=8

Orientation

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

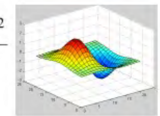


$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

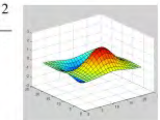


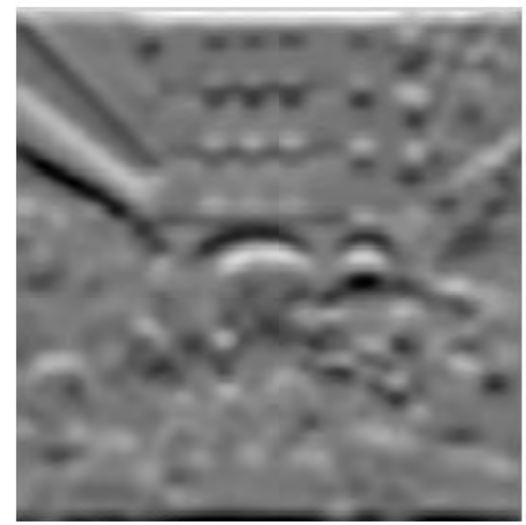
Orientation

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



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

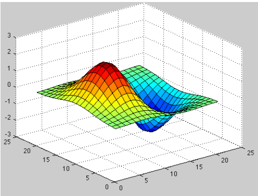




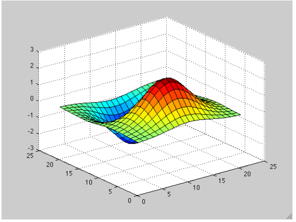
What about other orientations not axis aligned?

Orientation

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



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



The smoothed directional gradient is a linear combination of two kernels

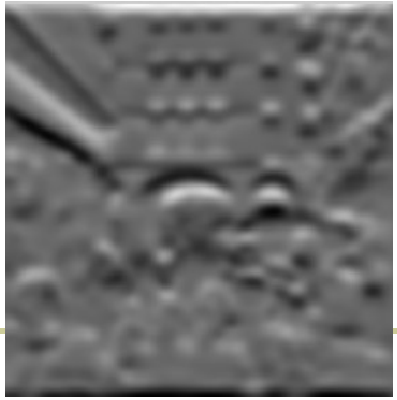
$$u^T \nabla g \otimes I = \left(\cos(\alpha) g_x(x,y) + \sin(\alpha) g_y(x,y)\right) \otimes I(x,y) =$$

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(α)

+sin(α)



=

Steereability of gaussian derivatives, Freeman & Adelson 92

Simple example of steerable filter

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

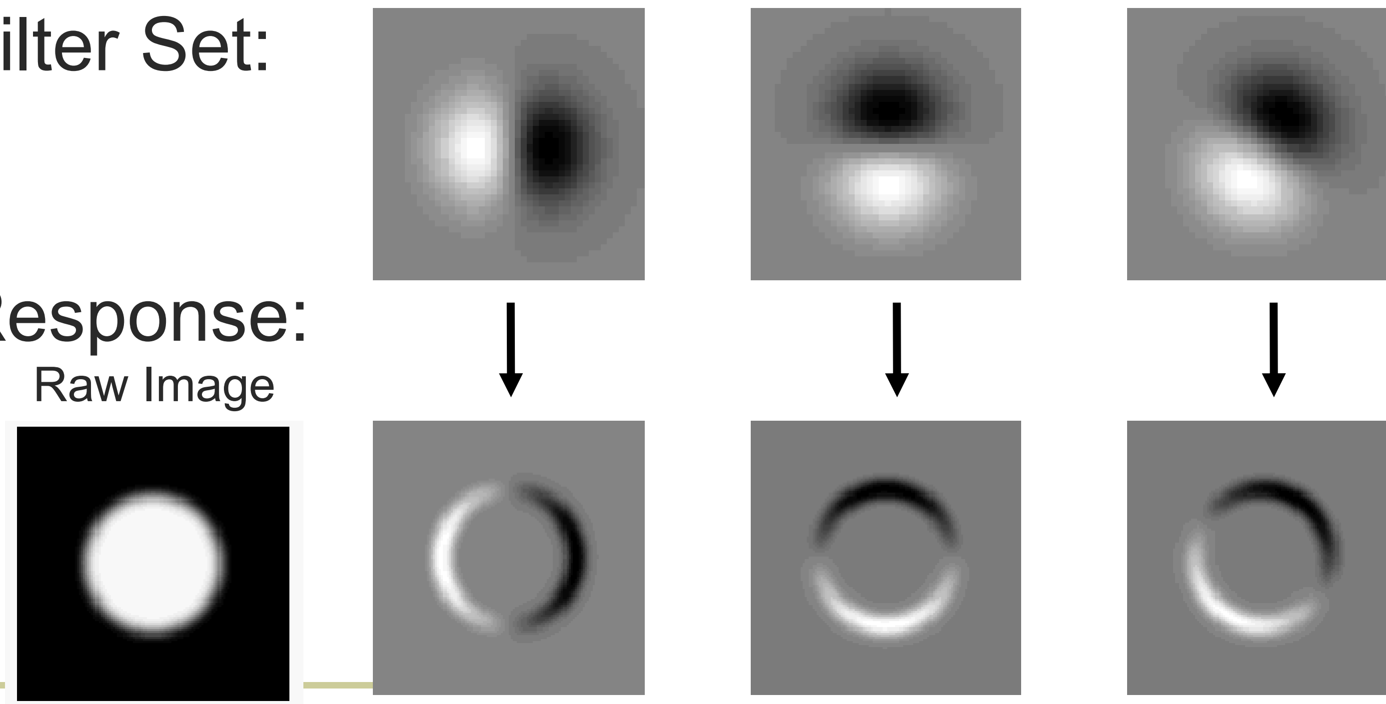
$$G_\theta^1 = \cos(\theta) G_0^1 + \sin(\theta) G_{90}^1$$

0^o | 90^o | Synthesized 30^o

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

Orientation analysis

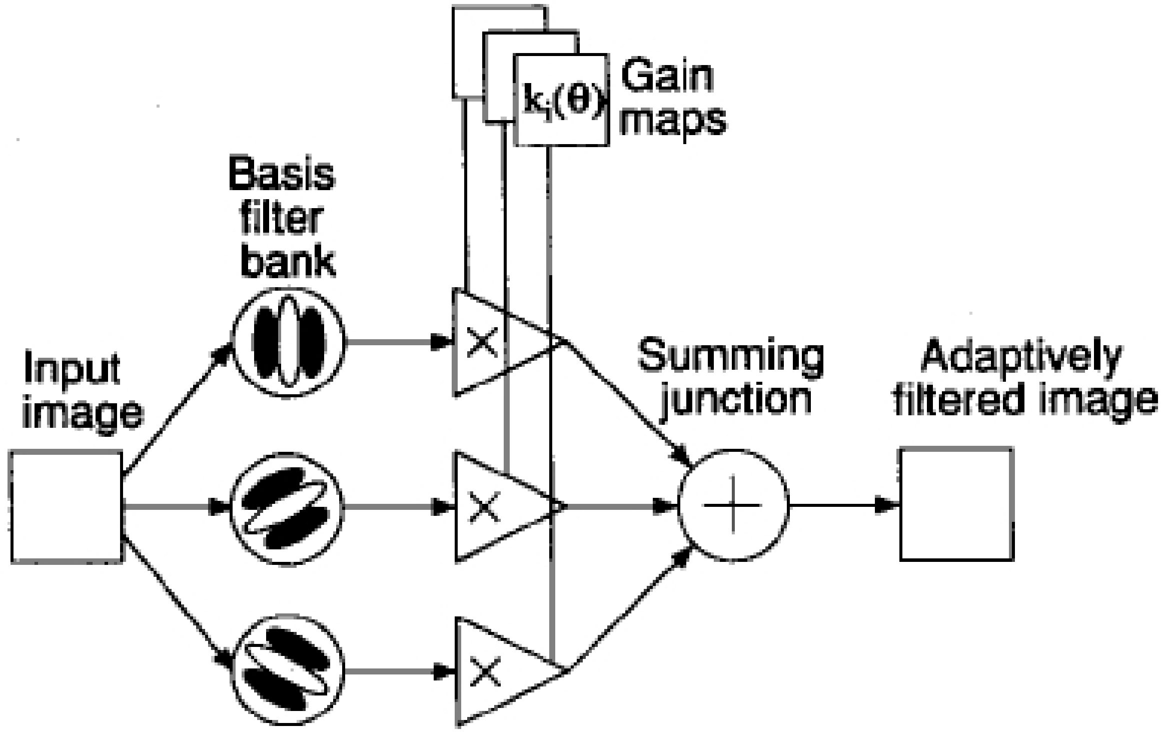
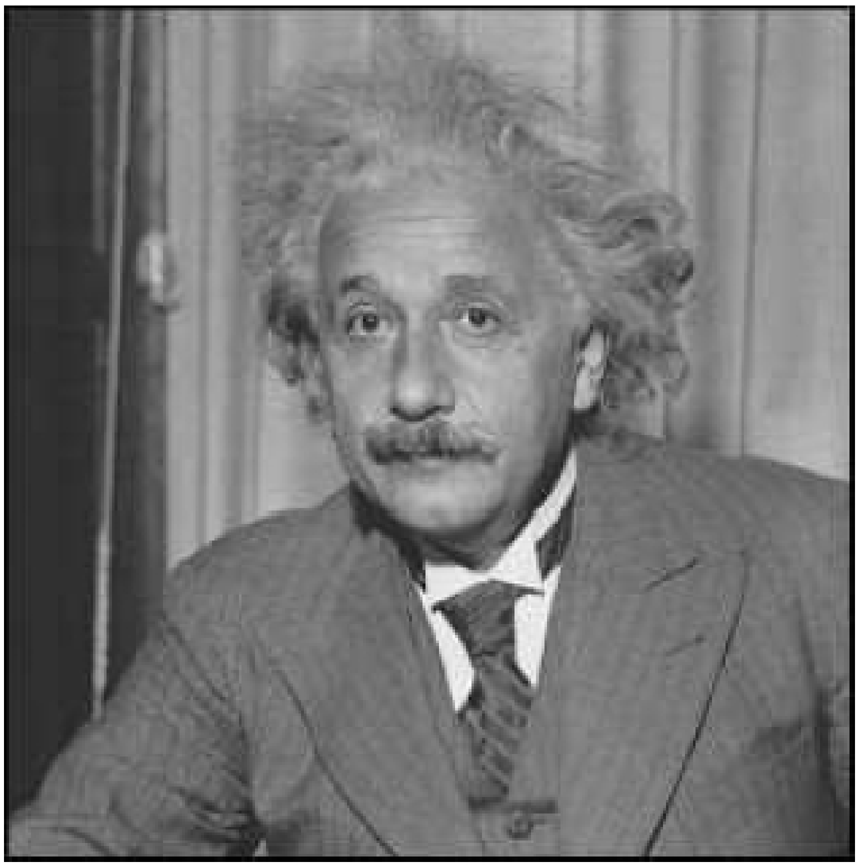


Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.

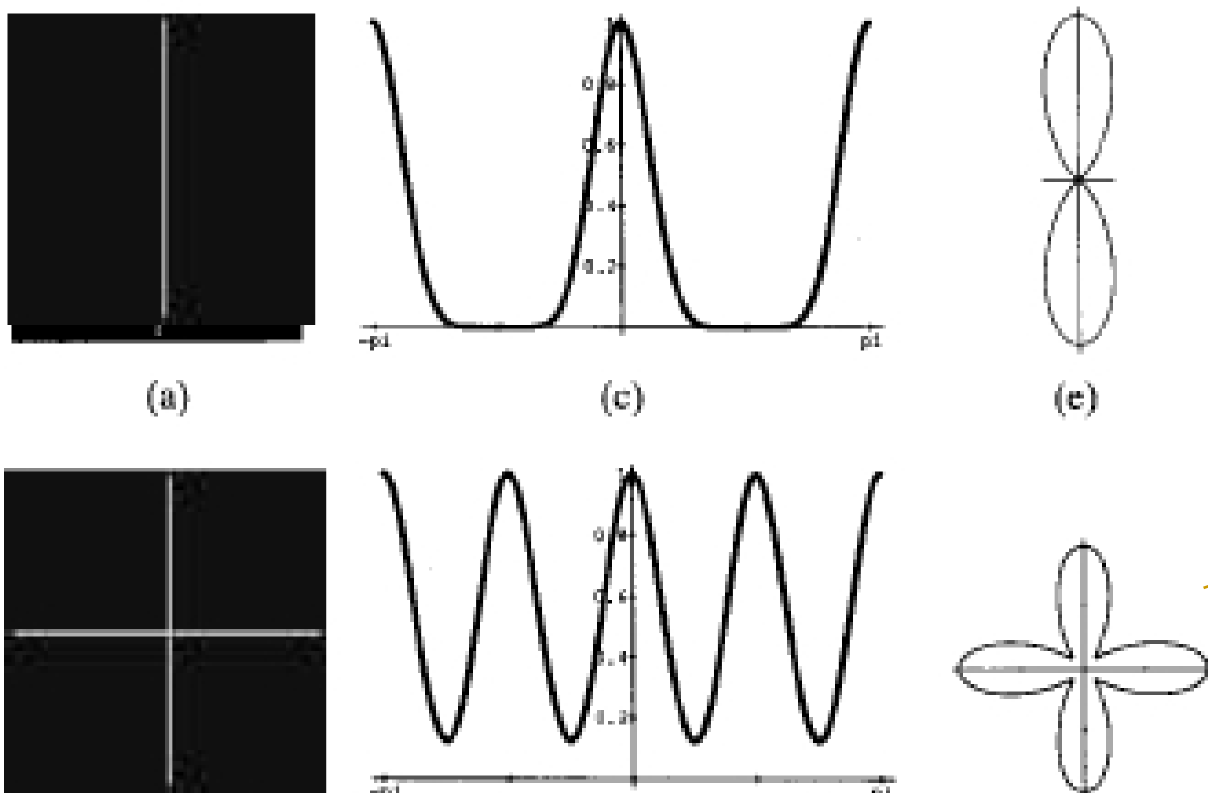
Orientation analysis



(a)

(b)

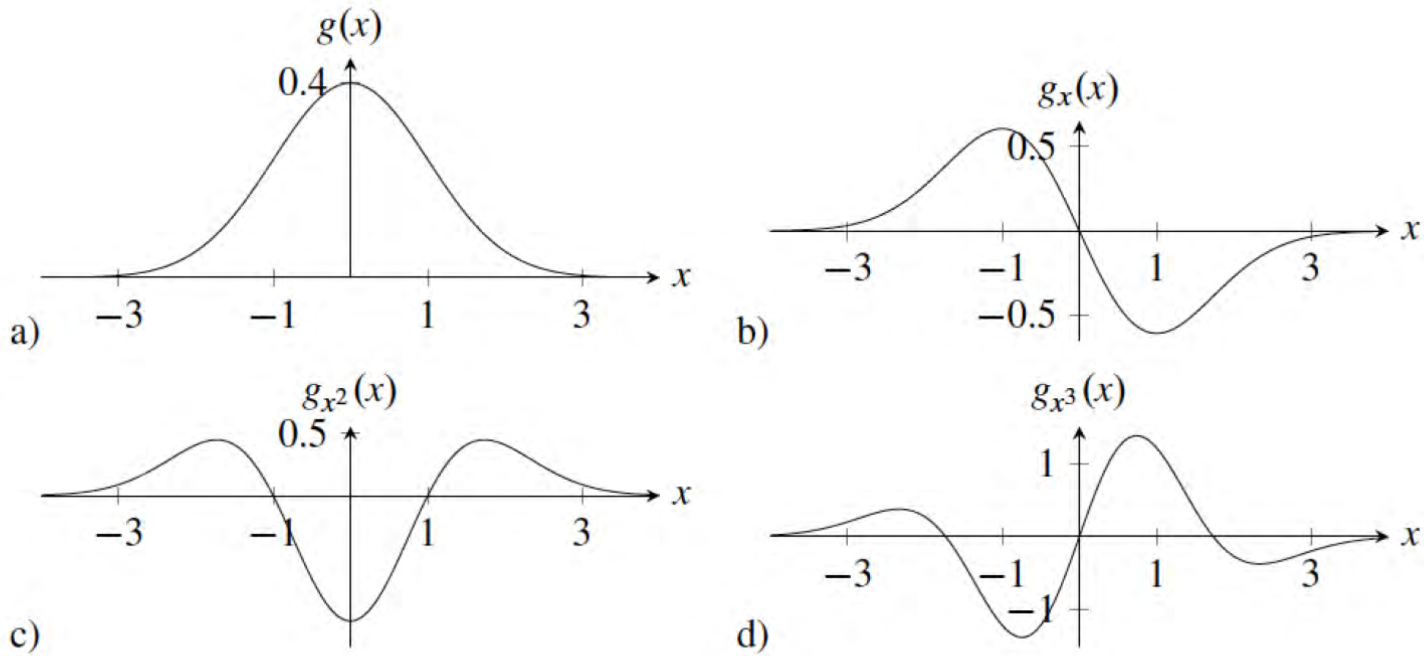
Orientation analysis



(a) (c) (e)
(b) (d) (f)

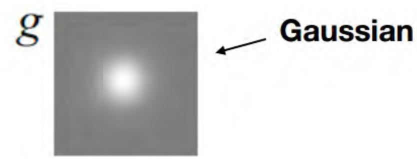
High resolution in orientation requires many oriented filters as basis (high order gaussian derivatives or fine-tuned Gabor wavelets).

N-th order Gaussian derivatives



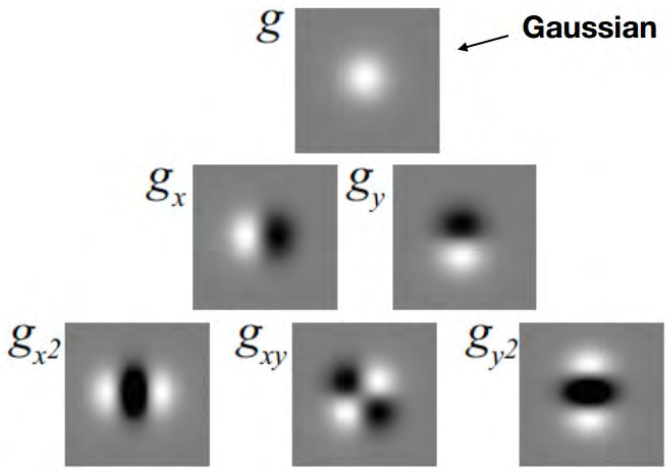
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

N-th order Gaussian derivatives



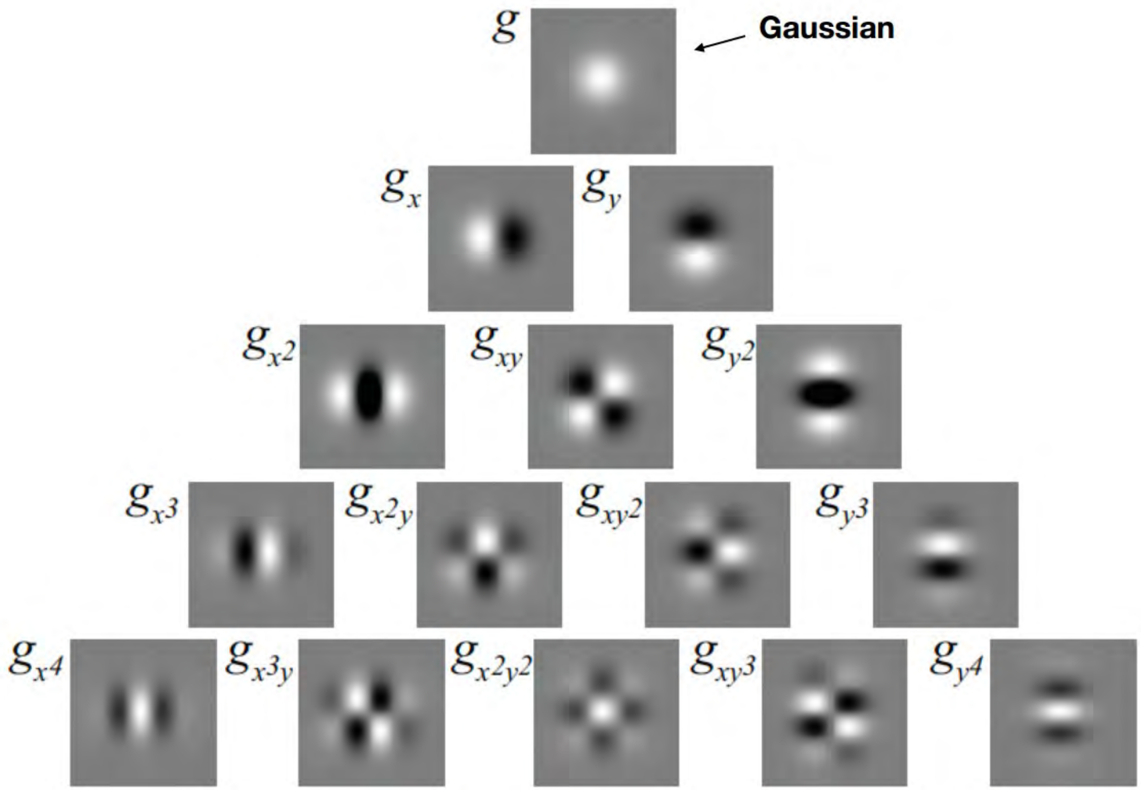
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

N-th order Gaussian derivatives



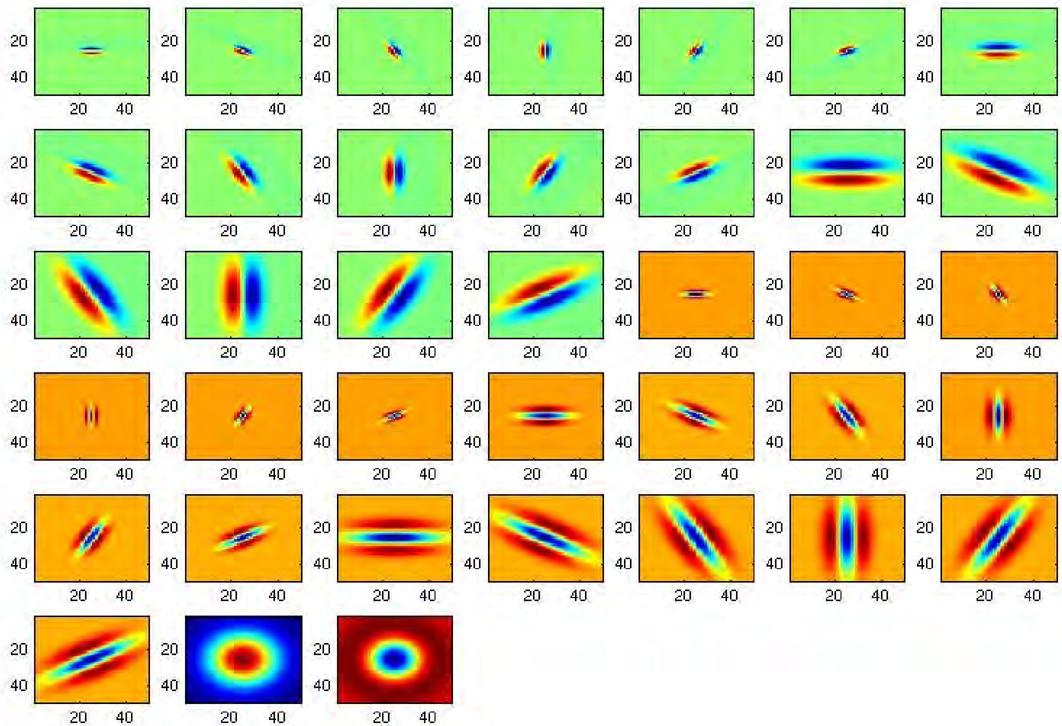
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

N-th order Gaussian derivatives



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

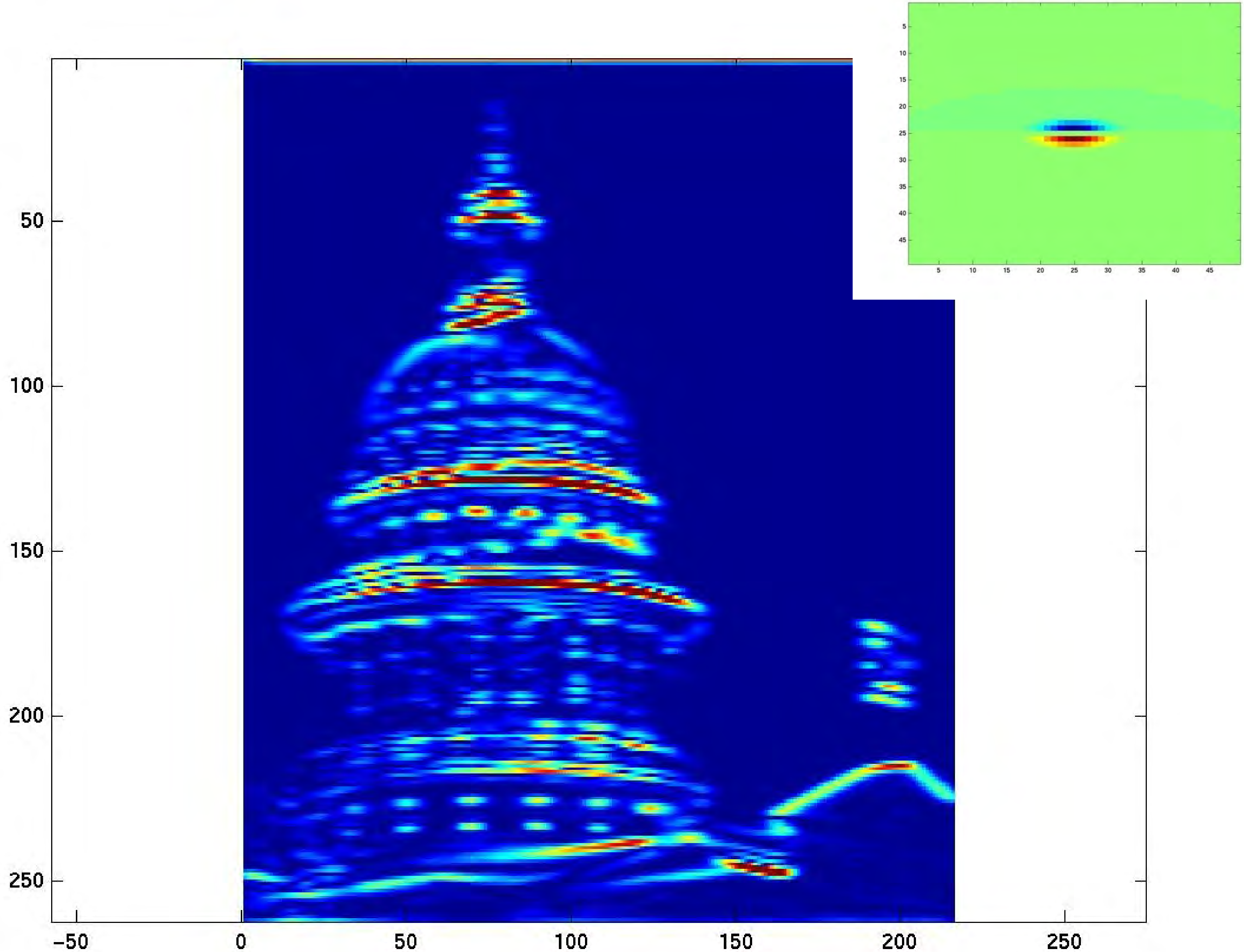
Filter bank



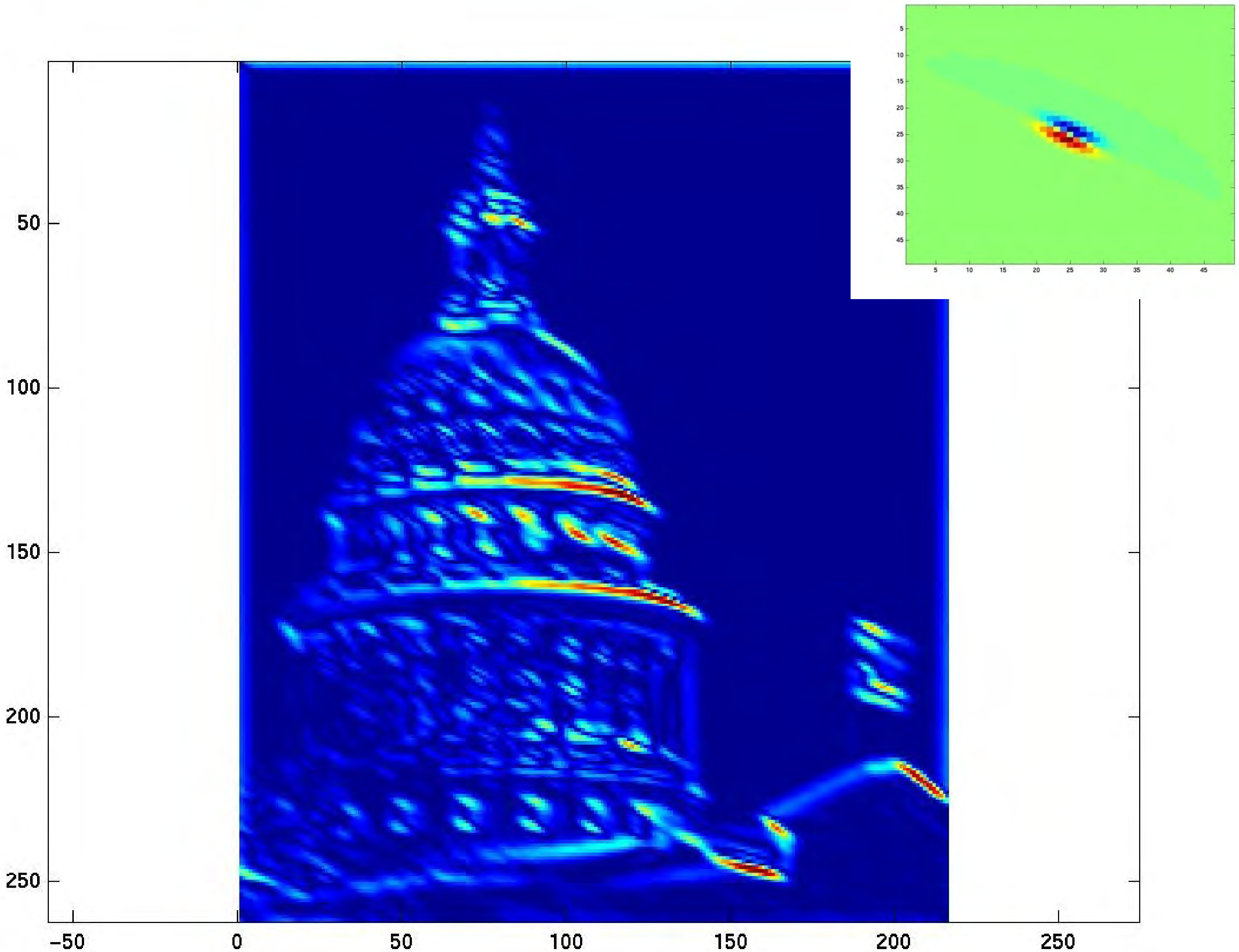
Slide credit:
Kristen Grauman

Image from http://www.texasexplorer.com/austincap2.jpg

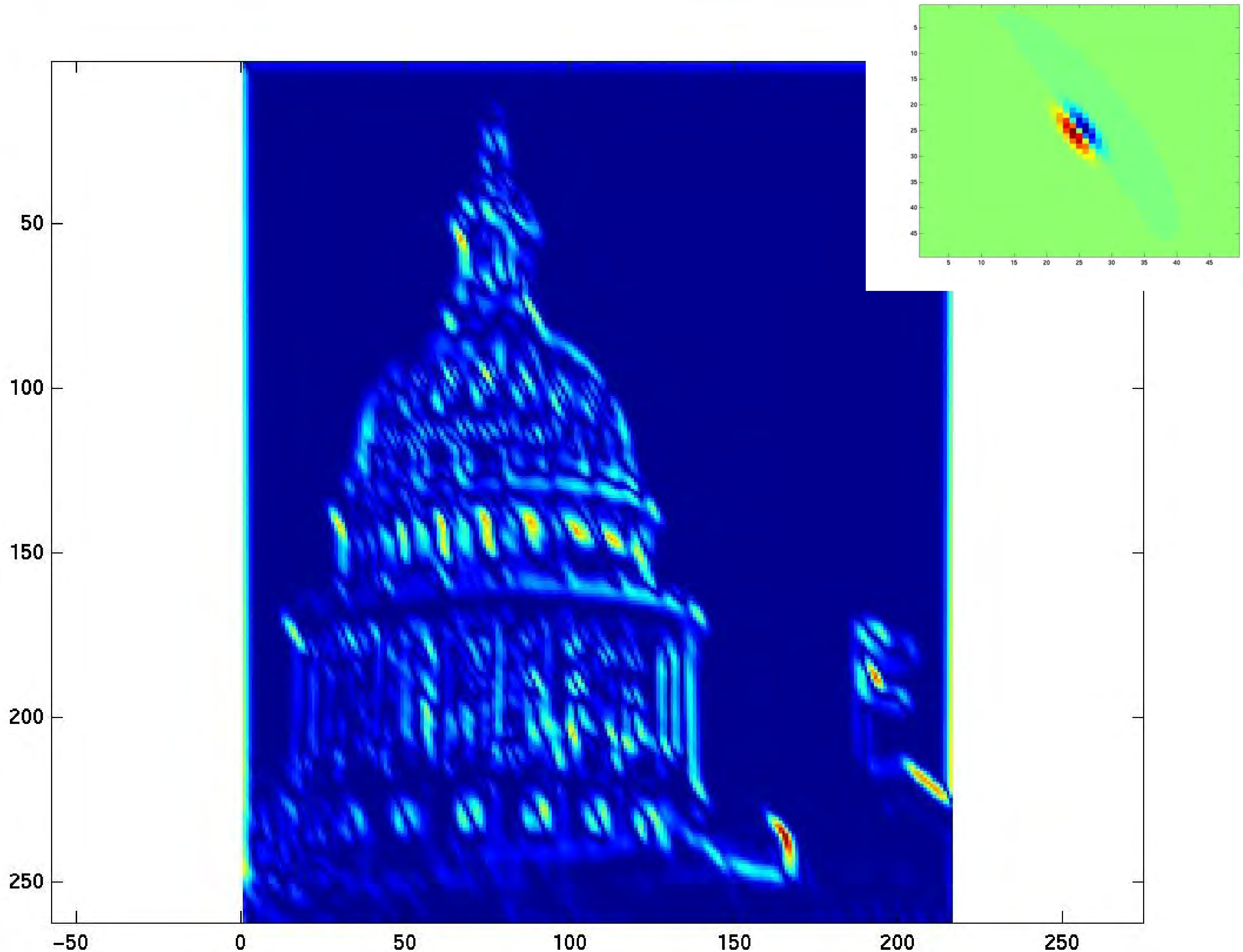
Slide credit:
Kristen Grauman



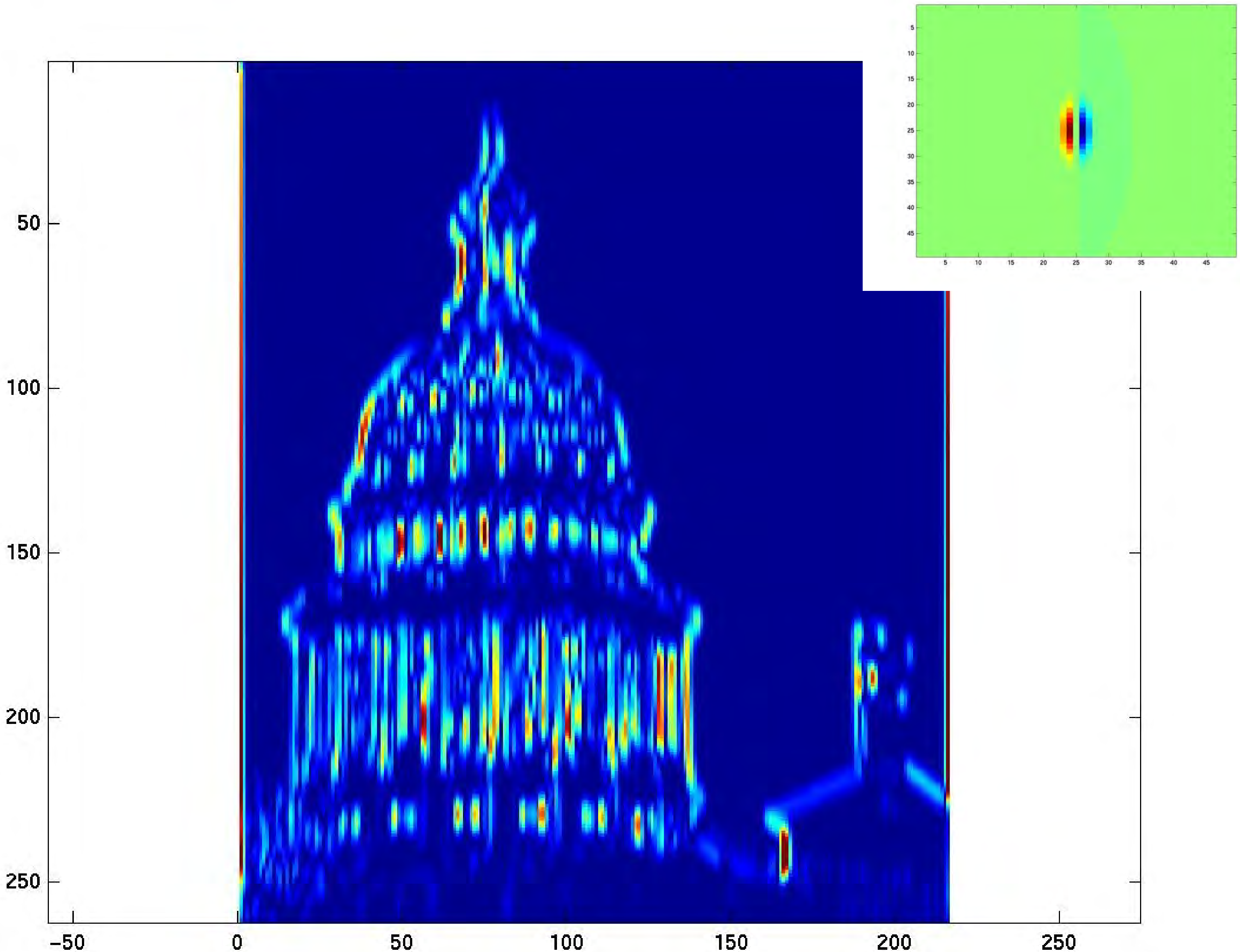
Slide credit:
Kristen Grauman



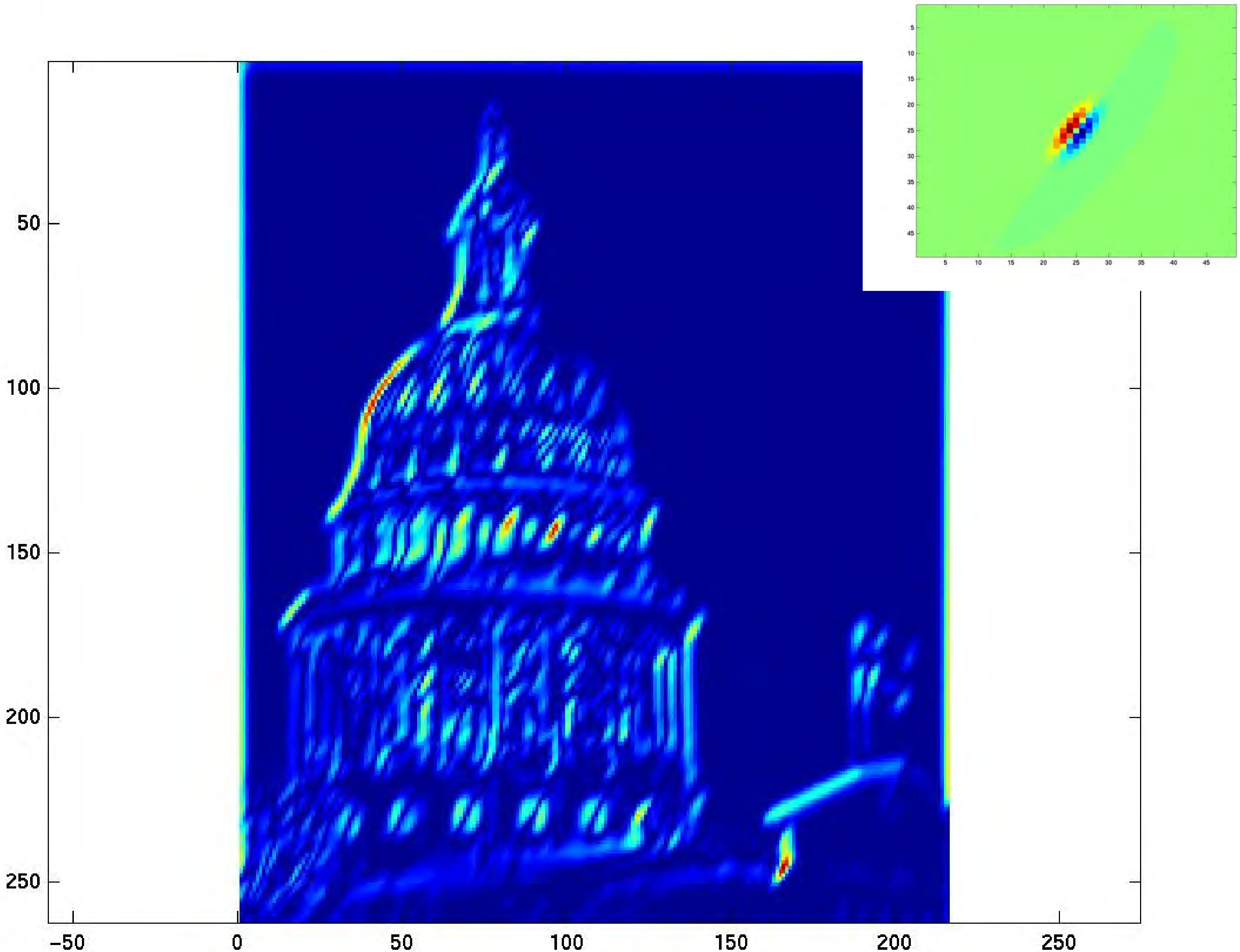
Slide credit:
Kristen Grauman



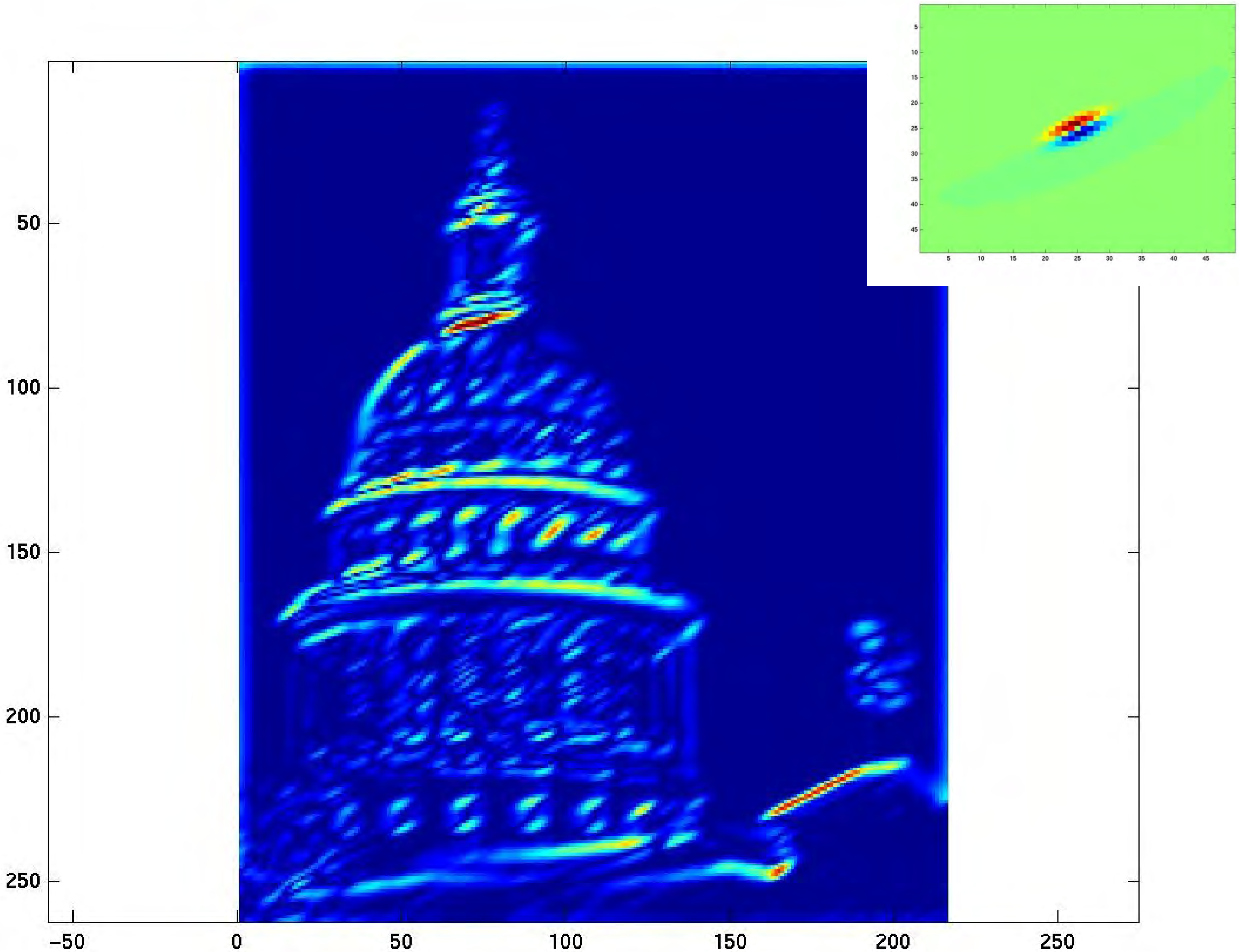
Slide credit:
Kristen Grauman



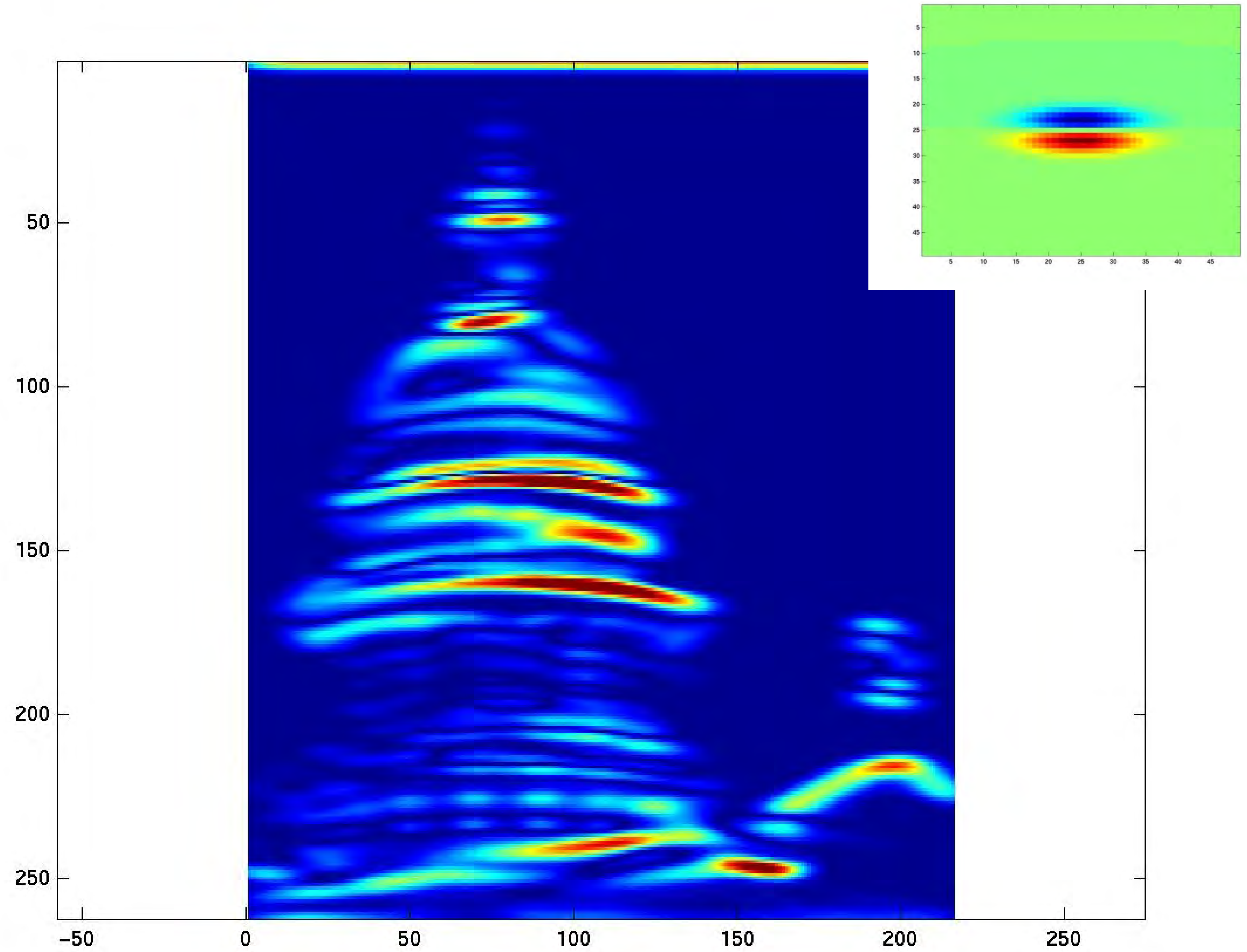
Slide credit:
Kristen Grauman



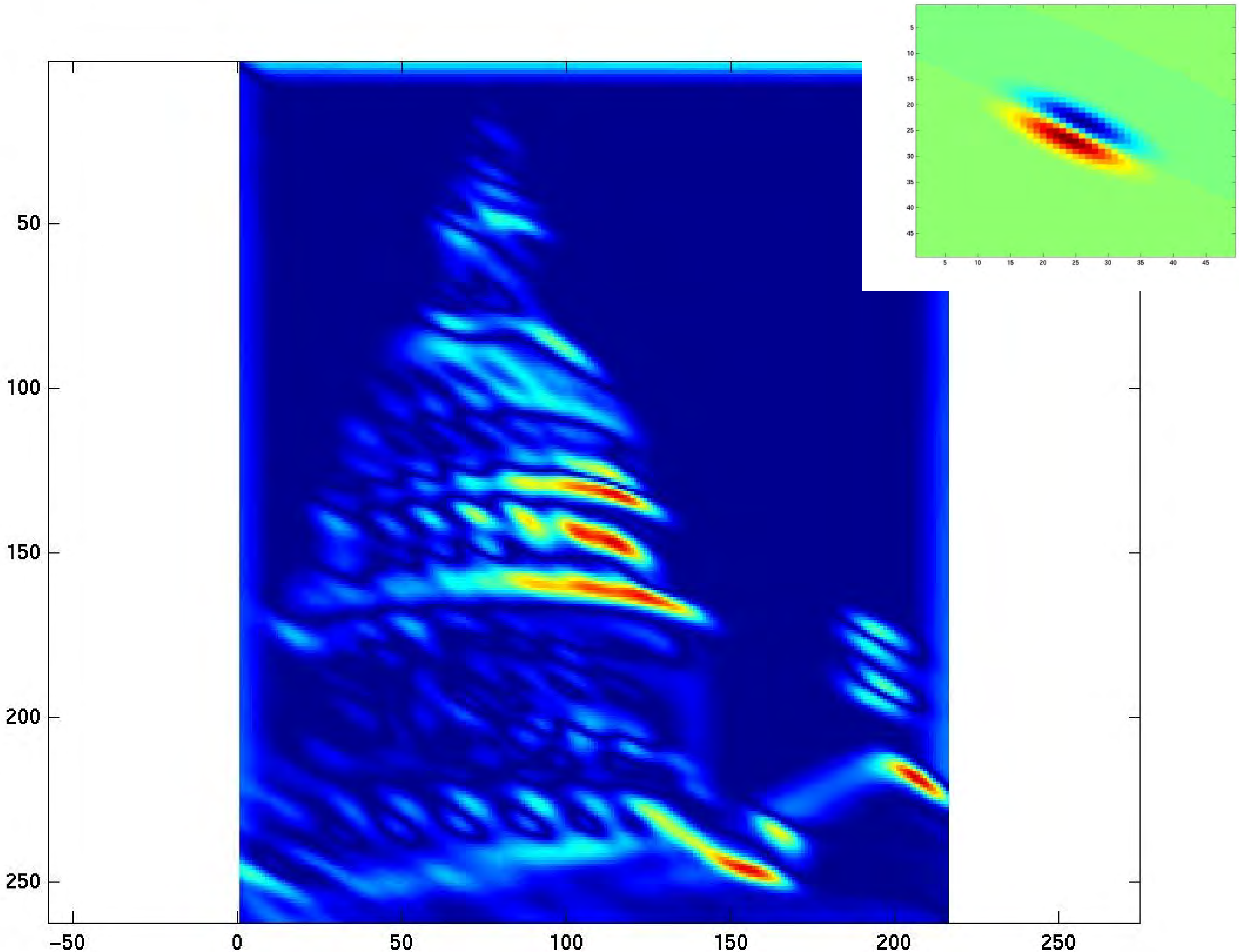
Slide credit:
Kristen Grauman



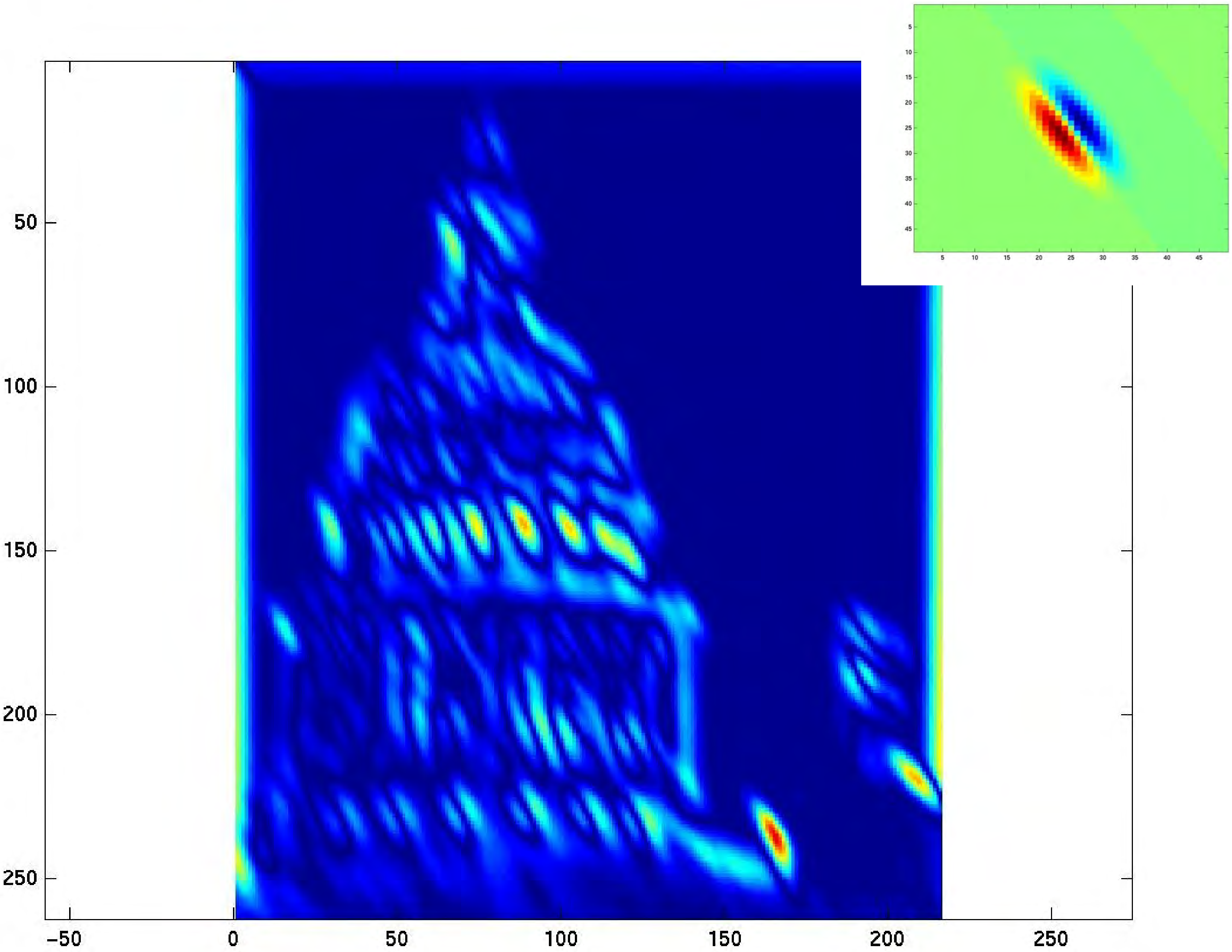
Slide credit:
Kristen Grauman



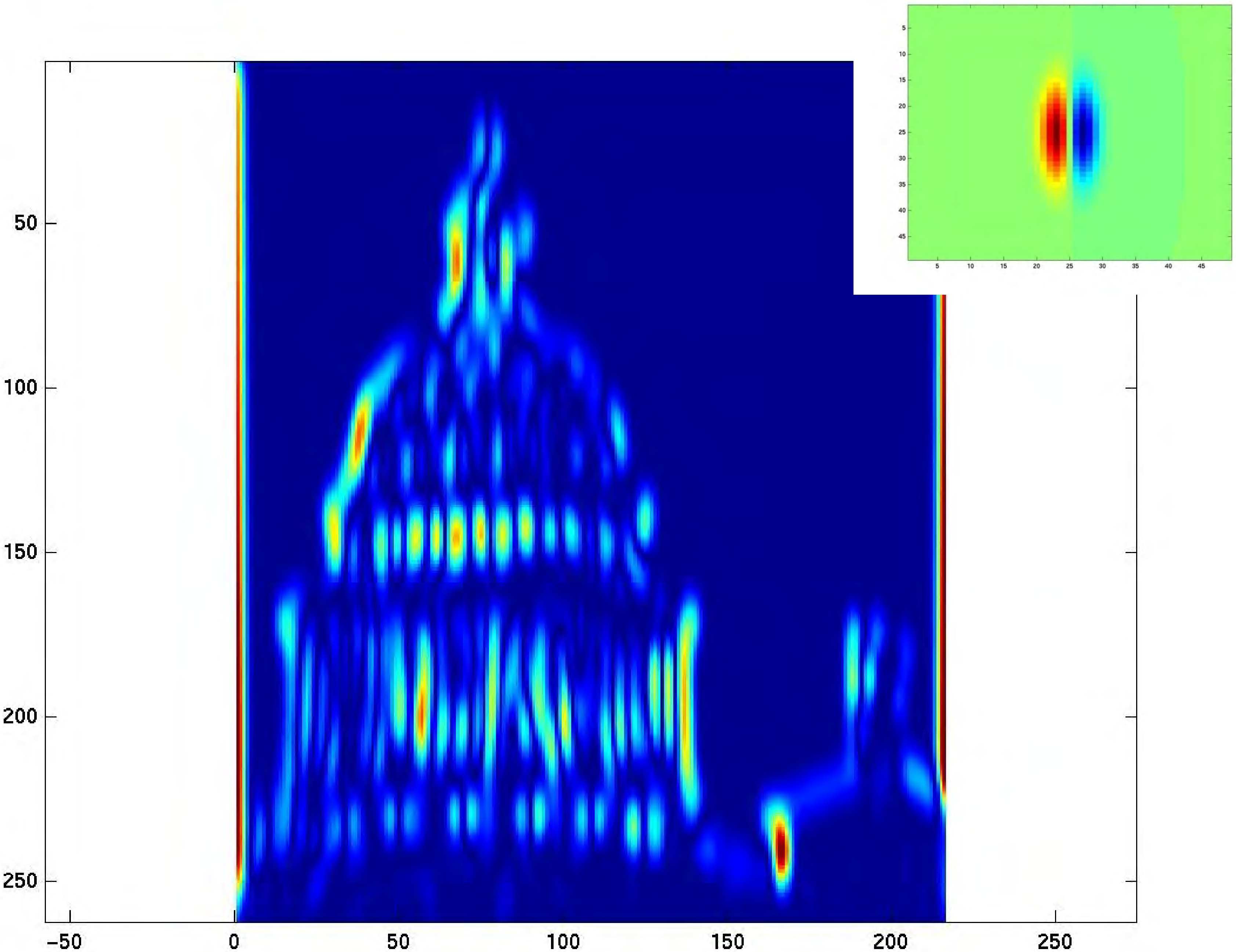
Slide credit:
Kristen Grauman



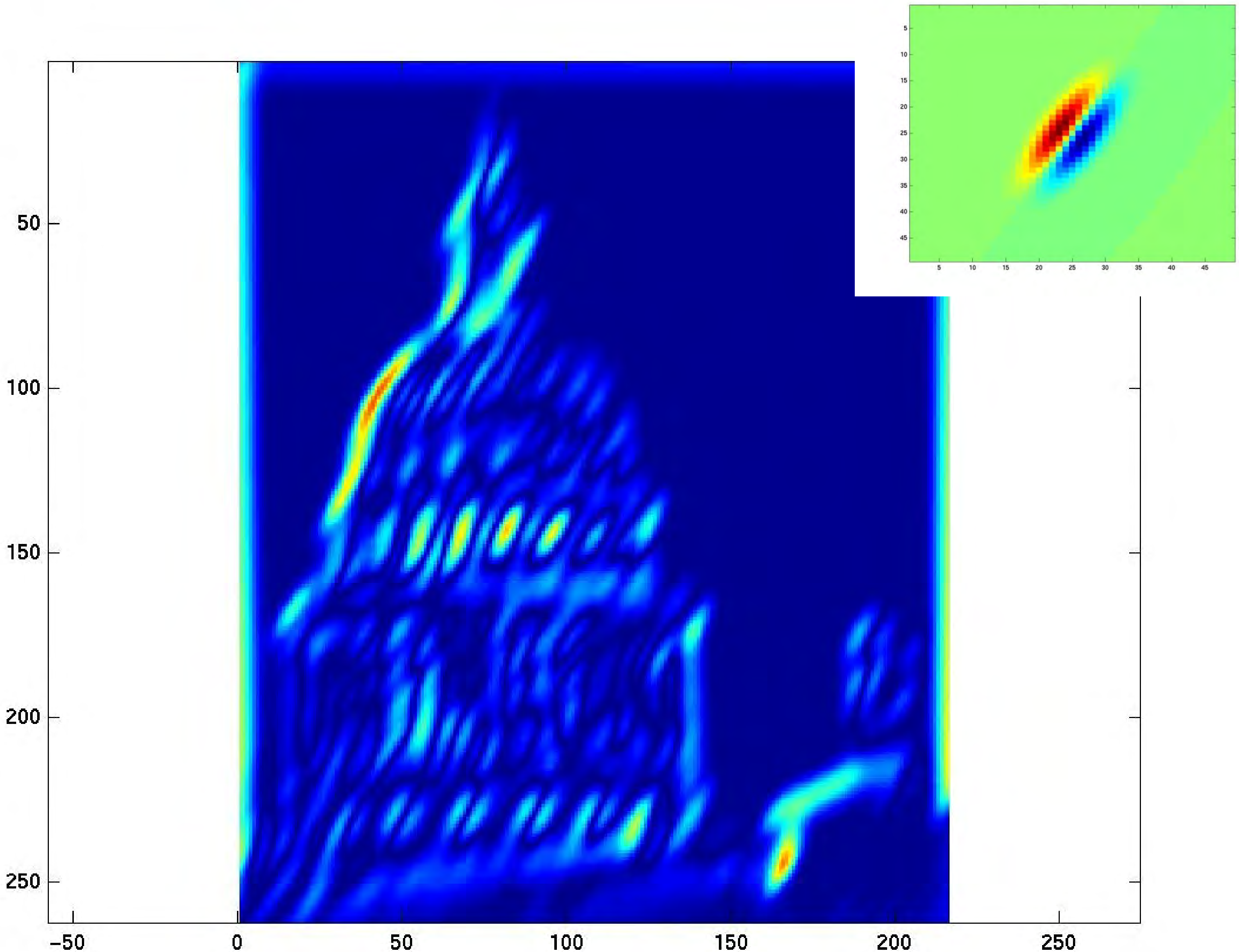
Slide credit:
Kristen Grauman



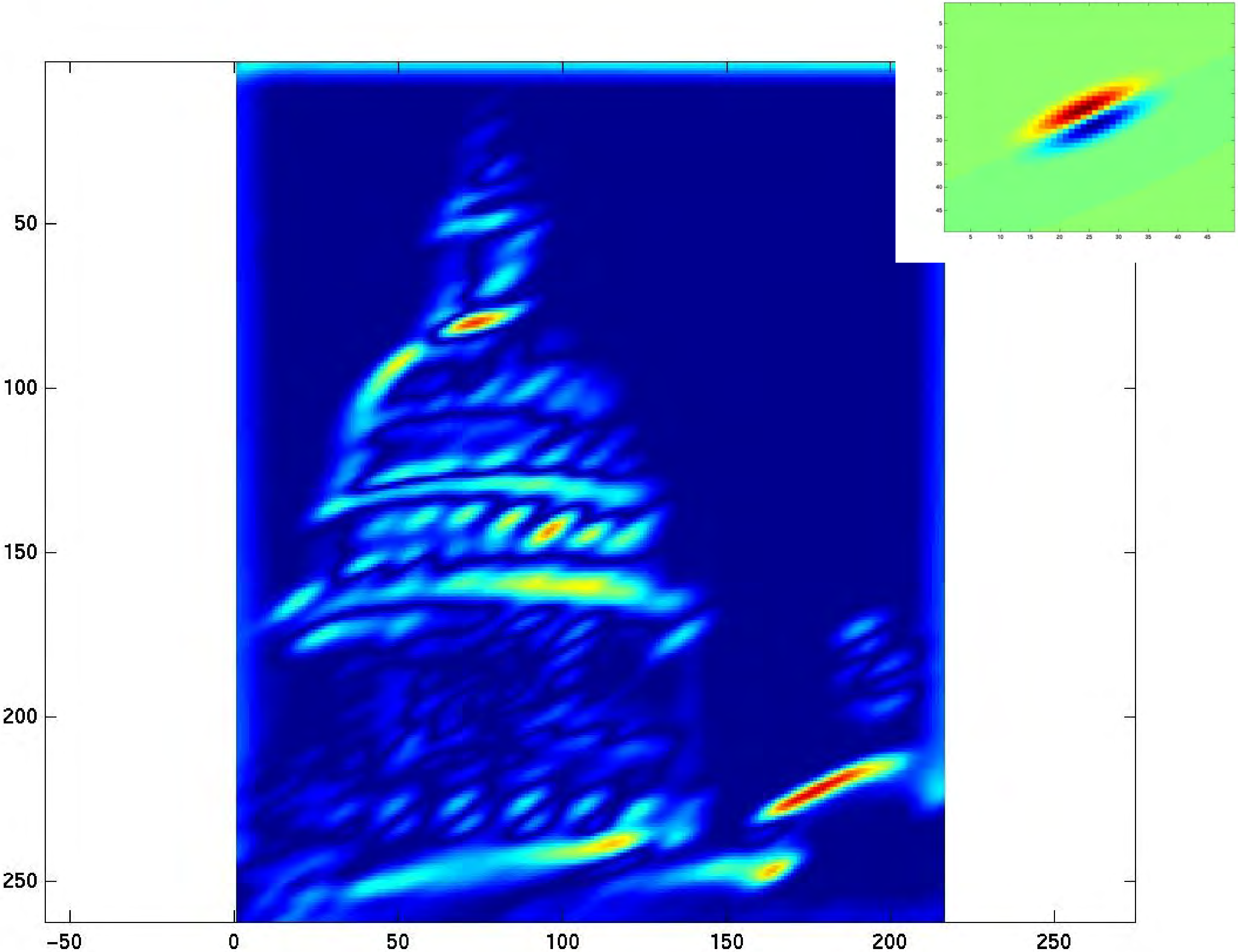
Slide credit:
Kristen Grauman



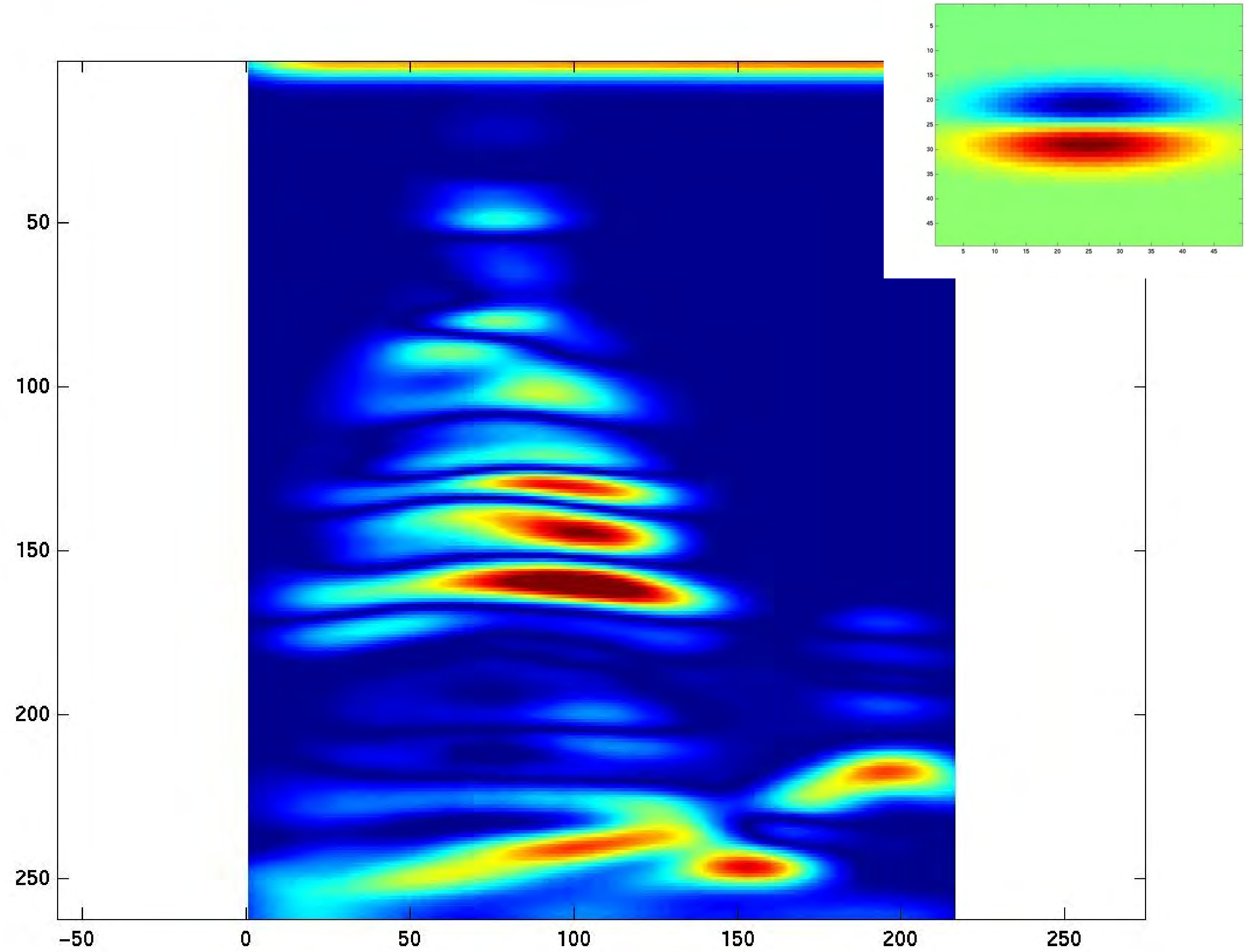
Slide credit:
Kristen Grauman



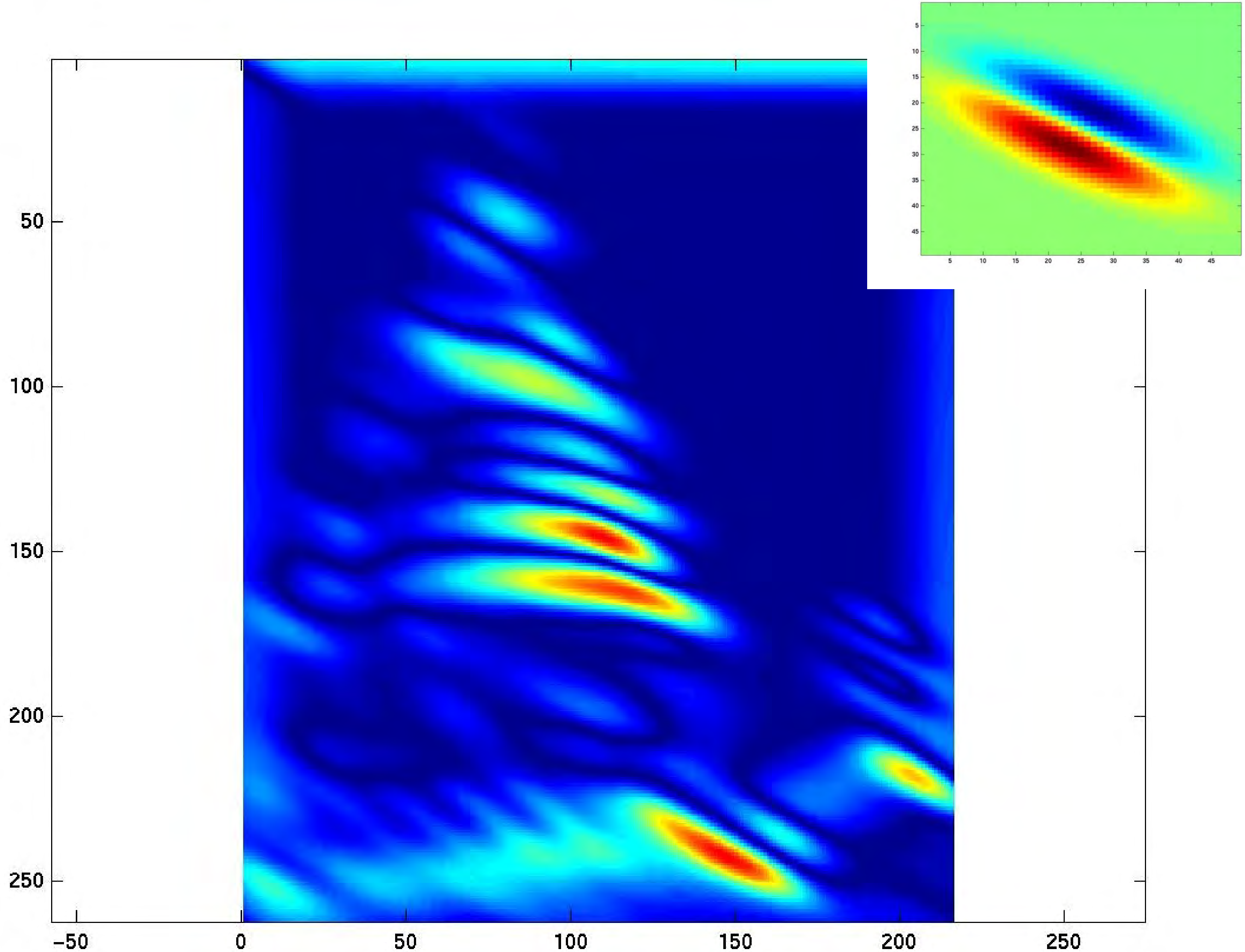
Slide credit:
Kristen Grauman



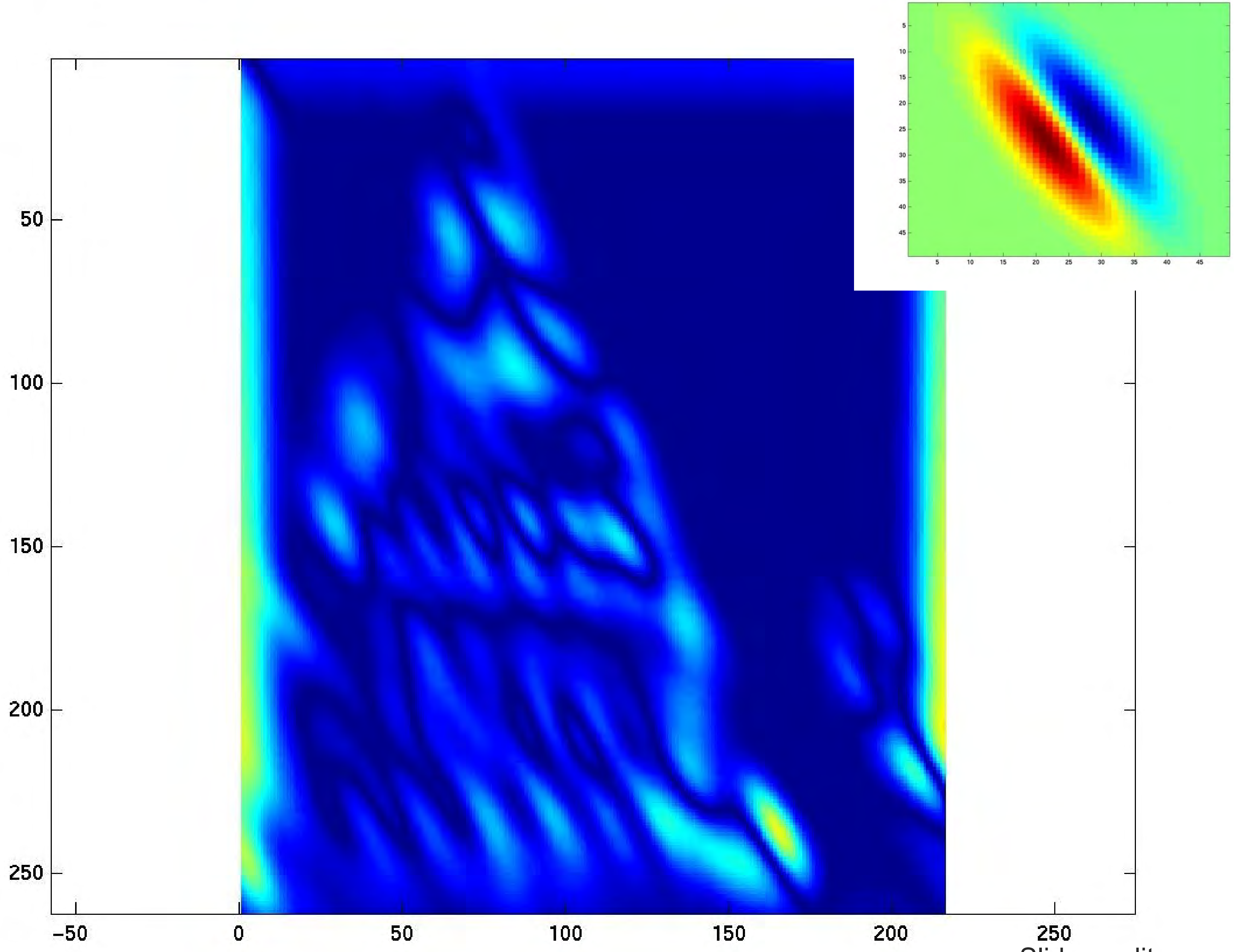
Slide credit:
Kristen Grauman



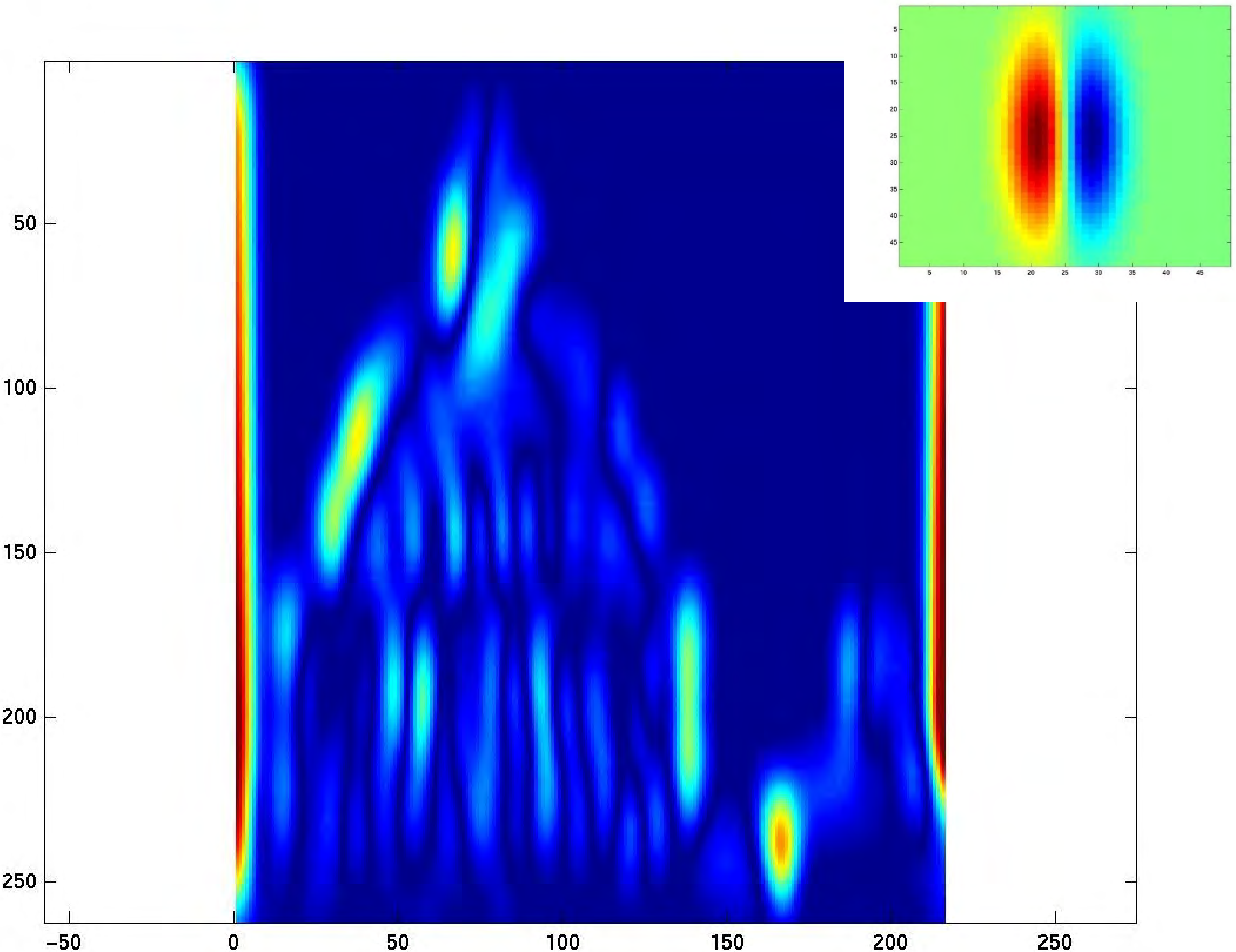
Slide credit:
Kristen Grauman



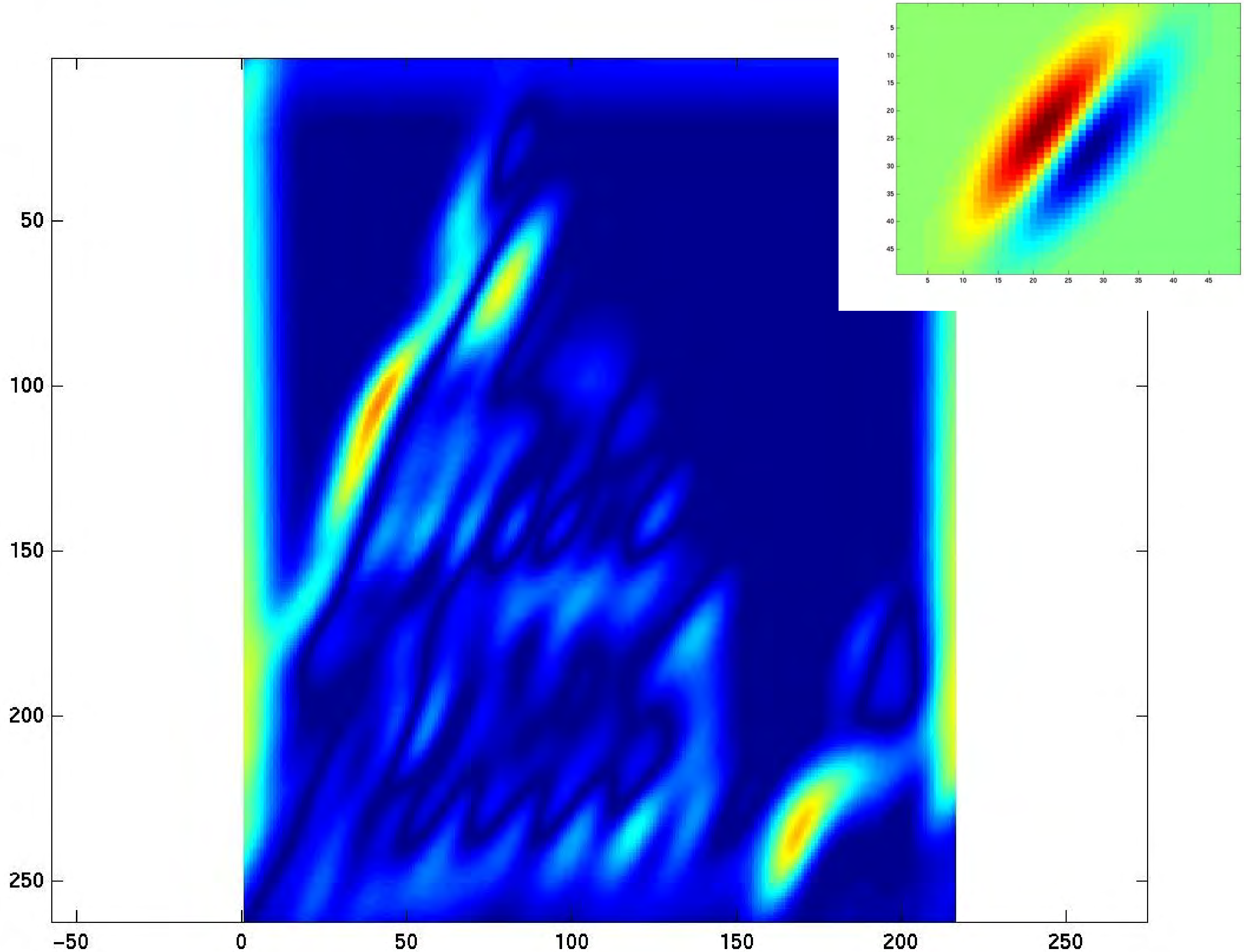
Slide credit:
Kristen Grauman



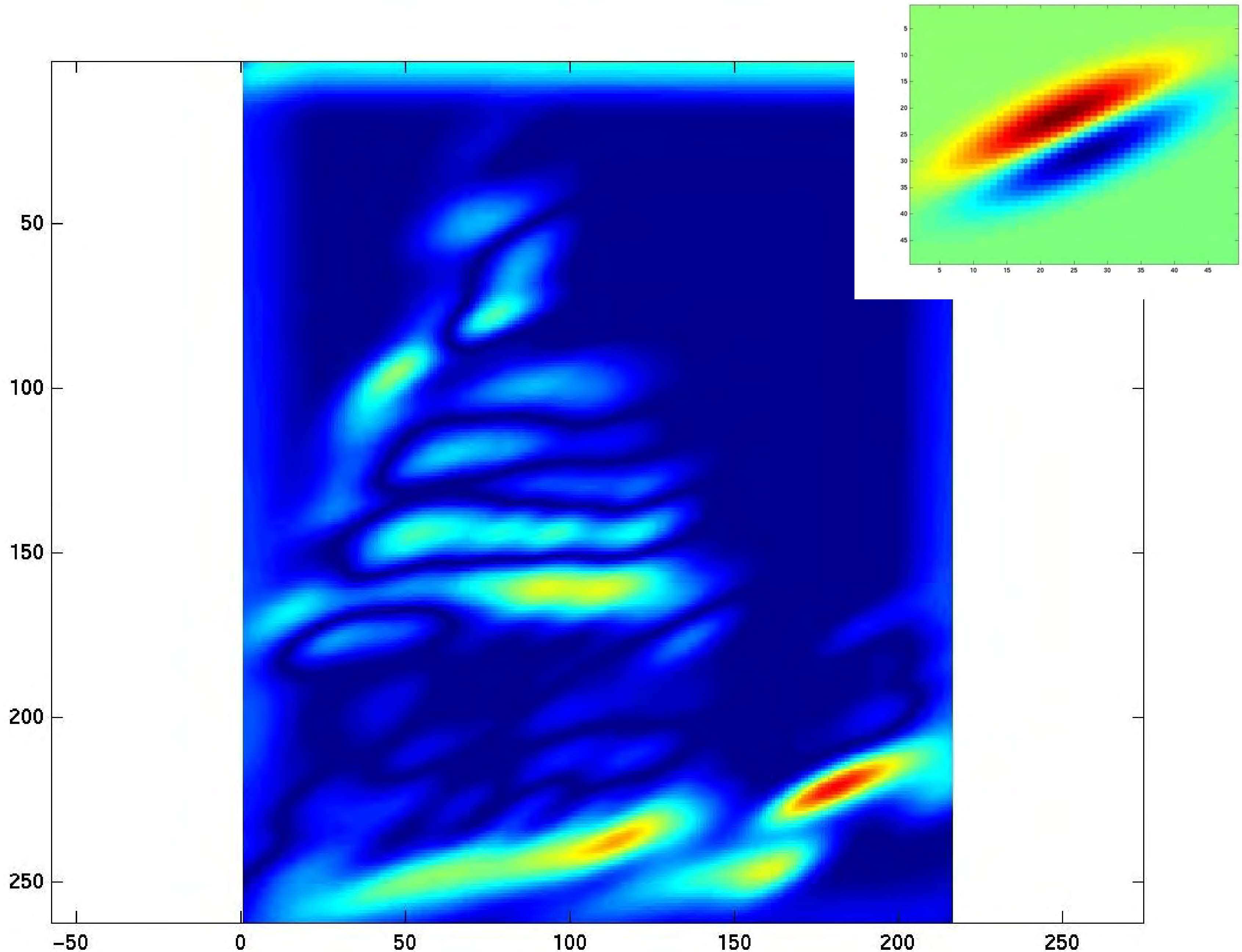
Slide credit:
Kristen Grauman



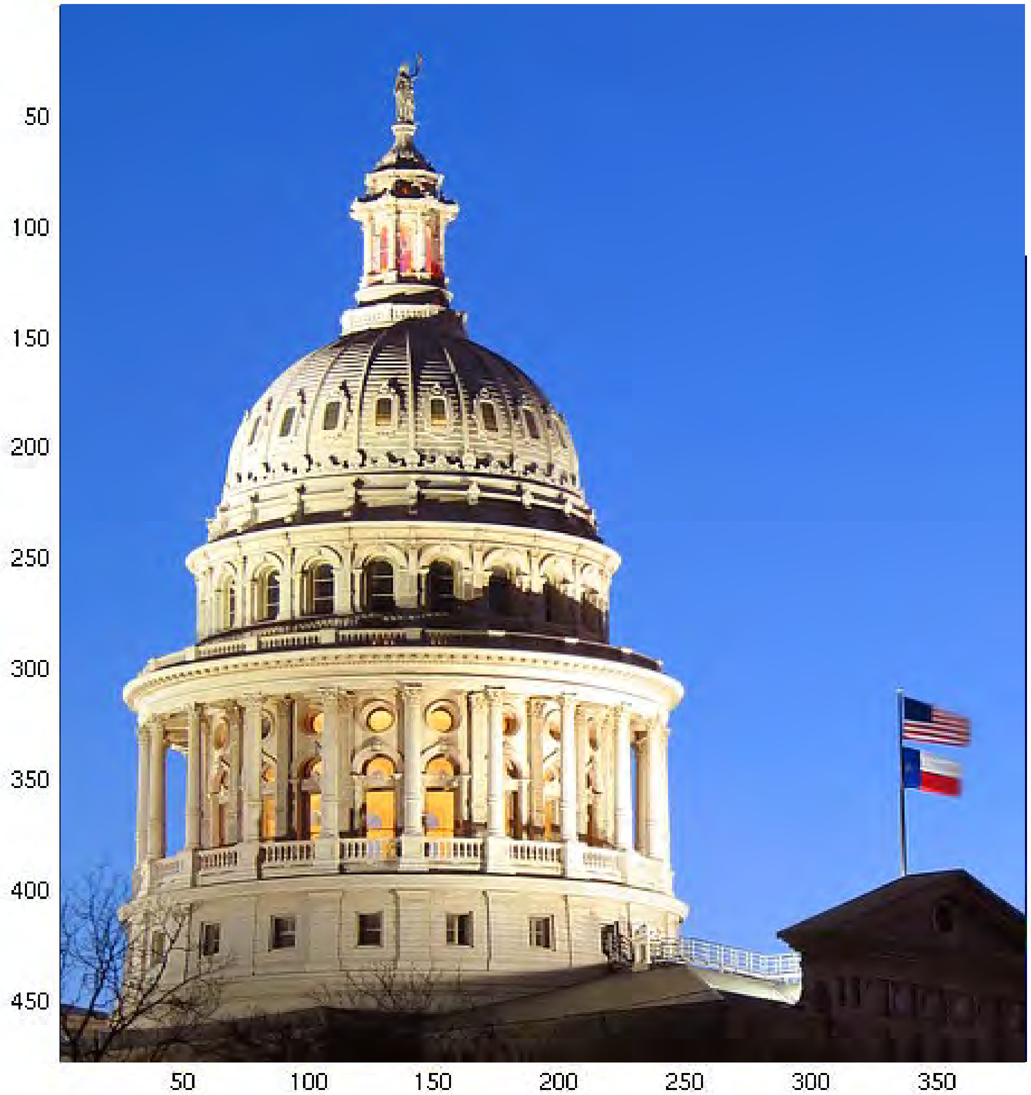
Slide credit:
Kristen Grauman

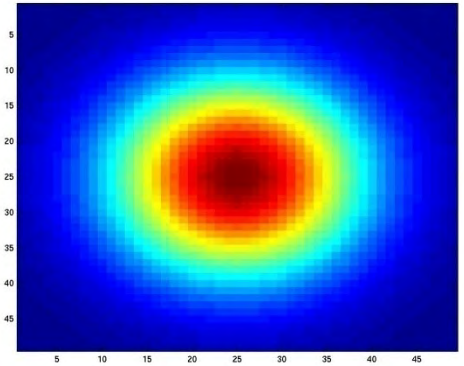


Slide credit:
Kristen Grauman



Slide credit:
Kristen Grauman





Slide credit:
Kristen Grauman

Steerable pyramid applications

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

Linear Image Transforms

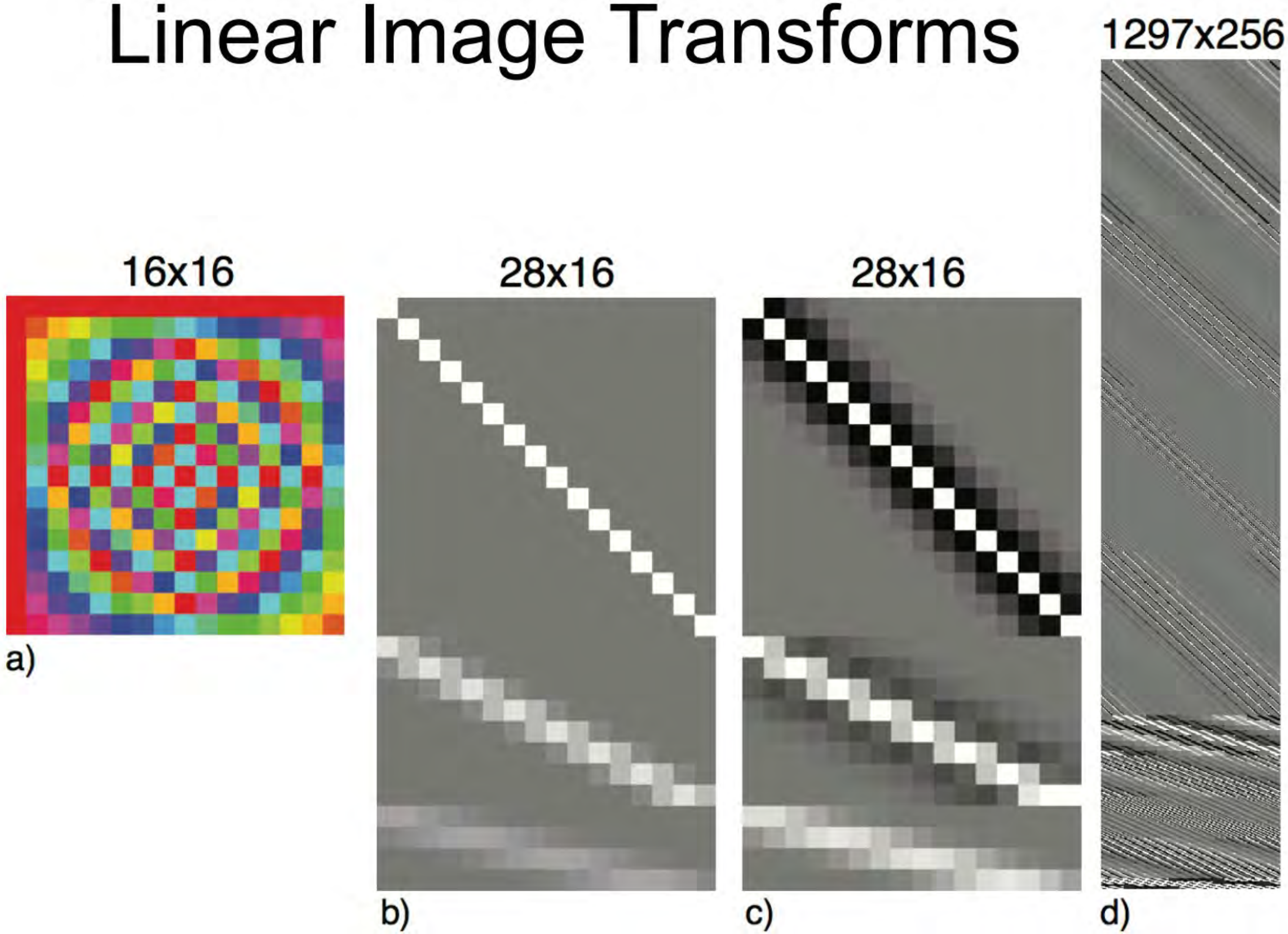


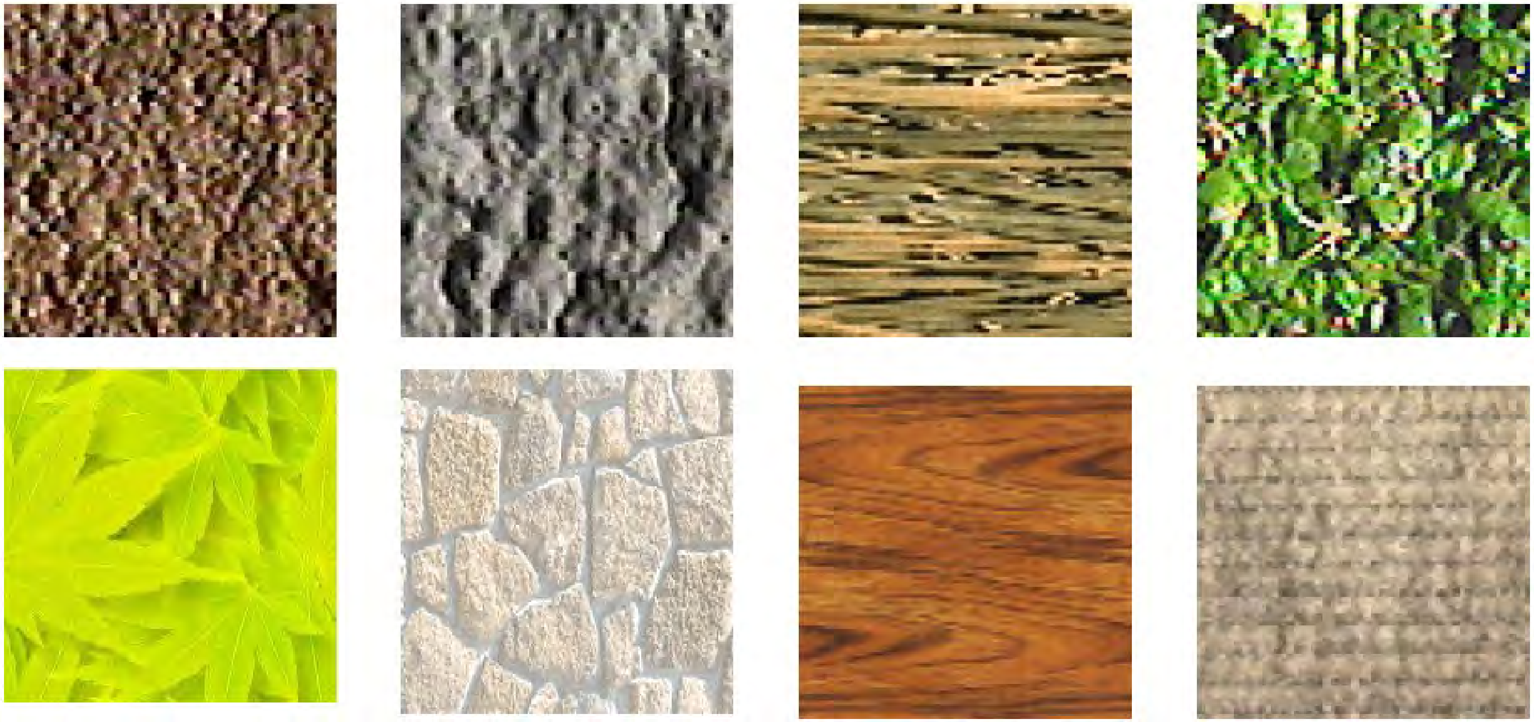
Image representation

- 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

Application: Representing Texture

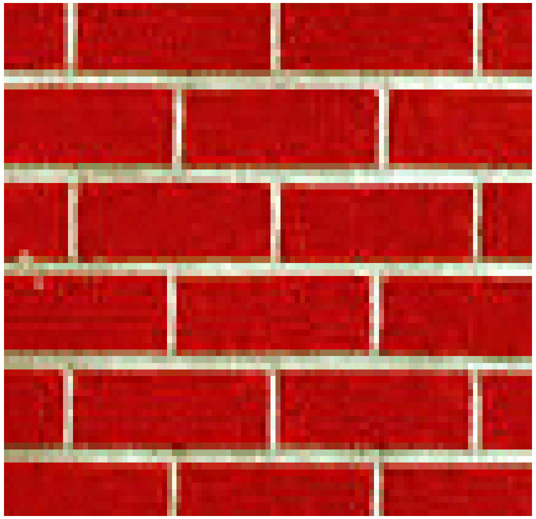
Source: Forsyth

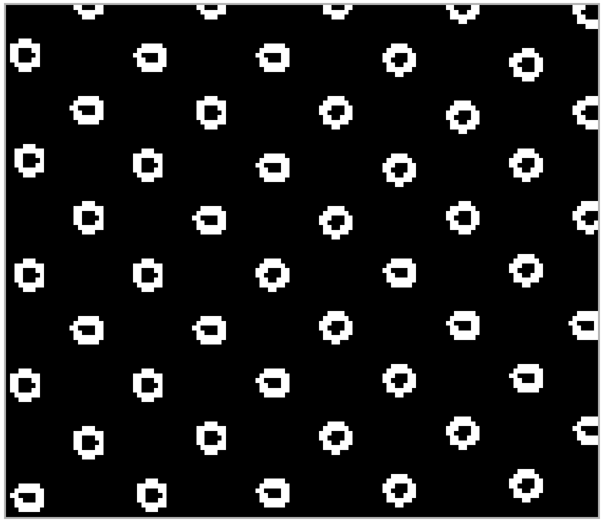
Texture

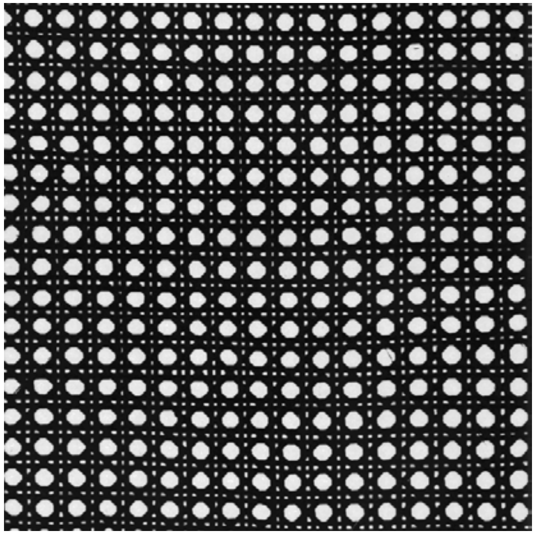


What defines a texture?

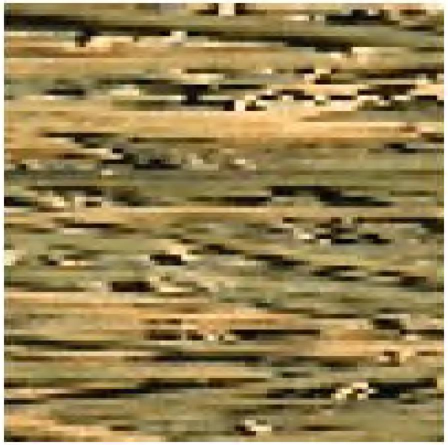
Includes: more regular patterns

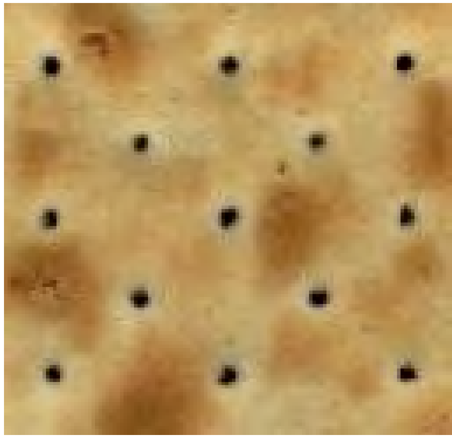




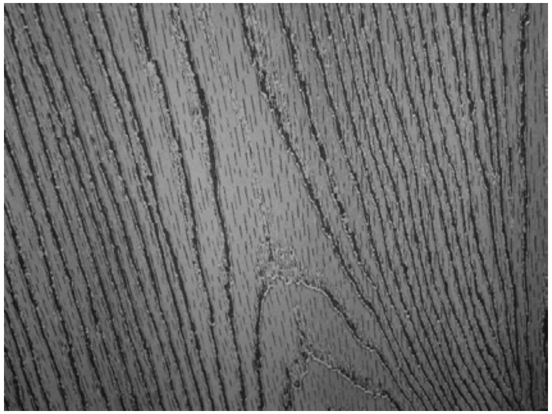


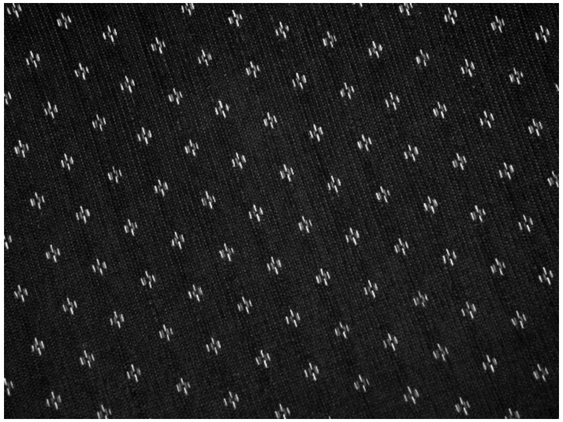
Includes: more random patterns

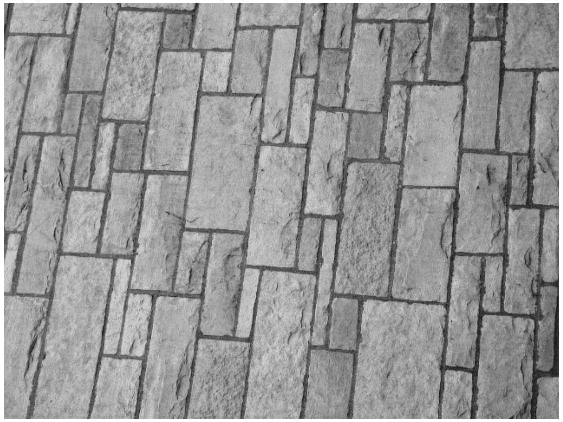




Texture and Material

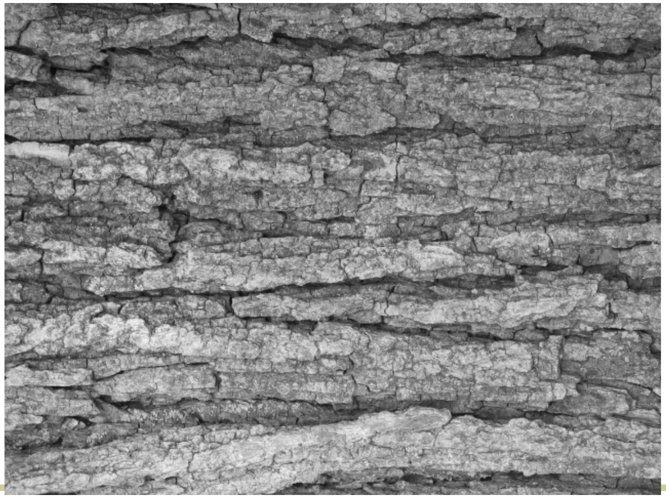






http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Orientation

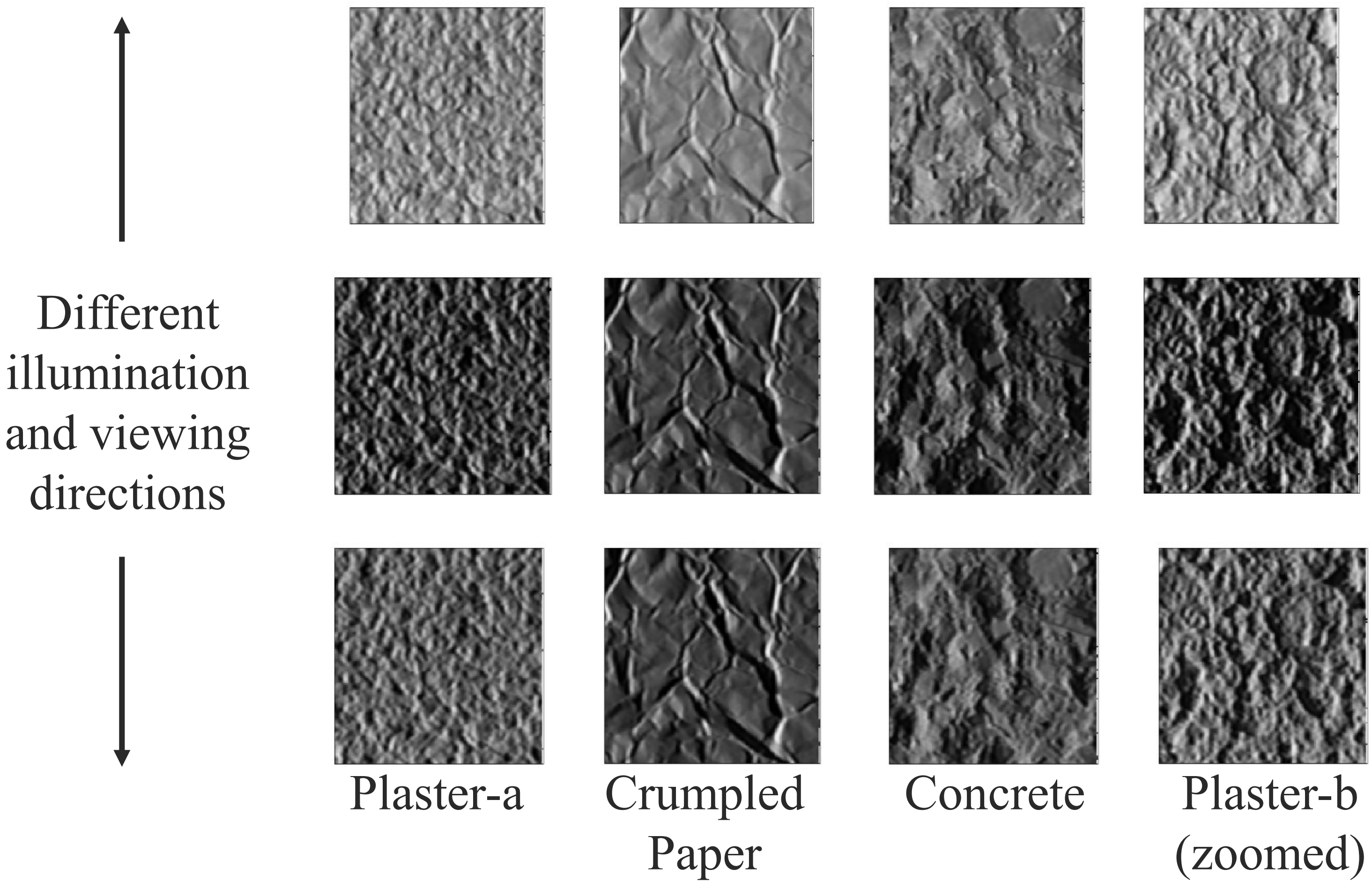


http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Scale

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

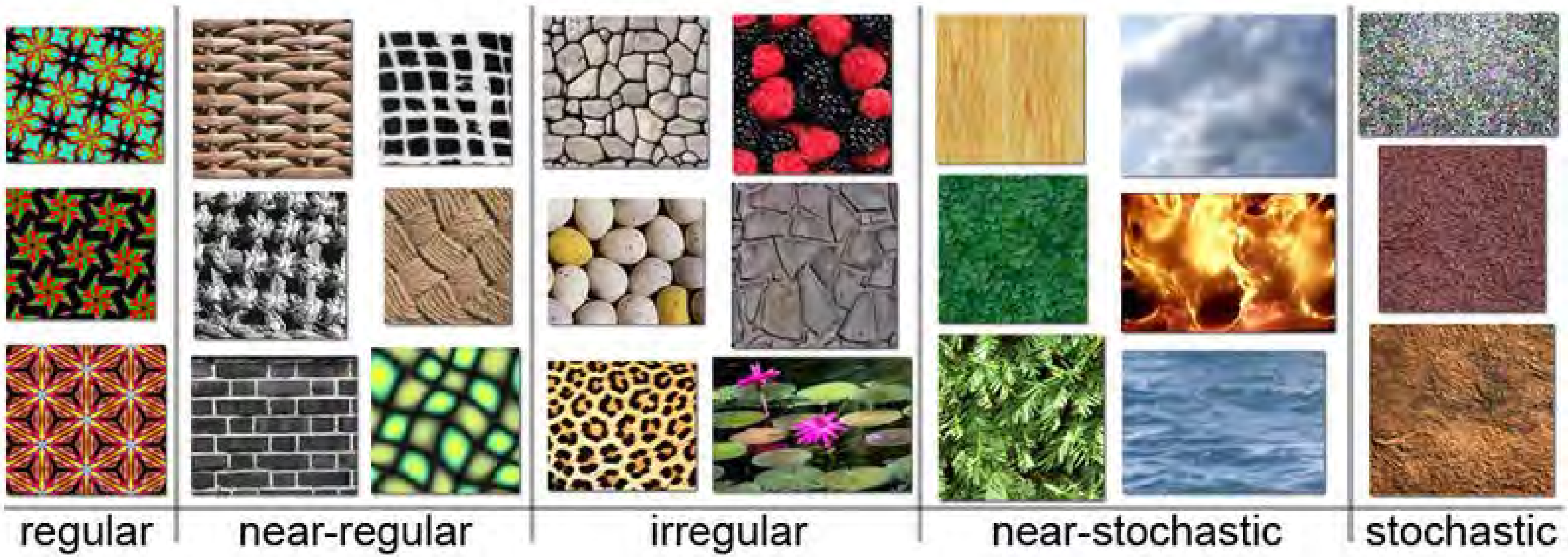
Materials under different illumination and viewing directions



What is texture?

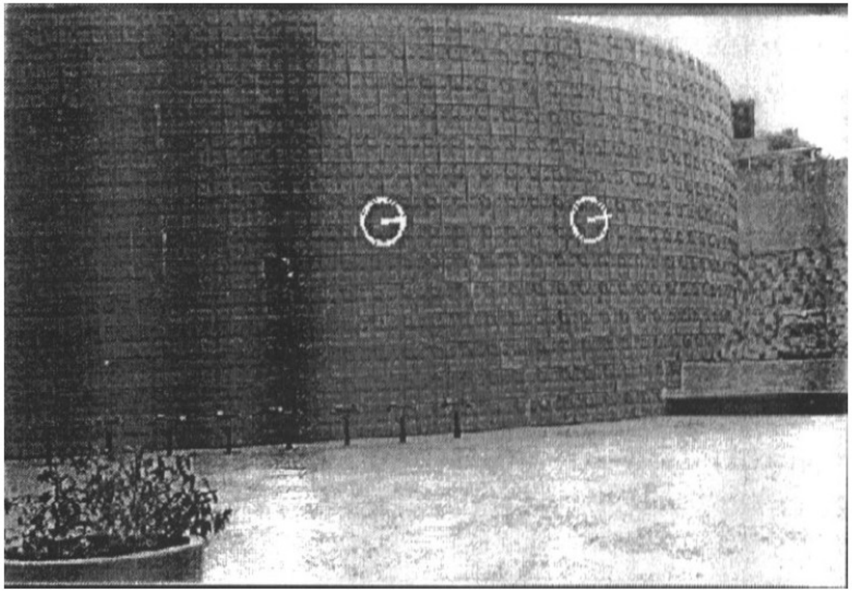
- **Texture is a phenomenon that is widespread, easy to recognize, and hard to define.**
- Views of large numbers of small objects
- Regular or stochastic patterns caused by bumps, grooves, and/or markings
- Textures tend to show repetition: the same local patch appears again and again.

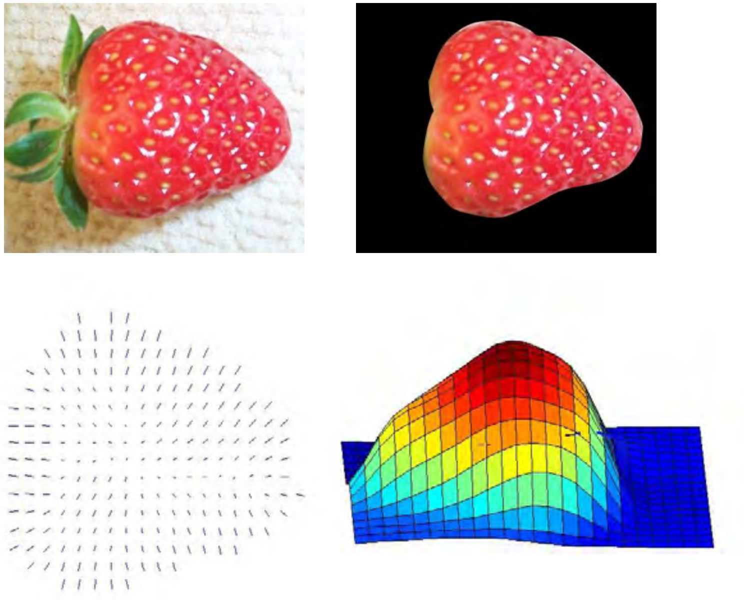
Texture overview



Shape from texture

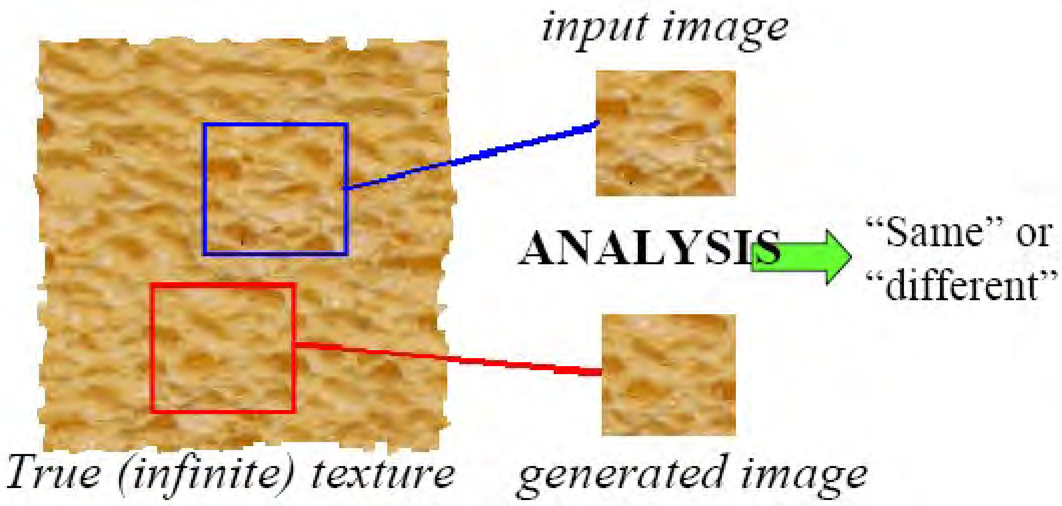
- Use deformation of texture from point to point to estimate surface shape



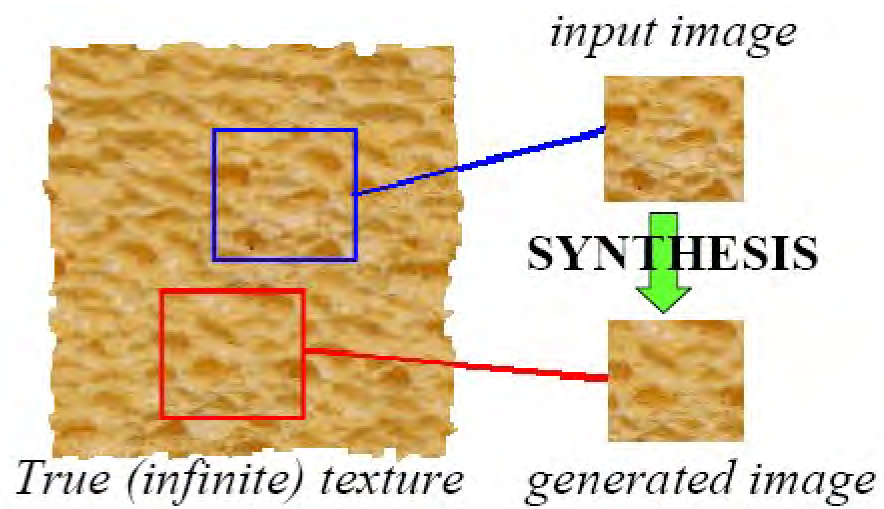


Pics from A. Loh: http://www.csse.uwa.edu.au/~angie/phdpics1.html

Analysis vs. Synthesis



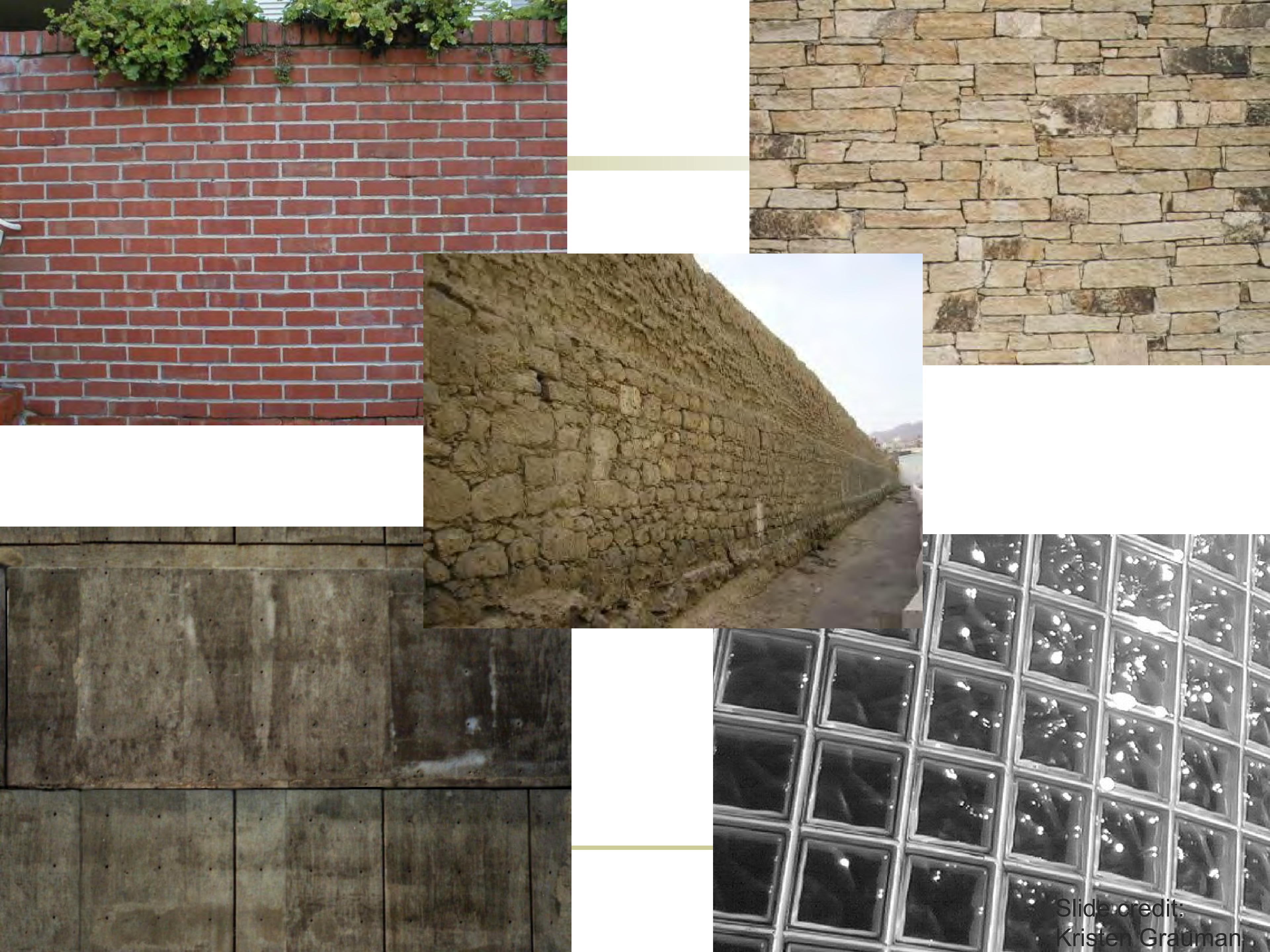
Why analyze texture?



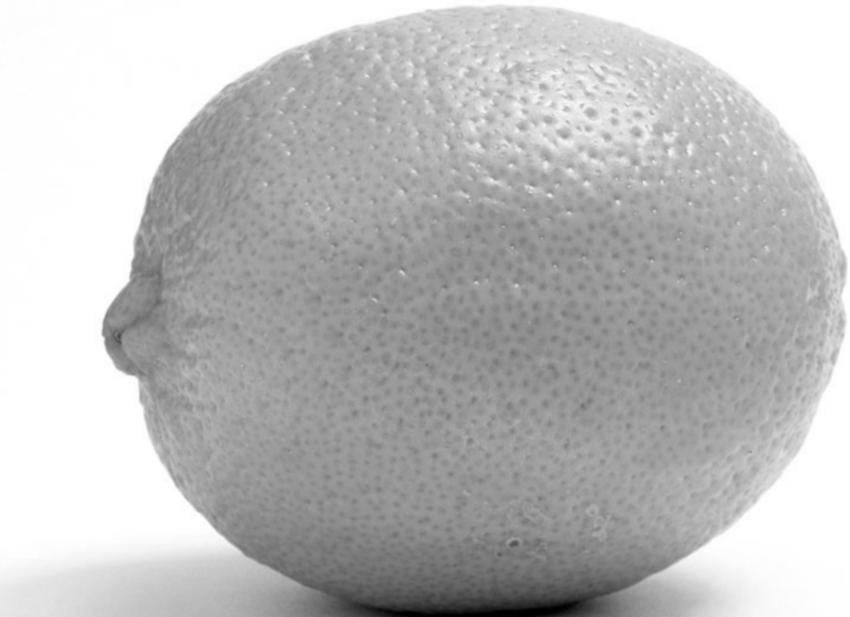
Images:Bill Freeman, A. Efros

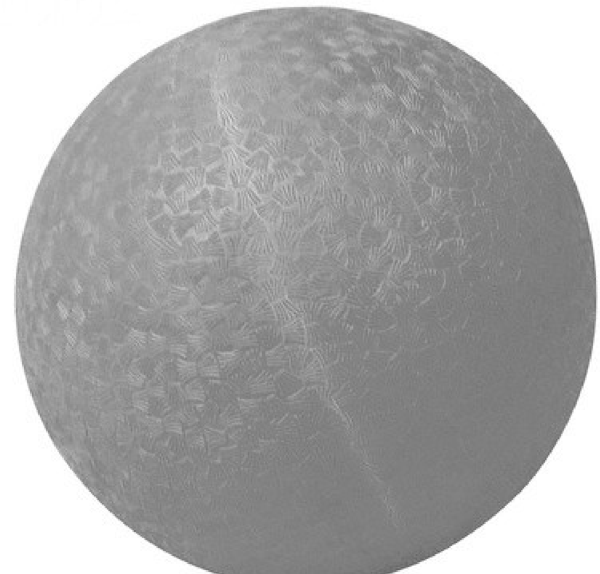
Texture-related tasks

- **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



Slide credit:
Kristen Grauman





Slide credit:
Kristen Grauman

http://animals.nationalgeographic.com/

Slide credit:
Kristen Grauman

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

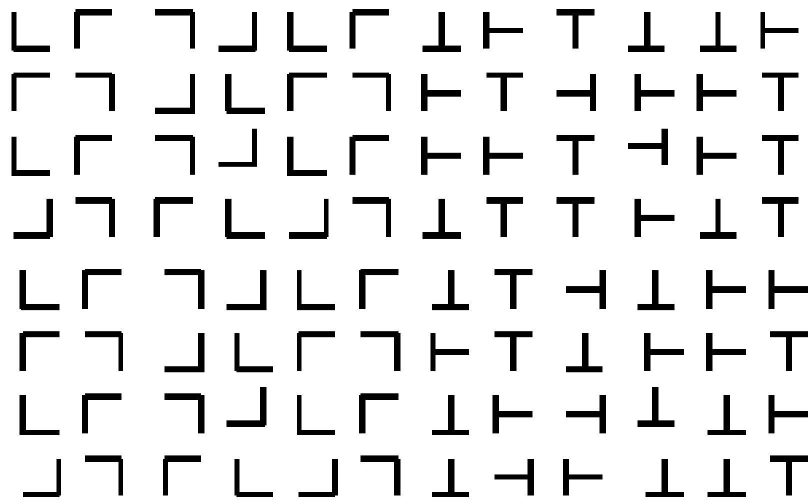
Technically:

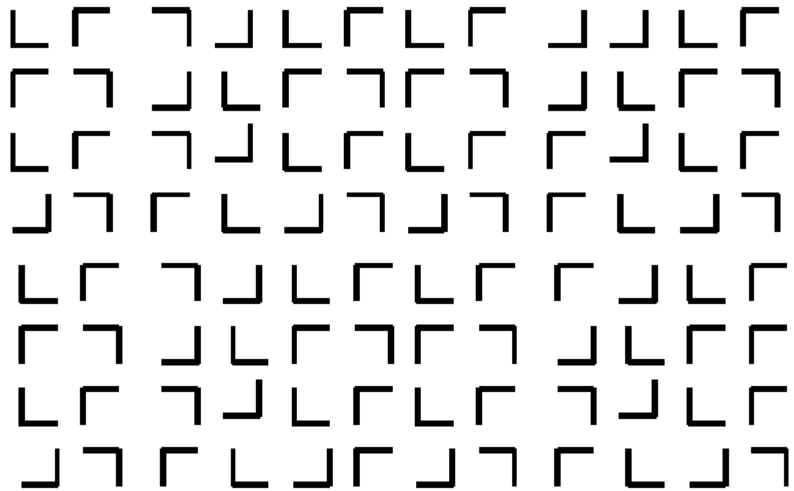
- Representation-wise, we want a feature one step above "building blocks" of filters, edges.

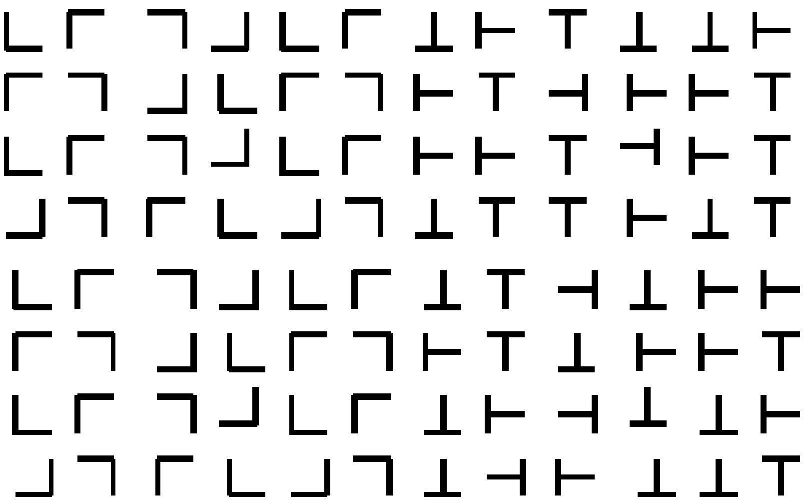
Psychophysics of texture

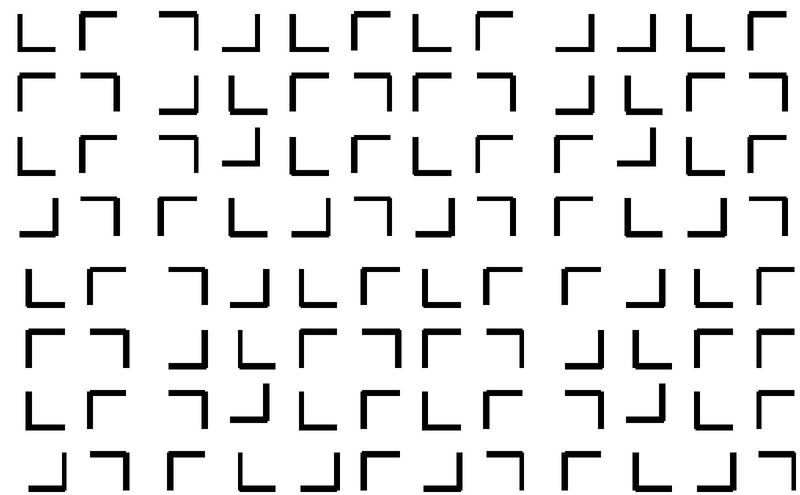
- Some textures distinguishable with *preattentive* perception– without scrutiny, eye movements [Julesz 1975]

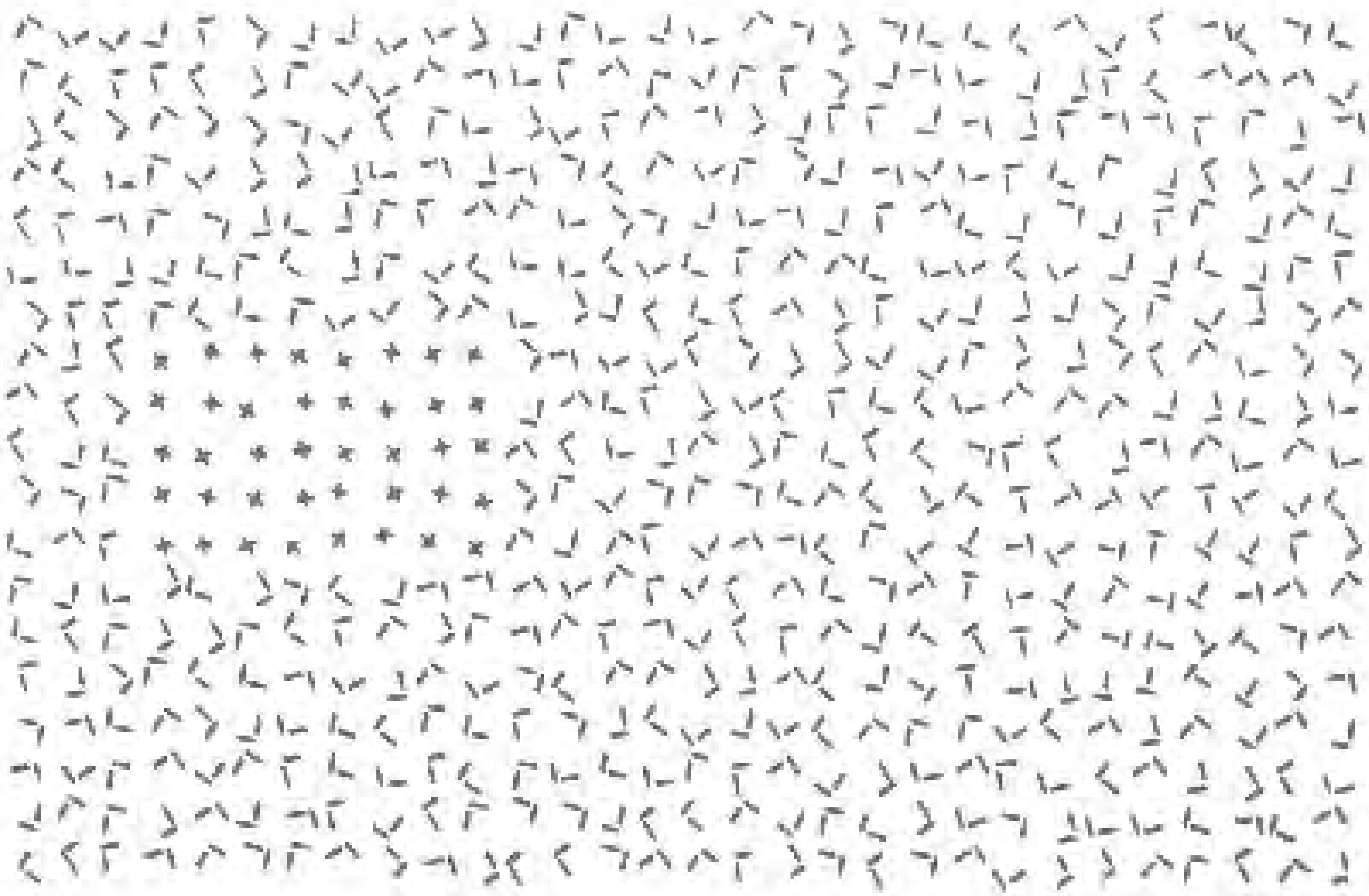
Same or different?



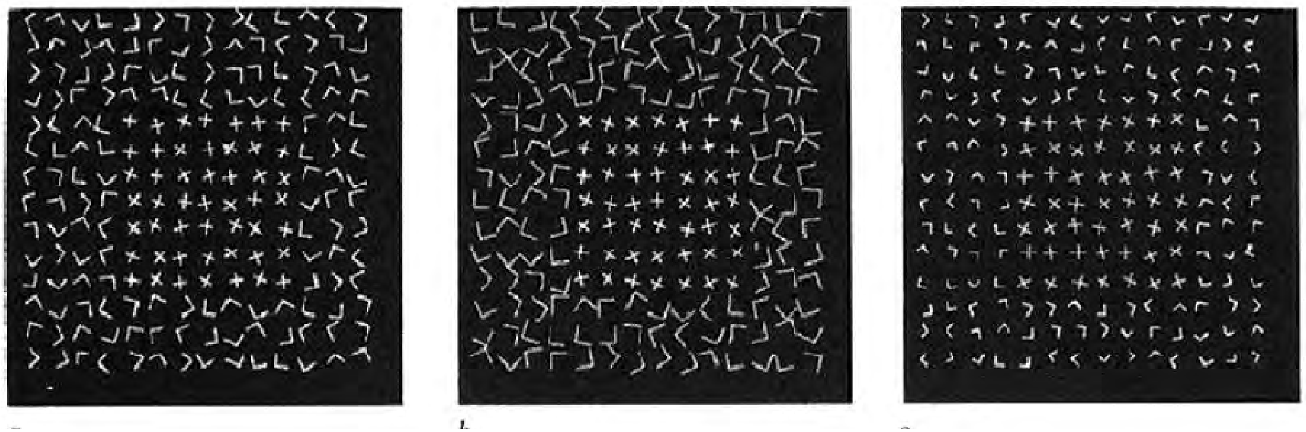








Capturing the local patterns with image measurements



[Bergen & Adelson, *Nature* 1988]

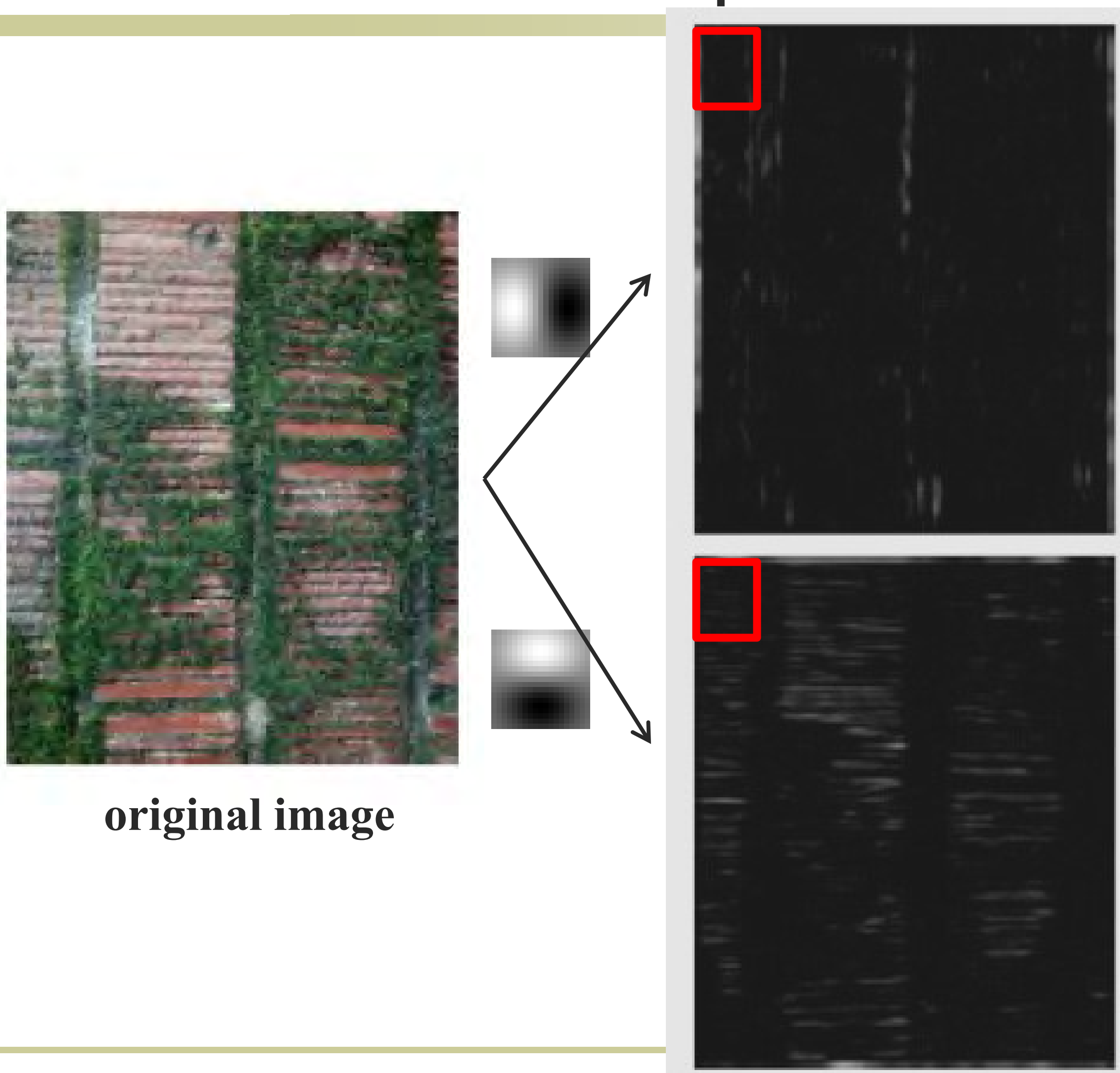
Scale of patterns influences discriminability

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

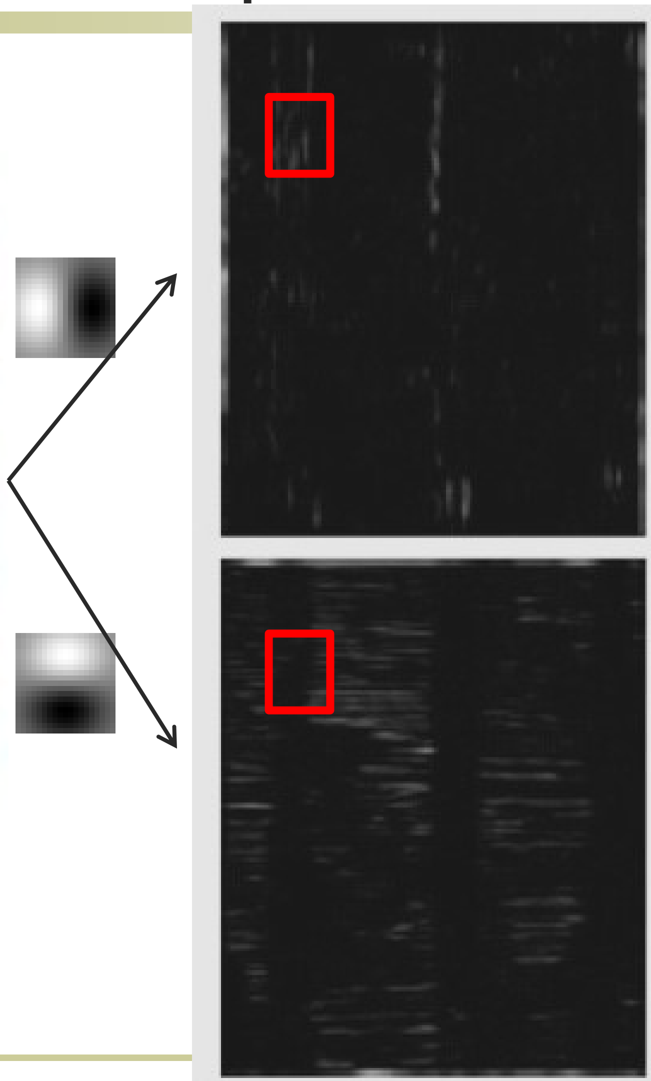
⋮

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

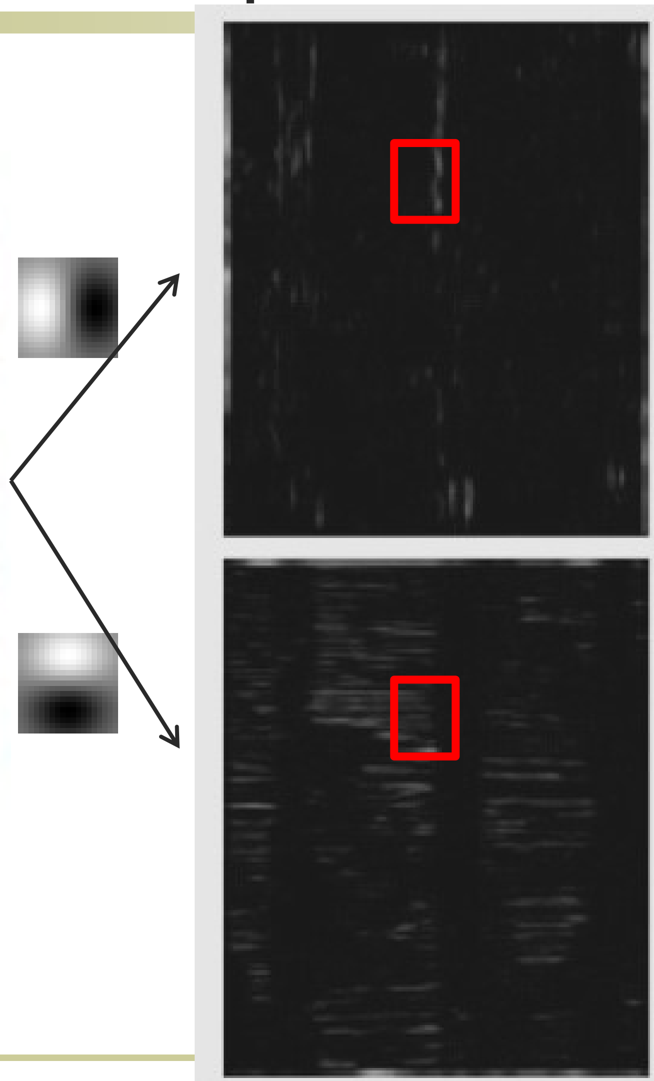
⋮

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

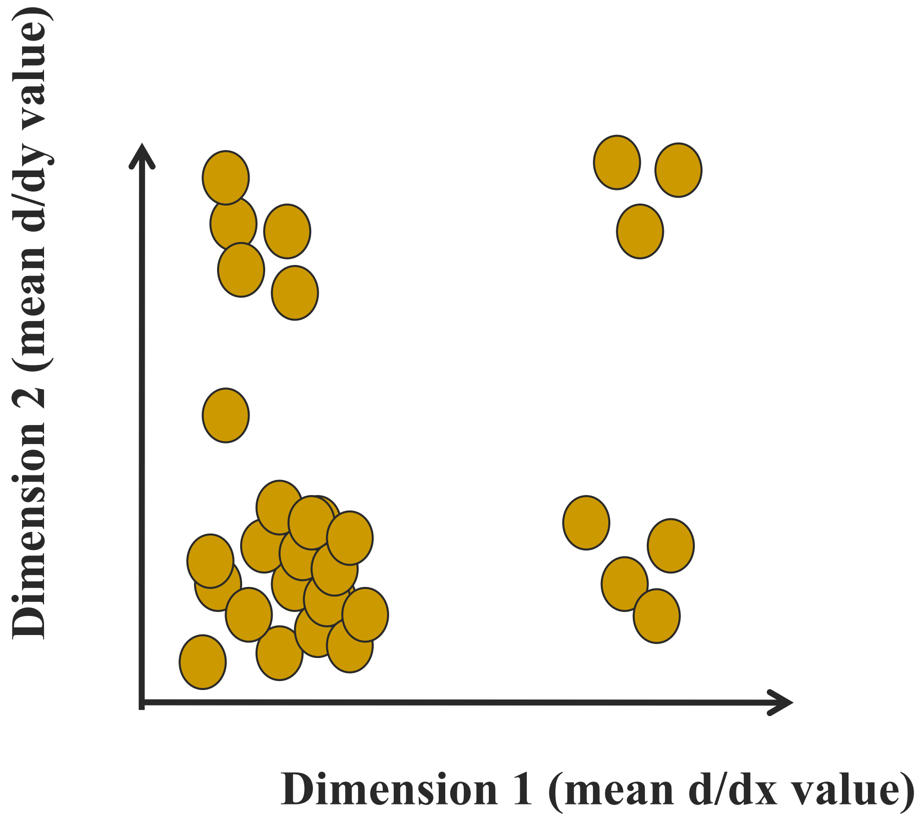
⋮

⋮

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

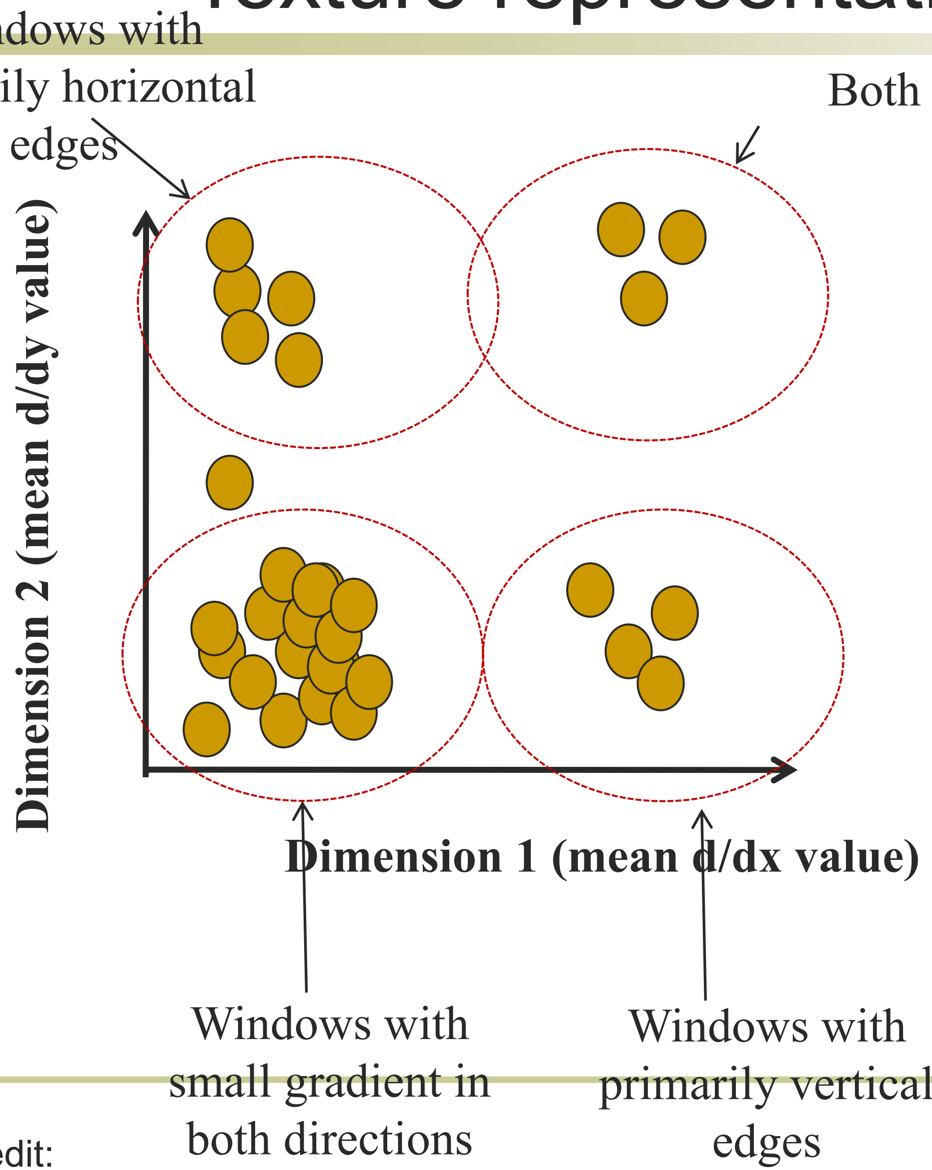
⋮

⋮

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

⋮

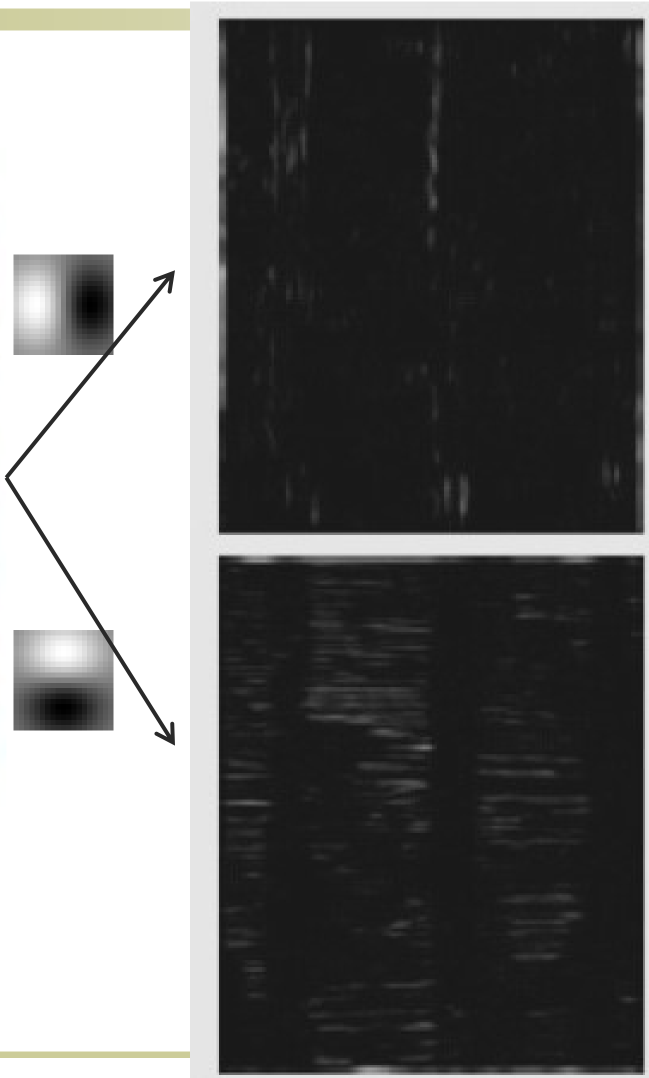
⋮

statistics to summarize patterns in small windows

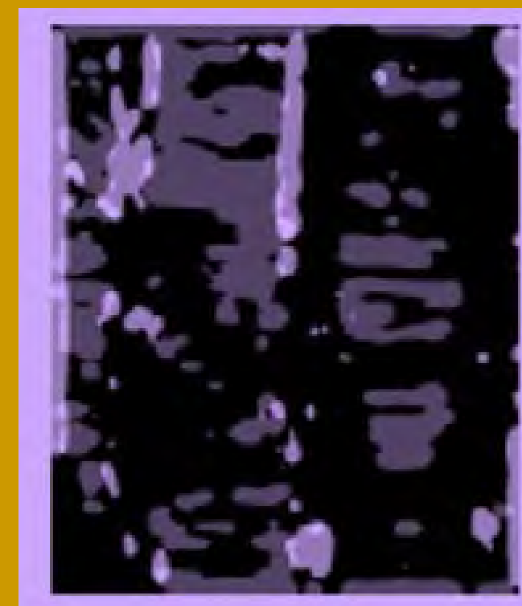
Slide credit: Kristen Grauman

Texture representation: example

original image



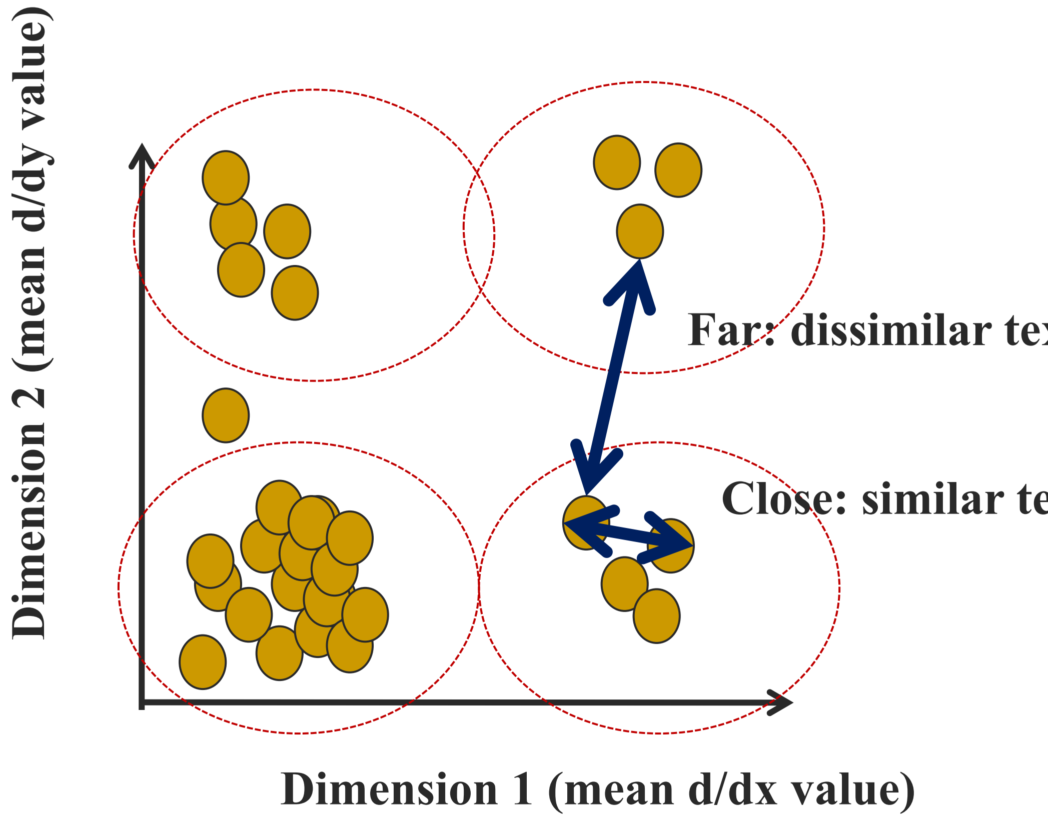
derivative filter responses, squared



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

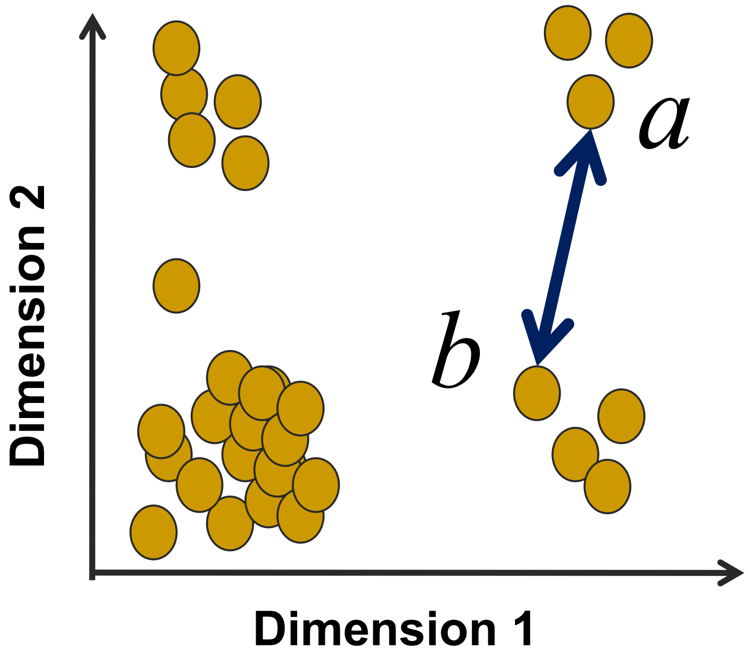
⋮

⋮

statistics to summarize patterns in small windows

Slide credit: Kristen Grauman

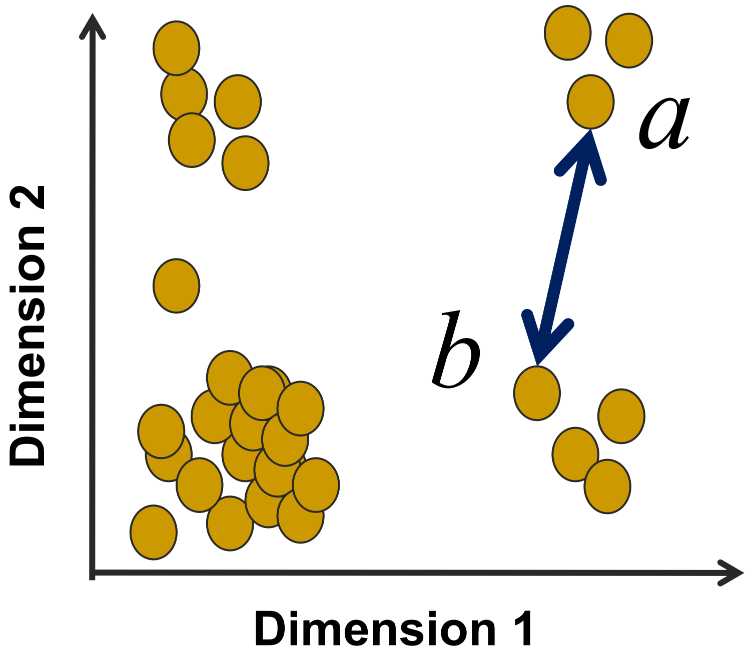
Texture representation: example

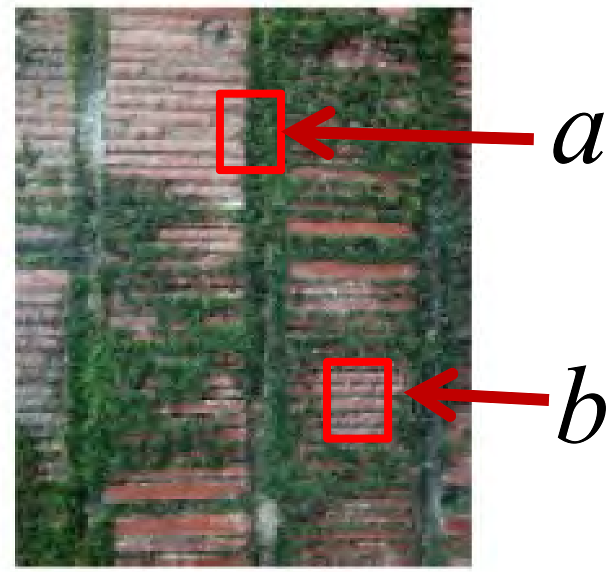


$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Slide credit:
Kristen Grauman

Texture representation: example



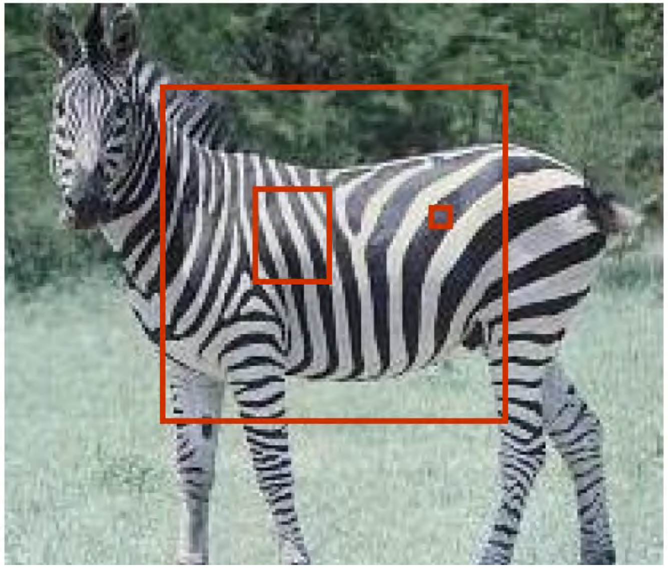


Distance reveals how dissimilar texture from window a is from texture in window b.

Slide credit:
Kristen Grauman

Texture representation: window scale

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



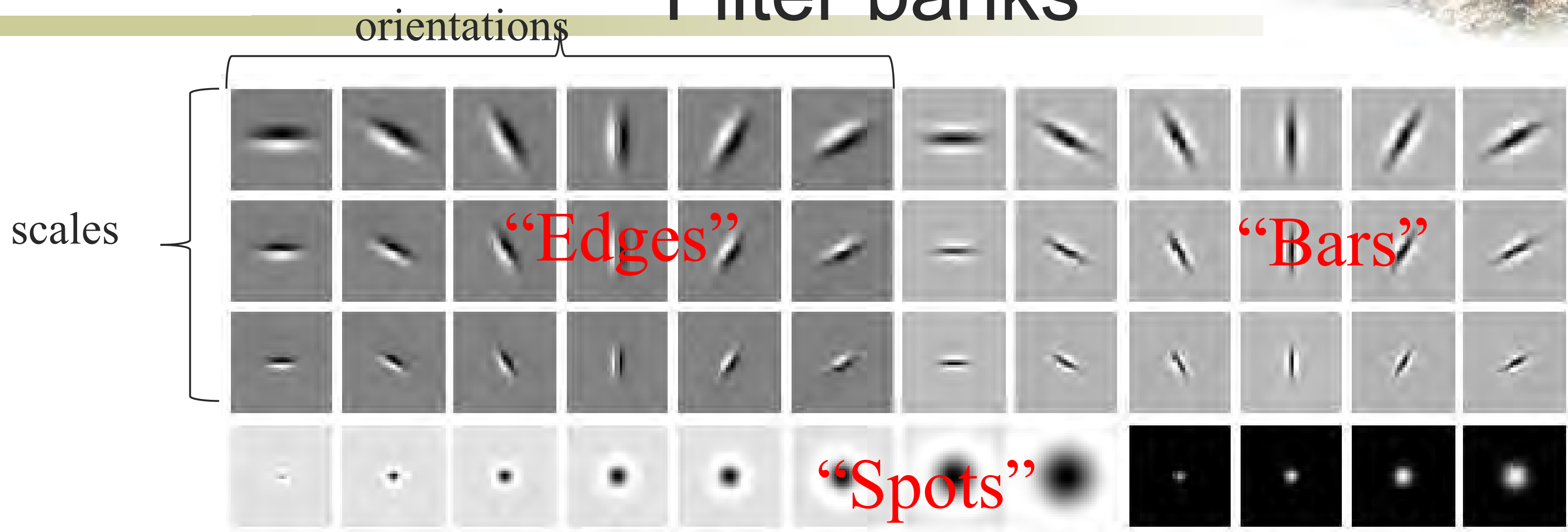
Possible to perform **scale selection** by looking for window scale where texture description not changing.

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

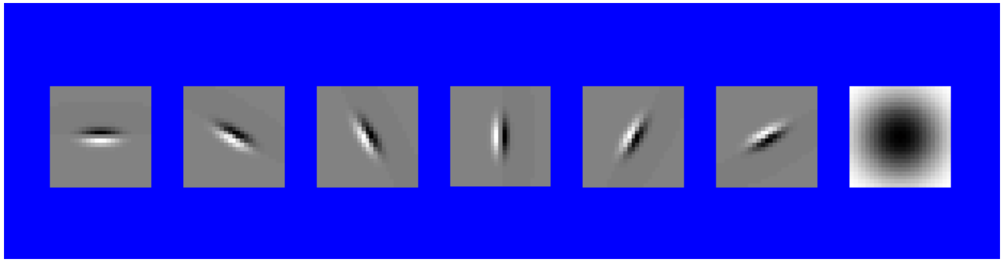


- What filters to put in the bank?
    - Typically we want a combination of scales and orientations, different types of patterns.

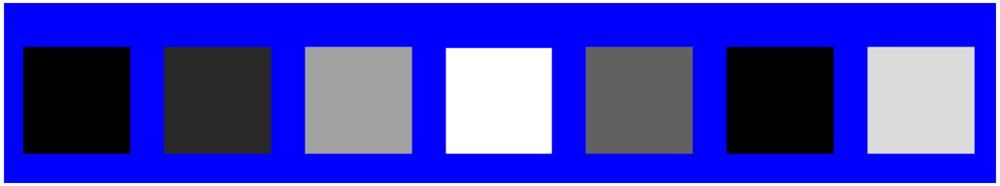
Matlab code available for these examples:
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

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

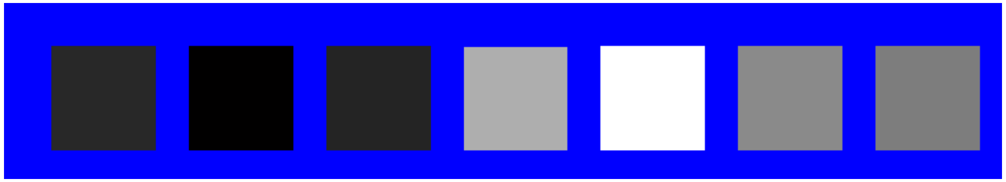
Filters



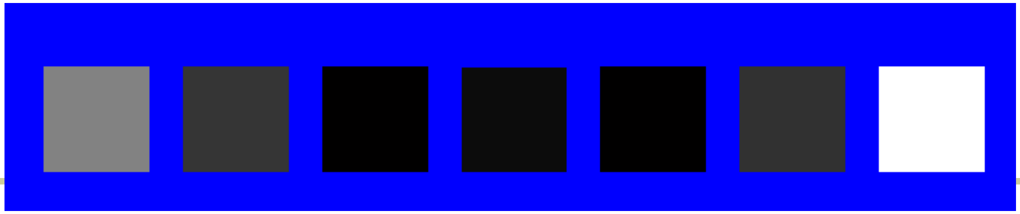
1



2

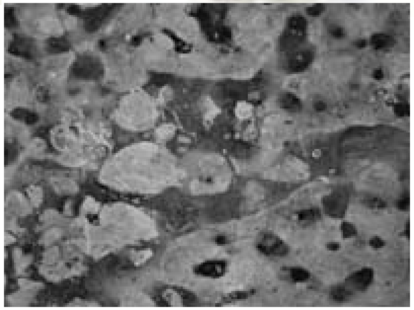


3



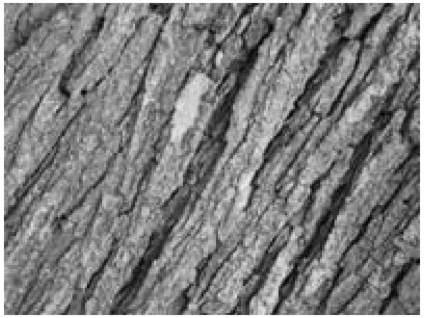
Mean abs responses

A



B

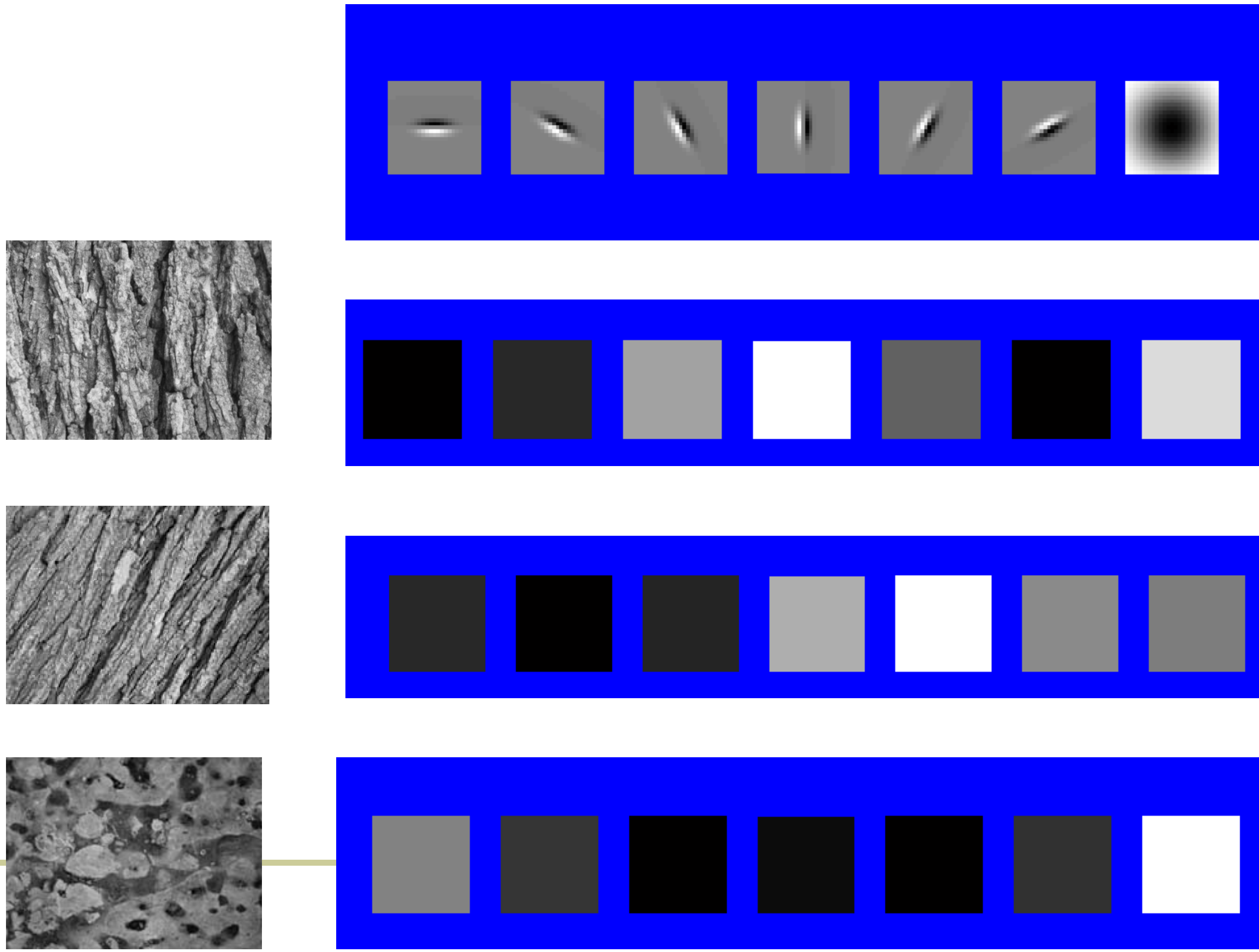
C



Derek Hoiem

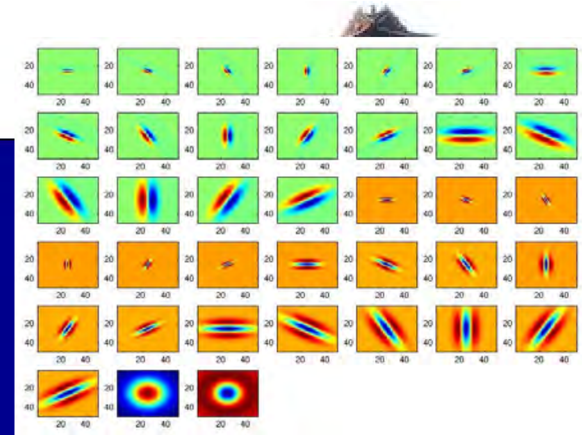
Representing texture by mean abs response

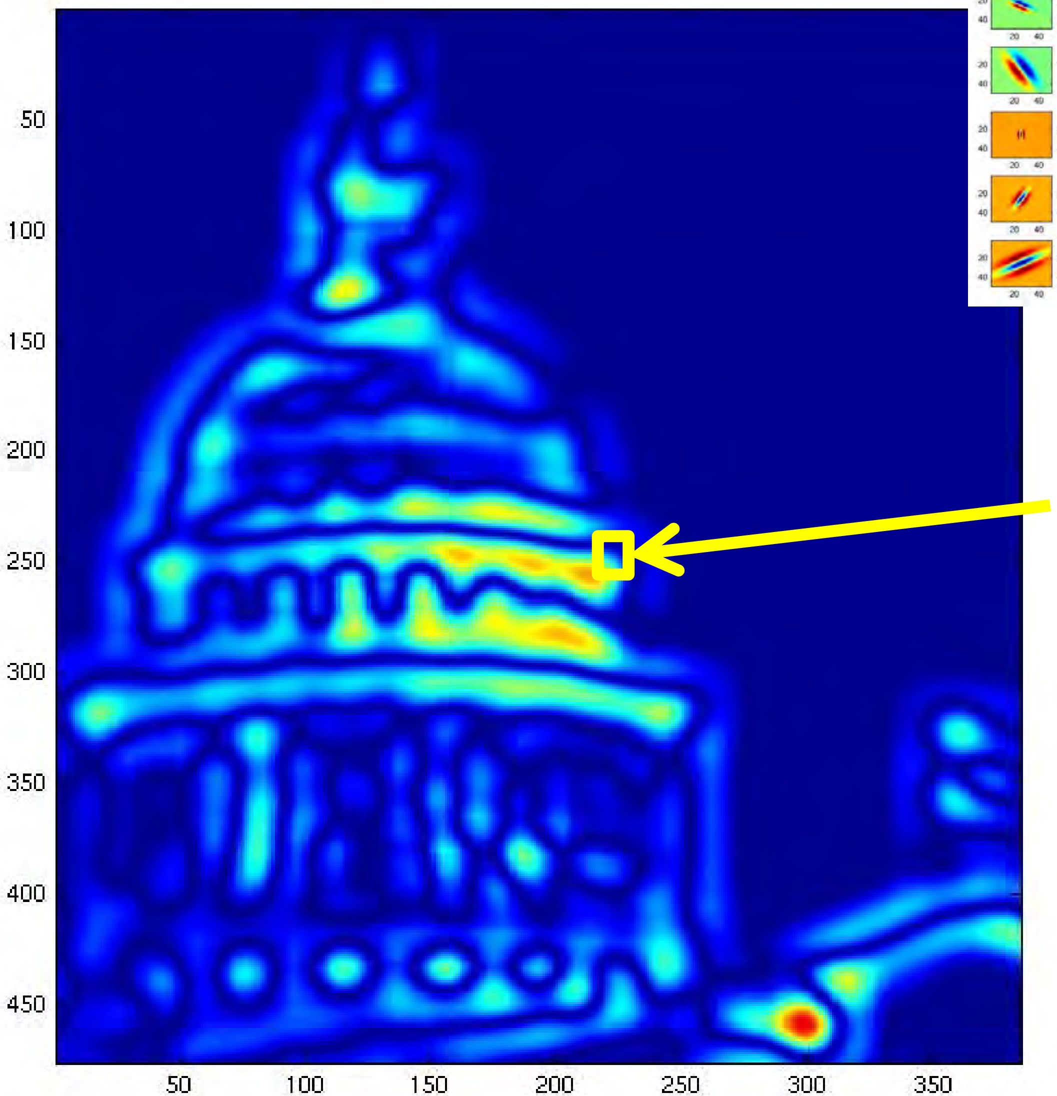
Filters



Mean abs responses

Derek Hoiem





[r1, r2, …, r38

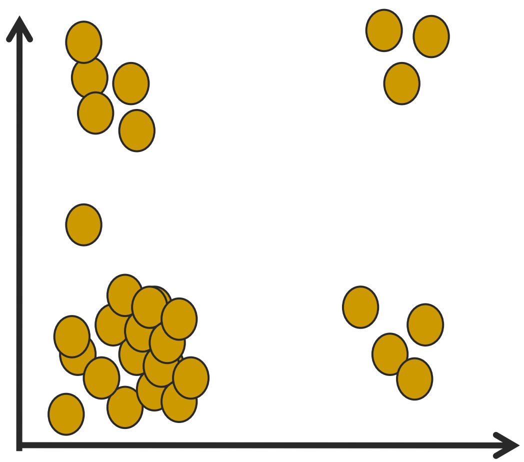
We can form a feature vector from the list of responses at each pixel.

Slide credit:
Kristen Grauman

d-dimensional features

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

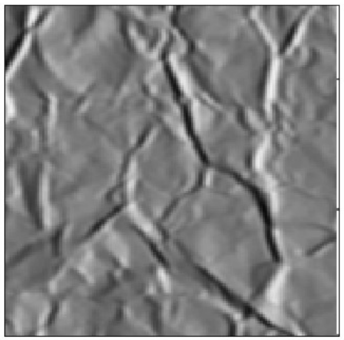
Euclidean distance (L_2)



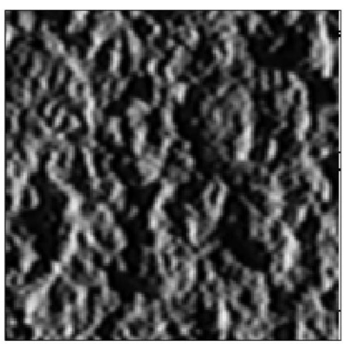
2d

Slide credit:
Kristen Grauman

Texture Recognition



?



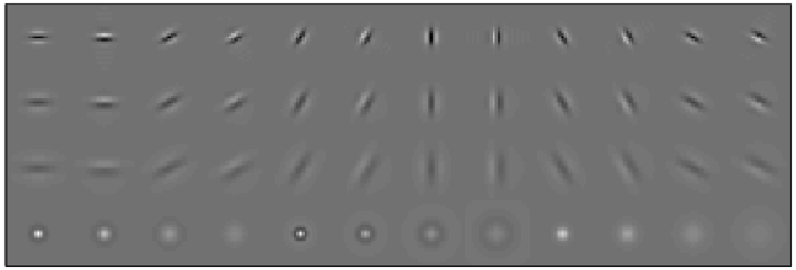
?

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:

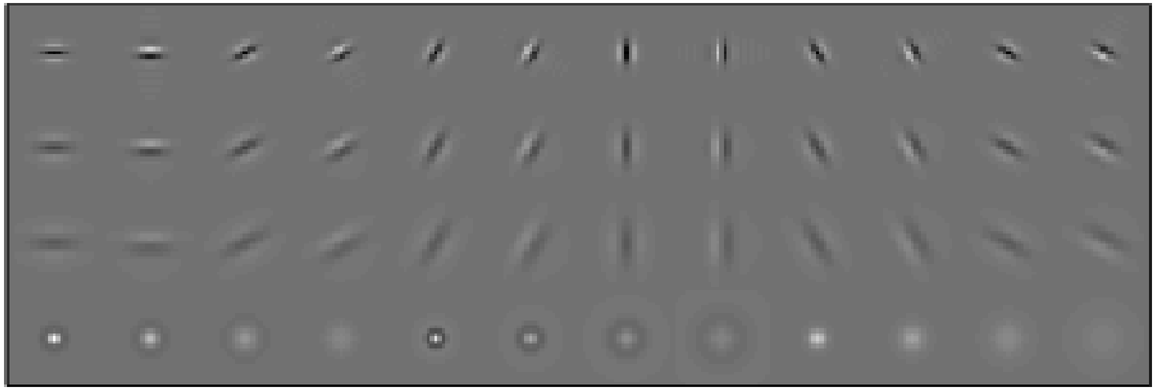


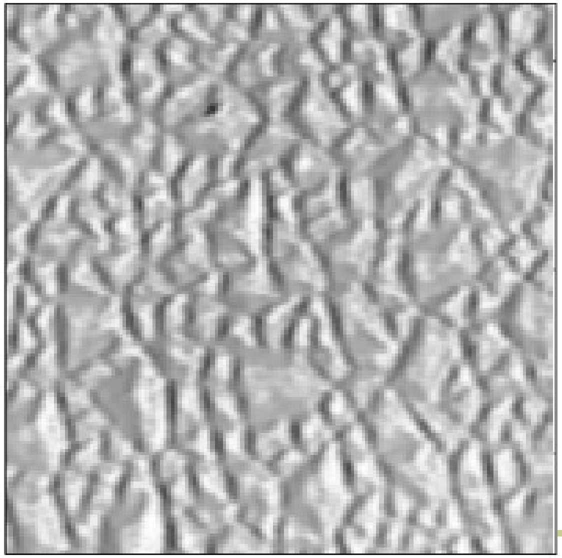
  2) Vector quantization (k-means) on filter outputs;

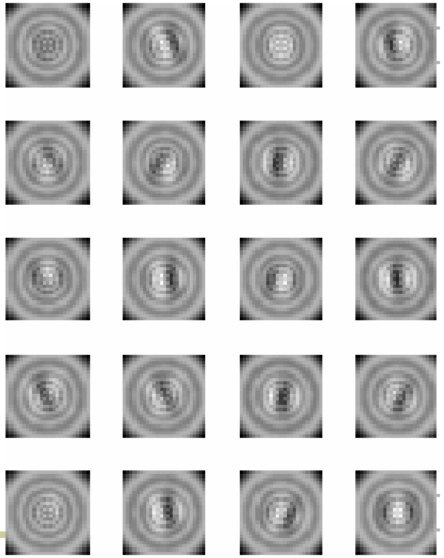
  3) Quantization centers are the textons.

- Spatial distribution of textons defines the texture;

2D Textons (cont'd)

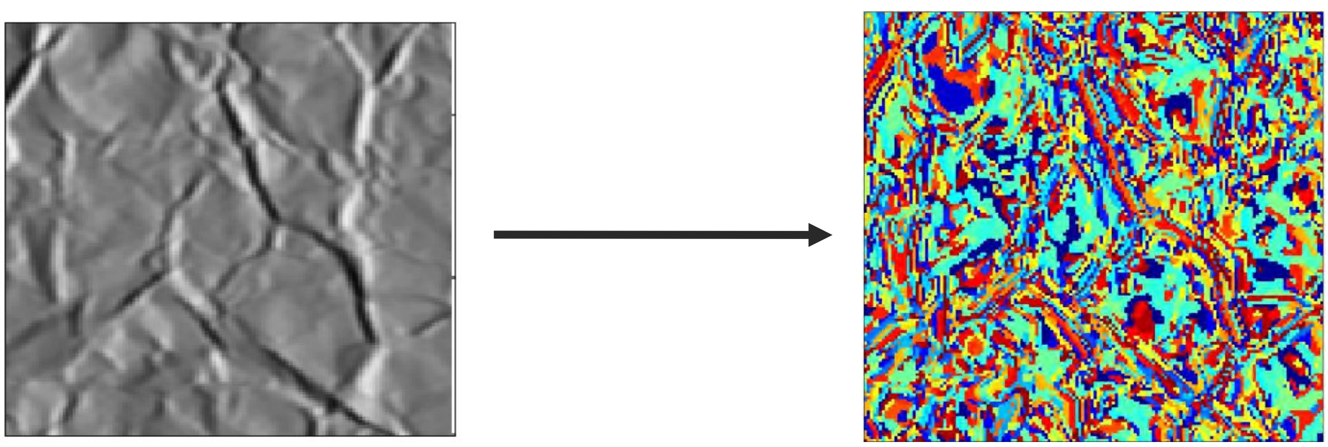






Texton Labeling

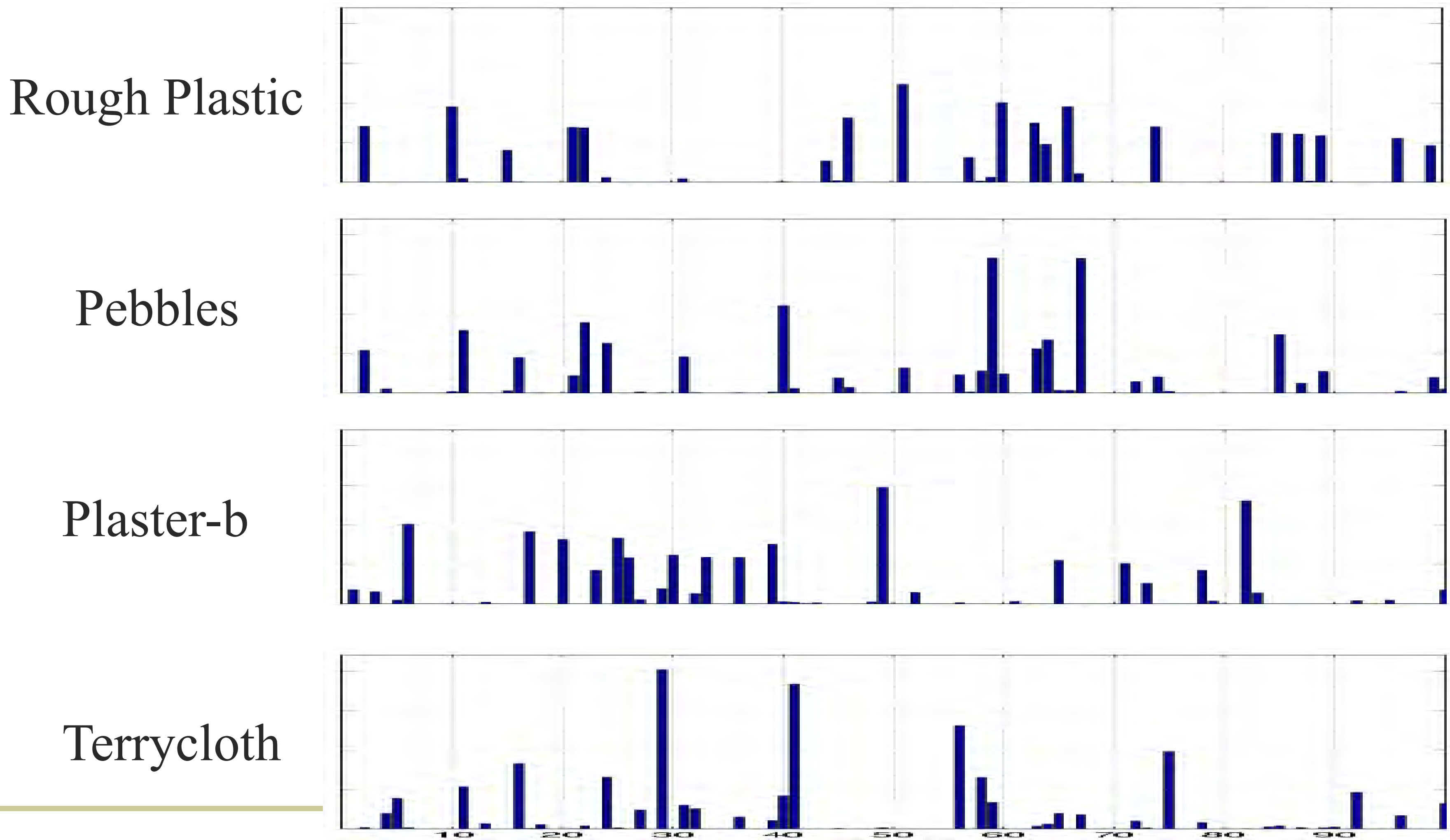
- Each pixel labeled to texton *i* (1 to K) which is *most similar in appearance*;
- Similarity measured by the Euclidean distance between the filter responses;



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)



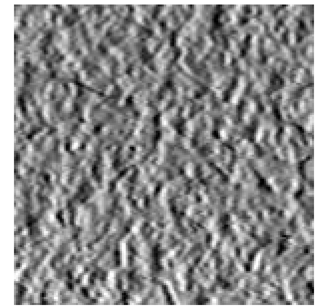
Similarity of materials

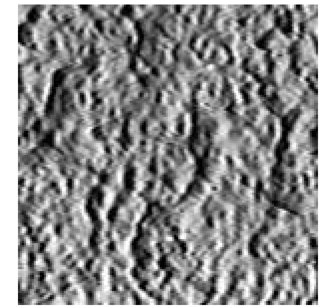
- Similarity between histograms measured using chi-square difference:

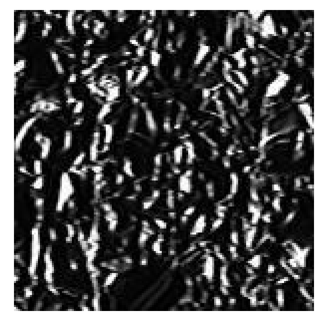
$$\chi^2(h_1, h_2) = \sum_{n=1}^{N} \frac{(h_1(n) - h_2(n))^2}{h_1(n) + h_2(n)}$$

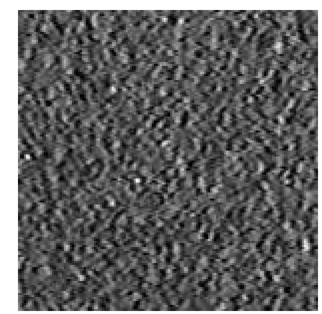
Similarity Matrix

	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

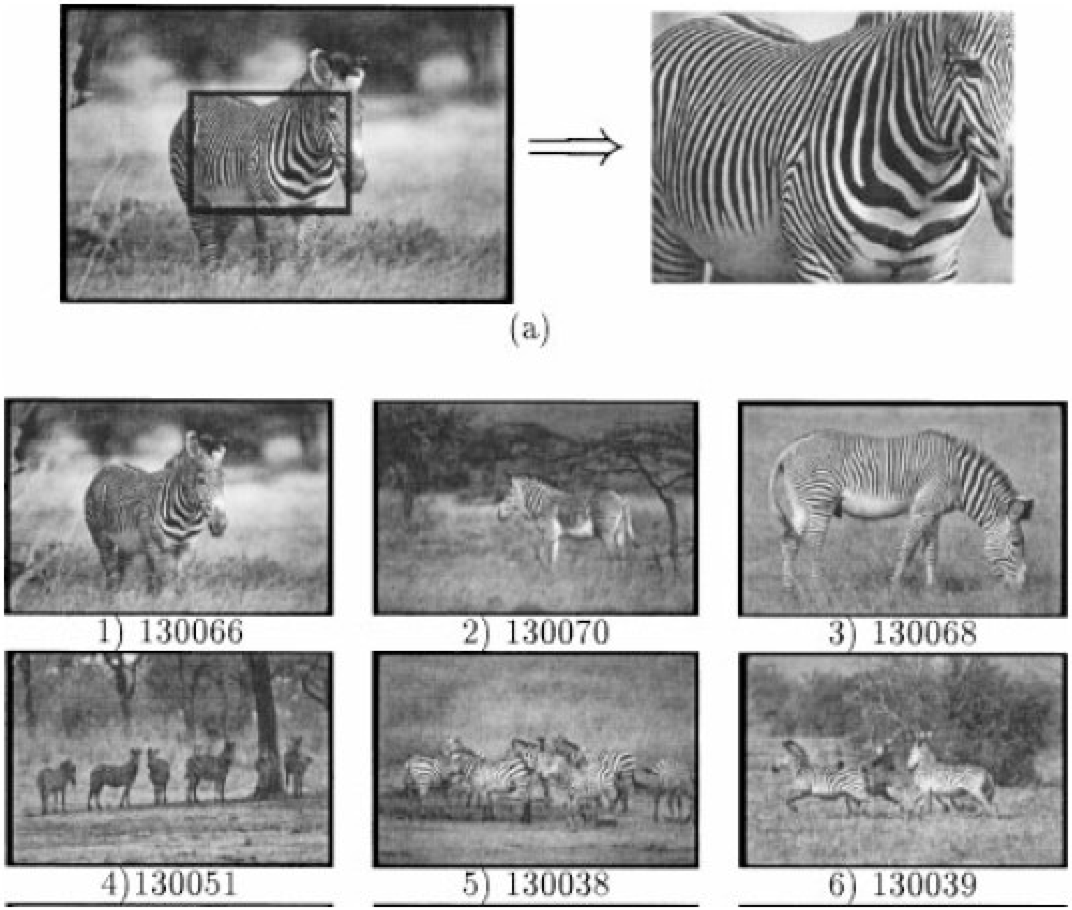
$$e_{ij} = \text{Similarity}(\text{material} = \text{i}\,,\ \text{sample} = \text{j})$$

Example uses of texture in vision: analysis

Classifying materials, "stuff"

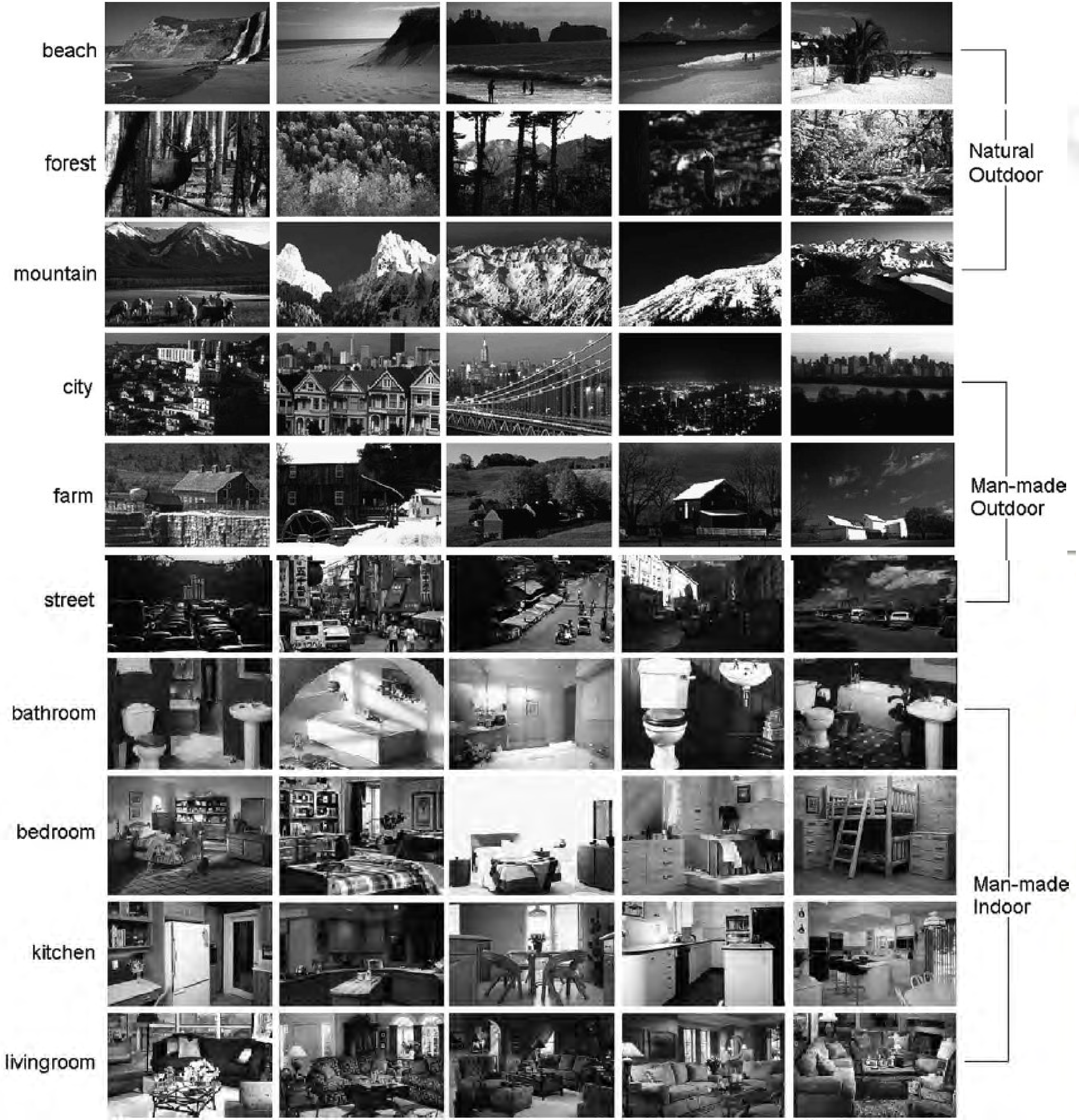
Figure by Varma & Zisserman

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

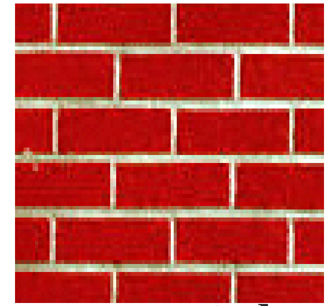
Texture synthesis

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

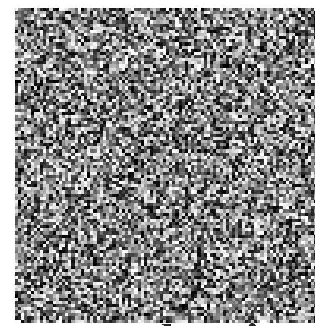
The Challenge

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

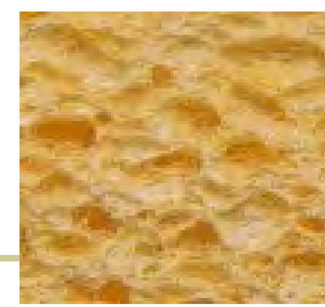
Alexei A. Efros and Thomas K. Leung, “Texture Synthesis by Non-parametric Sampling,” Proc. International Conference on Computer Vision (ICCV), 1999.



repeated

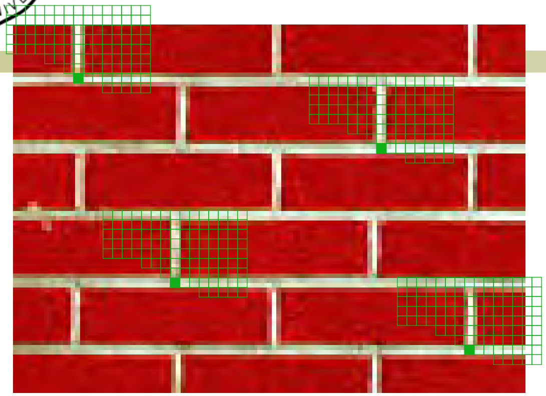


stochastic

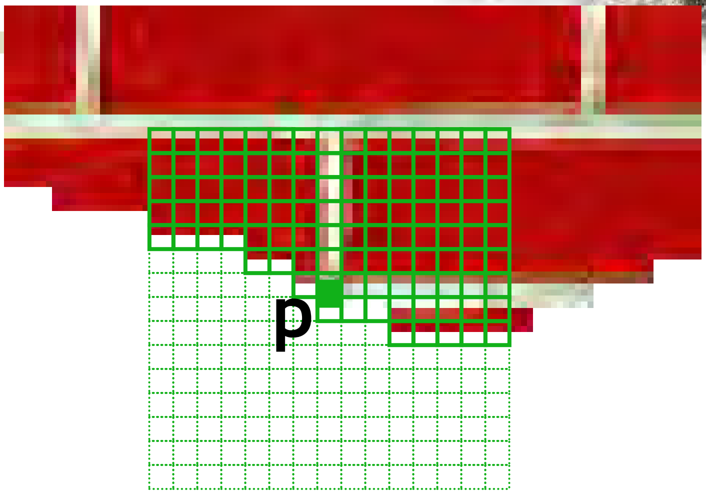


Both?

Synthesizing One Pixel



input image

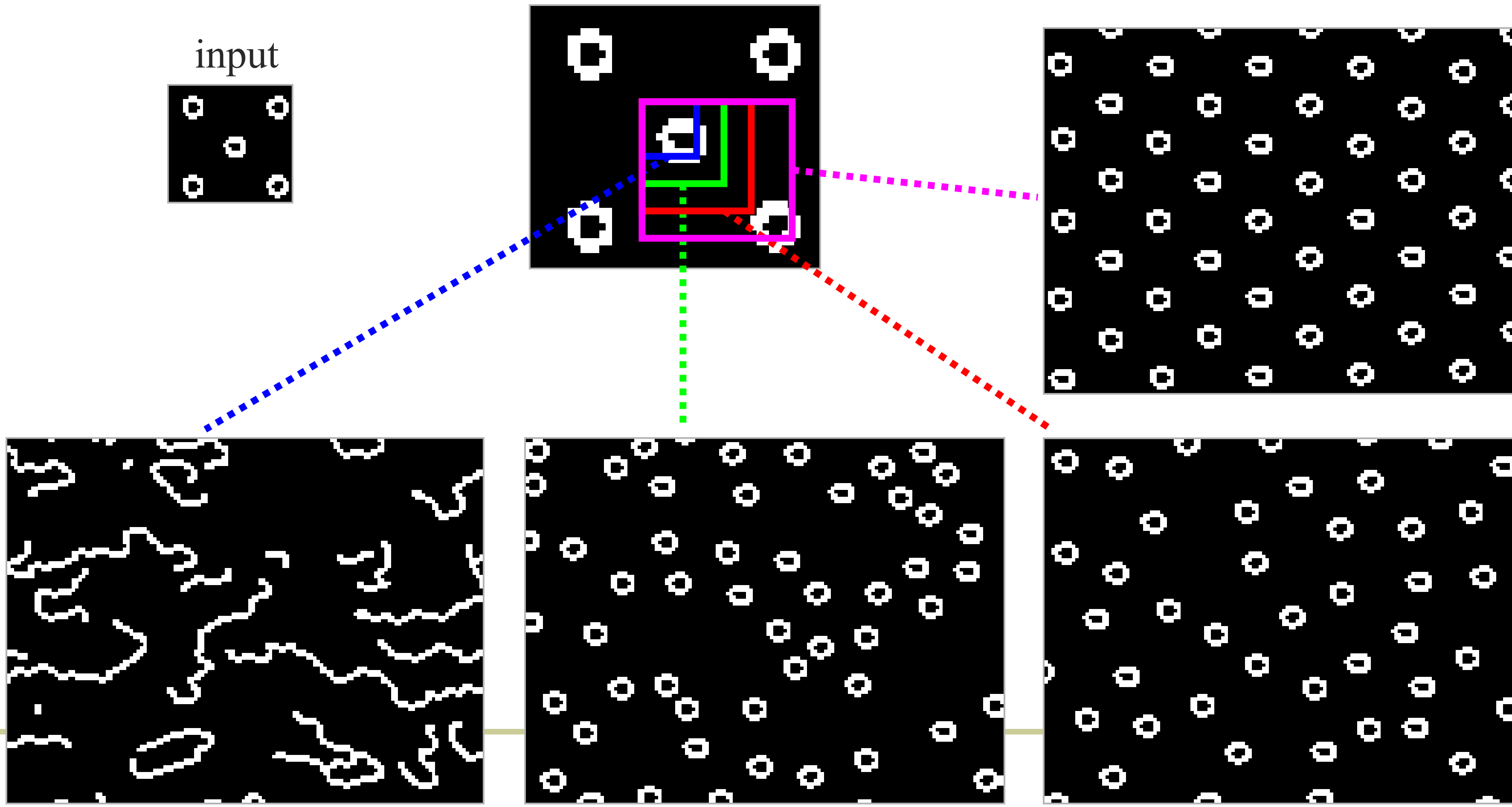


synthesized image

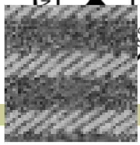
- What is $P(\mathbf{x}|\text{neighborhood of pixels around x})$?
- Find all the windows in the image that match the neighborhood
- To synthesize **x**
  - pick one matching window at random
  - assign **x** to be the center pixel of that window

Slide from Alyosha Efros, ICCV 1999

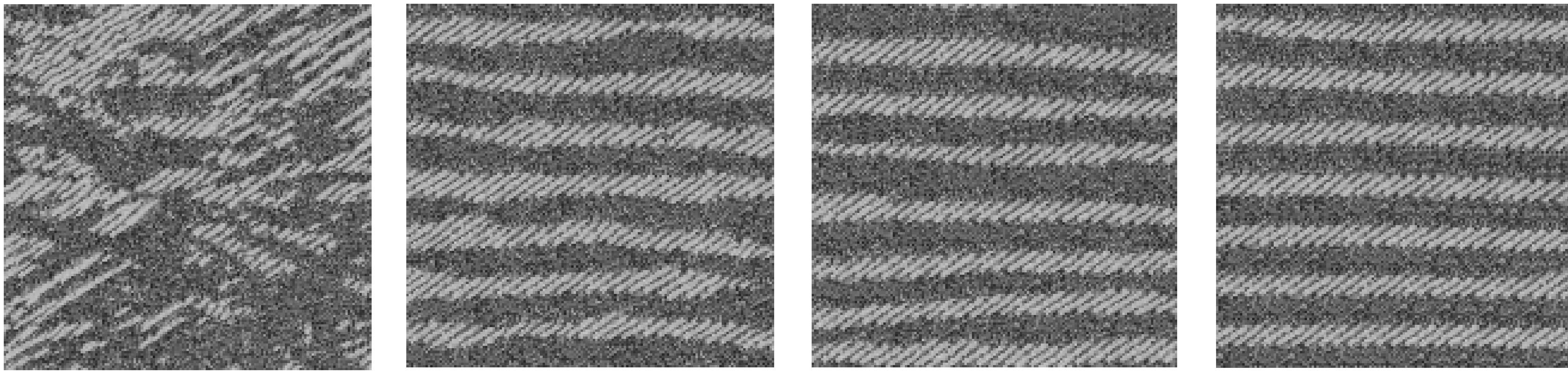
Neighborhood Window

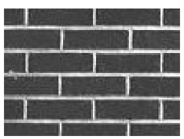


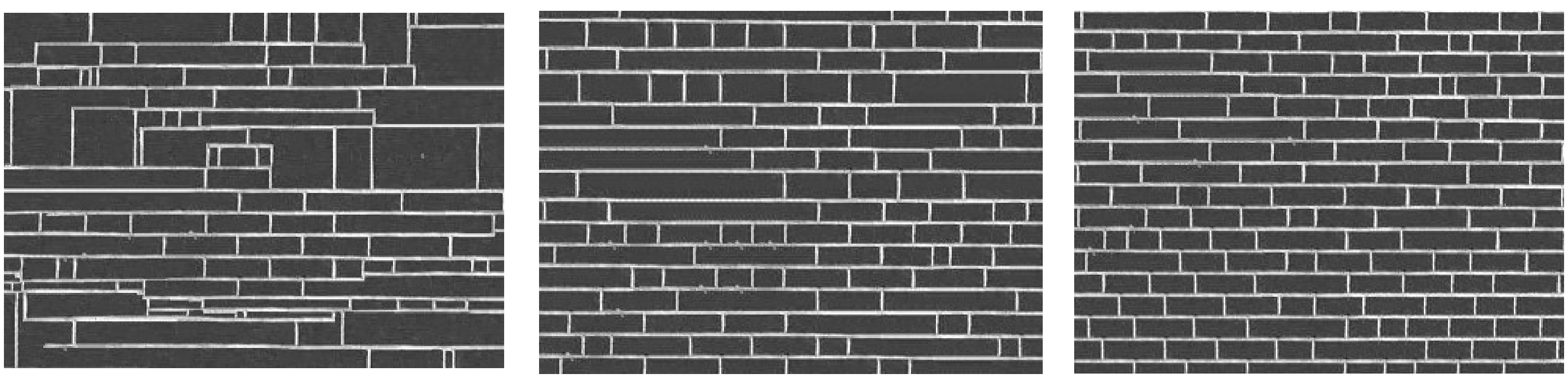
Slide from Alyosha Efros, ICCV 1999



Varying Window Size





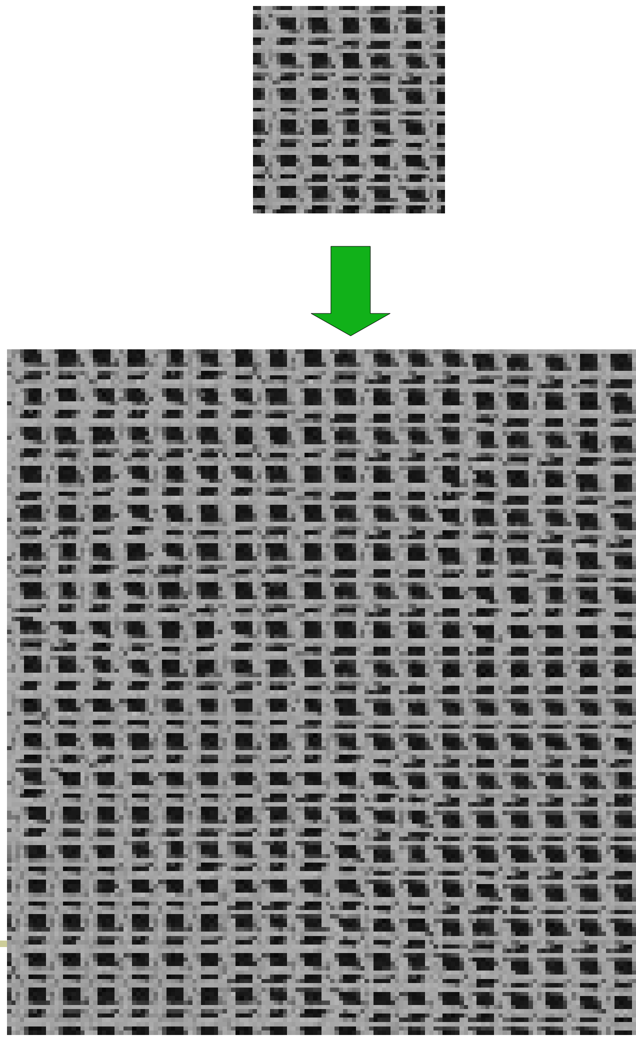


Increasing window size

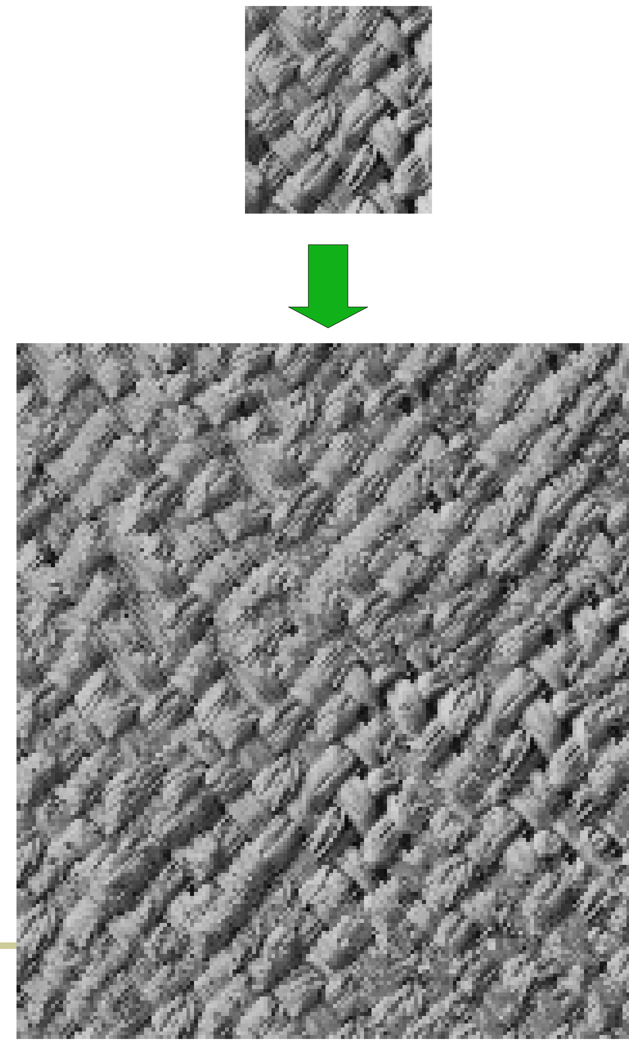
Slide from Alyosha Efros, ICCV 1999

Synthesis results

french canvas



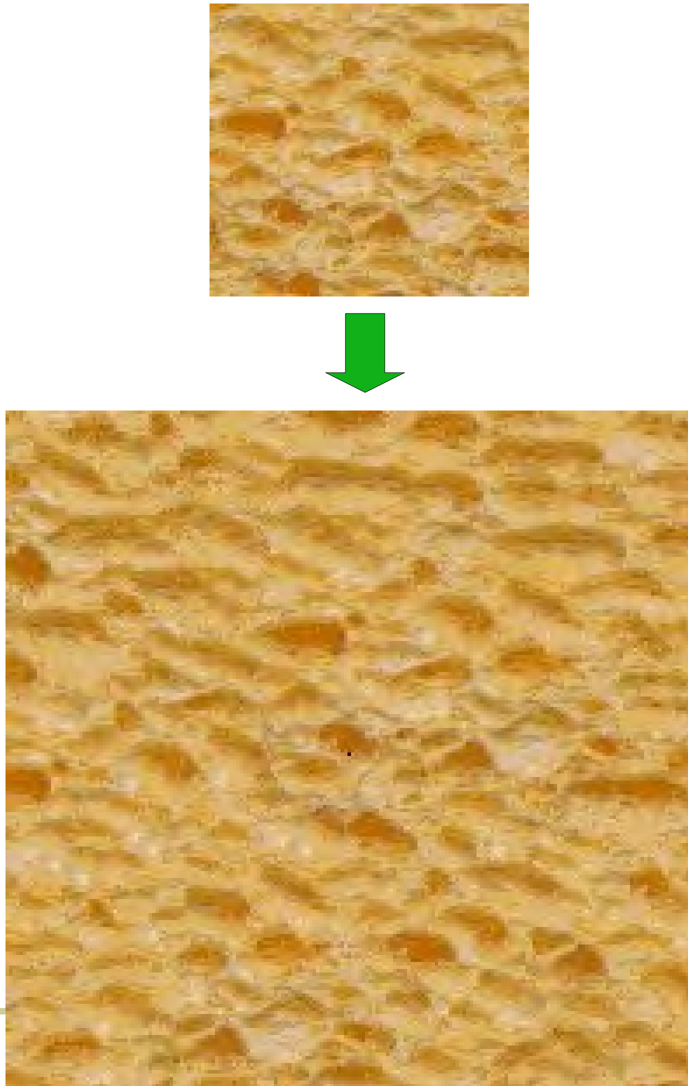
rafia weave



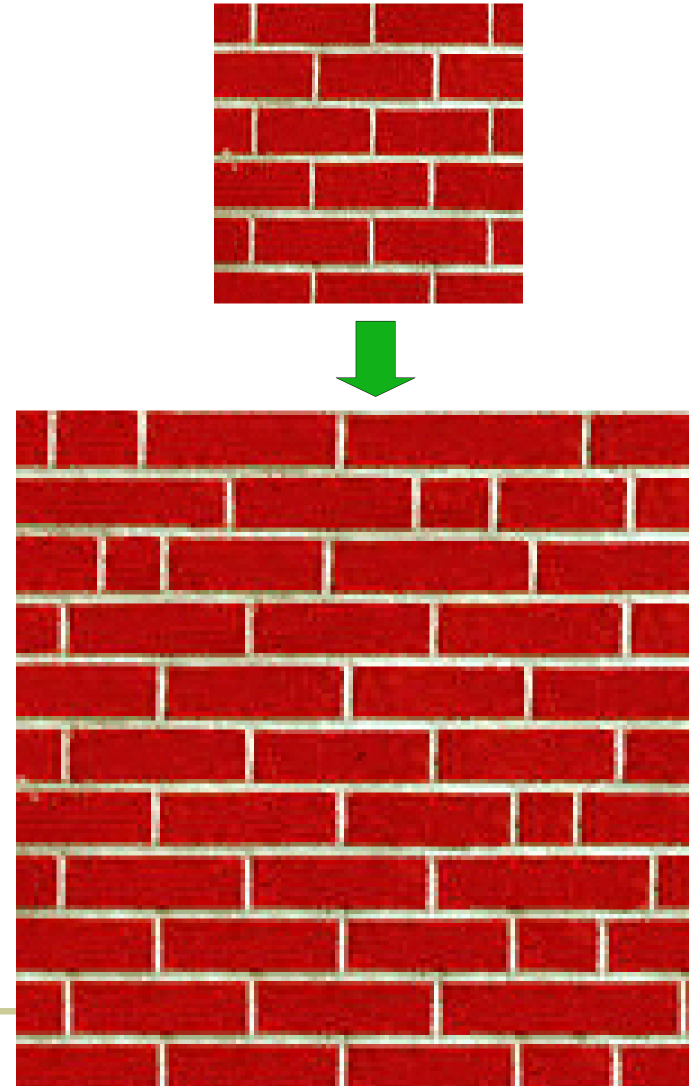
Slide from Alyosha Efros, ICCV 1999

Synthesis results

white bread

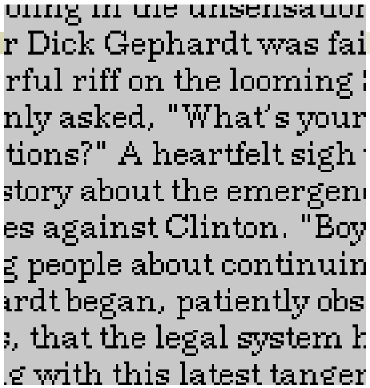


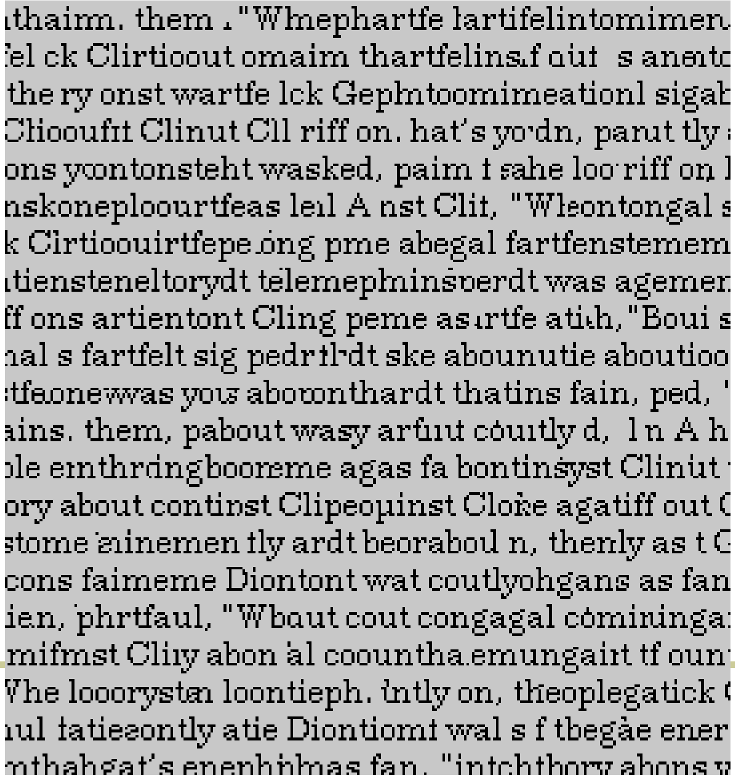
brick wall



Slide from Alyosha Efros, ICCV 1999

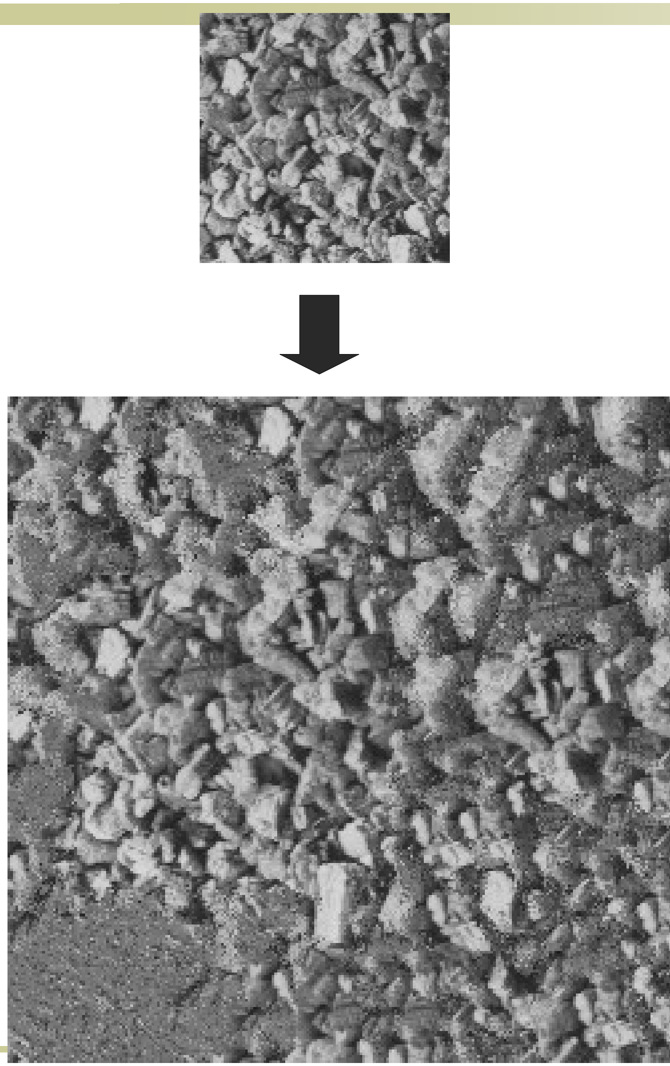
Synthesis results



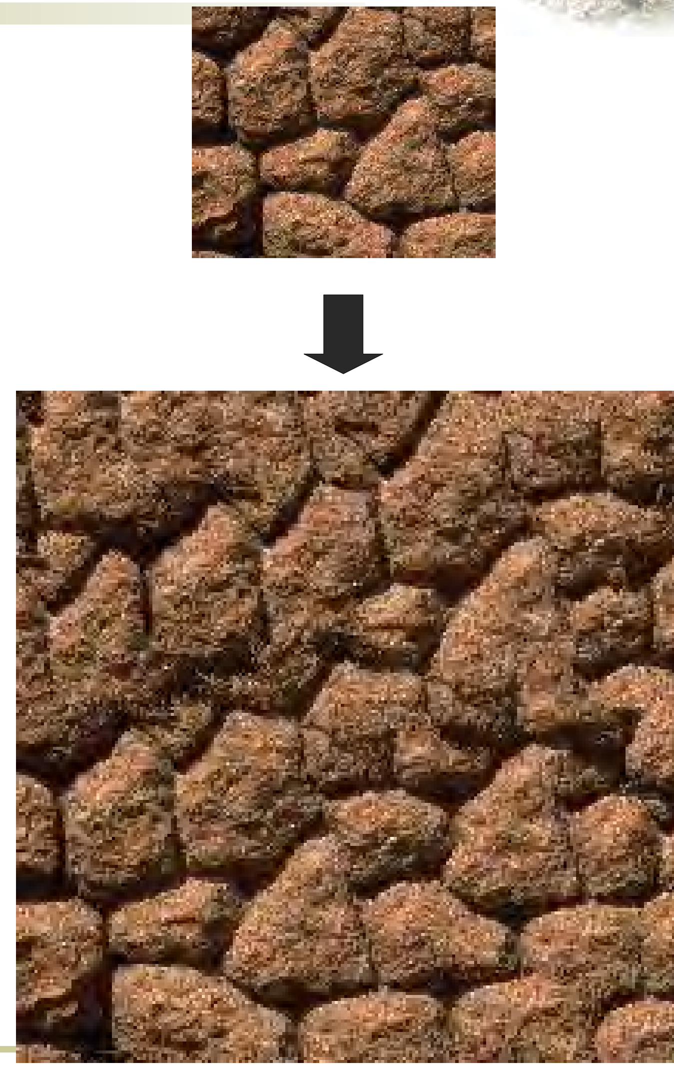


Slide from Alyosha Efros, ICCV 1999

Failure Cases



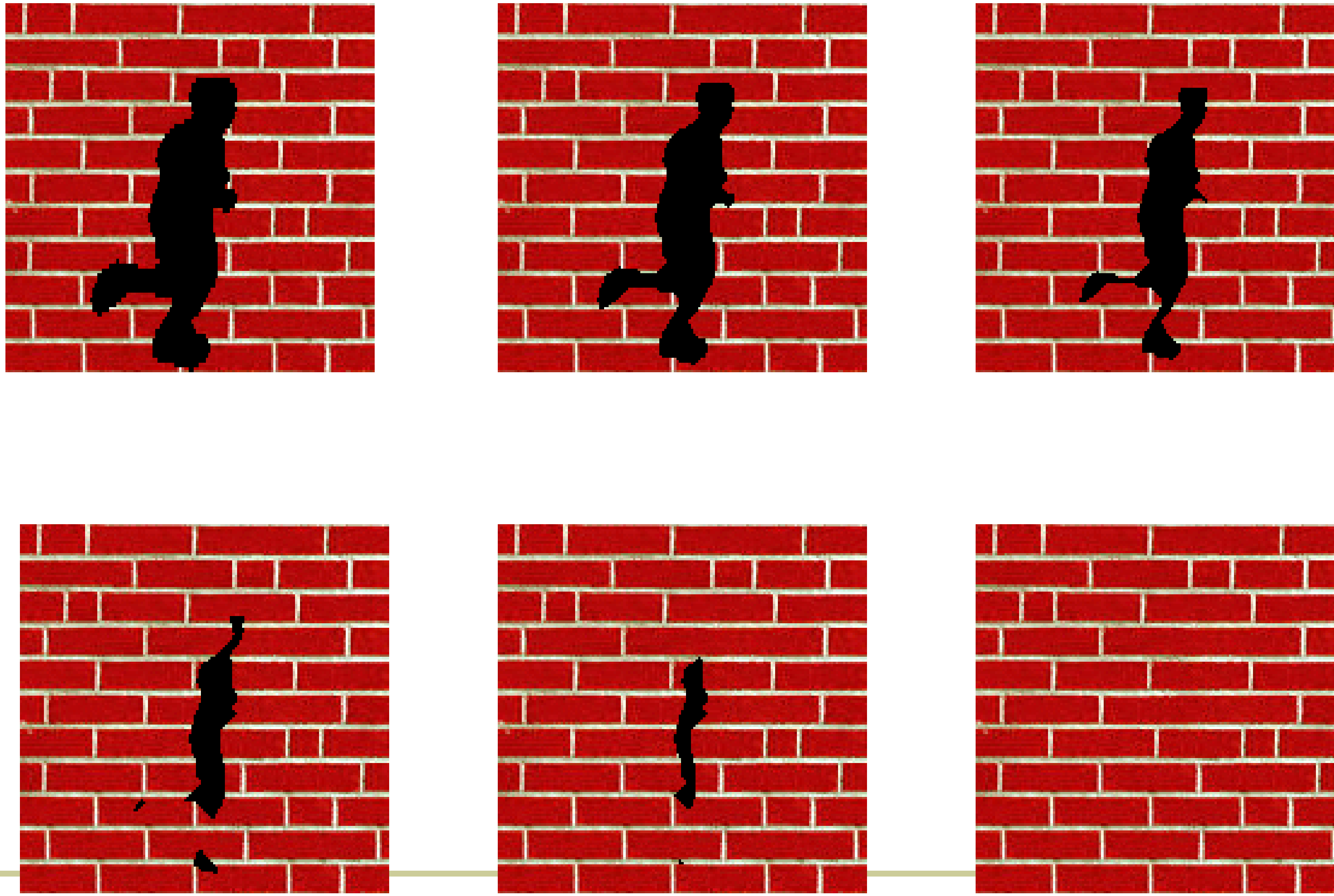
Growing garbage



Verbatim copying

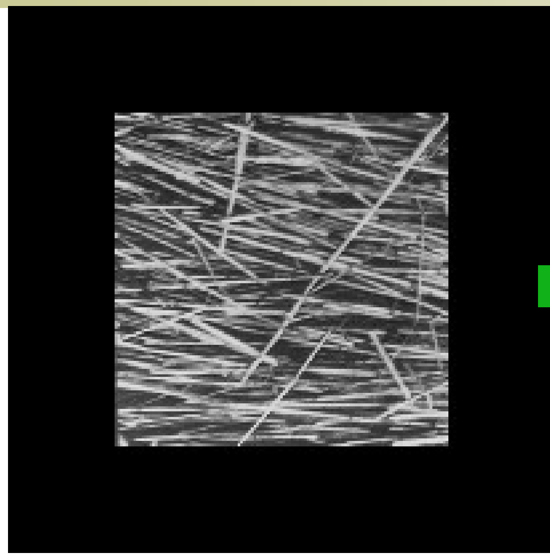
Slide from Alyosha Efros, ICCV 1999

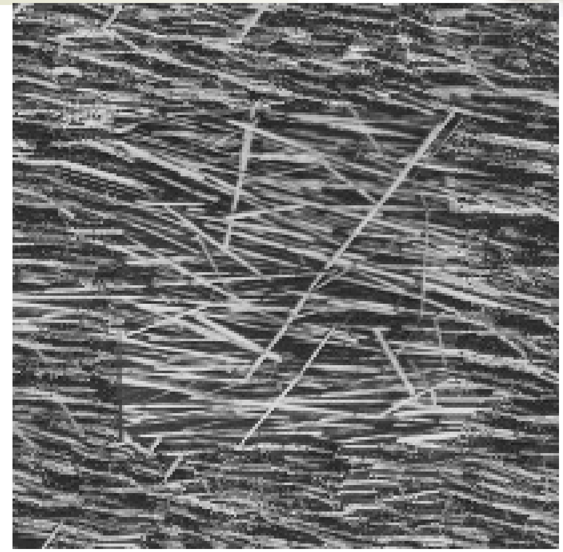
Hole Filling

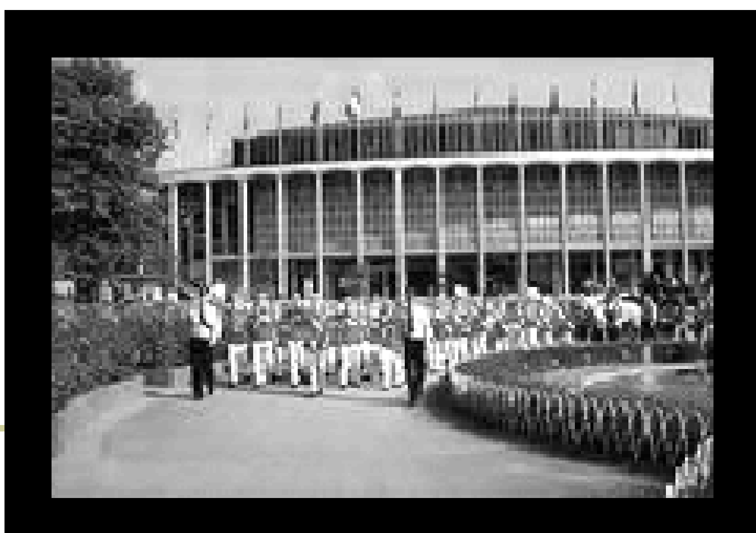


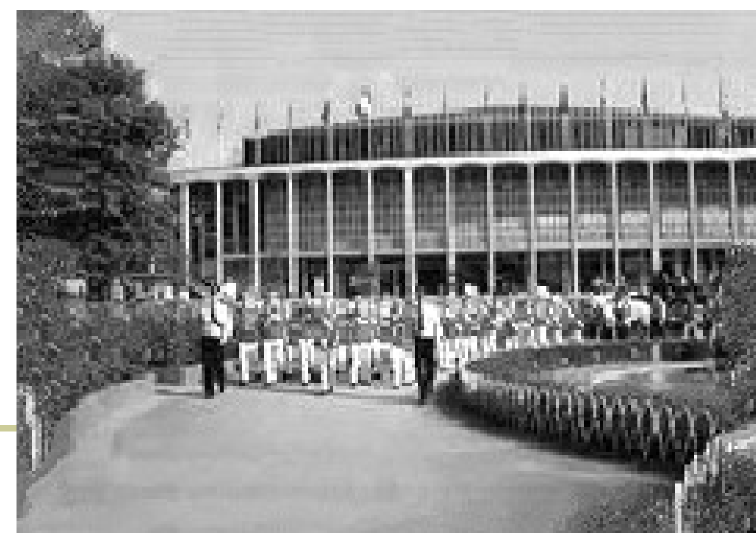
Slide from Alyosha Efros, ICCV 1999

Extrapolation









Slide from Alyosha Efros, ICCV 1999

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