# Pixels and Image Filtering

Computational Photography

Derek Hoiem (presented by Kevin Karsch)

Graphic: http://www.notcot.org/post/4068/

# Administrative stuff

- Any questions?

- Matlab + Linear algebra tutorial
- Office hours
  - Email coming out soon

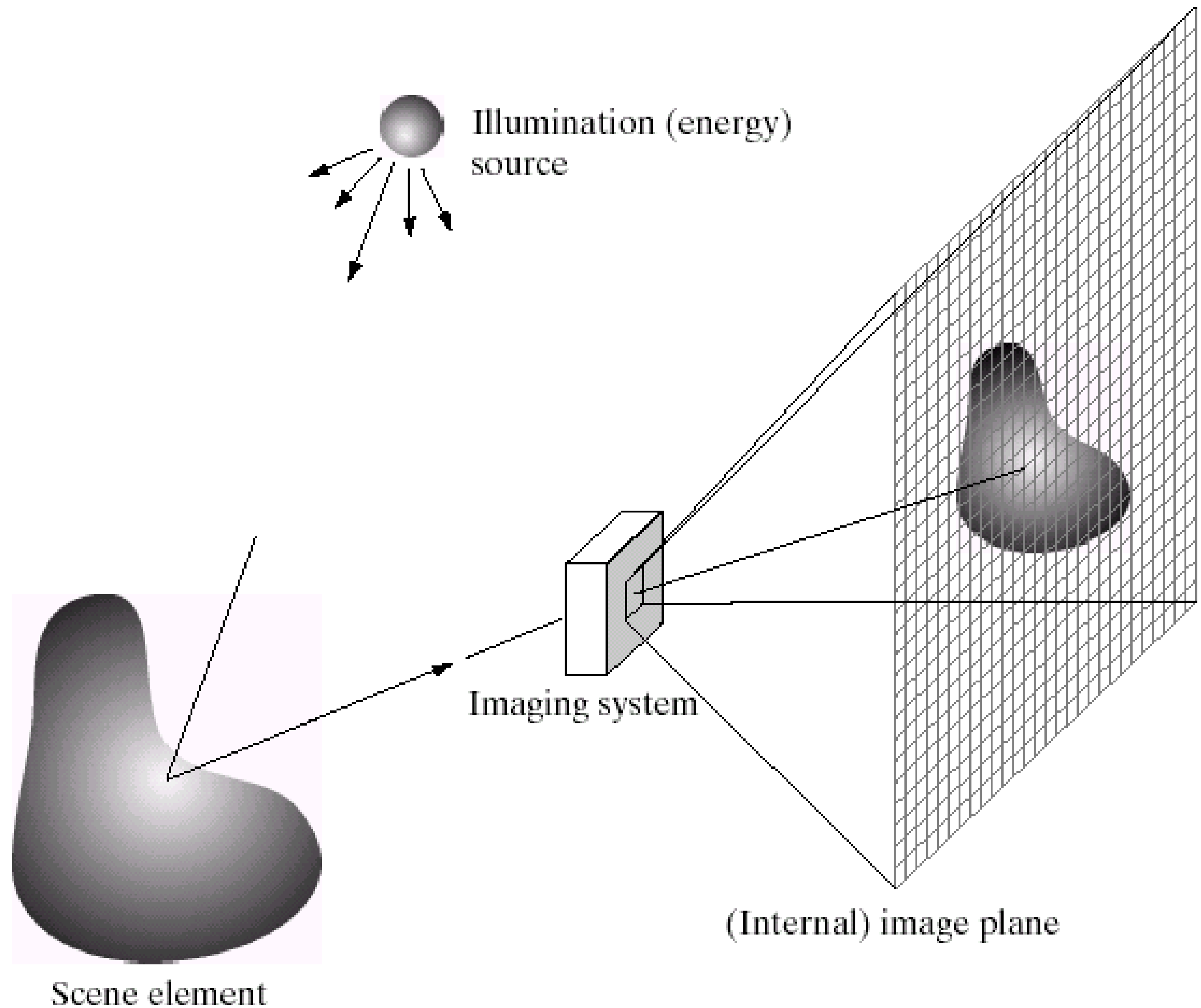
# Today’s Class: Pixels and Linear Filters

- What is a pixel?  How is an image represented?
- What is image filtering and how do we do it?
- Introduce Project 1: Hybrid Images

# Next three classes

- Image filters in spatial domain
  - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
  - Denoising, sampling, image compression
- Templates and Image Pyramids
  - Detection, coarse-to-fine registration

# Image Formation



# Digital camera


## A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons
- Two common types:
  - Charge Coupled Device (CCD): larger yet slower, better quality
  - Complementary Metal Oxide Semiconductor (CMOS): high bandwidth, lower quality
- http://electronics.howstuffworks.com/digital-camera.htm

Slide by Steve Seitz

# Sensor Array

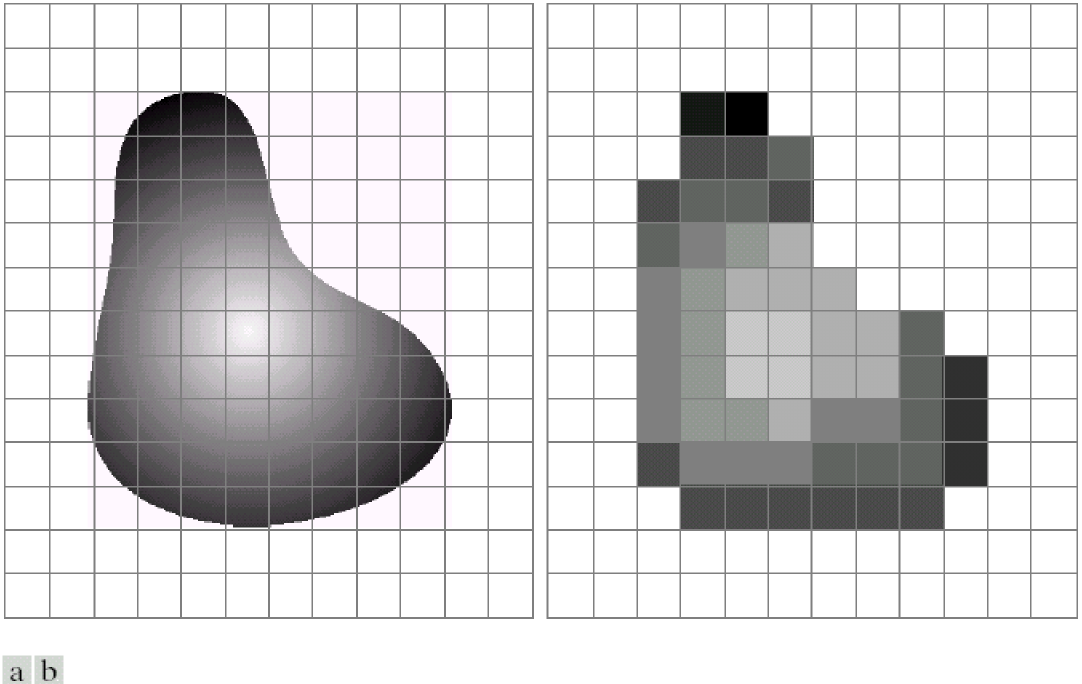
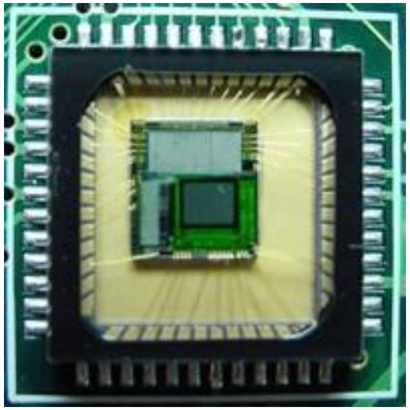
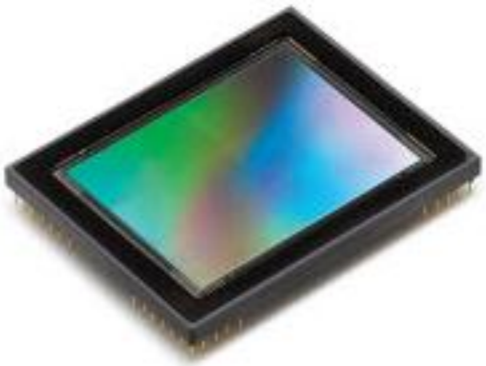



**FIGURE 2.17** (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

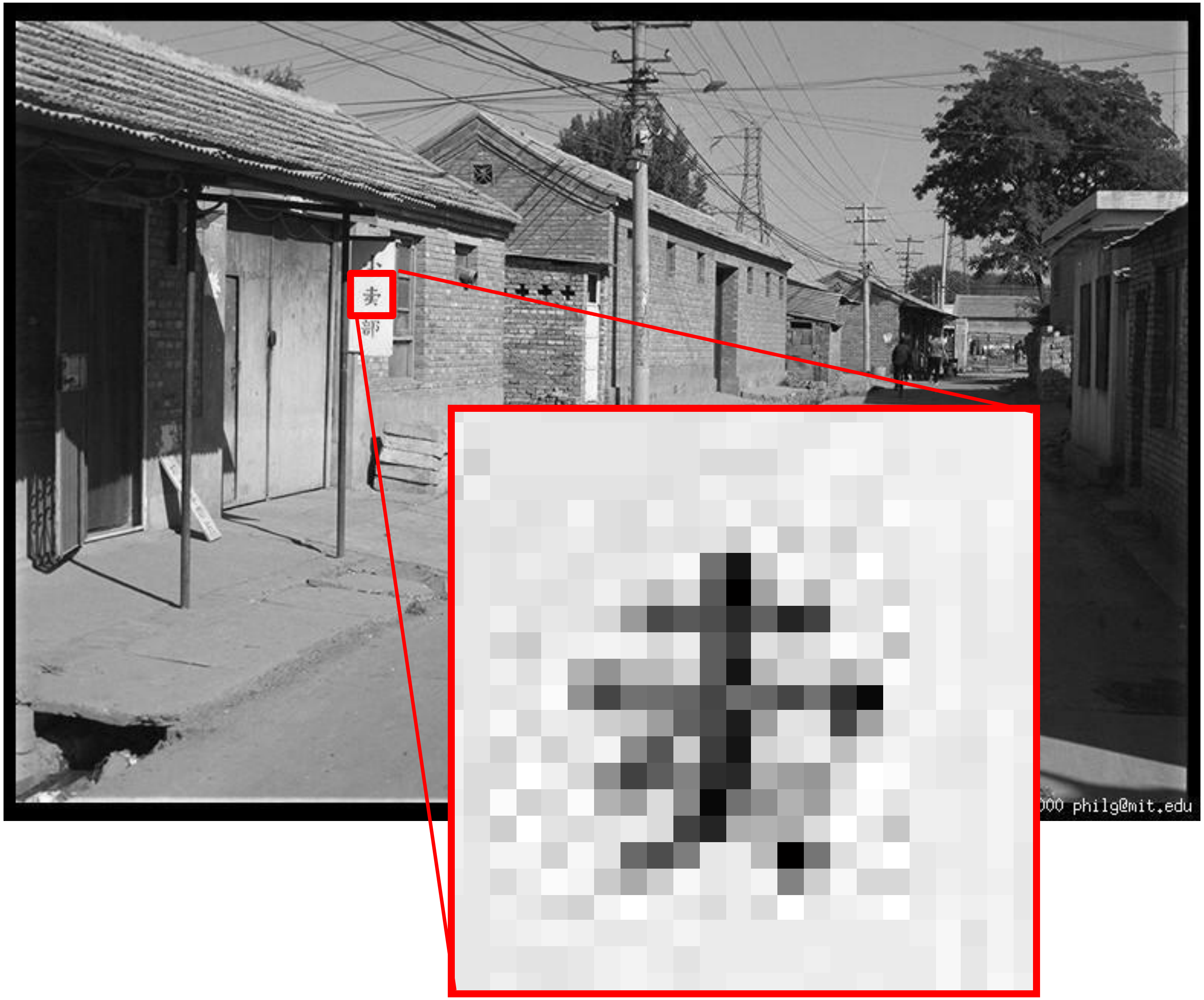


CMOS sensor

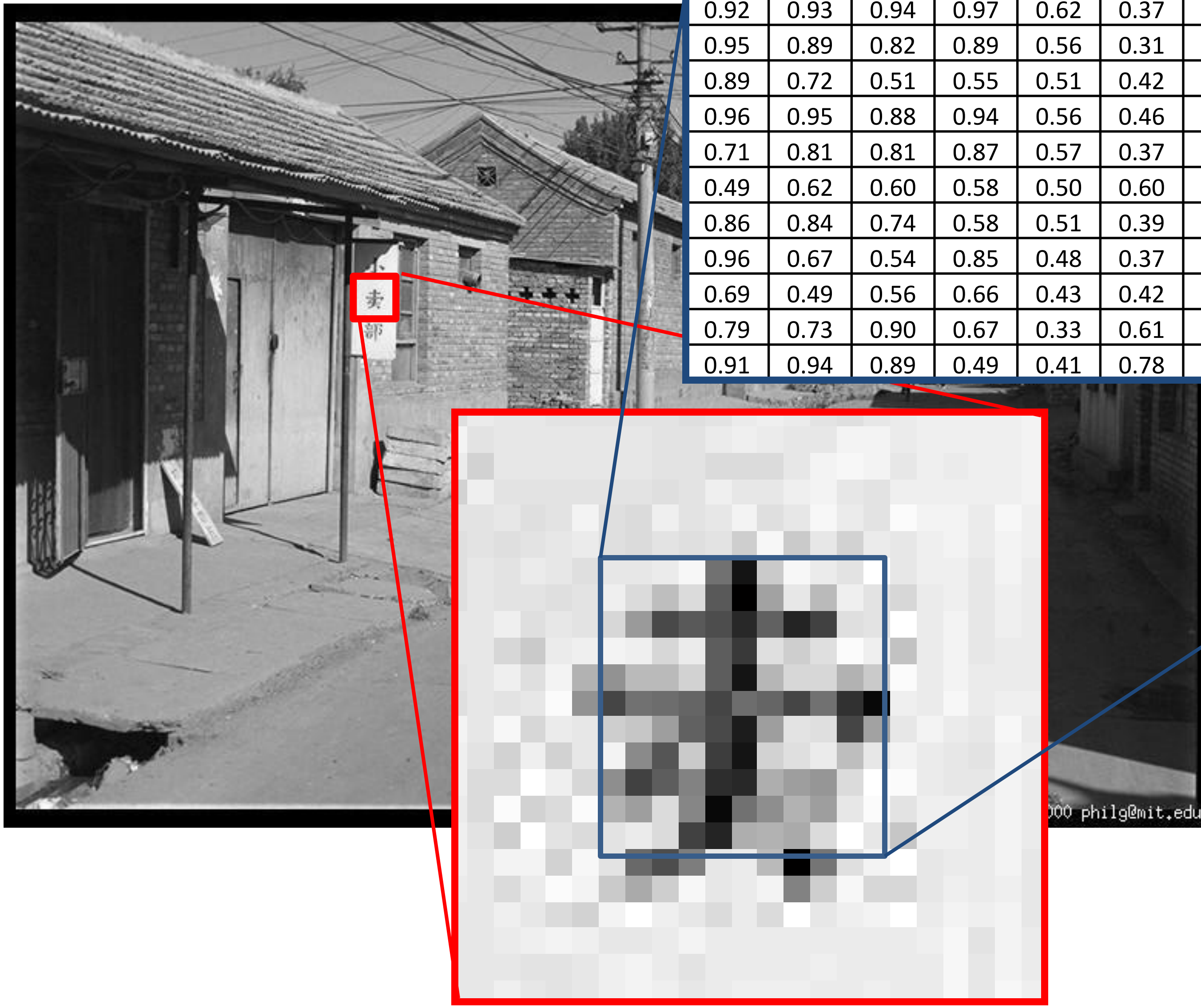


CCD sensor

# The raster image (pixel matrix)

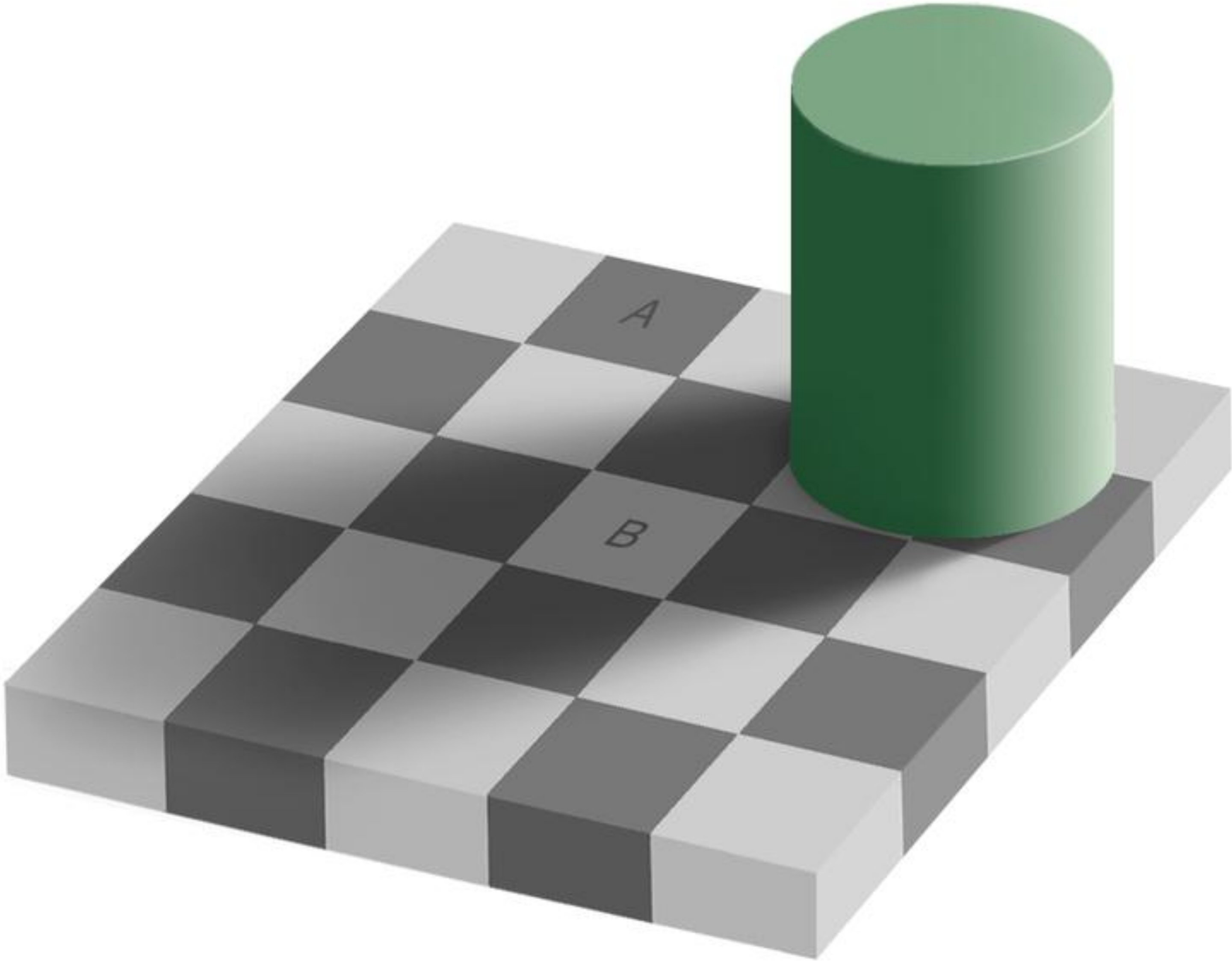


# The raster image (pixel matrix)



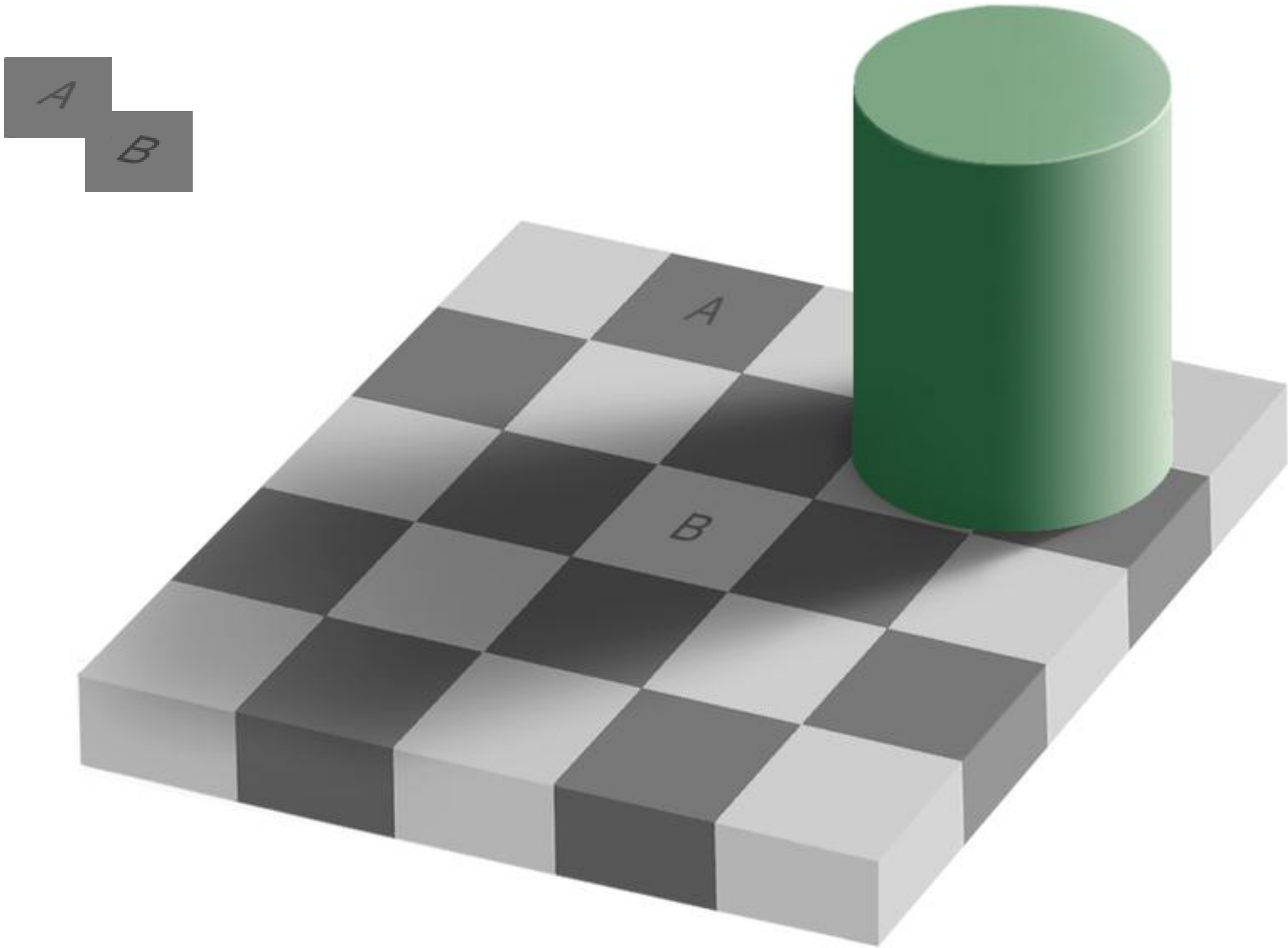

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.92 | 0.93 | 0.94 | 0.97 | 0.62 | 0.37 | 0.85 | 0.97 | 0.93 | 0.92 | 0.99 |
| 0.95 | 0.89 | 0.82 | 0.89 | 0.56 | 0.31 | 0.75 | 0.92 | 0.81 | 0.95 | 0.91 |
| 0.89 | 0.72 | 0.51 | 0.55 | 0.51 | 0.42 | 0.57 | 0.41 | 0.49 | 0.91 | 0.92 |
| 0.96 | 0.95 | 0.88 | 0.94 | 0.56 | 0.46 | 0.91 | 0.87 | 0.90 | 0.97 | 0.95 |
| 0.71 | 0.81 | 0.81 | 0.87 | 0.57 | 0.37 | 0.80 | 0.88 | 0.89 | 0.79 | 0.85 |
| 0.49 | 0.62 | 0.60 | 0.58 | 0.50 | 0.60 | 0.58 | 0.50 | 0.61 | 0.45 | 0.33 |
| 0.86 | 0.84 | 0.74 | 0.58 | 0.51 | 0.39 | 0.73 | 0.92 | 0.91 | 0.49 | 0.74 |
| 0.96 | 0.67 | 0.54 | 0.85 | 0.48 | 0.37 | 0.88 | 0.90 | 0.94 | 0.82 | 0.93 |
| 0.69 | 0.49 | 0.56 | 0.66 | 0.43 | 0.42 | 0.77 | 0.73 | 0.71 | 0.90 | 0.99 |
| 0.79 | 0.73 | 0.90 | 0.67 | 0.33 | 0.61 | 0.69 | 0.79 | 0.73 | 0.93 | 0.97 |
| 0.91 | 0.94 | 0.89 | 0.49 | 0.41 | 0.78 | 0.78 | 0.77 | 0.89 | 0.99 | 0.93 |

# Perception of Intensity



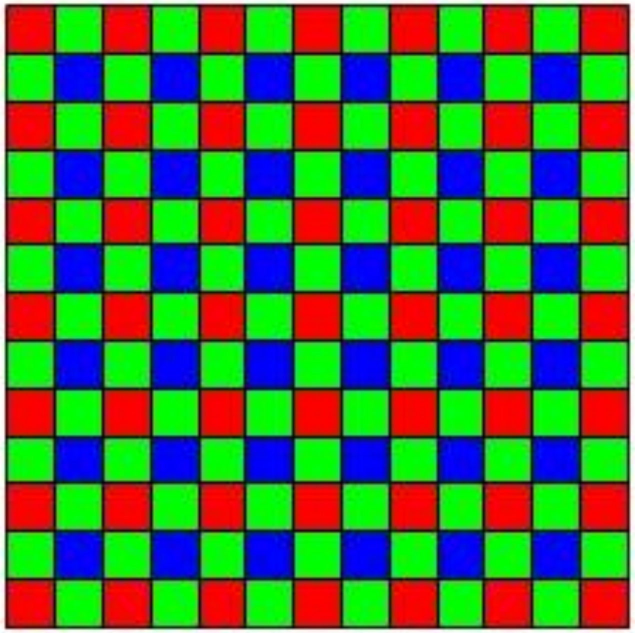

from Ted Adelson

# Perception of Intensity



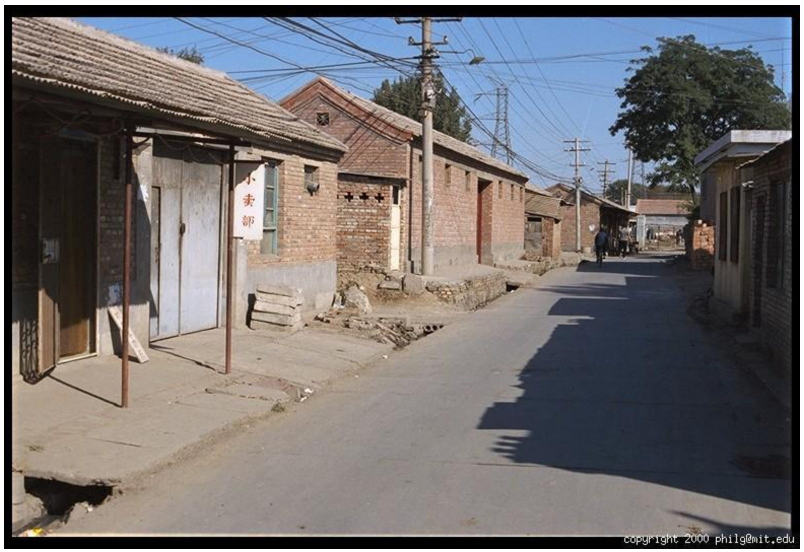

from Ted Adelson

# Digital Color Images



Bayer filter

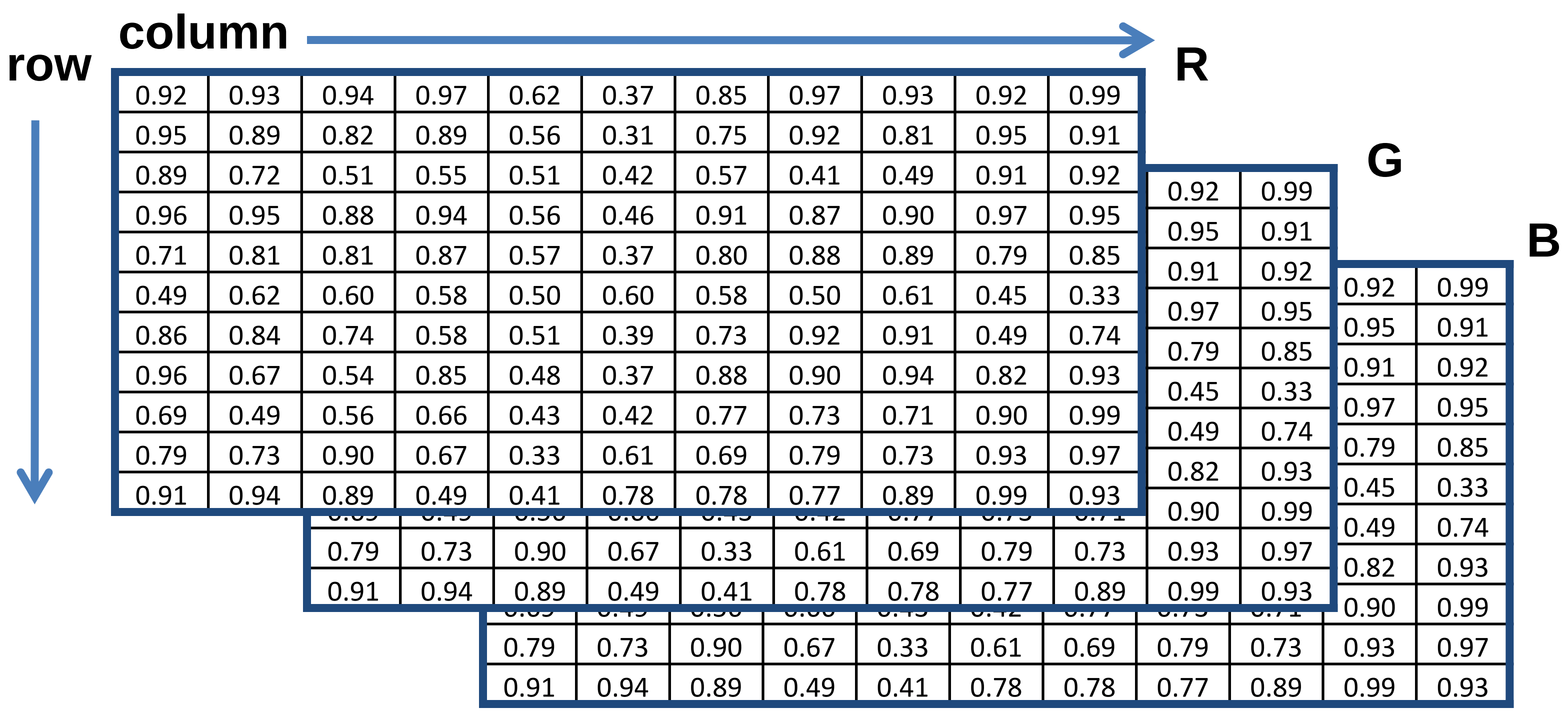
# Color Image


copyright 2000 philg@mit.edu


R
G
B
copyright 2000 philg@mit.edu

# Images in Matlab

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
  - im(1,1,1) = top-left pixel value in R-channel
  - im(y, x, b) = y pixels down, x pixels to right in the b$^{th}$ channel
  - im(N, M, 3) = bottom-right pixel in B-channel
- imread(filename) returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with im2double




| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.92 | 0.93 | 0.94 | 0.97 | 0.62 | 0.37 | 0.85 | 0.97 | 0.93 | 0.92 | 0.99 |
| 0.95 | 0.89 | 0.82 | 0.89 | 0.56 | 0.31 | 0.75 | 0.92 | 0.81 | 0.95 | 0.91 |
| 0.89 | 0.72 | 0.51 | 0.55 | 0.51 | 0.42 | 0.57 | 0.41 | 0.49 | 0.91 | 0.92 |
| 0.96 | 0.95 | 0.88 | 0.94 | 0.56 | 0.46 | 0.91 | 0.87 | 0.90 | 0.97 | 0.95 |
| 0.71 | 0.81 | 0.81 | 0.87 | 0.57 | 0.37 | 0.80 | 0.88 | 0.89 | 0.79 | 0.85 |
| 0.49 | 0.62 | 0.60 | 0.58 | 0.50 | 0.60 | 0.58 | 0.50 | 0.61 | 0.45 | 0.33 |
| 0.86 | 0.84 | 0.74 | 0.58 | 0.51 | 0.39 | 0.73 | 0.92 | 0.91 | 0.49 | 0.74 |
| 0.96 | 0.67 | 0.54 | 0.85 | 0.48 | 0.37 | 0.88 | 0.90 | 0.94 | 0.82 | 0.93 |
| 0.69 | 0.49 | 0.56 | 0.66 | 0.43 | 0.42 | 0.77 | 0.73 | 0.71 | 0.90 | 0.99 |
| 0.79 | 0.73 | 0.90 | 0.67 | 0.33 | 0.61 | 0.69 | 0.79 | 0.73 | 0.93 | 0.97 |
| 0.91 | 0.94 | 0.89 | 0.49 | 0.41 | 0.78 | 0.78 | 0.77 | 0.89 | 0.99 | 0.93 |

# Image filtering

- Image filtering: compute function of local neighborhood at each position
- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

# Example: box filter

$$g[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slide credit: David Lowe (UBC)

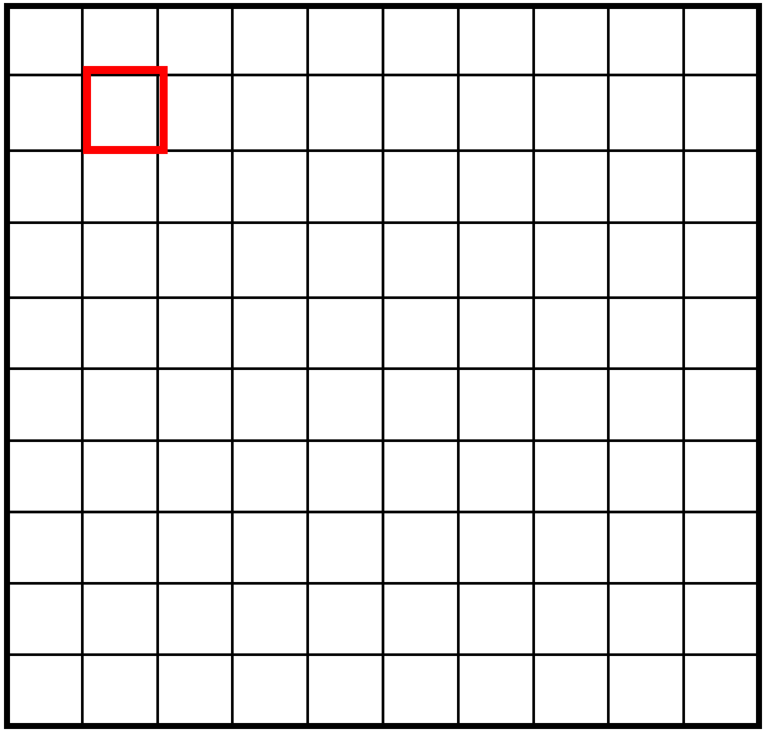
# Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$



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

Credit: S. Seitz

# Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$f[.,.]$$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$h[.,.]$$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Image filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | ? | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | ? | | | |
| | | | | | | | | | |
| | | | | 50 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | | | | | | | |
| 0 | 0 | 0 | | | | | | | |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | 50 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

**Intuitively, what's happening?**

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

Credit: S. Seitz

# Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| | 0 | 30 | 60 | 90 | 90 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| | 10 | 20 | 30 | 30 | 30 | 30 | 20 | 10 | |
| | 10 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | |

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

Credit: S. Seitz

# Box Filter

## What does it do?

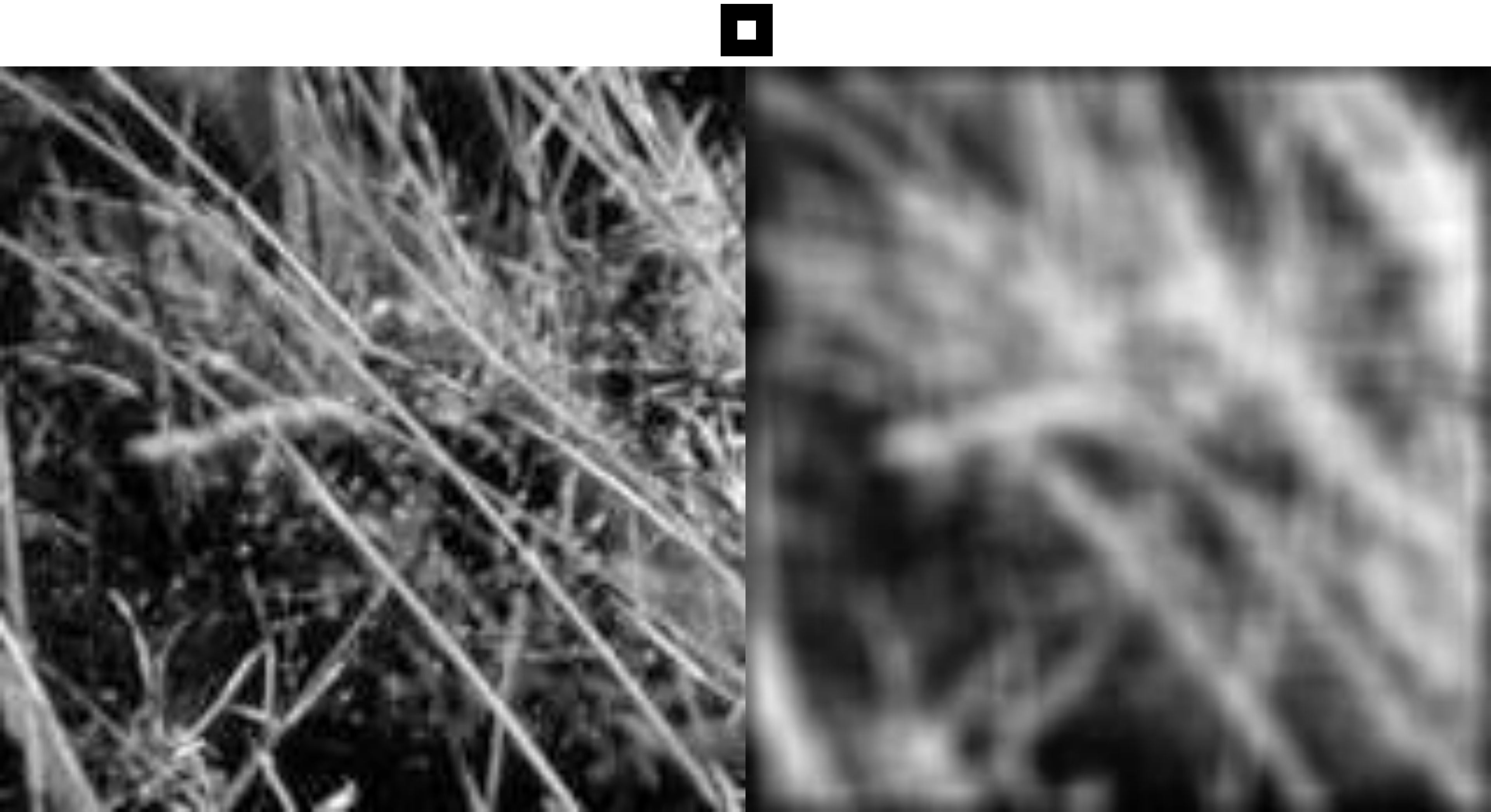
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$$g[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Slide credit: David Lowe (UBC)

# Smoothing with box filter



# One more on board…

# Practice with linear filters

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

?

Source: D. Lowe

# Practice with linear filters

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Filtered
(no change)

Source: D. Lowe

# Practice with linear filters

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

?

# Practice with linear filters

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

Shifted left
By 1 pixel

Source: D. Lowe

# Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad ?$$

(Note that filter sums to 1)

Source: D. Lowe

# Practice with linear filters

Original

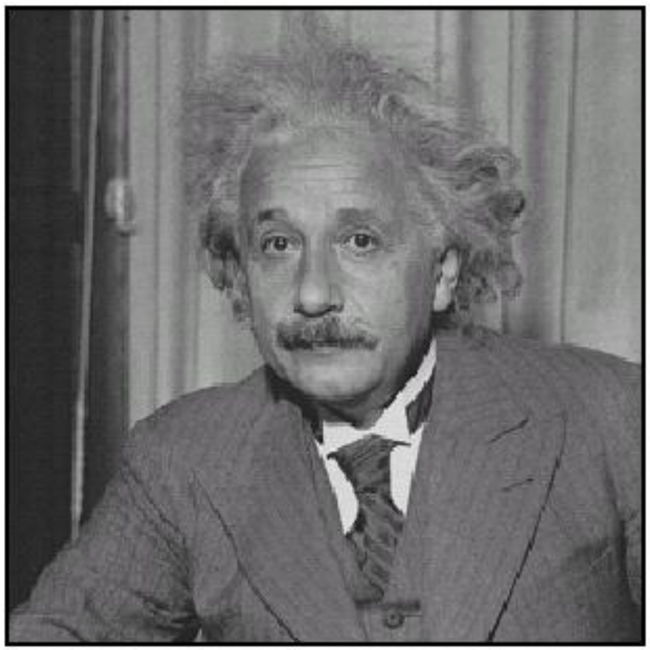
$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

**Sharpening filter**

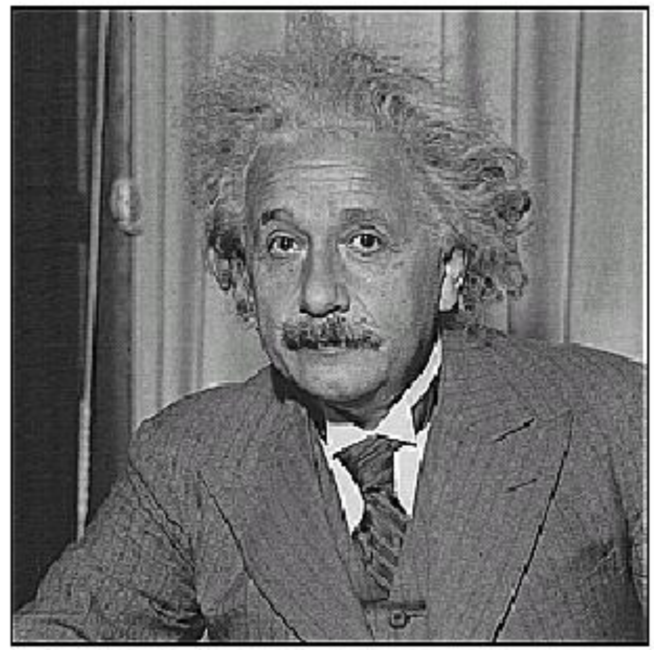
- Accentuates differences with local average

Source: D. Lowe

# Sharpening



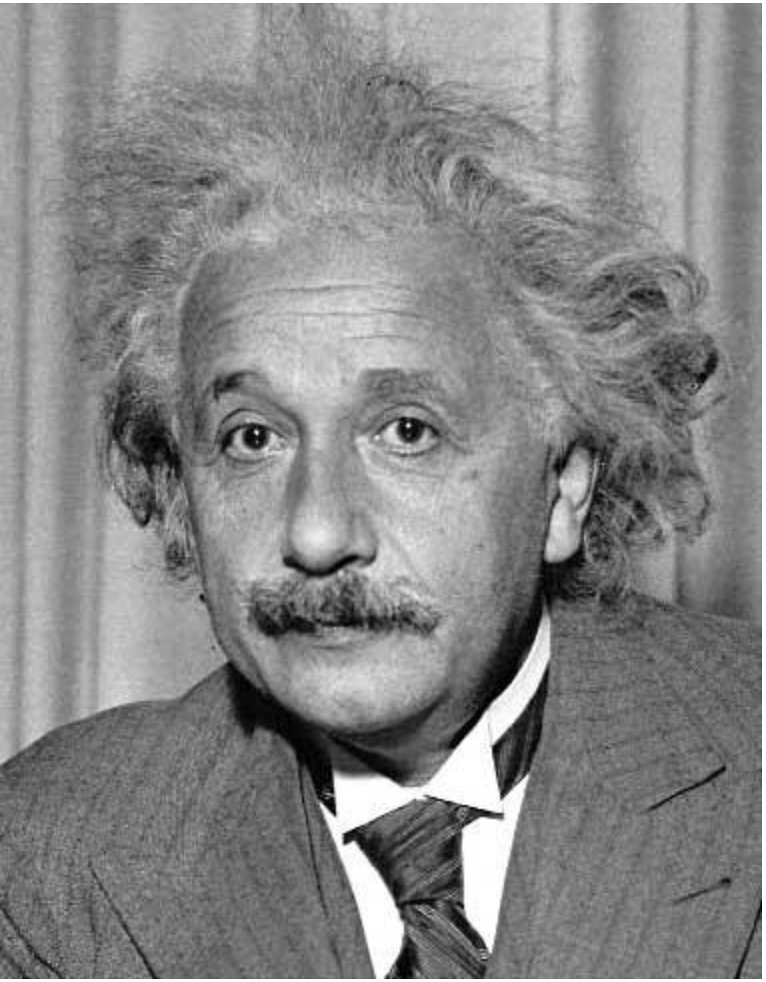
before



after

Source: D. Lowe

# Other filters



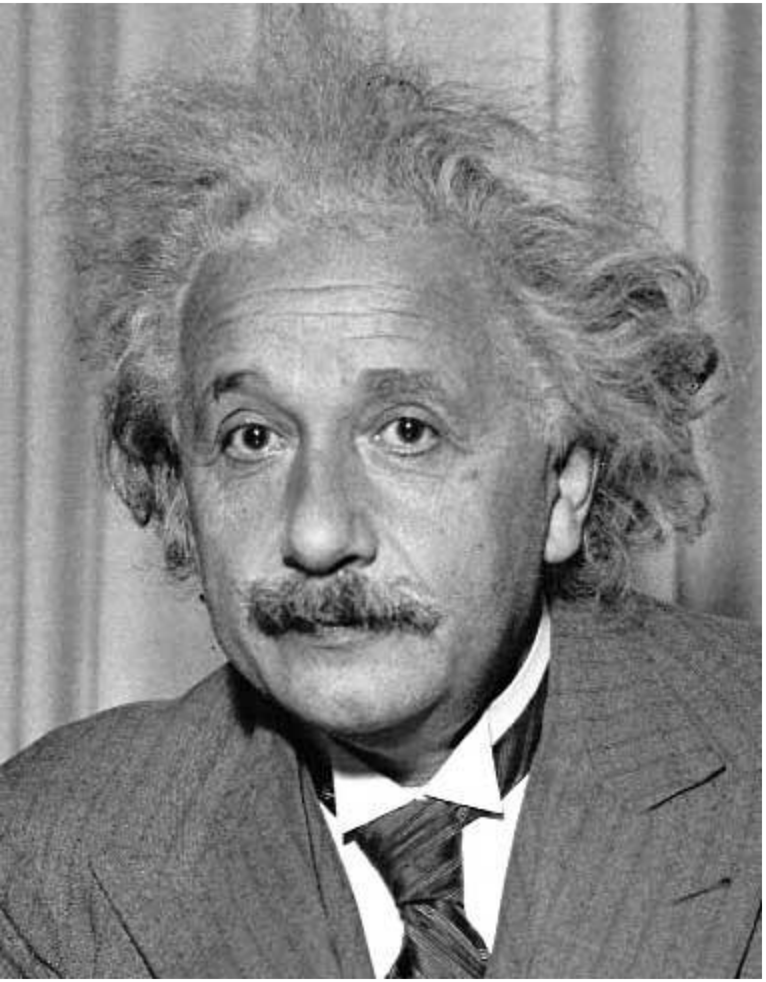
| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Sobel

Vertical Edge
(absolute value)

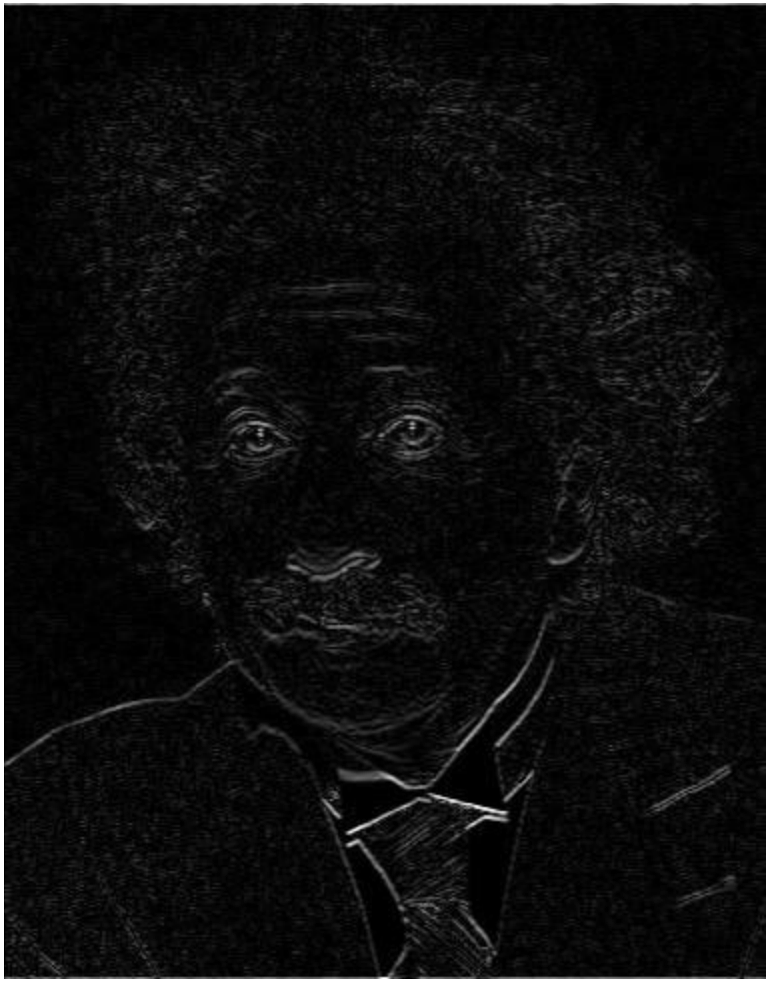
Q?

# Other filters



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

Sobel



Horizontal Edge
(absolute value)

# How could we synthesize motion blur?

```
theta = 30; len = 20;
fil = imrotate(ones(1, len), theta, 'bilinear');
fil = fil / sum(fil(:));
figure(2), imshow(imfilter(im, fil));
```

# Demo

# Filtering vs. Convolution

- 2d filtering

g=filter f=image

  - `h=filter2(g,f);` or
    `h=imfilter(f,g);`

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

- 2d convolution
  - `h=conv2(g,f);`

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

# Key properties of linear filters

**Linearity:**

$\texttt{filter}(f_1 + f_2) = \texttt{filter}(f_1) + \texttt{filter}(f_2)$

**Shift invariance:** same behavior regardless of pixel location

`filter(shift(f)) = shift(filter(f))`

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

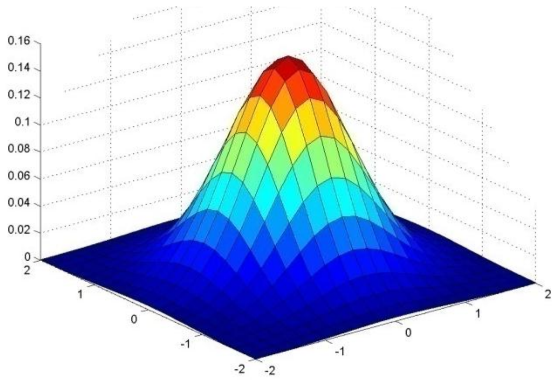
# More properties

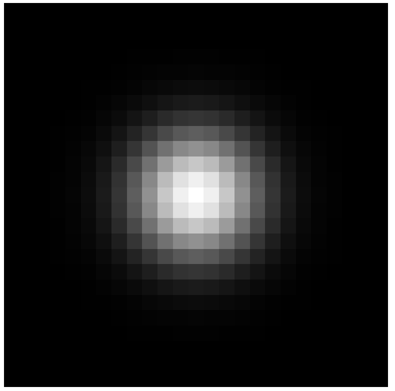
- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal (image)
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k\,(a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

Source: S. Lazebnik

# Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness



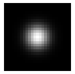
| | | | | |
|---|---|---|---|---|
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.022 | 0.097 | 0.159 | 0.097 | 0.022 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |

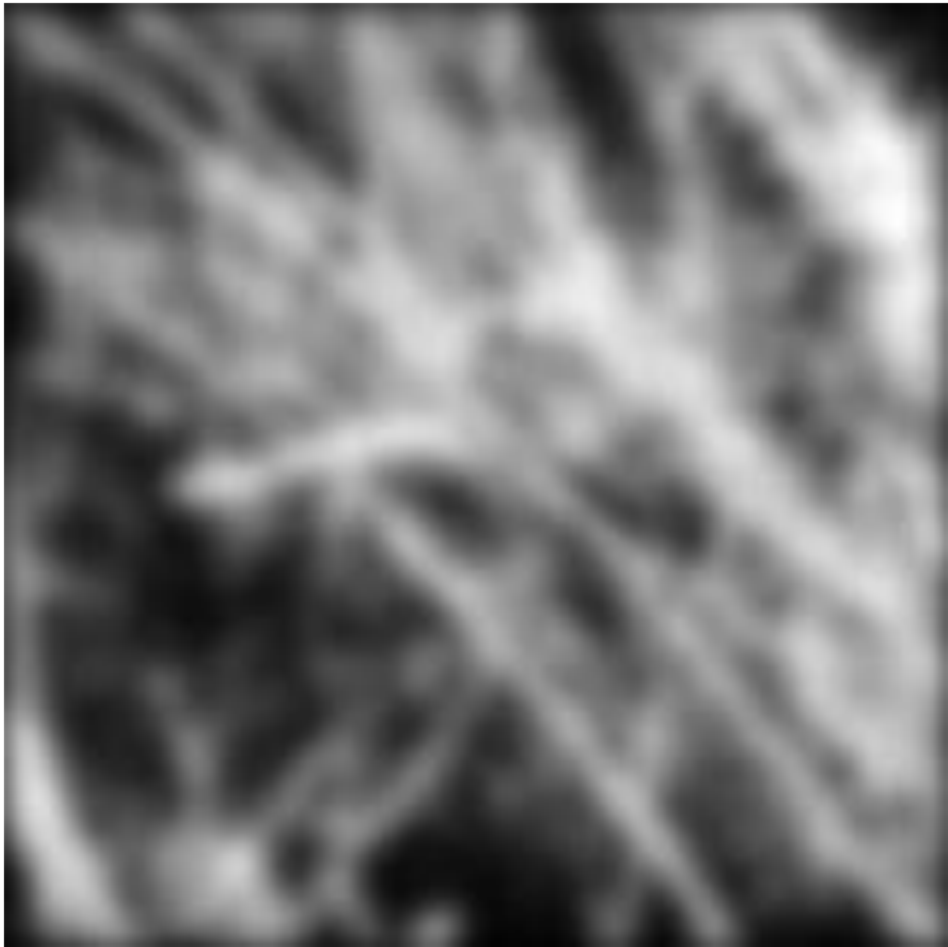
5 x 5, $\sigma$ = 1

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

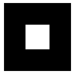
Slide credit: Christopher Rasmussen

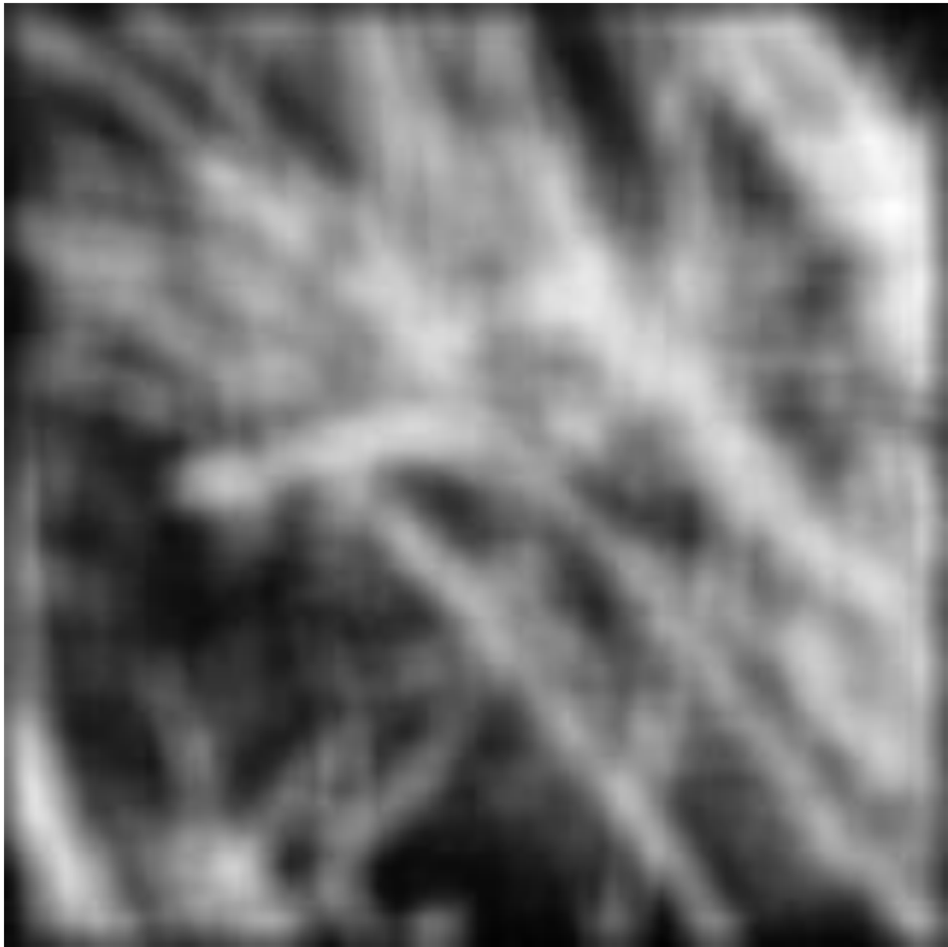
# Smoothing with Gaussian filter





# Smoothing with box filter





# Gaussian filters

- Remove "high-frequency" components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width $\sigma$ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

# Separability of the Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of $x$ and the other a function of $y$

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

# Separability example

2D filtering
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform filtering
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by filtering
along the remaining column:

Source: K. Grauman

# Demo

# Separability

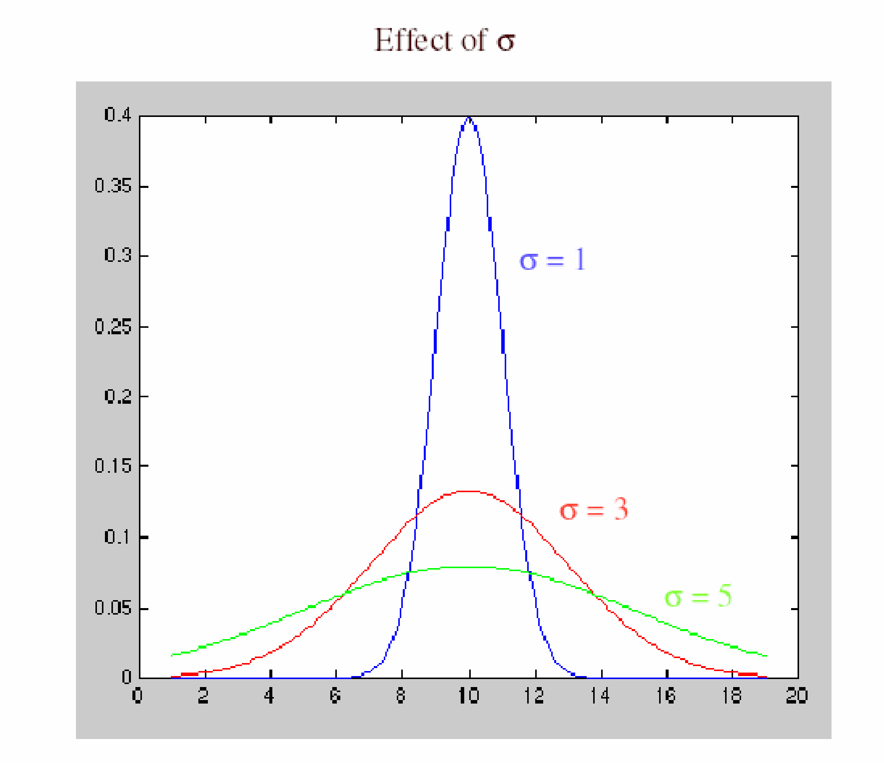
- Why is separability useful in practice?

# Some practical matters

# Practical matters

## How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about 3 $\sigma$



# Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

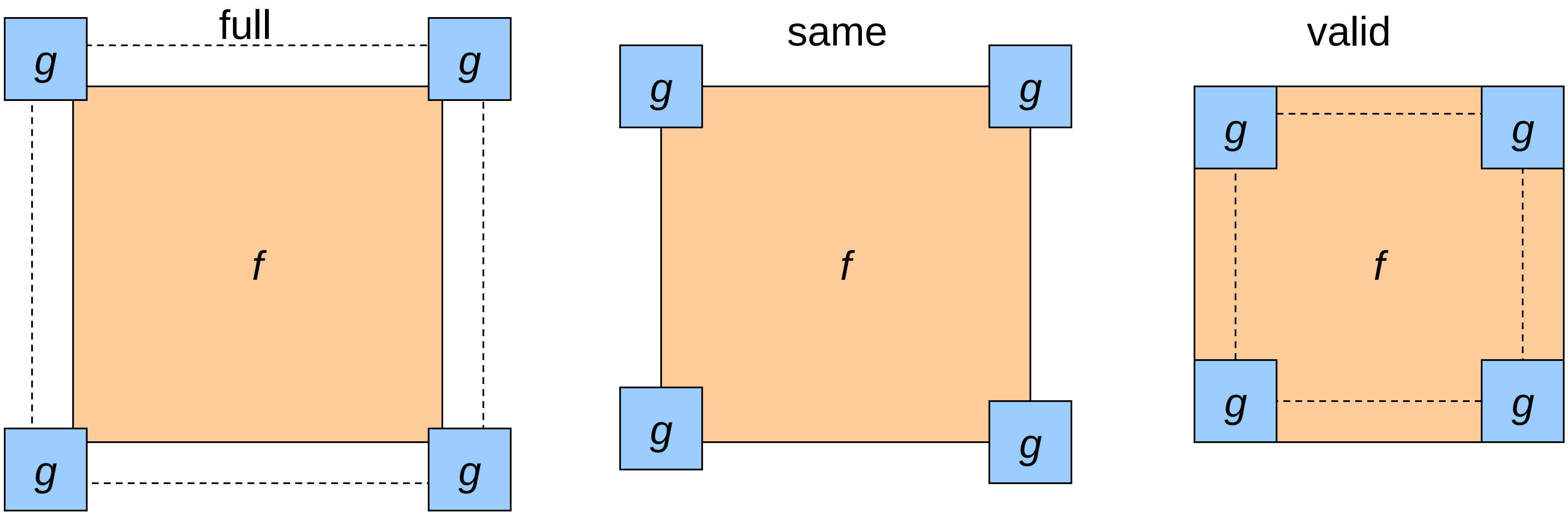
# Practical matters

- methods (MATLAB):
  - clip filter (black): imfilter(f, g, 0)
  - wrap around: imfilter(f, g, ‘circular’)
  - copy edge: imfilter(f, g, ‘replicate’)
  - reflect across edge: imfilter(f, g, ‘symmetric’)

Source: S. Marschner

# Practical matters

- What is the size of the output?
- MATLAB: filter2(g, f, *shape*)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g

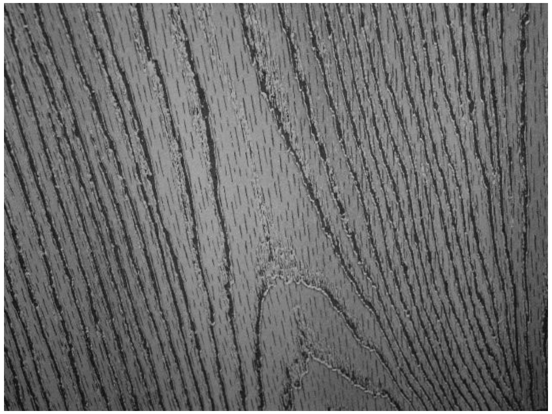


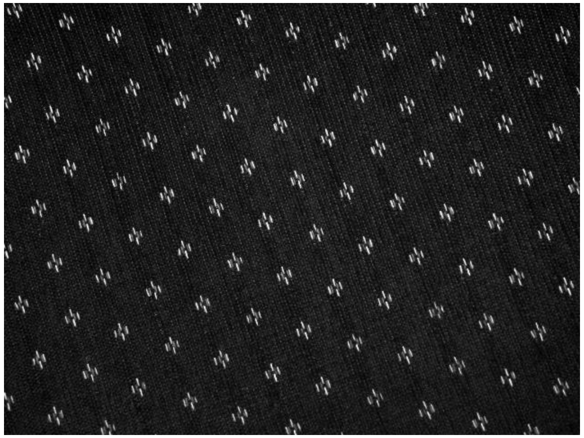

Source: S. Lazebnik

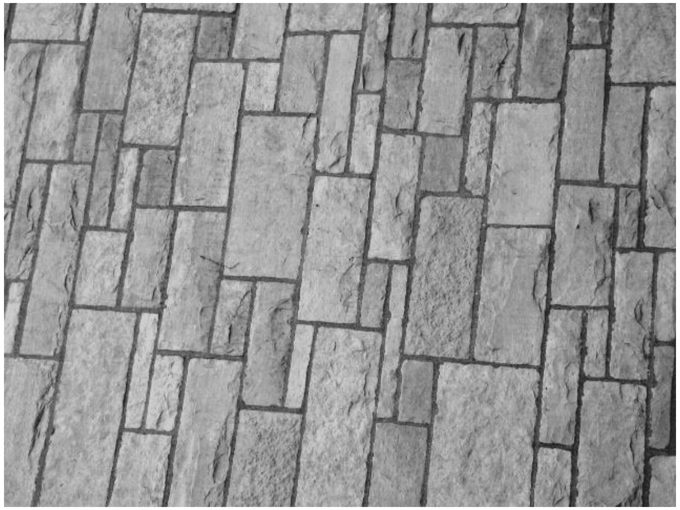
# Application: Representing Texture

Source: Forsyth

# Texture and Material

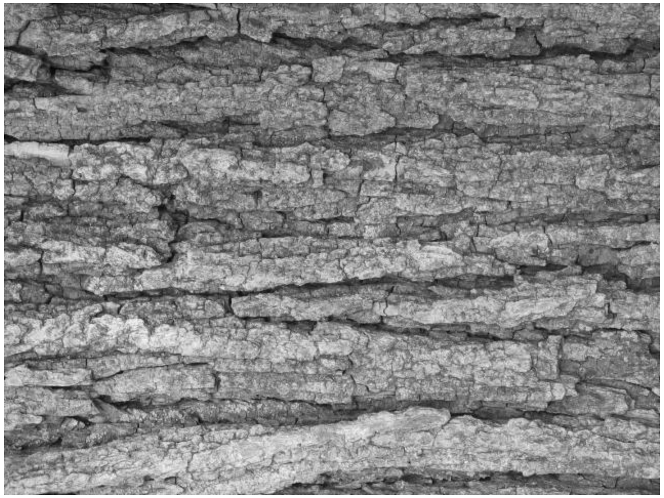






http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

# 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/

# What is texture?

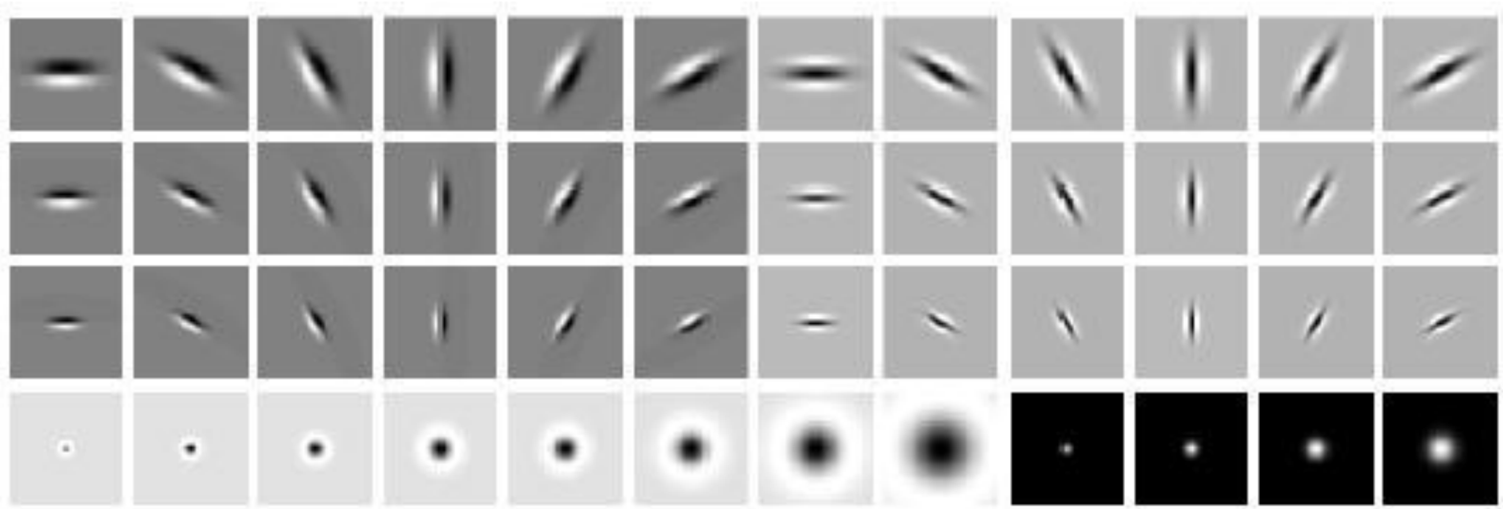
Regular or stochastic patterns caused by bumps, grooves, and/or markings

# How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales

# Overcomplete representation: filter banks

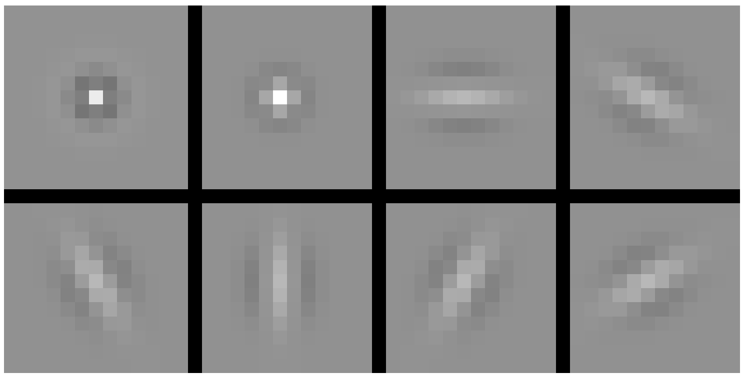
LM Filter Bank

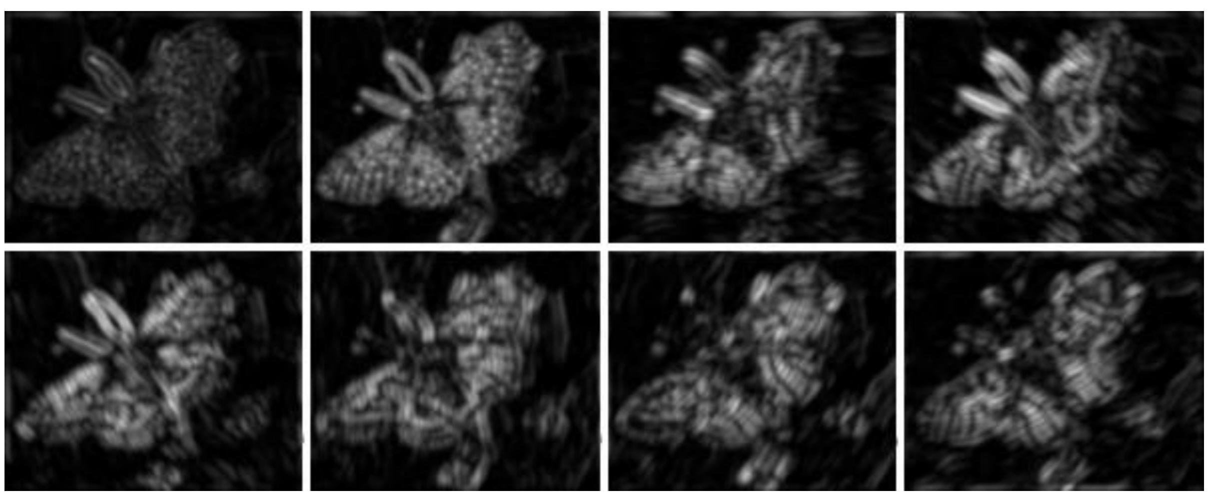


Code for filter banks: *www.robots.ox.ac.uk/~vgg/research/texclass/filters.html*

# Filter banks

- Process image with each filter and keep responses (or squared/abs responses)



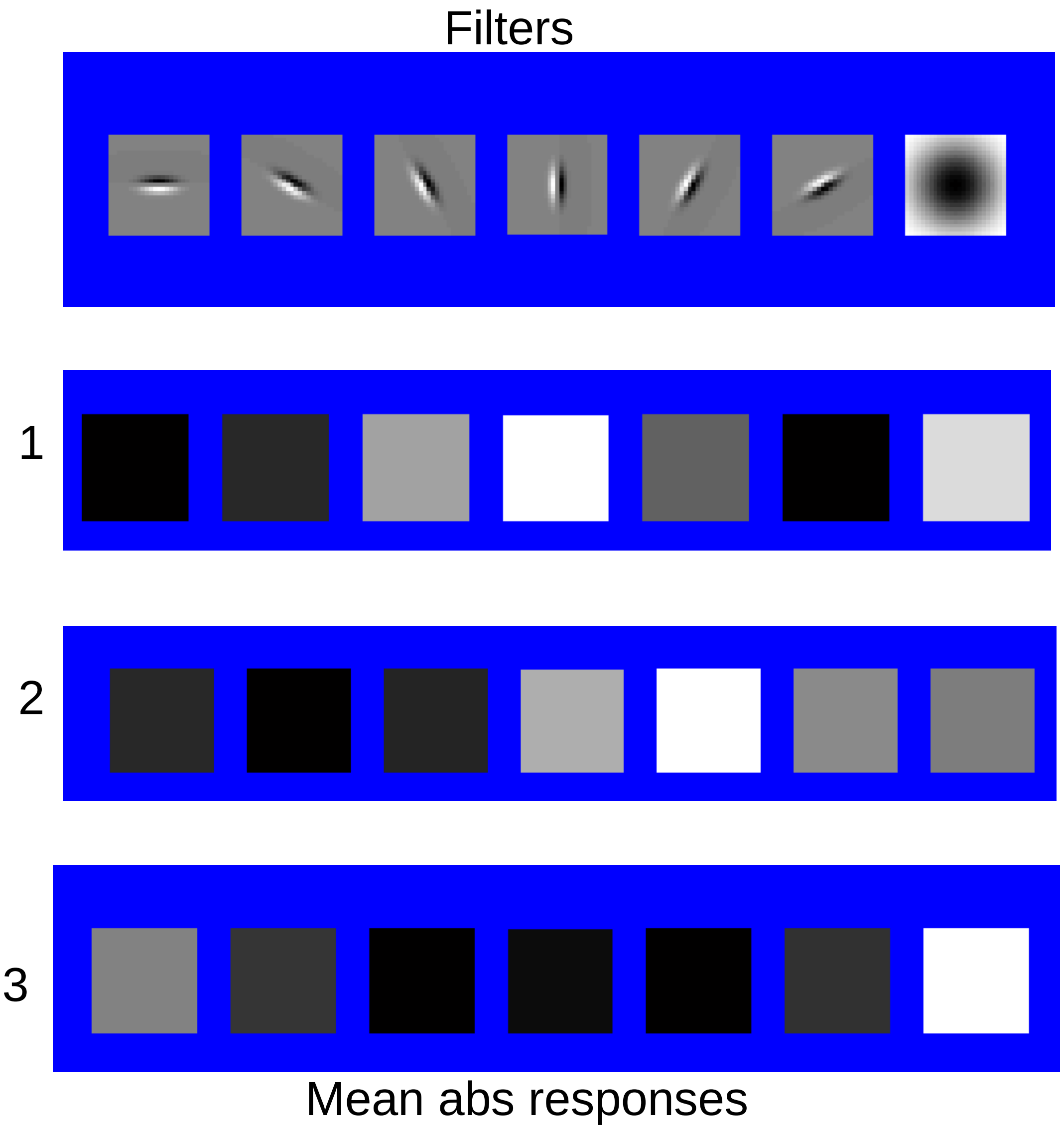


# How can we represent texture?

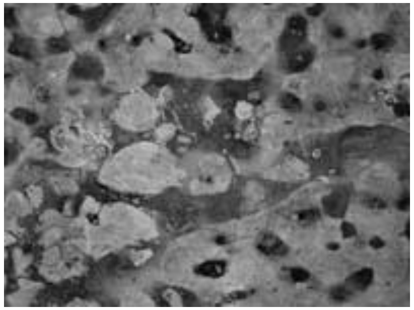
- Measure responses of blobs and edges at various orientations and scales

- Record simple statistics (e.g., mean, std.) of absolute filter responses

# Can you match the texture to the response?

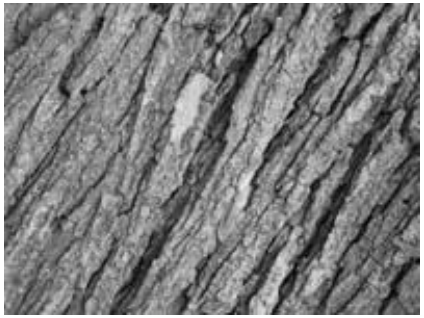



A

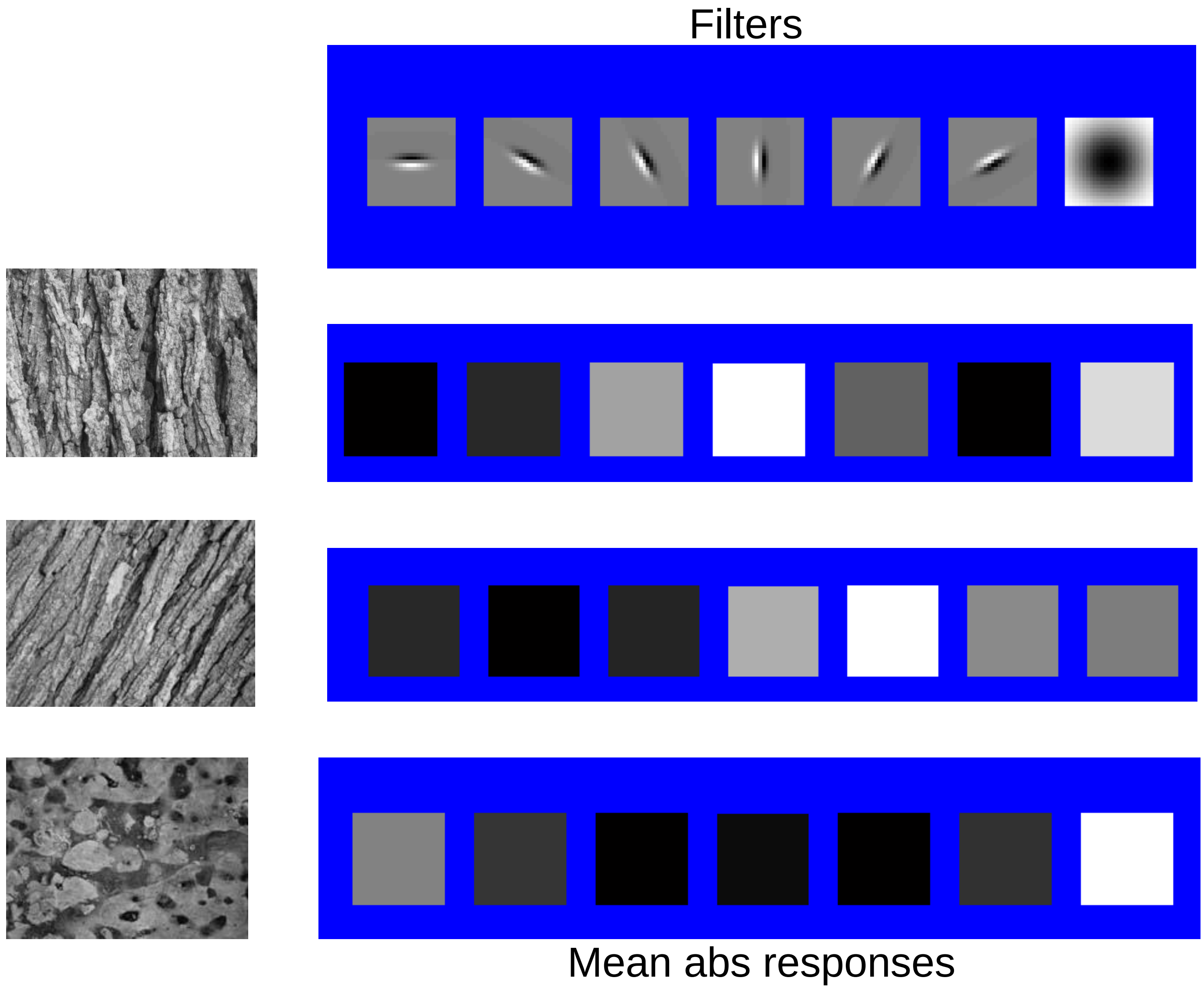


B

C



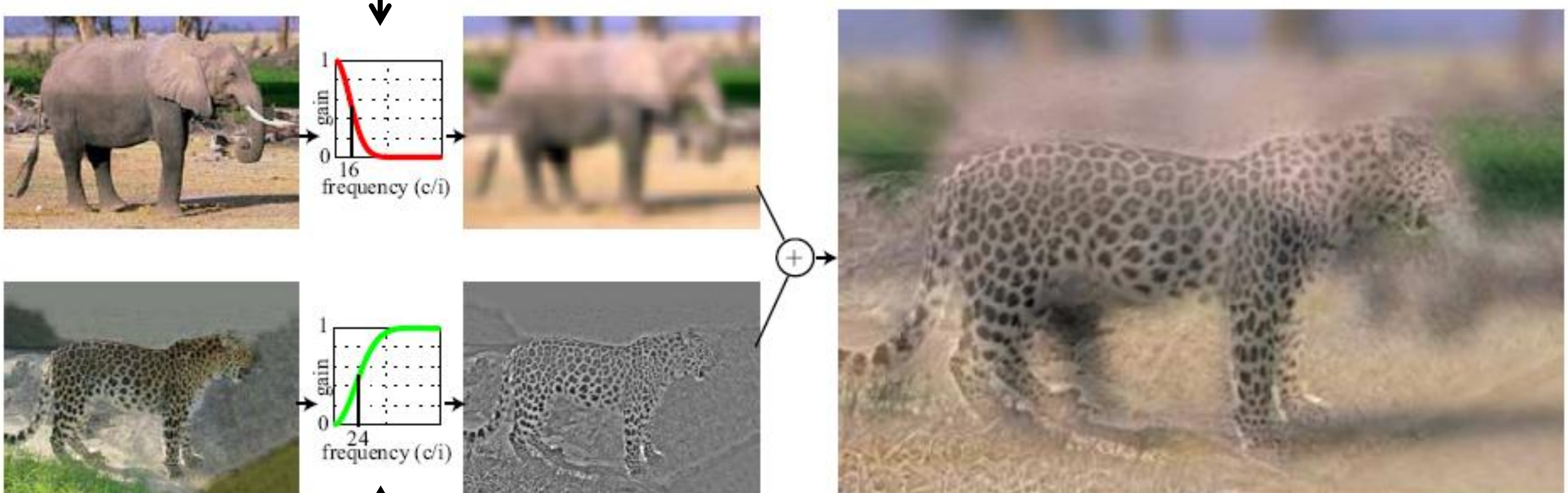
# Representing texture by mean abs response



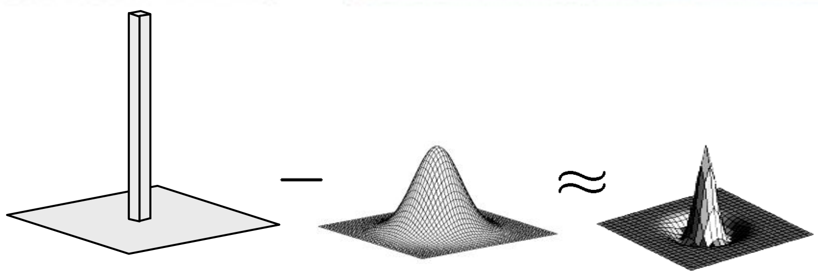
# Project 1: Hybrid Images

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

Gaussian Filter!




Laplacian Filter!



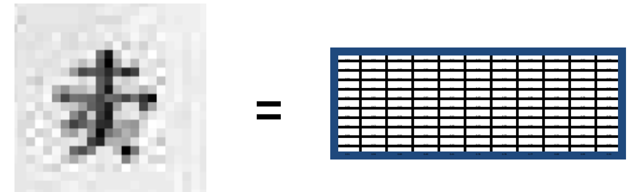
unit impulse — Gaussian ≈ Laplacian of Gaussian

Project Instructions:

http://courses.engr.illinois.edu/cs498dh3/projects/hybrid/ComputationalPhotography_ProjectHybrid.html

# Take-home messages

- Image is a matrix of numbers



- Linear filtering is a dot product at each position
  - Can smooth, sharpen, translate (among many other uses)

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

- Be aware of details for filter size, extrapolation, cropping

- Start thinking about project (read the paper, create a test project page)

# Take-home questions

1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise

2. Write down a filter that will compute the gradient in the x-direction:

```
gradx(y,x) = im(y,x+1)-im(y,x) for each x, y
```

# Take-home questions

## 3. Fill in the blanks:

Filtering Operator

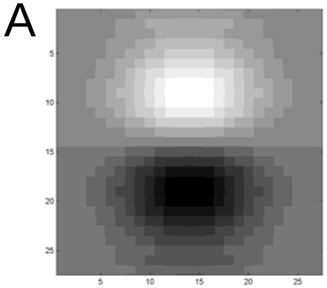
a) _ = D * B

b) A = _ * _

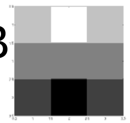
c) F = D * _

d) _ = D * D

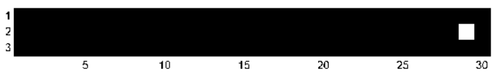
A



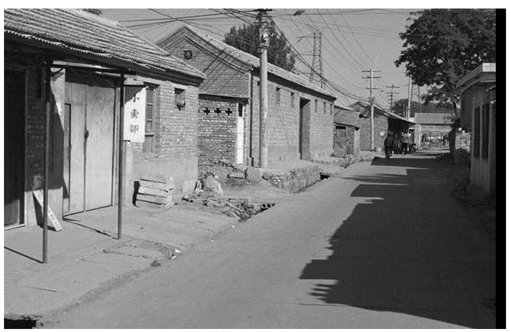
B



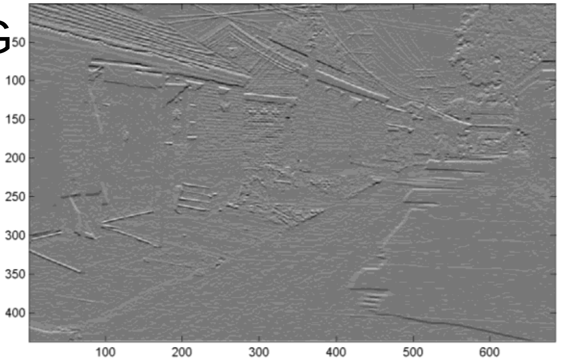
E



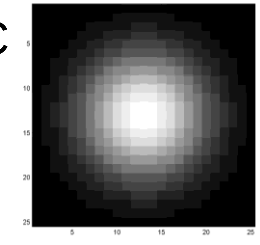

F



G

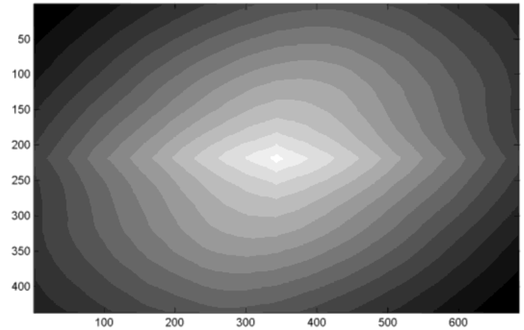



C

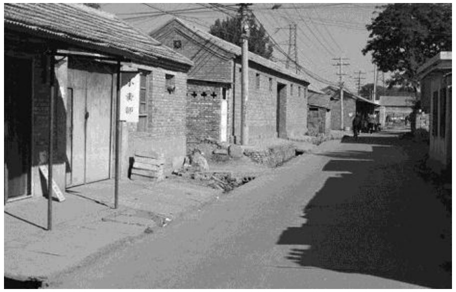


H

I




D



# Next class: Thinking in Frequency