COSC579: Image Processing Basics

Jeremy Bolton, PhD
Assistant Teaching Professor

GEORGETOWN UNIVERSITY

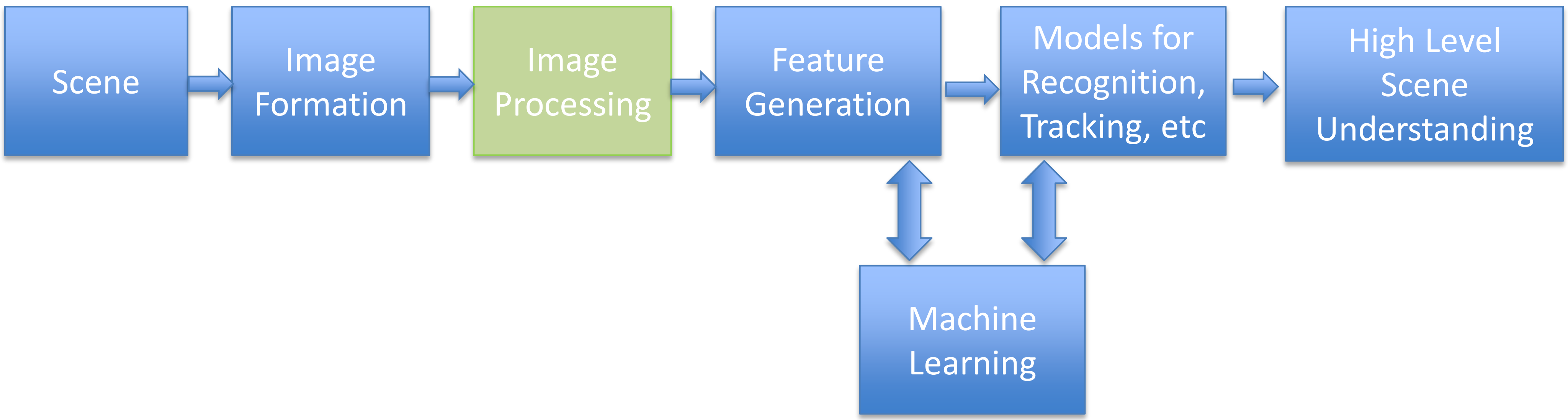
Outline

I. Point Operators
    I. Color Transforms
    II. Normalization / Whitening
    III. Histogram Normalization
II. Linear Filters
    I. Mathematical Formulation
        I. Cross Correlation
        II. Convolutions
    II. Common Filters
        I. Average
        II. Gaussian
        III. Laplacian

Reading

- Szeliski, Chapter 3.1-3.2
- Prince, Chapter 13.1.1 – 13.1.2

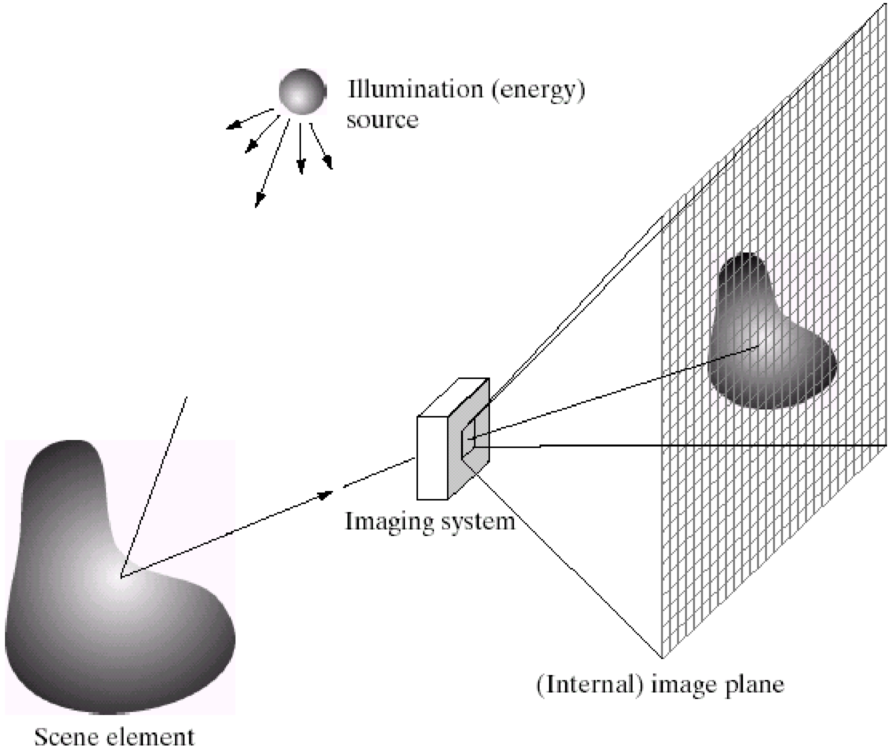
Our Progression Through Topics

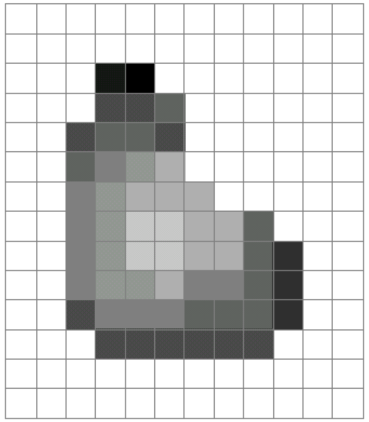


What is an image?

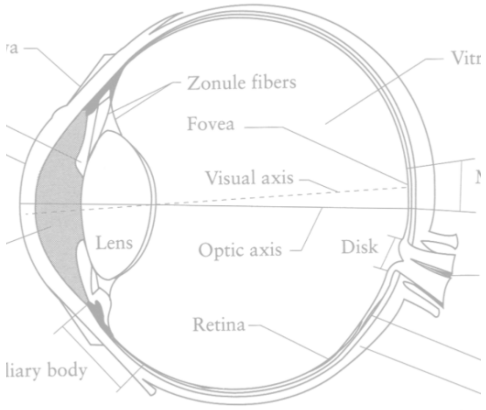
GEORGETOWN UNIVERSITY

What is an image?





Digital Image

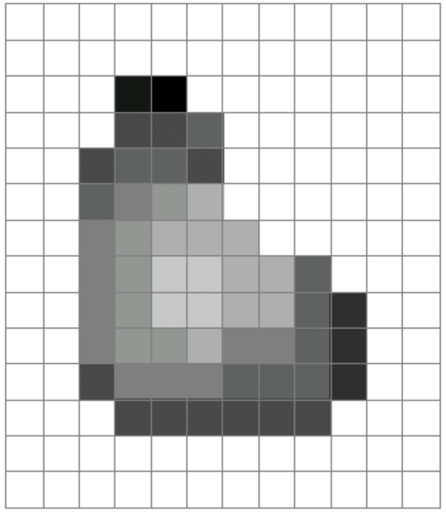


The Eye

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



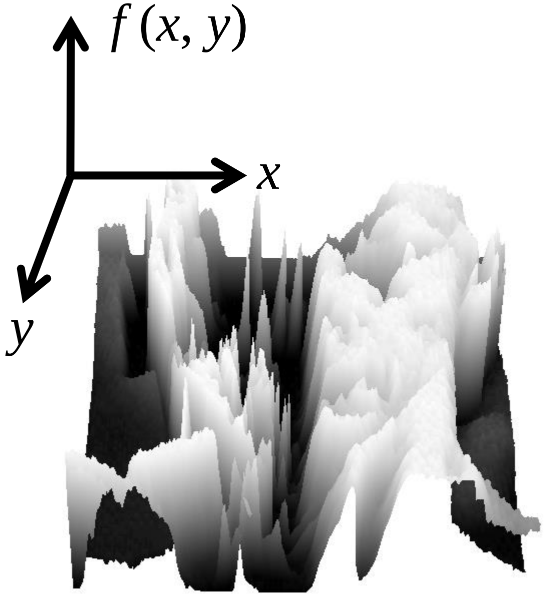
=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

What is an image?

- We can think of a (grayscale) image as a **function**, *f*, from R^2 to R (or a 2D *signal*):
  - $f\,(x,y)$ gives the **intensity** at position (x,y)

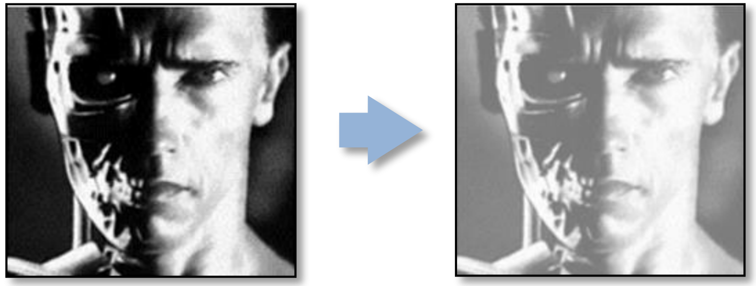


  - A **digital** image is a discrete (**sampled**, **quantized**) version of this function

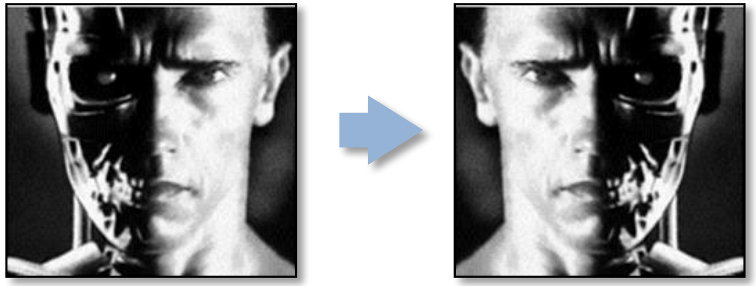
GEORGETOWN UNIVERSITY

Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

- Operators
  - *Point Operators*
    - Histogram equalization
    - Color or intensity transformations
  - *Linear Filtering:* convolution and cross correlation

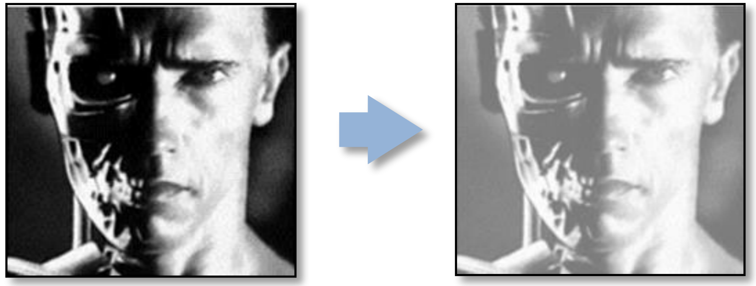
Pixel (intensity) transformation

- Transformation can be constant for each pixel

$$g\,(i,j) = a\,f\,(i,j) + b$$

- Transform can vary from pixel-by-pixel

$$g\,(i,j) = a(i,j)\,f\,(i,j) + b(i,j)$$



$g\,(i,j) = f\,(i,j) + 20$

GEORGETOWN UNIVERSITY

Pixel Transforms based on image statistics

- Often an image is assumed to be a collection of observations of some random variable (a sample).

- Image statistics can provide a means for a data-based transformation (rather than some arbitrary scaling).

Histogram Equalization

- Problem: Image has undesirable histogram.
  - Distribution of intensity values.

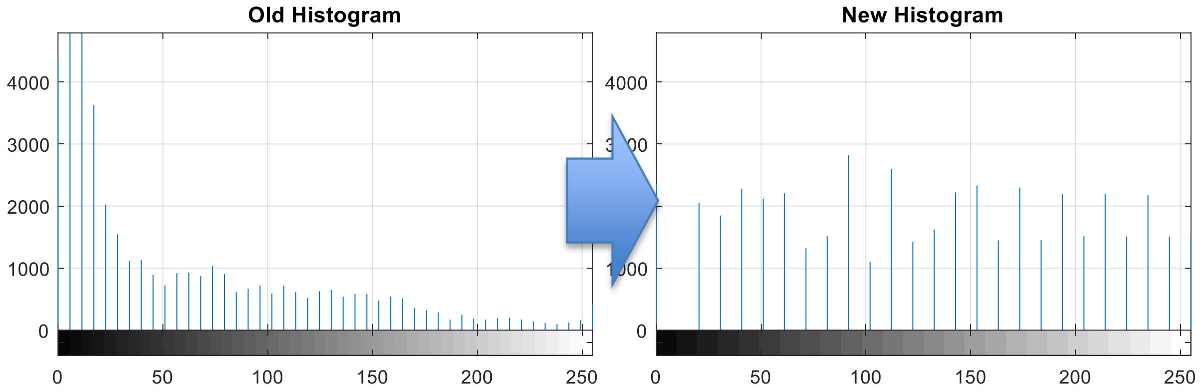
- Solution: Map intensities to a desirable histogram using Histogram (PDF) and Cumulative Histogram (CDF)

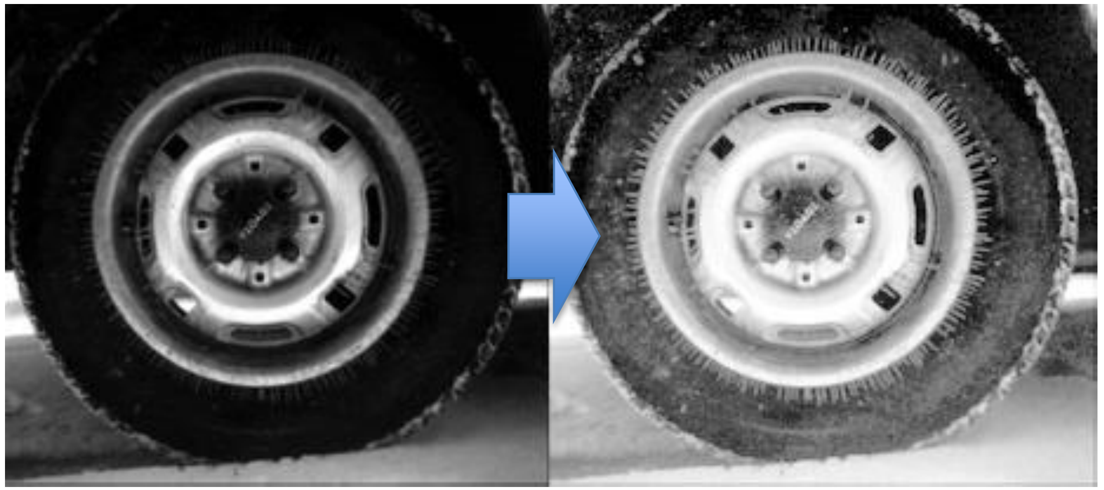
- An often desired histogram is a Uniform distribution (all intensities are equal in frequency)

Histogram Equalization

MATLAB

```
I = imread('tire.tif');
J = histeq(I); %Enhance the contrast of an intensity image using histogram equalization
imshowpair(I,J,'montage') %Display the original image and the adjusted image
axis off
figure;
imhist(I,64)
imhist(J,64)
```





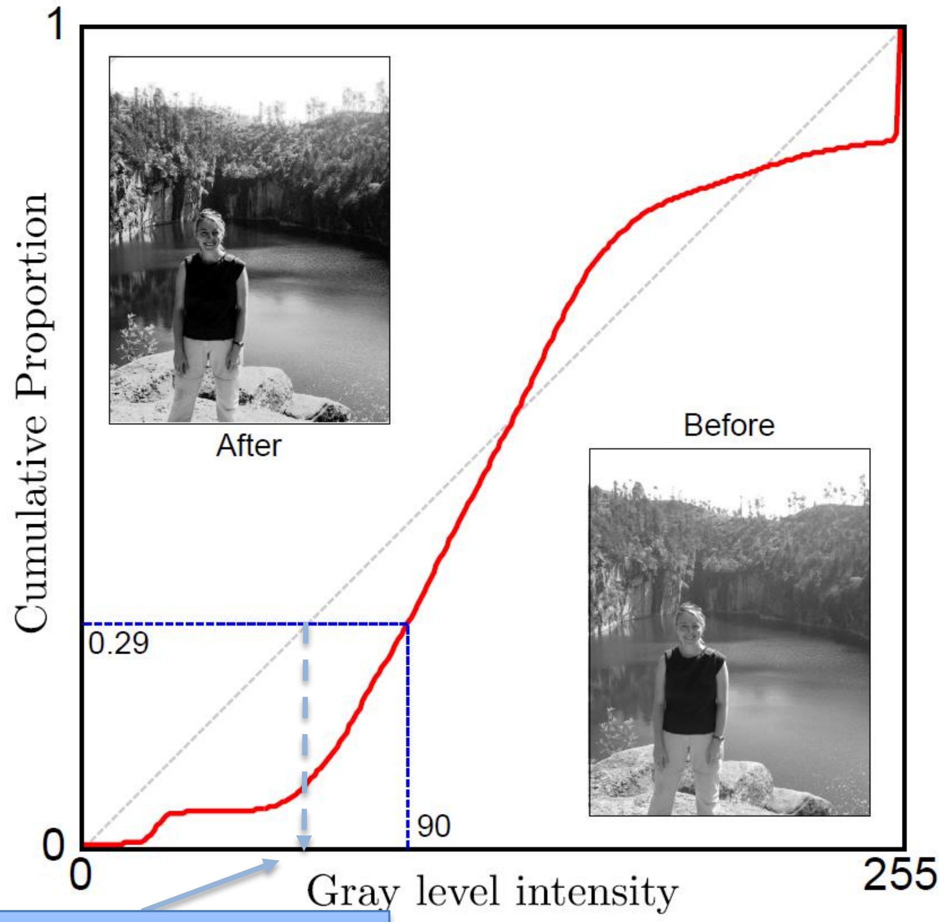
Histogram Equalization

1. Compute Image Histogram (PDF)

2. Compute CDF

3. Compute CDF of desired histogram. (Assume Uniform)

4. Transform to desired CDF



Map back to the original domain if you do not wish to change the range of intensities

GEORGETOWN UNIVERSITY

Histogram Equalization

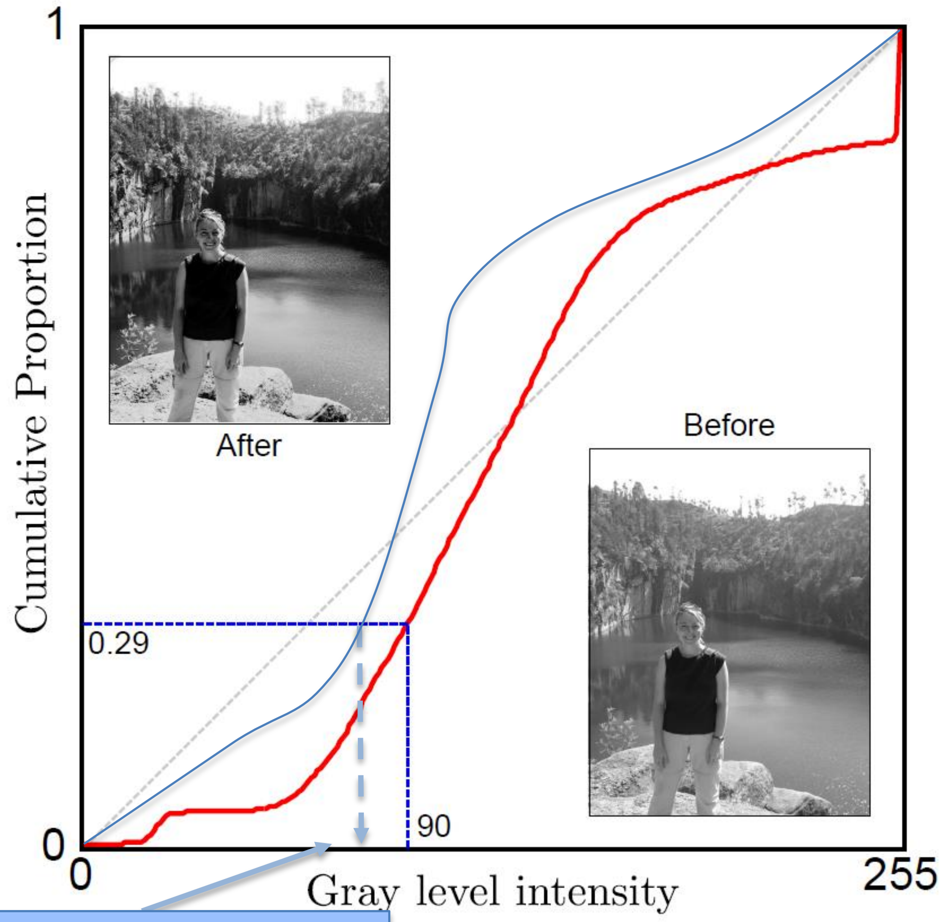
1. Compute Image Histogram (for all possible values k)

2. Compute CDF

3. Compute inverse of CDF of desired histogram.
   - Some CDFs are invertible (some are not)
   - If no closed form exists, then simply build a lookup table

4. Transform to desired CDF

$$f_{new}(x,y) = CDF^{-1}_{desired}(CDF(f(x,y)))$$



Map back to the original domain if you do not wish to change the range of intensities

Whitening

1. Compute Image Statistics
   – Mean and standard deviation

$$\mu = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} p_{ij}}{IJ}$$

$$\sigma^2 = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} (p_{ij} - \mu)^2}{IJ}.$$

2. Apply transform to each pixel
   – Use mean and standard deviation to “whiten” each pixel

$$x_{ij} = \frac{p_{ij} - \mu}{\sigma}.$$

Normalization Via Whitening

- Fix first and second moments to standard values
- Reduces contrast and constant additive luminance variations

Before

After

Normalization and Histogram Equalization

Make all of the moments the same by forcing the histogram of intensities to be the same

Before/ normalized/ Histogram Equalized

Linear Filter

- Basic Point Operators perform the same operation on each pixel
  - Not necessarily based on image data (local or global)

- Normalization / Whitening
  - Processing based on global (or local) image statistics

- Local Operators AKA (neighborhood operators)
  - Often implemented as **Linear Filters**
  - Local or neighborhood information is used in transformation

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel
  - Example: 3x3 neighborhood

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

Source: L. Zhang

Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination of its neighbors
- This linear combination is an binary operator which maps two functions to one function.
  - First operand is the input image
  - Second operand is called the “kernel” (or “mask”, “filter”)
  - The resulting function is the output image

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Cross-correlation

Let F be the image, H be the kernel (of size 2k+1 x 2k+1), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

Convolution

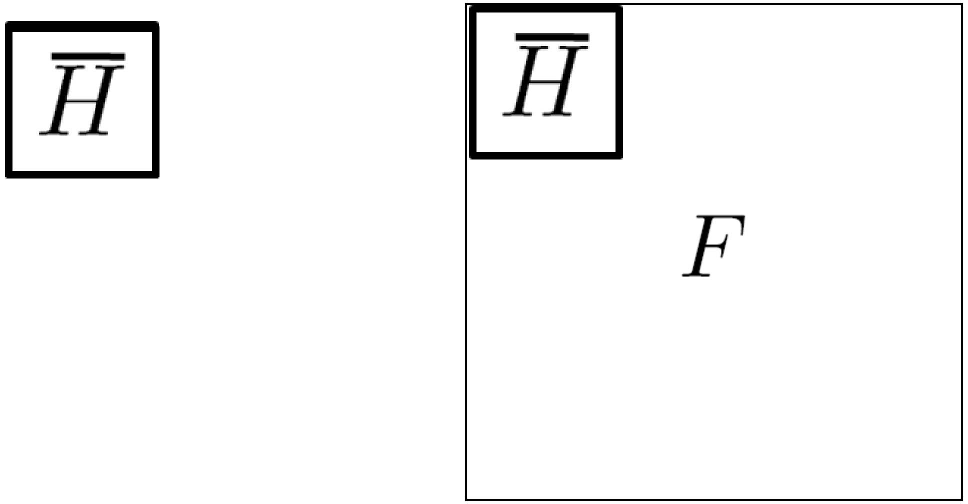
- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation: $G = H * F$

- Convolution is **commutative** and **associative**

Convolution



Adapted from F. Durand

Mean filtering

H

$*$

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

F

$=$

	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	

G

Linear filters: examples

Original

*

0	0	0
0	1	0
0	0	0

=

Identical image

Source: D. Lowe

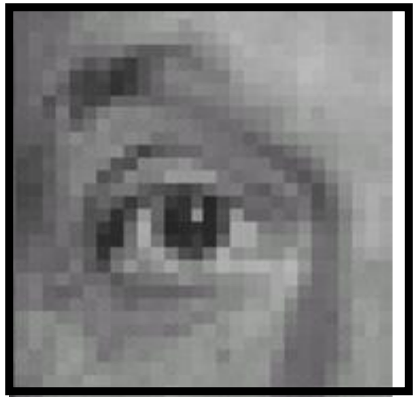
Linear filters: examples

Original

*

0	0	0
1	0	0
0	0	0

=



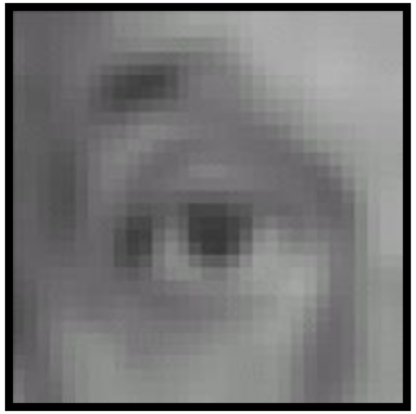
Shifted left
By 1 pixel

Source: D. Lowe

Linear filters: examples

Original

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



Blur (with a mean filter)

Source: D. Lowe

Linear filters: examples

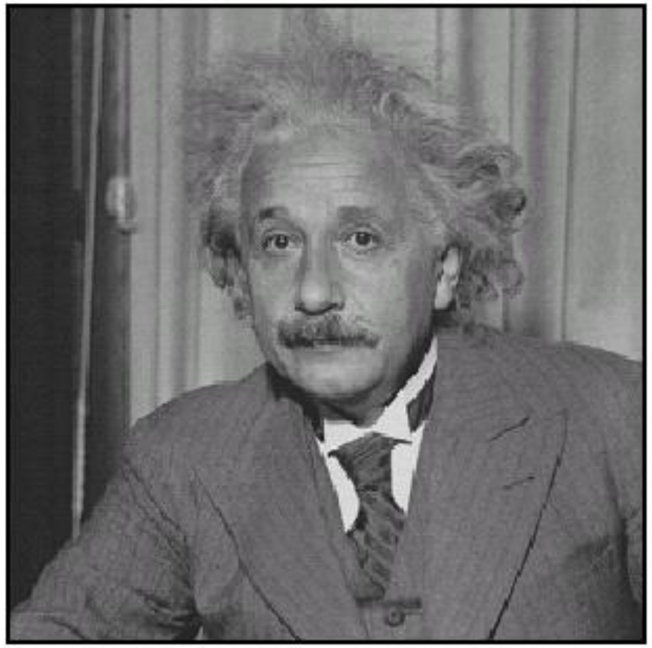
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

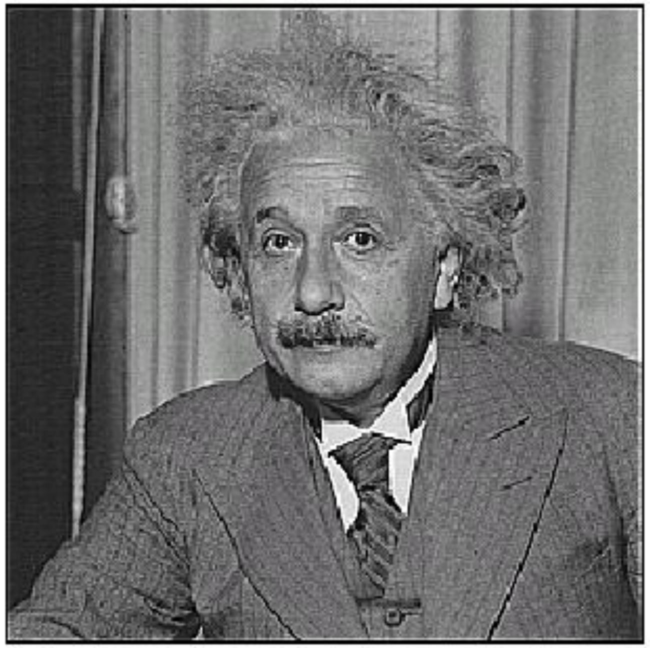
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



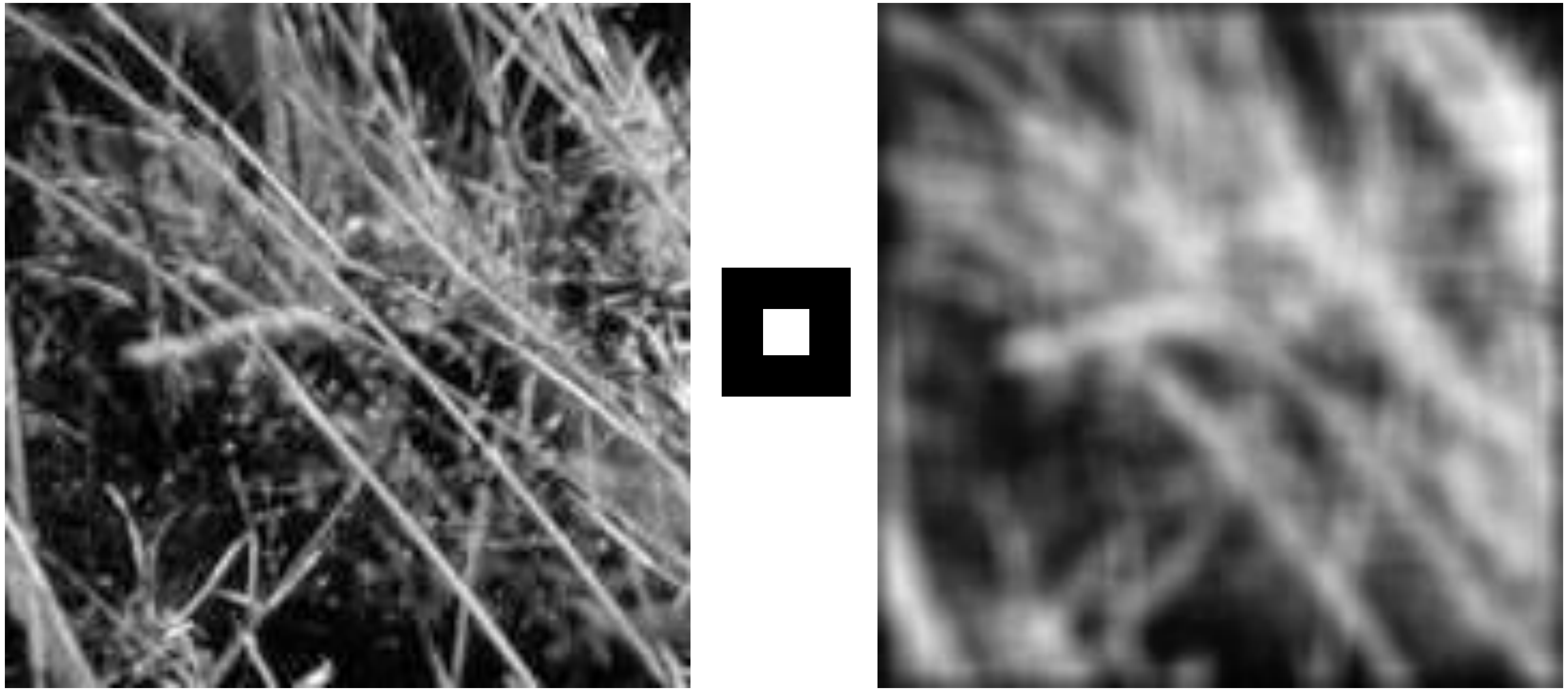
before



after

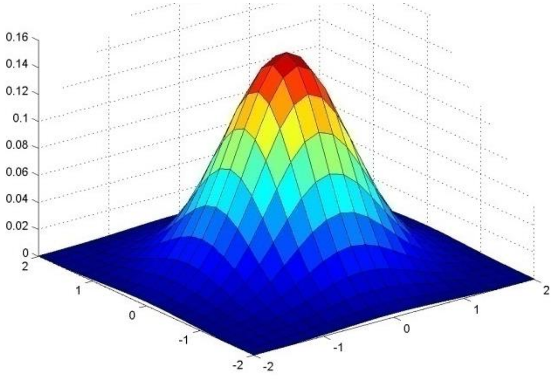
Source: D. Lowe

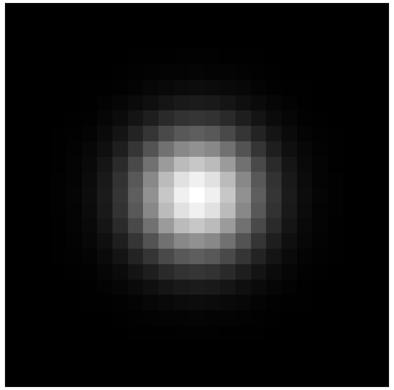
Smoothing with box filter revisited



Source: D. Forsyth

Gaussian Kernel

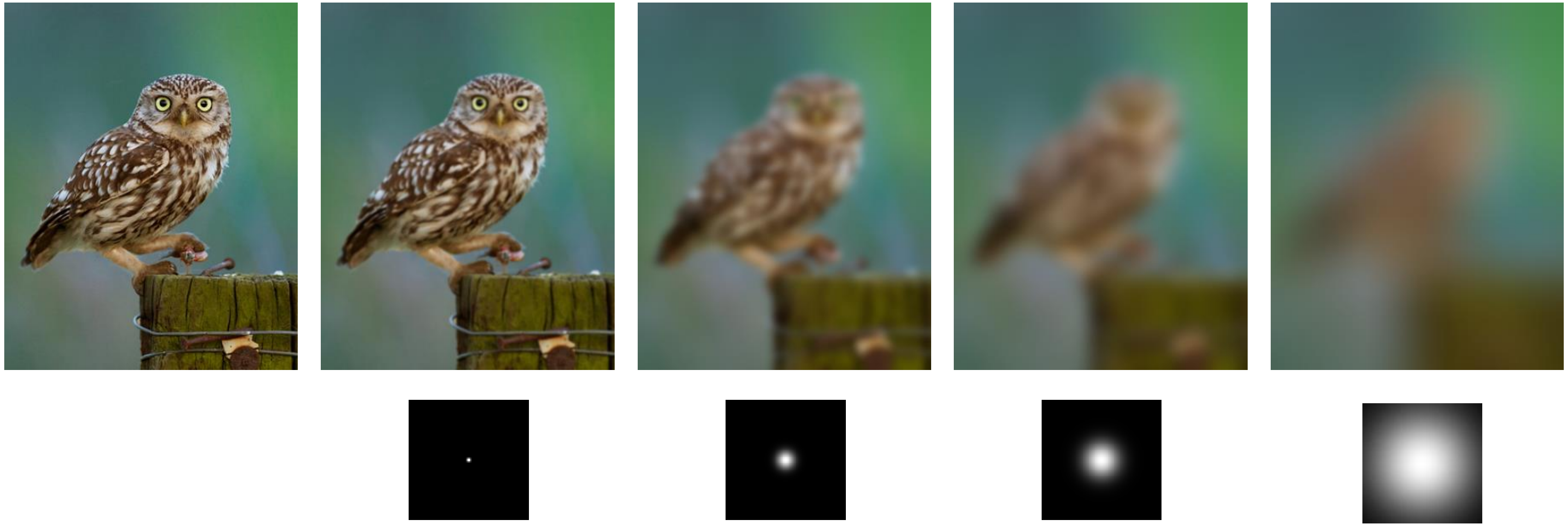




$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

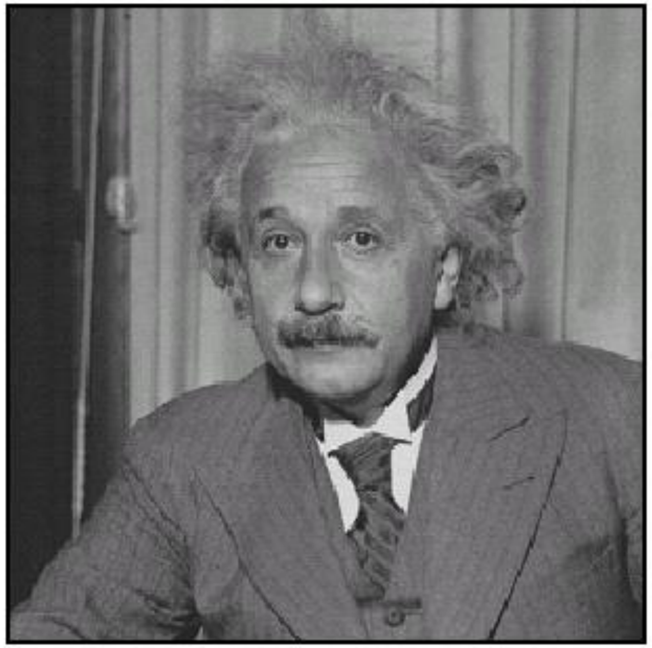
Gaussian filters



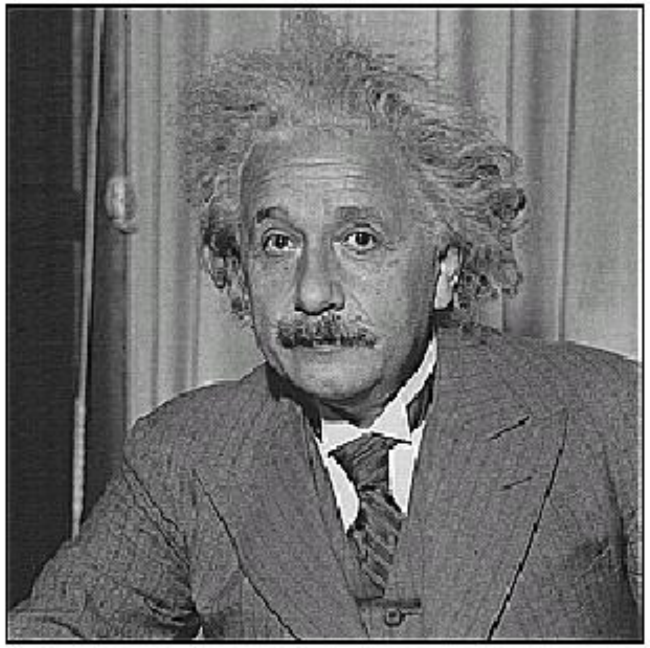
σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

GEORGETOWN UNIVERSITY

Sharpening



before



after

Source: D. Lowe

Sharpening revisited

- What does blurring take away?

–

=

Let’s add it back:

+ α

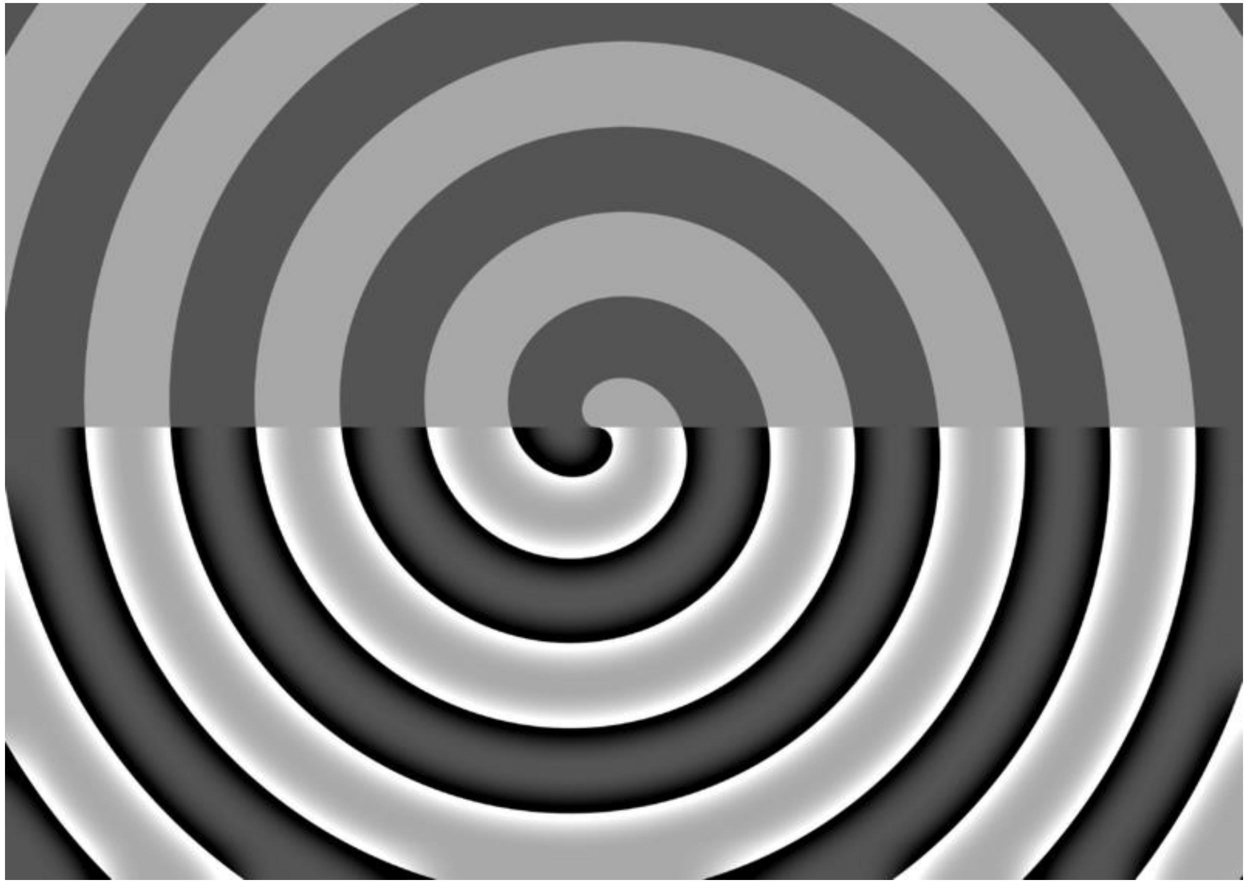
=

Source: S. Lazebnik

Sharpen filter

unfiltered

filtered



GEORGETOWN UNIVERSITY

Blurring (convolve with Gaussian)

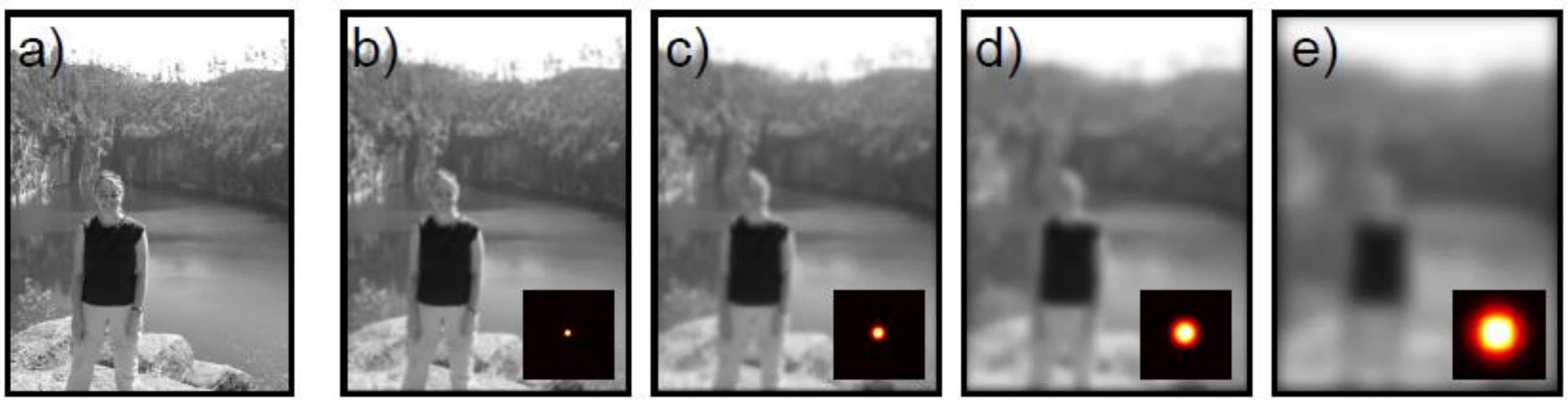
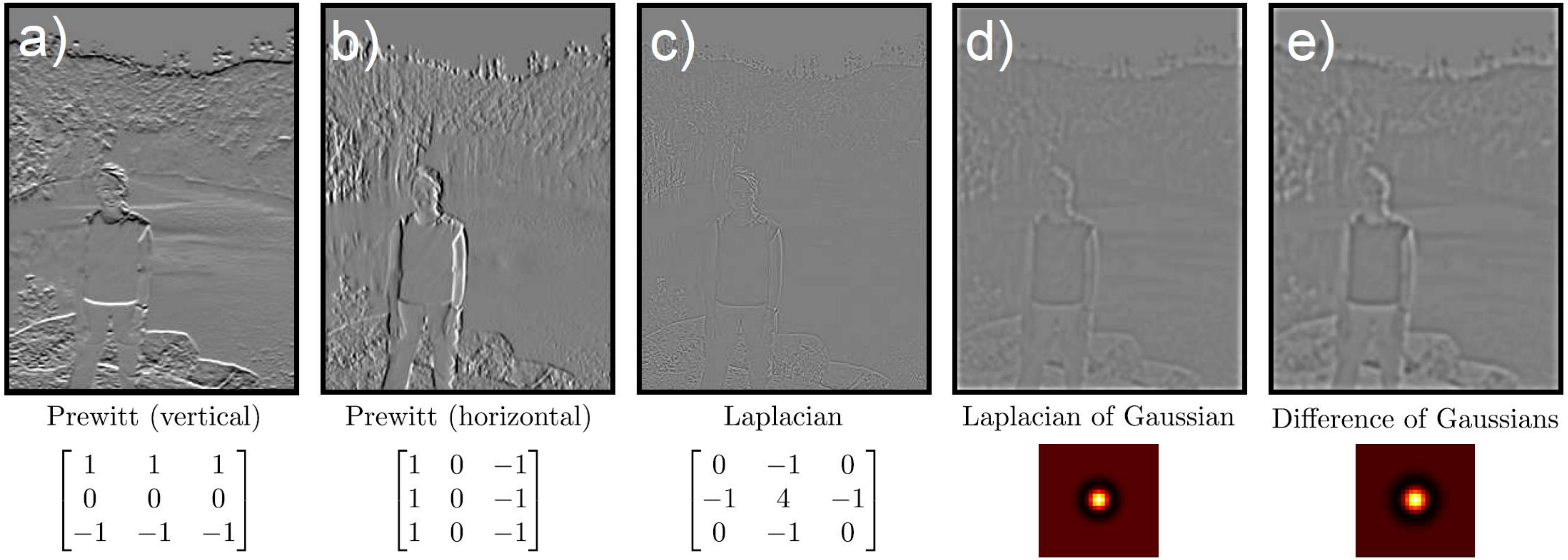


Figure B.3 Image blurring. a) Original image. b) Result of convolving with a Gaussian filter (filter shown in bottom right of image). The image is slightly blurred. c-e) Convolving with a filter of increasing standard deviation causes the resulting image to be increasingly blurred.

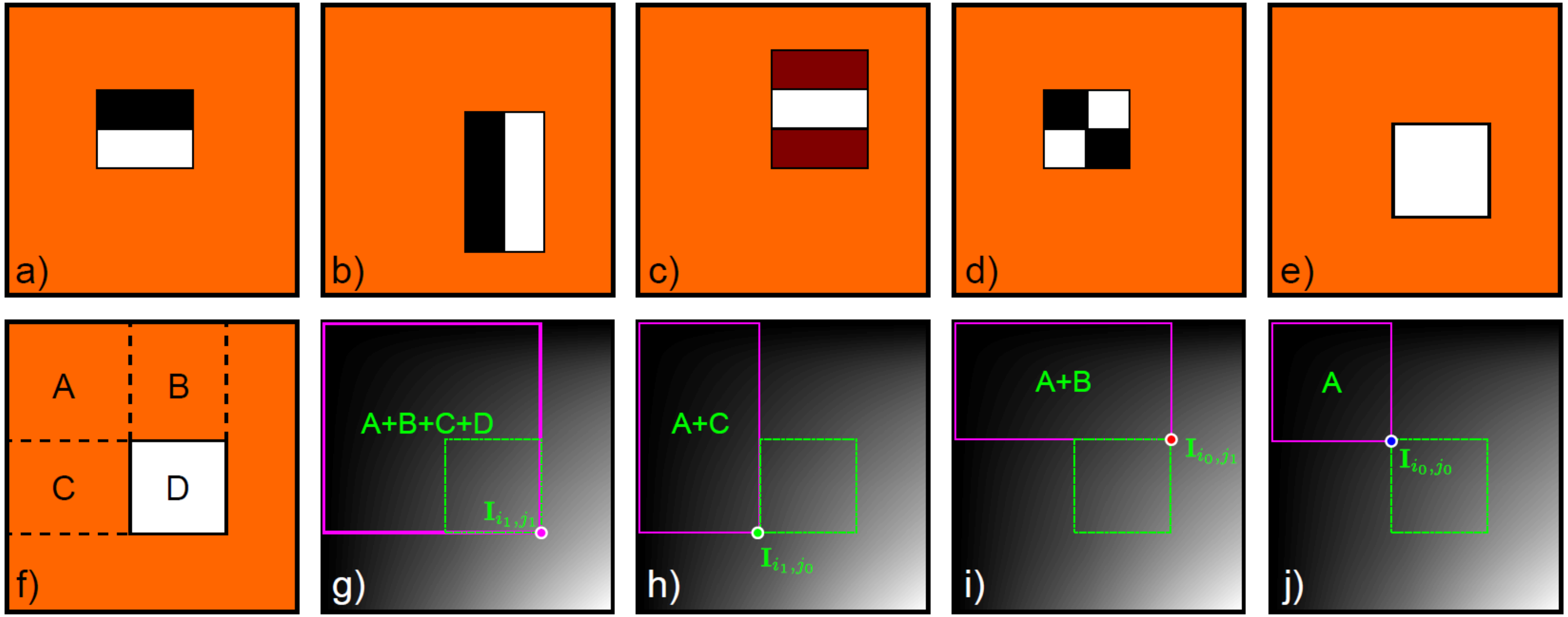
Looking Ahead: Gradient Filters (Edge Detection)



$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Rule of thumb: big response when image matches filter

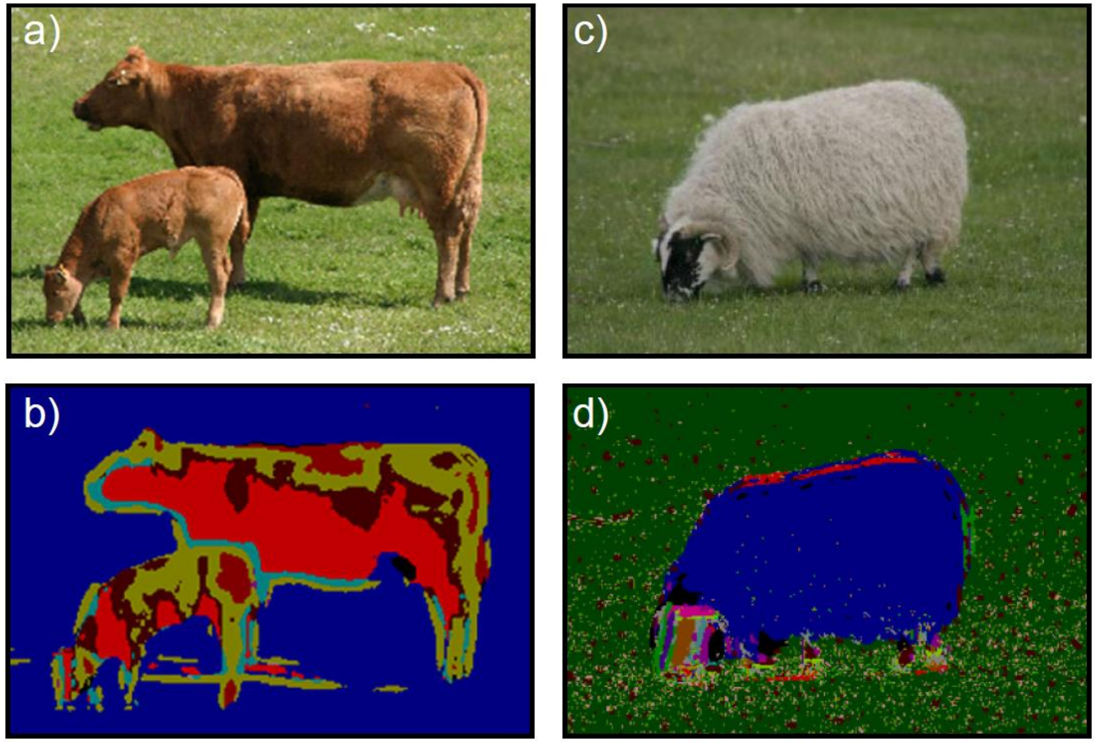
Haar Filters



GEORGETOWN UNIVERSITY

Looking Ahead: Filtering for Texture (Textons)

- An attempt to characterize texture
- Replace each pixel with integer representing the texture 'type'

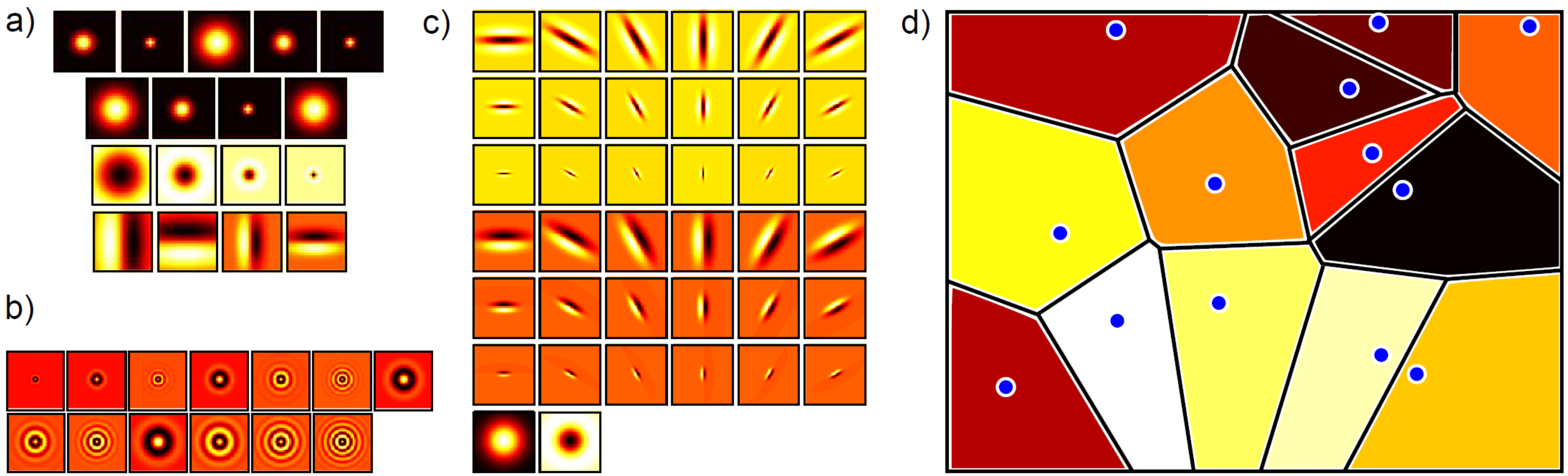


GEORGETOWN UNIVERSITY

Computing Textons

Take a bank of filters and apply to lots of images

Cluster in filter space



For new pixel, filter surrounding region with same bank, and assign to nearest cluster