Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and interacting with photographs.

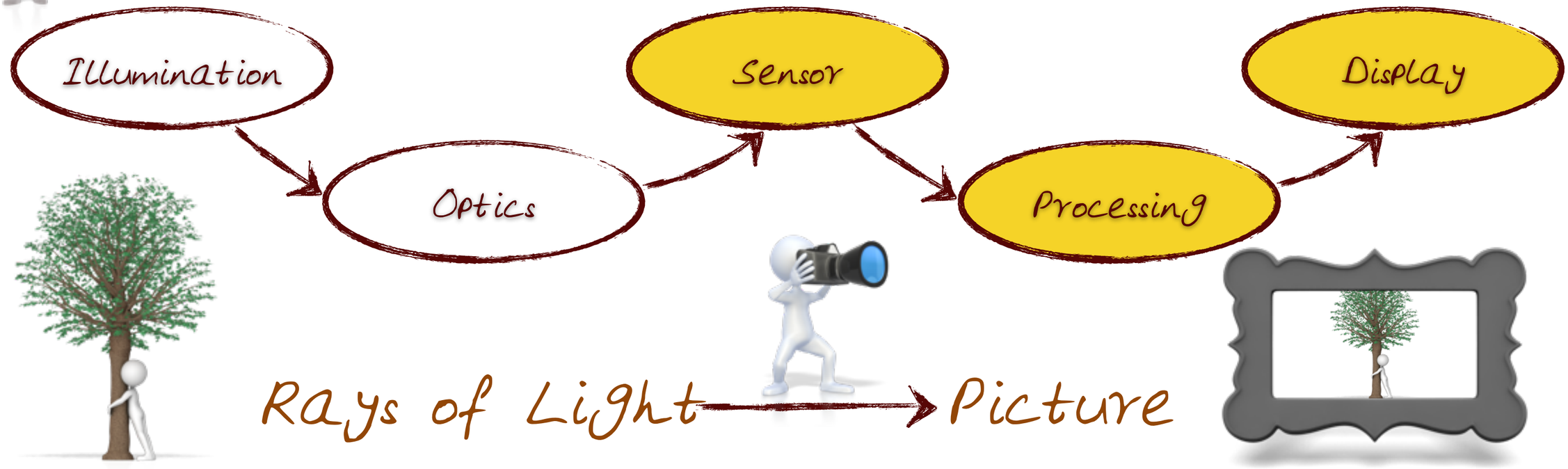
Georgia Tech College of Computing

Representing an Image

* What is a Digital Image?
* How to make an Image a Computable Entity

Georgia Tech College of Computing

Recall the Comp. Photography Pipeline

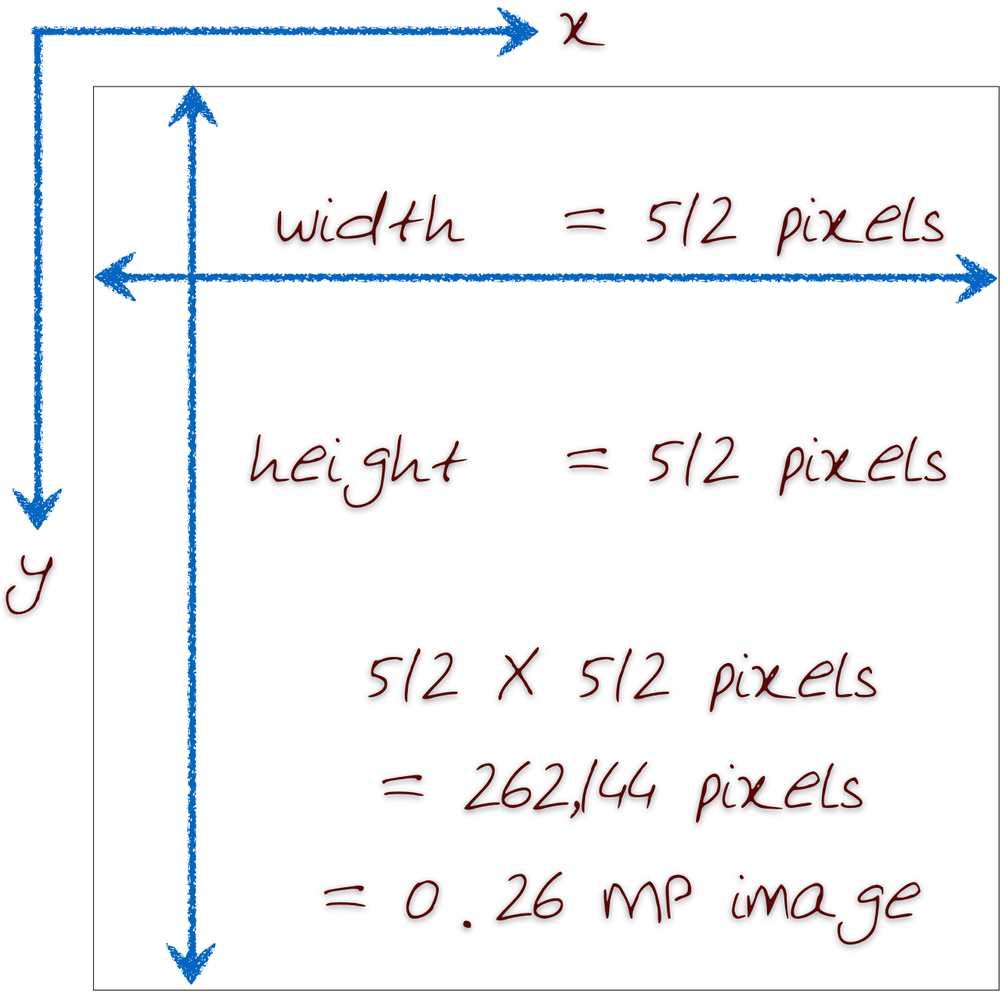


* make an Image a "computable" object
* ie., A Digital Image

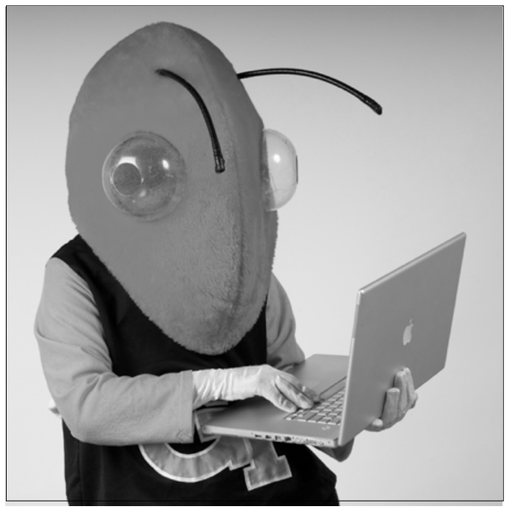
1. Digital Image - pixels and image resolution
2. Discrete (matrix) and Continuous (function) representations
3. Grayscale and Color Images
4. Digital Image formats

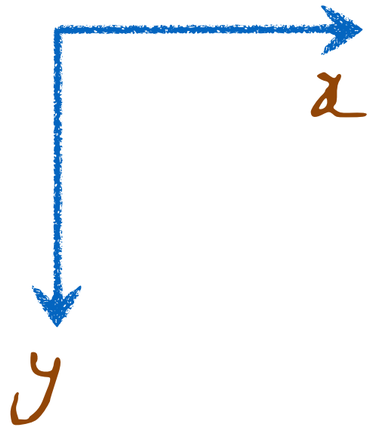
A Digital Image (W X H)

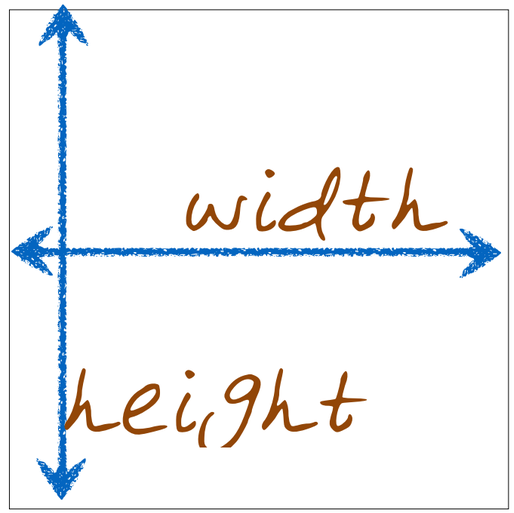
Georgia Tech's mascot Buzz in Black and White



A Digital Image!



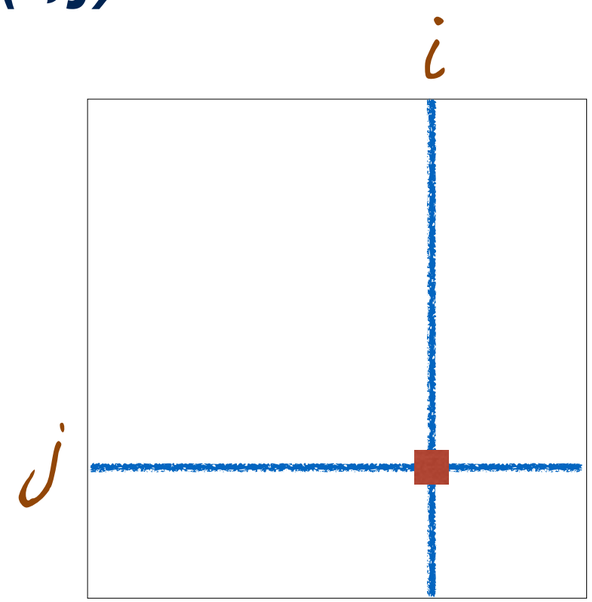




* Numeric representation in 2-D (x and y)
* Referred to as **I(x,y)** in continuous function form, **I(i,j)** in discrete
* Image Resolution: expressed in terms of Width and Height of the image

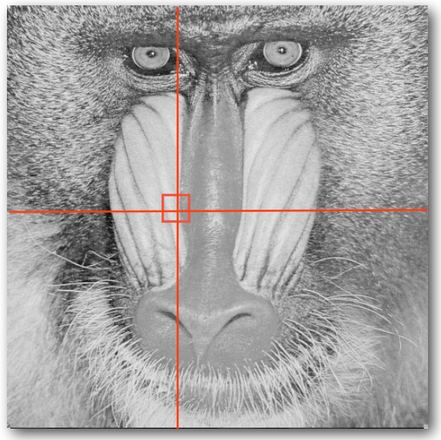
Pixel

A "**picture element**" that contains the light intensity at some location **(i,j)** in the image



I(i,j) = Some Numeric Value

Characteristics of a Digital Image

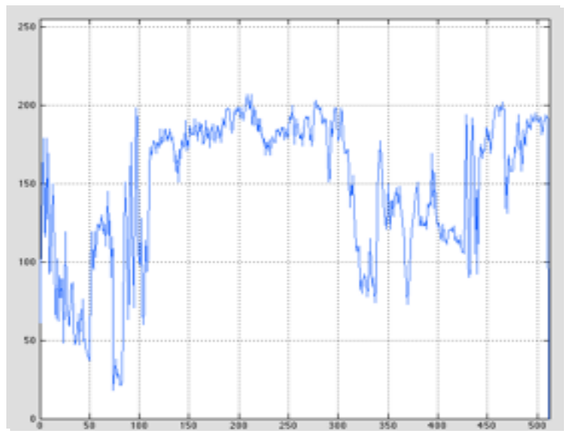


Original Image

Zoomed In

159	166	149	134	142	145	152	156
154	147	127	126	134	139	144	142
163	153	157	137	132	145	155	144
178	160	166	152	142	151	166	136
179	164	156	153	138	136	145	134
184	164	159	168	154	129	128	141
186	171	160	164	148	132	130	139
161	177	160	146	131	131	132	129
173	167	158	152	144	136	130	129
184	175	170	164	144	138	136	135
188	188	160	151	147	146	144	143
190	193	189	172	140	149	157	143
171	182	174	168	136	141	150	144

Values



Plots of Values at a Slice

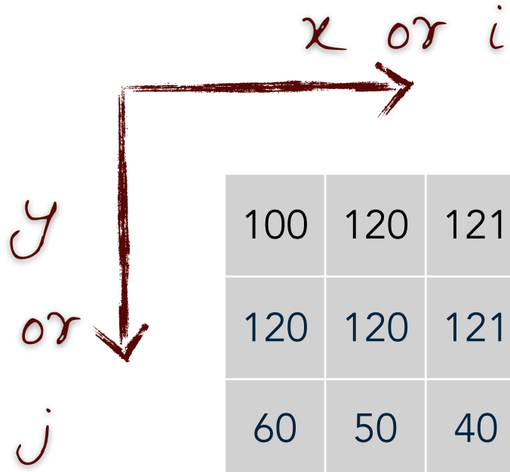
* A two-dimensional array of pixels and respective intensities
* Image can be represented as a matrix
* Intensity Values range from 0 = Black to 255 = White

Common data types

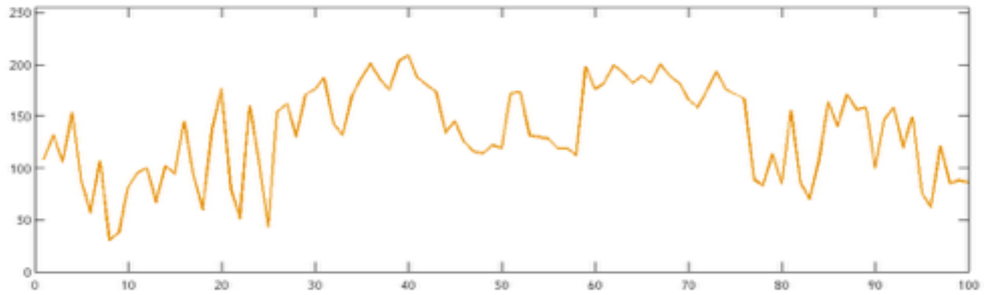
Data types used to store pixel values:

- unsigned char
- uint8
- unsigned char 8bit
- 2^n (2^1, 2^2, 2^4, 2^8, etc.)

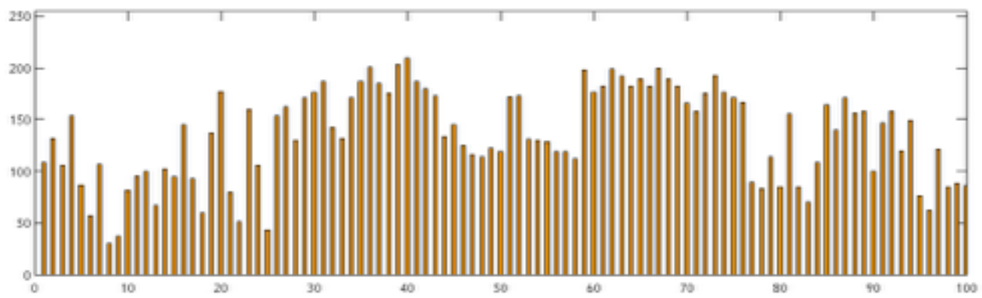
Digital Image is a Function



100	120	121	122	30	40
120	120	121	122	70	40
60	50	40	41	7	8
100	120	121	122	1	0
200	120	200	122	12	14
200	220	225	250	30	40



Continuous Signal



Discrete Signal

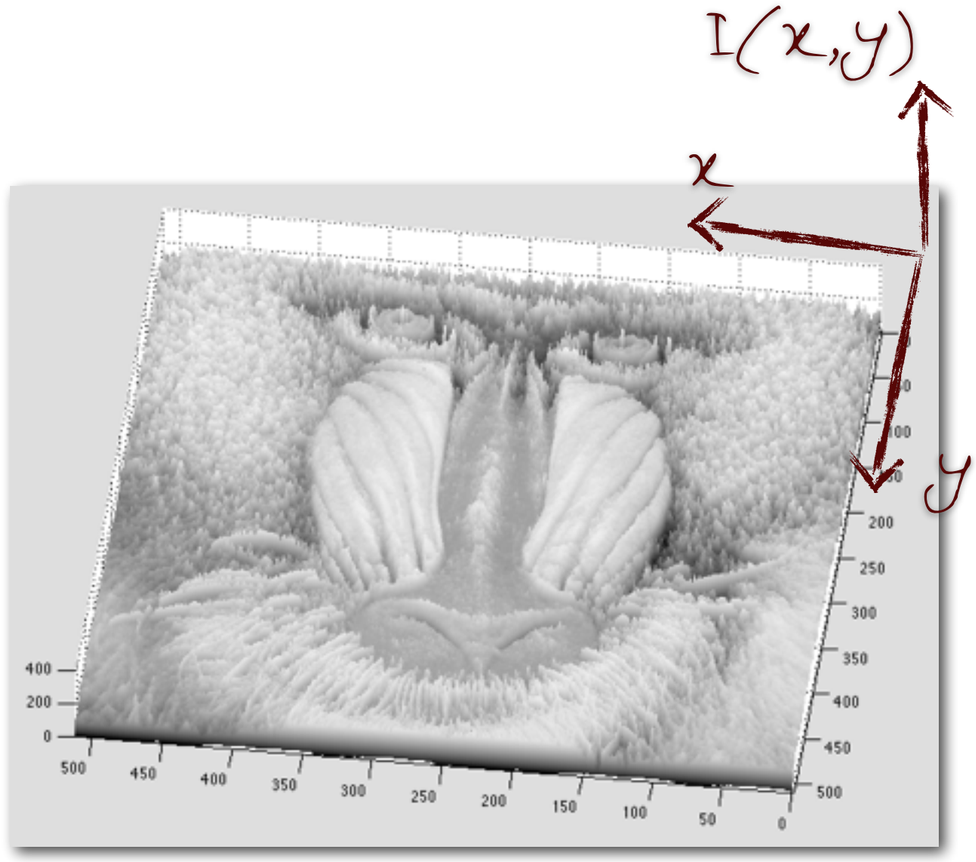
Slide adapted from Steve Seitz and Aaron Bobick

Digital Image is a Function

x or i

y or j

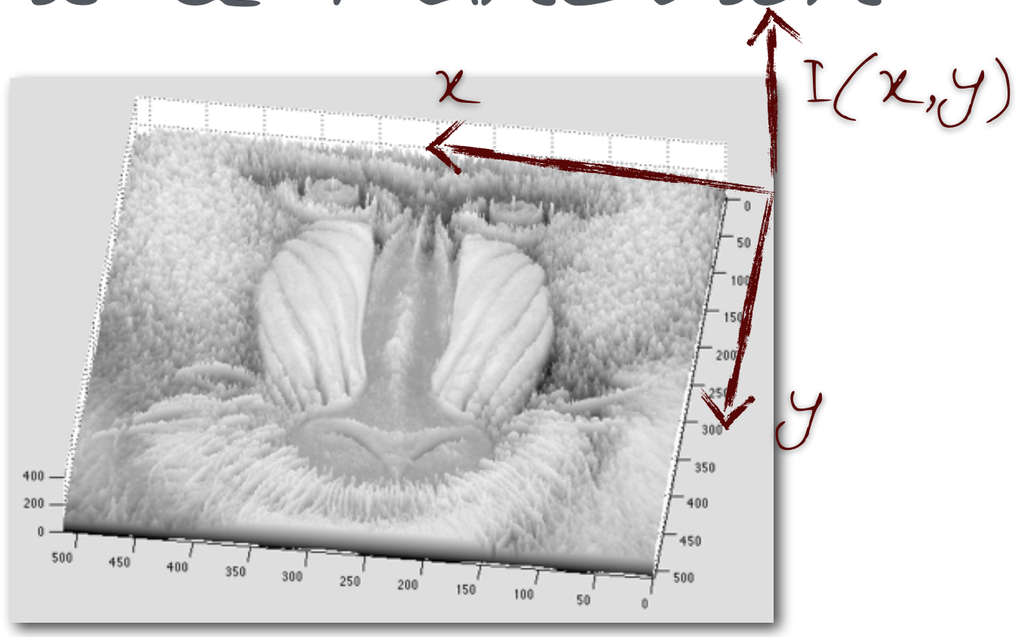
100	120	121	122	30	40
120	120	121	122	70	40
60	50	40	41	7	8
100	120	121	122	1	0
200	120	200	122	12	14
200	220	225	250	30	40



Slide adapted from Steve Seitz and Aaron Bobick

Digital Image is a Function

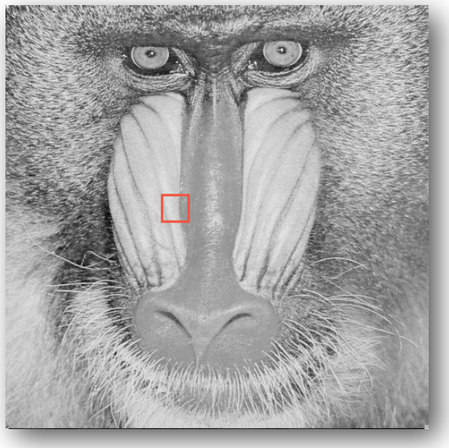
100	120	121	122	30	40
120	120	121	122	70	40
60	50	40	41	7	8
100	120	121	122	1	0
200	120	200	122	12	14
200	220	225	250	30	40



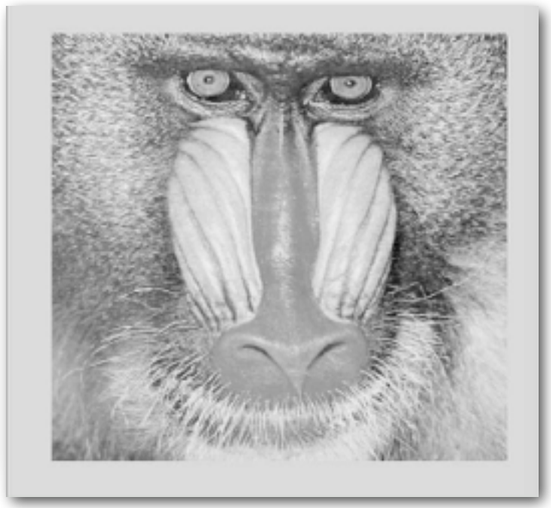
- Typically, the functional operation requires discrete values
  - Sample the two-dimensional (2D) space on a regular grid
  - Quantize each sample (rounded to "nearest integer")
- matrix of integer values (Range: 0-255)

Slide adapted from Steve Seitz and Aaron Bobick

Black/White Digital Image: An Example



Image



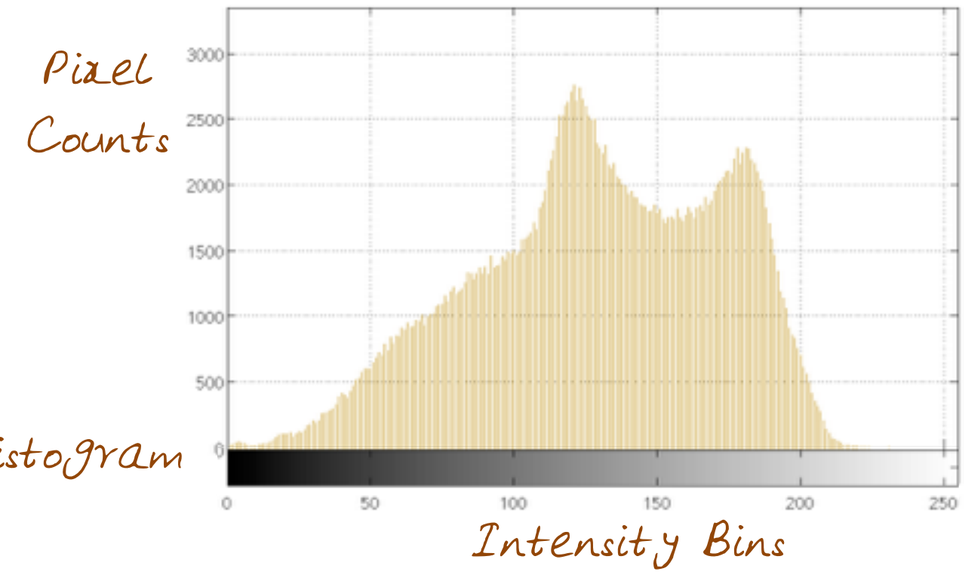
Height Map

Height Map

Zoomed In

Using matlab

Digital Image Statistics



- Image statistics - average, median, mode
  - Scope - entire image or smaller windows/regions
- Histogram - distribution of pixel intensities in the image
  - Can be separate for each channel, or region-based too

Color Digital Image: An Example

Color Red Channel Green Channel Blue Channel

- Color image = 3 color channels (images, with their own intensities) blended together
- Makes 3D data structure of size: Width X Height X Channels
- Each pixel has therefore 3 intensities: Red (R), Green (G), Blue (B)

Digital Image Formats

Raster image formats store a series of colored dots "pixels"

Number of bits for each pixel represents the depth of color

- 1 bit-per-pixel: 2 colors (black or white, binary)
- 4 bits-per-pixel: 16 colors
- 8 bits-per-pixel: 256 different colors

Digital Image Formats

Images can also be 16, 24, 32 bits-per-pixel:

- 24 bits per pixel usually means 8 bits per color
- At the two highest levels, the pixels themselves can carry up to 16,777,216 different colors

Common raster image formats:

- GIF, JPG, PPM, TIF, BMP, etc.
- Will discuss Camera RAW format later

Exercises to do on your own

- matlab
  - (mathworks.com)
- OpenCV/Python
  - (opencv.org, python.org)
- Processing
  - (processing.org)

Exercises: Read and write images

In Python-OpenCV:

```
> import cv2
> img = cv2.imread('input.png')
> cv2.imwrite('output.png', img)
```

Exercises: Read and write images

In matlab:

```
> img = imread('input.png')
> imwrite('output.png', img)
```

Other functions to consider:

```
> imshow(img)
> imageinfo('input.png')
> help images
```

Extract image properties

Read image, then find out:

- Dimensions of the image
- Number of color channels
- Bits per pixel

```
> print img.shape % (w, h, #channels)
> print img.dtype % NumPy types
```

Understand image formats

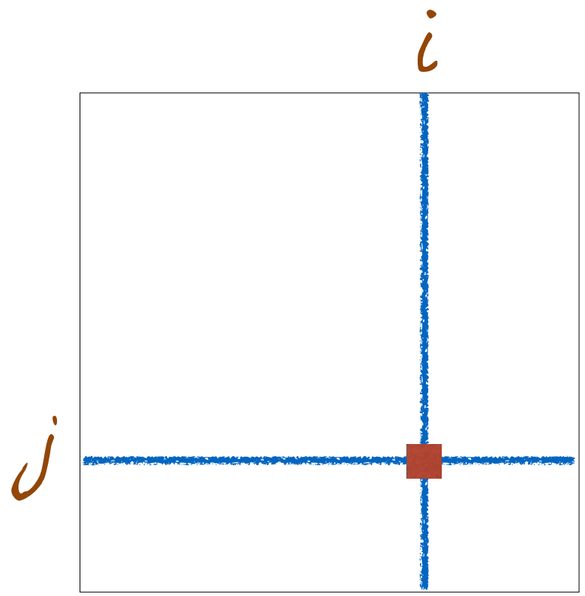
- Order of color channels
- Image compression information
- metadata about pictures
  (Exchangeable Image File Format:
  EXIF, etc.)

Summary

- Digital Image, and how it can be made into a computable object.
- Different formats of images using examples of black/white and color images.
- Introduced histograms, and simple computational methods over images to compute image statistics.

Next!

Digital Image: Processing and Filtering



Credits

- matlab software by Mathworks Inc.
- Some slides adapted from Steve Seitz and Aaron Bobick
- Mandrill Image from Signal and Image Processing Institute, University of Southern California,
  - http://sipi.usc.edu/database/download.php?vol=misc&img=4.2.03
- Georgia Tech's mascot Buzz image, courtesy Georgia Tech.

Computational Photography

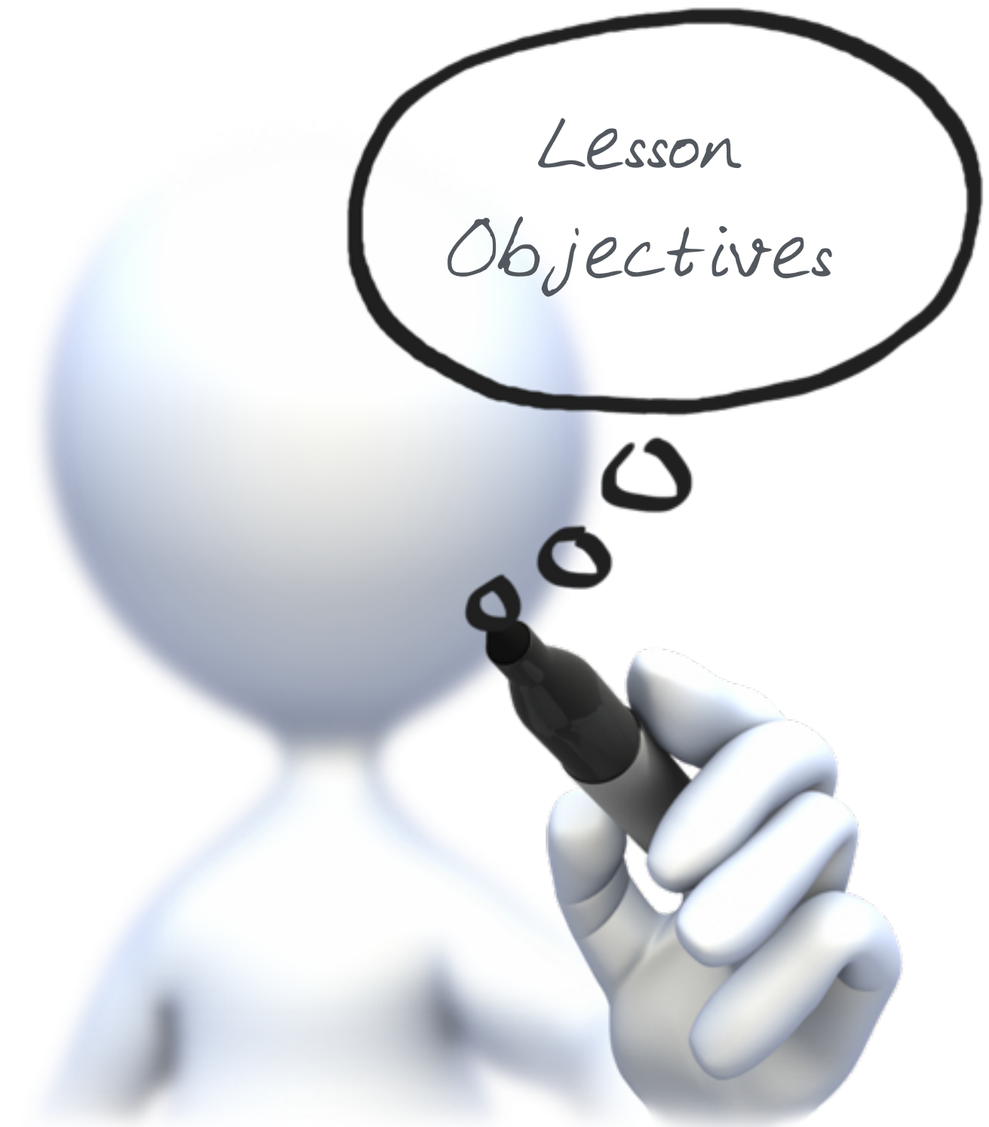
* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Image Processing and Filtering

- Point-process Computations on an Image
- How to combine intensities from 2 images?

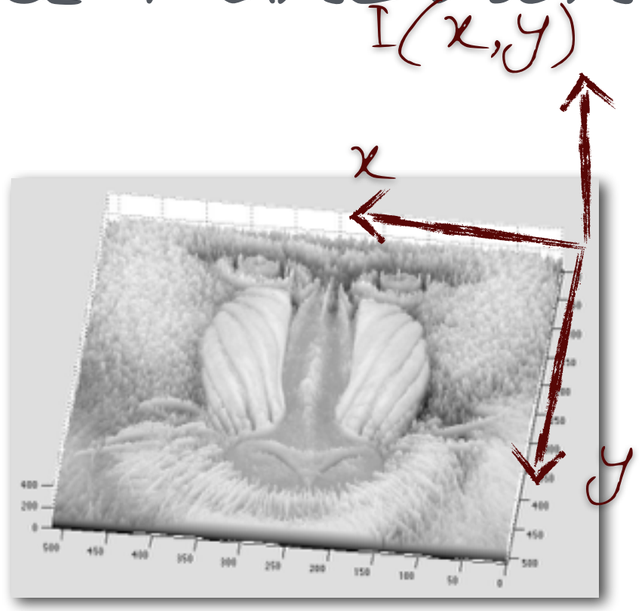
Georgia Tech College of Computing



* Point-process Computations.
* Add/Subtract Images
* α-blending & its applications
* Image Histograms

Recall: Digital Image is a Function

100	120	121	122	30	40
120	120	121	122	70	40
60	50	40	41	7	8
100	120	121	122	1	0
200	120	200	122	12	14
200	220	225	250	30	40



* How to obtain discrete values?
  * Sample the two-dimensional (2D) space on a regular grid
  * Quantize each sample (round to "nearest integer")
* Result: matrix of integer values (range, e.g.: 0-255)

Slide adapted from Steve Seitz and Aaron Bobick

Point-Process: Pixel/Point Arithmetic

120	122	140	142	143
121	120	141	144	147
122	121	144	146	11
125	121	144	145	10
126	121	145	147	13

+

120	122	140	142	143
121	80	40	144	10
122	81	40	0	151
125	80	40	0	152
126	70	40	0	153

=

240	244	280	284	286
121	200	181	288	157
122	202	184	146	162
125	201	184	145	164
126	191	185	147	166

120	122	140	142	143
121	120	141	144	147
122	121	144	146	11
125	121	144	145	10
126	121	145	147	13

−

120	122	140	142	143
121	80	40	144	10
122	81	40	0	151
125	80	40	0	152
126	70	40	0	153

=

0	0	0	0	0
0	40	101	0	137
0	40	104	146	-140
0	40	104	145	-142
0	191	185	147	-140

Pixel/Point Arithmetic: An Example

Image 1

Image 2

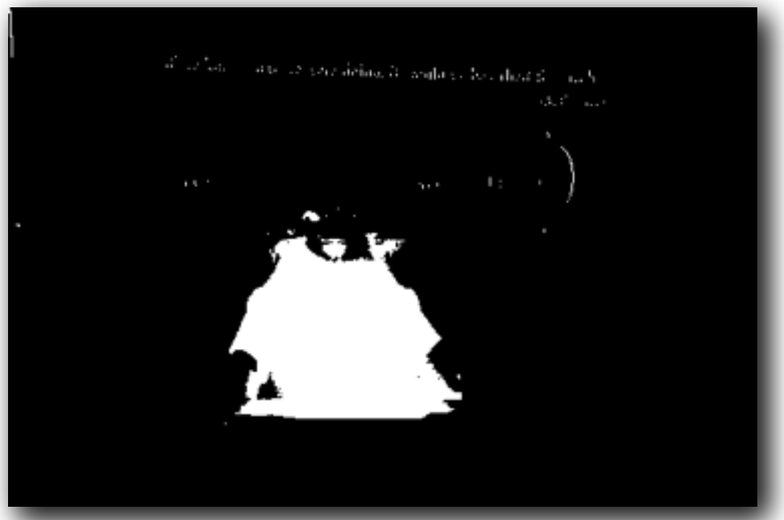
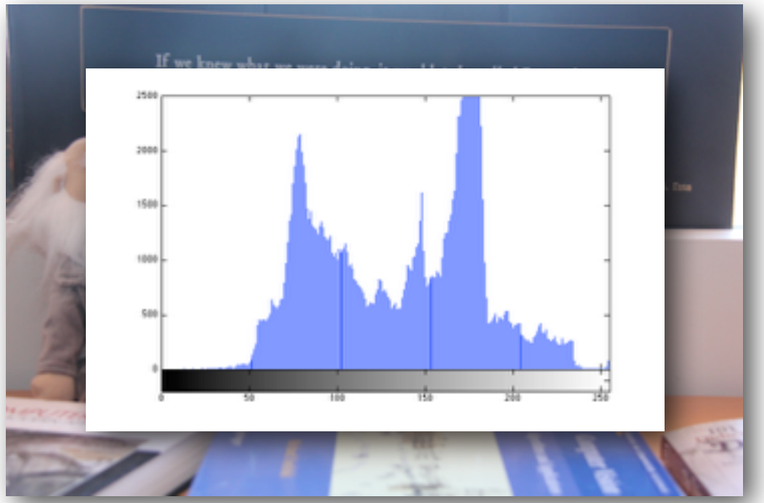


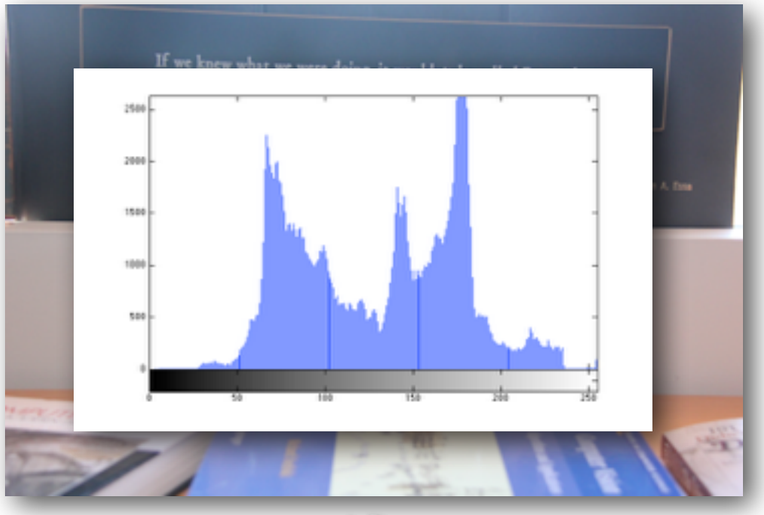
Image 1 - Image 2

Binary(Image 1 - Image 2)

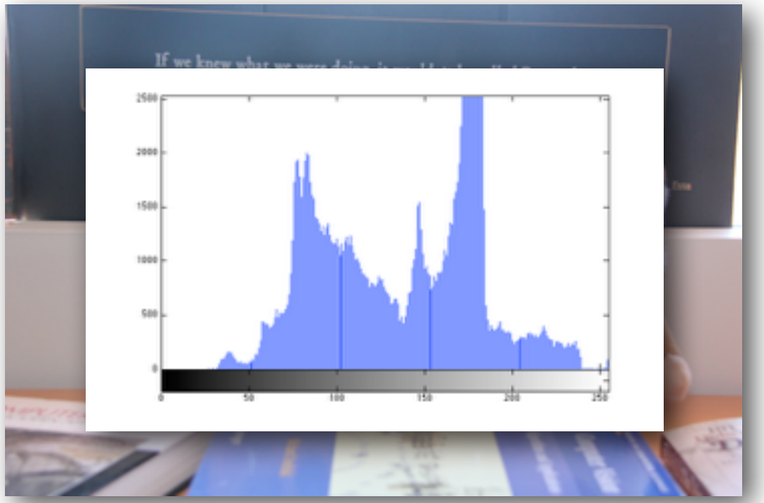
Pixel Operations: Another Example



CD

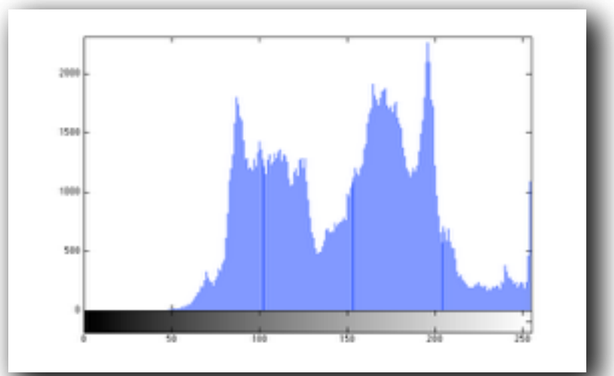


AE



LD

CD+AE+LD=



.34 X CD+ .34 X AE+ .34 X LD=

Alpha-blending (α-blending)

.34 x CD + .34 x AE + .34 x LD =

* Transparency (Conversely, Opacity!)
* Usually represented as: **α**
* **α** varies from 0 to 1.0 (0=invisible, 1.0=fully visible)
* RGB → **α**RGB

Summary

* Point-process Computations on Images to Add and Subtract images.
* Showed an example of **α**-blending commonly used in Image Processing.
* Use of Image Histogram in Image Processing.

Next Class

Image Processing and Filtering, via Convolution and Correlation

Credits

* Matlab software by Mathworks Inc.
* Some content adapted from Steve Seitz and Aaron Bobick
* Images
  * by Irfan Essa

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Point Processes: Pixel Blending Modes

* A Real Life example of point arithmetic
* Variety of Blending modes built on concept of Point Processes

Georgia Tech College of Computing

Point Processes: "What is with the weird artifacts in the lecture videos?

* A Real Life example of point arithmetic
* Variety of Blending modes built on concept of Point Processes

Georgia Tech College of Computing

Pixel/Point Arithmetic: An Example

Image 1

Image 2

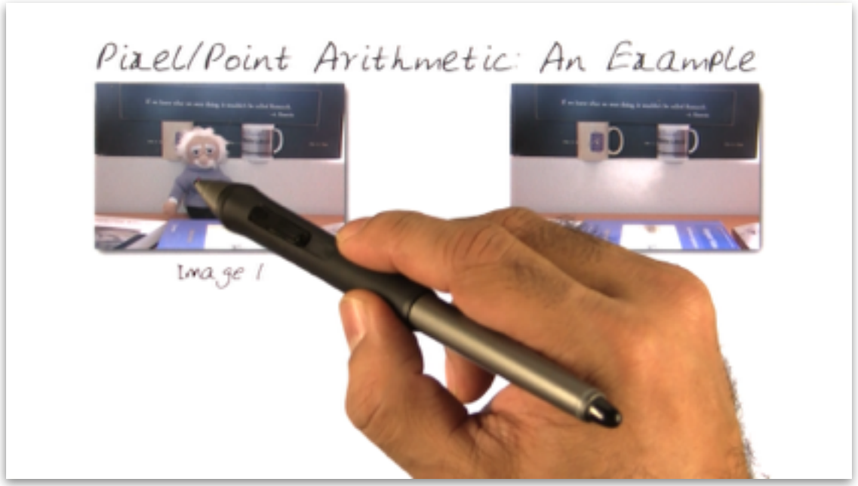
1. Explain what is happening on the videos you are seeing

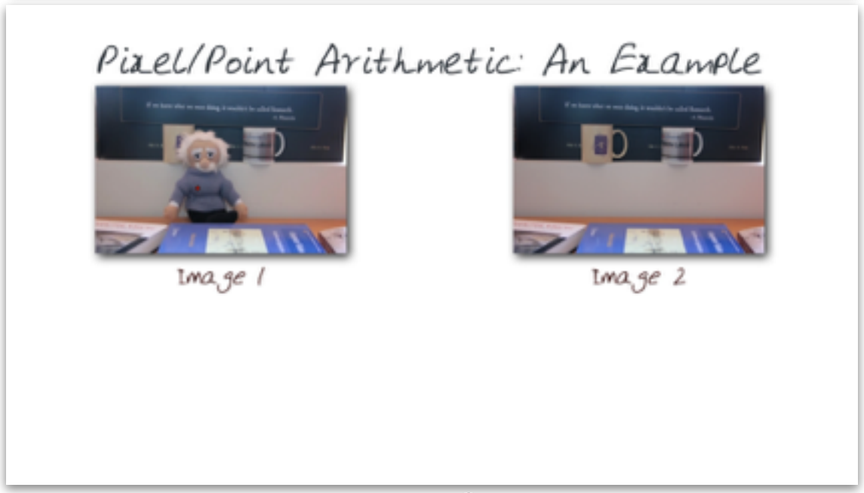
2. Introduce the variety of blending modes in wide use

A bit about the setup

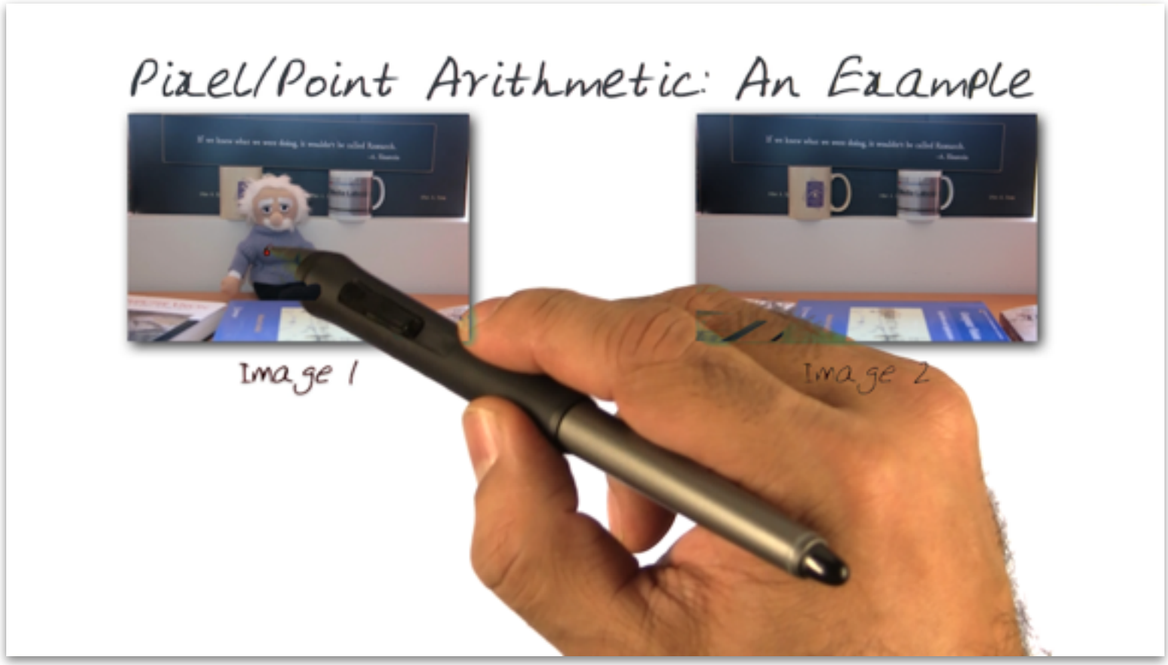
The final output is a pixel blend

Camera





Screen Capture



Screen Capture

+

Camera

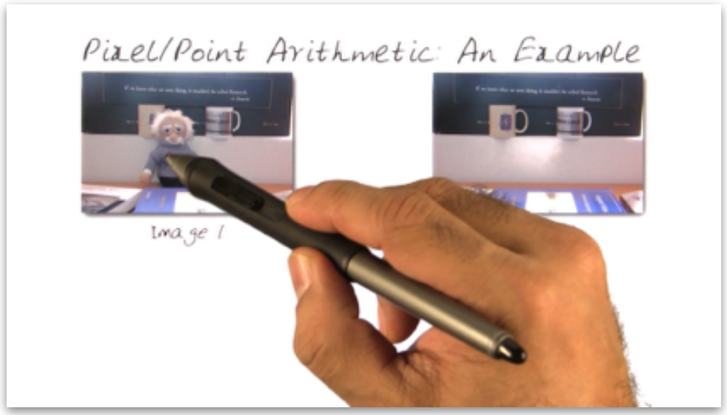
Blending Pixels

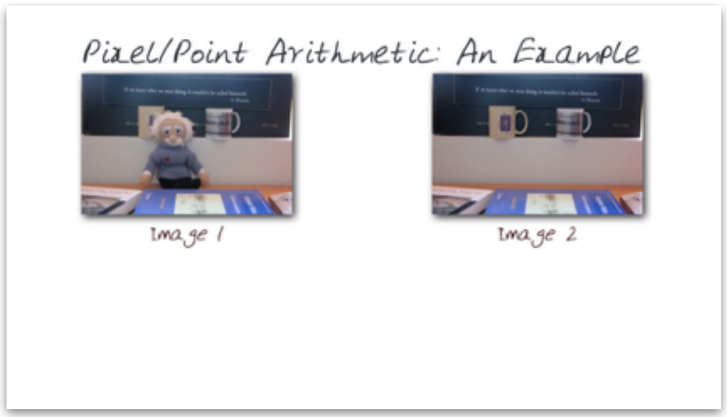
* Blend two pixels from two images: $f_{blend}(a, b)$
* For example
  * average:

$$f_{blend}(a, b) = (a + b)/2$$

  * normal:

$$f_{blend}(a, b) = b$$





Arithmetic Blend Modes

* Divide (brightens photos)
* Addition (too many whites)
* Subtract (too many blacks)
* Difference (subtract with scaling)
* Darken: $f_{blend}(a, b) = min(a, b)$ for RGB
* Lighten: $f_{blend}(a, b) = max(a, b)$ for RGB

Advanced Modes

* multiply $f_{blend}(a,b) = ab$
  * darker
* screen $f_{blend}(a,b) = 1 - (1-a)(1-b)$
  * brighter
* Overlay $f_{blend}(a,b) = \begin{cases} 2ab, & \text{if } a < 0.5 \\ 1 - 2(1-a)(1-b), & \text{otherwise} \end{cases}$
  * The parts of the top layer where base layer is light become lighter, the parts where the base layer is dark become darker

Dodge and Burn

* Dodge and burn change the lightness of the pictures
* Dodging lightens an image, while burning darkens it.
* Dodge builds on Screen mode
* Burn builds on Multiply mode
* There are numerous variations of each!

Darken

Pixel/Point Arithmetic: An Example

Summary

* Introduced Pixel/Layer Blending
* Explained variety of Blending approaches
* Showed why some of the videos look ODD!

Credits

* For more information, see:
  * http://en.wikipedia.org/wiki/Blend_modes
  * http://blog.udacity.com/2014/09/udacity-videos-transparent-hand.html

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Computational Photography

- Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

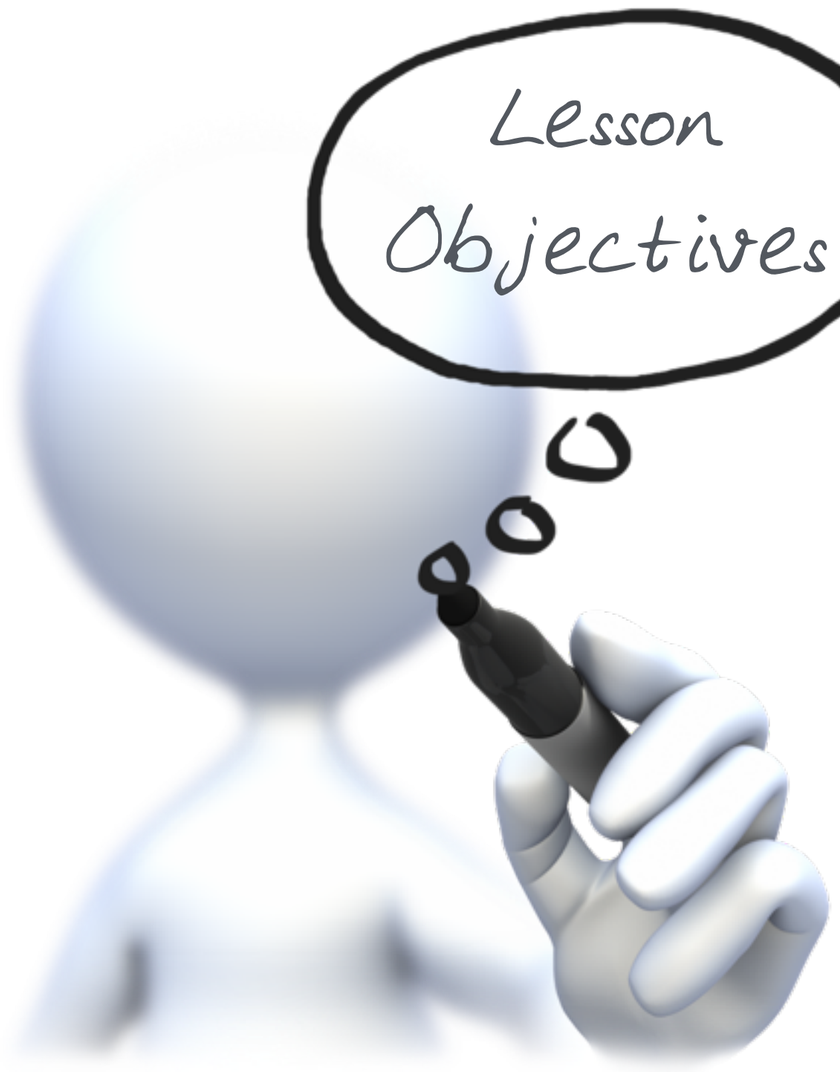
Georgia Tech College of Computing

Image Processing and Filtering: Smoothing

- Point-process and Neighboring Pixels

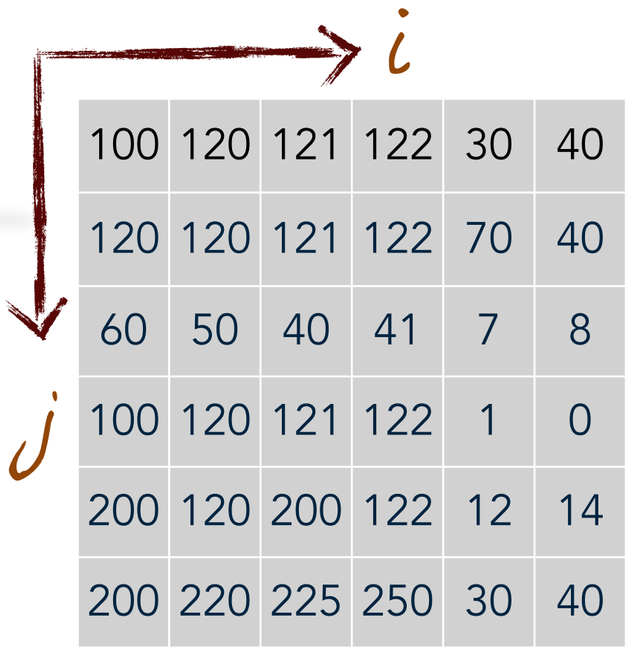
Computations on an Image for Image Smoothing

Georgia Tech College of Computing

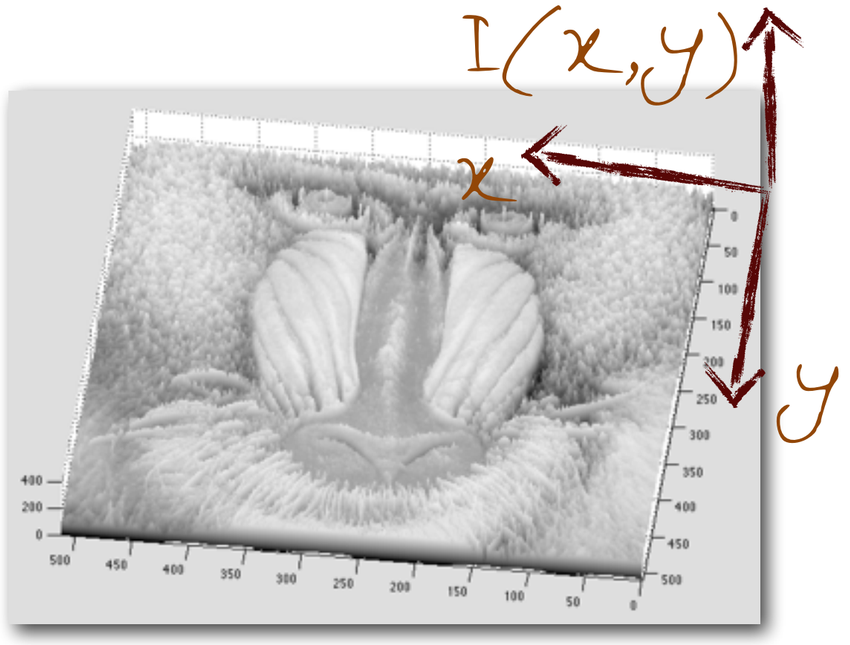


1. Smooth an image over a neighborhood of pixels
2. median filtering as a special non-linear filtering and smoothing approach

Recall: Digital Image is a Function



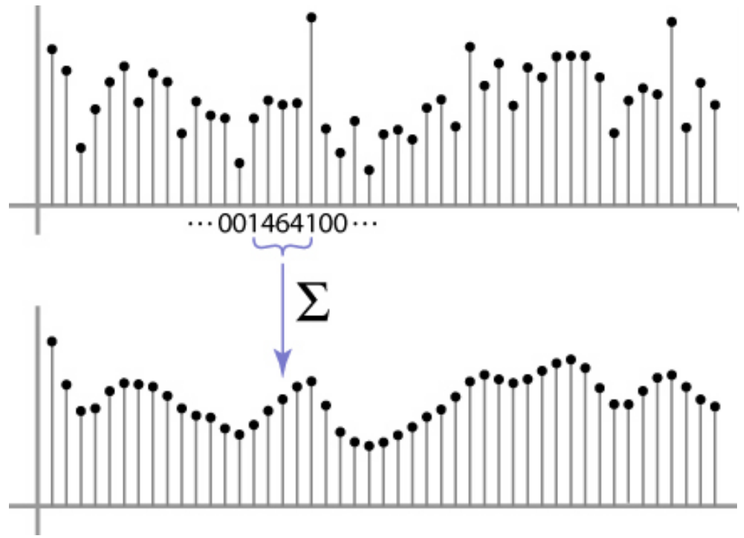
100	120	121	122	30	40
120	120	121	122	70	40
60	50	40	41	7	8
100	120	121	122	1	0
200	120	200	122	12	14
200	220	225	250	30	40

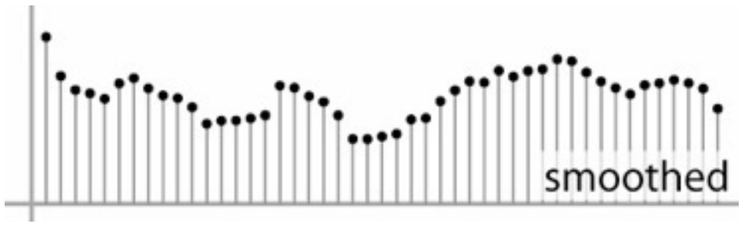


Slide adapted from Steve Seitz and Aaron Bobick

From Pixel/Point Operations to Groups of Pixels

120	122	140	142	143	10
121	120	141	144	147	151
122	121	144	146	11	152
125	121	144	145	10	153
126	121	145	147	13	12
120	122	140	142	143	10





- How to Smooth a Signal?
- Determine the Average of Neighboring Values

1. Moving Average [1 1 1 1 1] X 1/5
2. Weighted Moving Average [1 4 6 4 1] X 1/16

Slide adapted from Aaron Bobick and Steve Marschner

Smoothing Process over an Image using Averages

input (i,j)

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	0	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	90	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	0

output (i,j)

Slide adapted from Aaron Bobick and Steve Marschner

Smoothing Process over an Image using Averages

input (i,j)

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	0	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	90	90	90	90	0	0
0	90	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	0

output (i,j)

	0	10	20	30	30	30	20	10	
	0	20	30	50					

Slide adapted from Steve Seitz and Aaron Bobick

Smoothing Process over an Image using Averages

input (i,j)

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	0	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	90	90	90	90	0	0
0	90	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	0

output (i,j)

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

Slide adapted from Steve Seitz and Aaron Bobick

Smoothing Process over an Image using Averages

input (i,j)

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	0	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	90	90	90	90	0	0
0	90	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	0

output (i,j)

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

Results in Shades of Gray

Slide adapted from Aaron Bobick and Steve Marschner

Smoothing Process for the Edges of an Image

20	20	0	20	10	20	10	20	10	0	10	10
20	20	0	20	10	20	10	20	10	0	10	10
30	30	0	0	0	0	0	0	0	0	10	10
20	20	0	0	90	90	90	90	90	0	20	20
20	20	0	0	90	0	90	90	90	0	10	10
10	10	0	0	90	90	90	90	90	0	20	20
10	10	0	0	90	90	90	90	90	0	10	10
10	10	0	0	90	90	90	90	90	0	10	10
20	20	90	0	0	0	0	0	0	0	10	10
20	20	0	0	0	0	0	0	0	0	10	10
20	20	0	20	10	20	10	20	10	0	10	10
20	20	0	20	10	20	10	20	10	0	10	10

Some Options

1. wrap around
2. copy edge
3. reflect across edge

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

Gray values are for demonstration only, not accurate

Smoothing Process for the Edges of an Image

20	20	0	20	10	20	10	20	10	0	10	10
20	20	0	20	10	20	10	20	10	0	10	10
30	30	0	0	0	0	0	0	0	0	10	10
20	20	0	0	90	90	90	90	90	0	20	20
20	20	0	0	90	0	90	90	90	0	10	10
10	10	0	0	90	90	90	90	90	0	20	20
10	10	0	0	90	90	90	90	90	0	10	10
10	10	0	0	90	90	90	90	90	0	10	10
20	20	90	0	0	0	0	0	0	0	10	10
20	20	0	0	0	0	0	0	0	0	10	10
20	20	0	20	10	20	10	20	10	0	10	10
20	20	0	20	10	20	10	20	10	0	10	10

Some Options

1. wrap around
2. copy edge
3. reflect across edge

Gray values are for demonstration only, not accurate

Some observations ...

* Small image "rubbed" over bigger image

kernel h(i,j)

a	b	c
d	e	f
g	h	i

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	90	90	90	90	90	0	20
20	0	10	20	30	90	90	0	20
10	0	90	80	90	90	90	0	10
10	0	90	70	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

* 3x3 area around each original pixel used (i.e., neighborhood size k=1)
* Window size is 2k+1, therefore our window is 3x3

Observations (continued)

Let's refer to F[i,j] as input and

G[i,j] as output, h[i,j] as the kernel

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

$$G[3,3] = a*A + b*B + c*C + d*D + e*E + f*F + h*H + i*I$$

a	b	c
d	e	f
g	h	i

A	B	C
D	E	F
G	H	I

$$G[3,3] = \frac{1}{9}(A + B + C + D + E + F + G + H + I)$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

A Mathematical Representation for Smoothing

$$G[3,3] = \frac{1}{9}(A+B+C+D+E+F+G+H+I)$$

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

Attribute uniform weight to each pixel

Loop over all pixels in neighborhood around image pixel $F[i,j]$

a	b	c
d	e	f
g	h	i

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

A Mathematical Representation for Smoothing

$$G[3,3] = \frac{1}{9}(A + B + C + D + E + F + G + H + I)$$

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

a	b	c
d	e	f
g	h	i

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

More Generally,

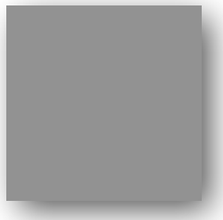
$$G[3,3] = a * A + b * B + c * C + d * D + e * E + f * F + g * G + h * H + i * I$$

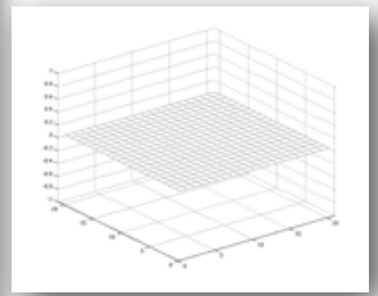
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i+u, j+v]$$

Attribute non-uniform weights

Referred to as Cross-correlation, which we will cover later

Example: Box Filter (Averaging!) for Smoothing





Box/Average Filter - 21 x 21

Special Case: Median Filtering

20	20	1	20	10	20	10	13	13
20	0	0	0	0	0	0	20	20
20	0	90	90	90	90	90	20	20
20	0	90	0	90	90	90	20	20
10	0	90	90	90	90	90	10	10
10	0	90	90	90	90	90	10	10
10	0	90	90	90	90	90	10	10
20	0	0	0	0	0	0	20	20
20	20	10	20	10	20	10	13	13

F [i,j]

median(20 20 1 / 20 0 0 / 20 0 90)

median(0 0 0 1 20 20 20 20 90) = 20

average(0 0 0 1 20 20 20 20 90) = 19

Apply median filter to all 3x3 regions over the whole image (as we would for averaging)

Special Case: Median Filtering

Median filtering: Nonlinear operation often used in image processing

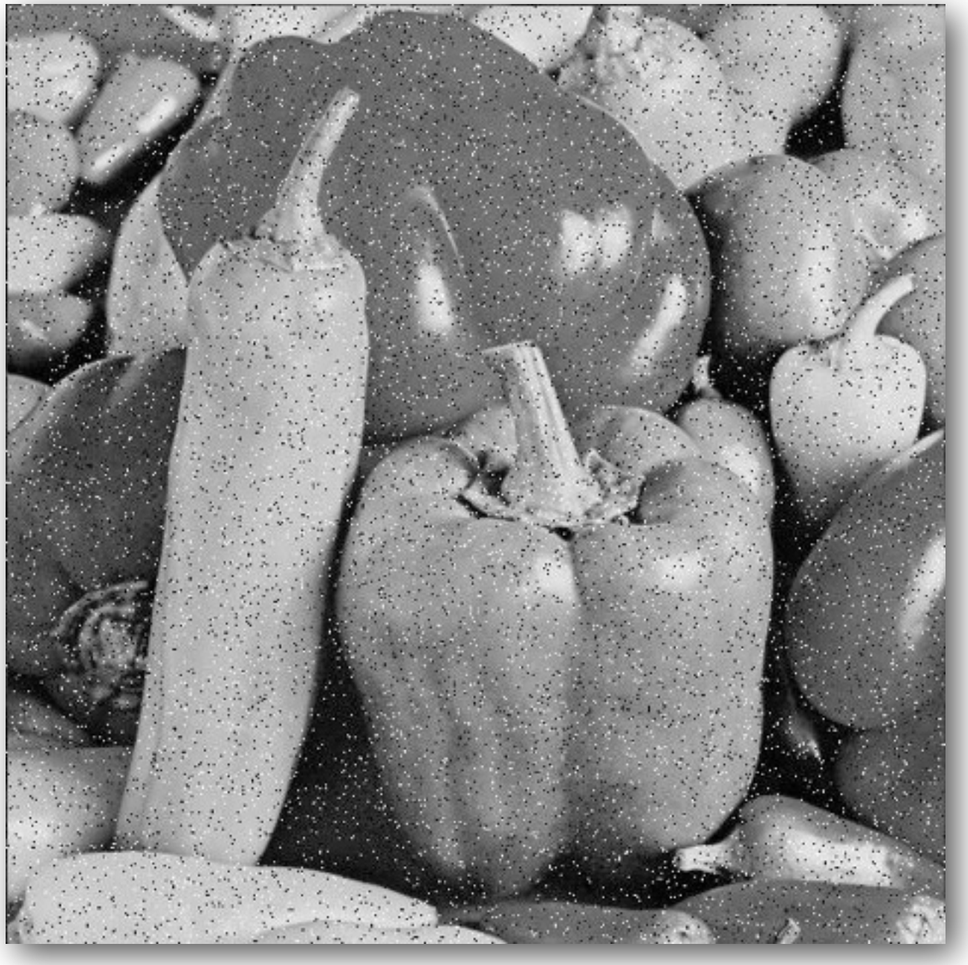
* reduces noise, but,
* preserves edges (sharp lines!)

Main idea: Use median of all pixels in kernel area, instead of mean

Example: Median Filtering for Smoothing Images

Median filtering (MF)

Example: Median Filtering for Noise Removal



'Salt & Pepper' Noise

Median Filtered (3 x 3)

Summary

1. Image smoothing
2. Applying a kernel to smooth an image
3. Averaging and median filtering

Next Class

Image Analysis: Cross-correlation and Convolution

Credits

- matlab software by Mathworks Inc.
- Some Slides adapted from Aaron Bobick, Steve Seitz, Steve Marschner.
- Images used from USC's Signal and Image Processing Institute's Image Database

Computational Photography

- Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Image Processing and Filtering, via Convolution and Cross-Correlation

* Point-process and Neighboring Pixels Computations on an Image using Cross-Correlation and Convolution

Georgia Tech College of Computing

1. Cross-Correlation
2. Convolution
3. Difference between Cross-Correlation and Convolution
4. Properties of these methods!

Recall A Mathematical Representation for Smoothing

$$G[3,3] = \frac{1}{9}(A + B + C + D + E + F + G + H + I)$$

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

a	b	c
d	e	f
g	h	i

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

More Generally,

$$G[3,3] = a * A + b * B + c * C + d * D + e * E + f * F + g * G + h * H + i * I$$

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

Referred to as Cross-correlation, which we will cover later

Cross-Correlation Method

★ In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them.

★ Also known as a sliding dot product or sliding inner-product.

Cross-Correlation Method

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

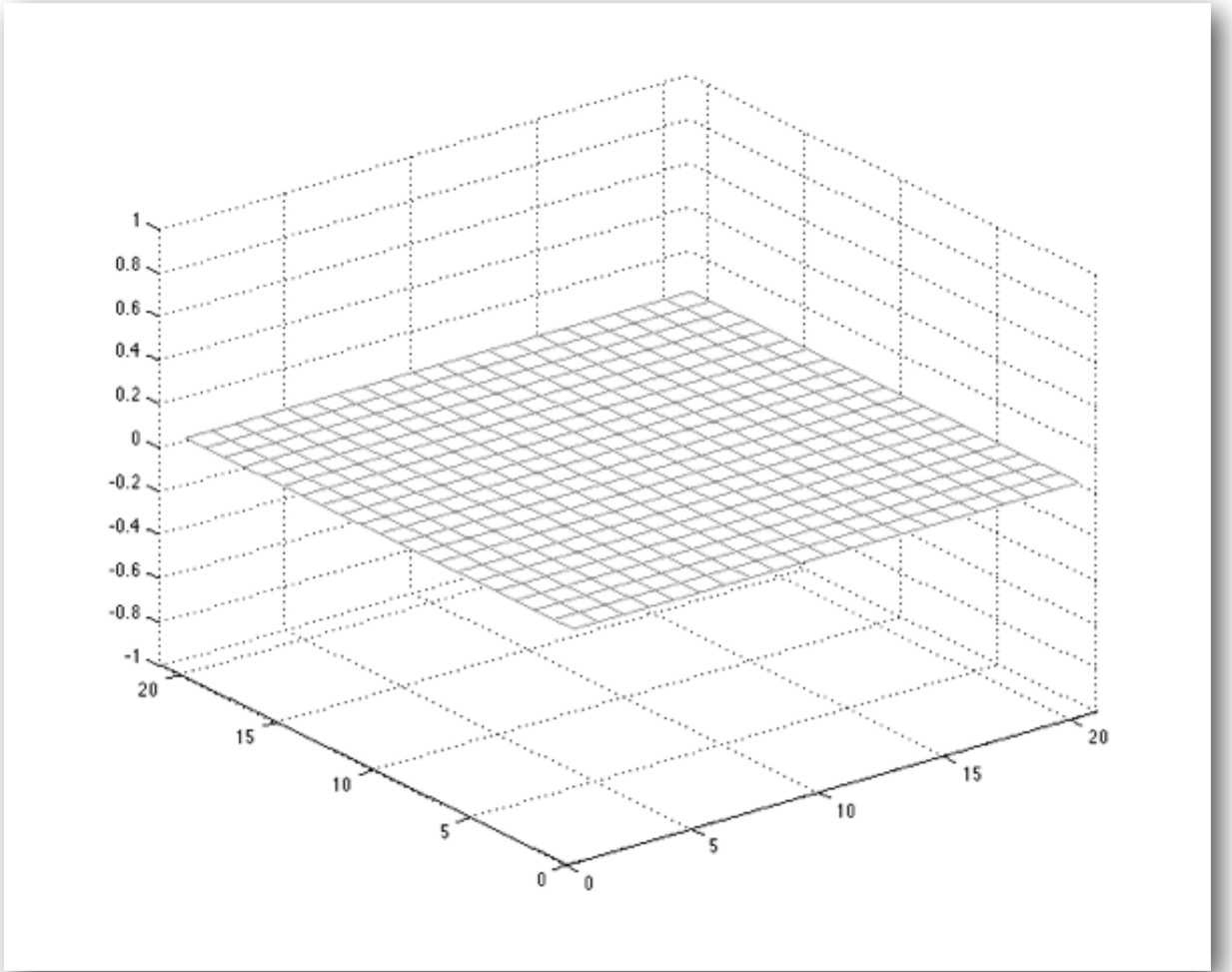
Denoted by $G = h \otimes F$

* Filtering an image:
  * Replace each pixel with a linear combination of its neighbors
* Filter "**kernel**" or "**mask**"
  * **h[u,v]** is the prescription for weights in the linear combination

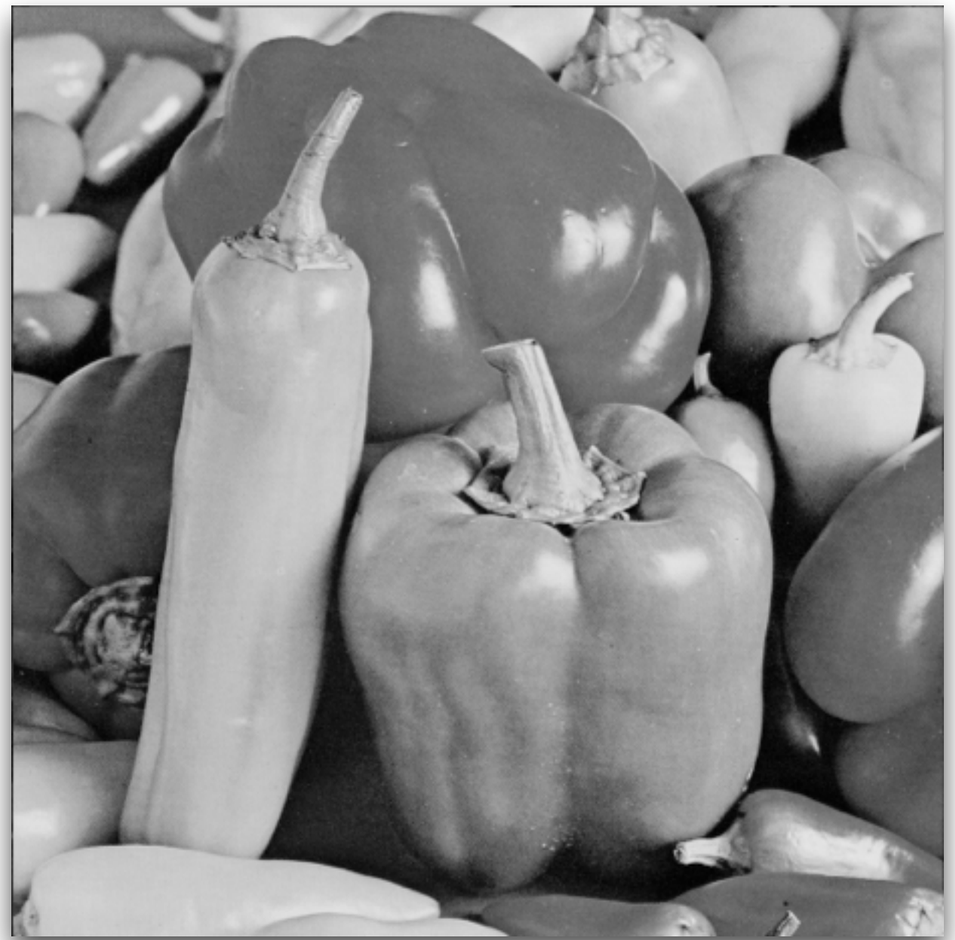
Example: Box Filter

Box/Average Filter

- Size: 21x21
- Values: Uniform



Example: Box Filter



Original

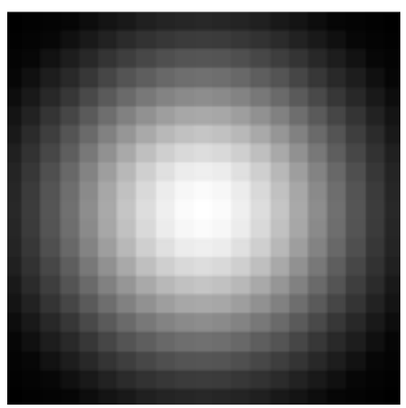
Filter

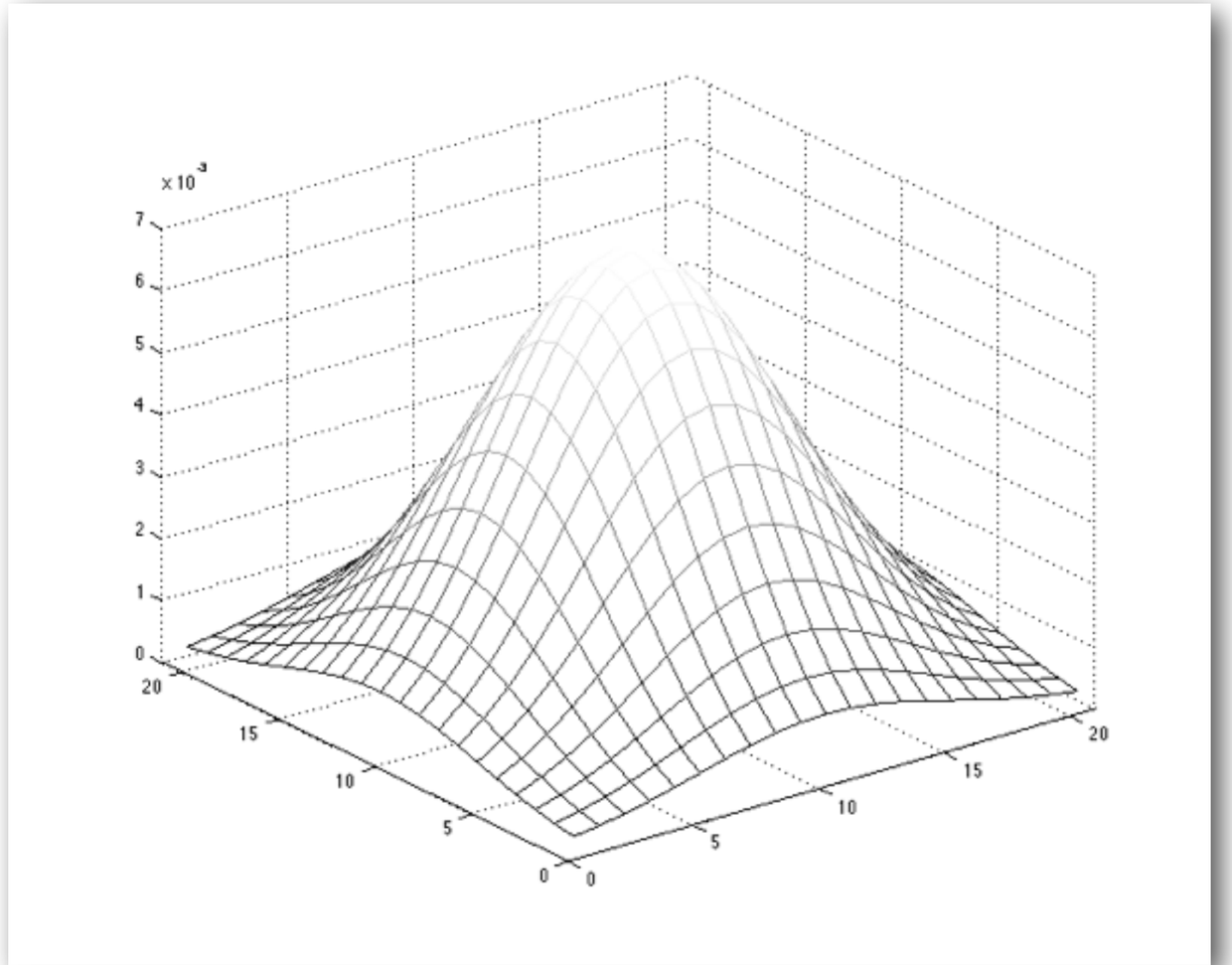
Result

Example: Gaussian Filter

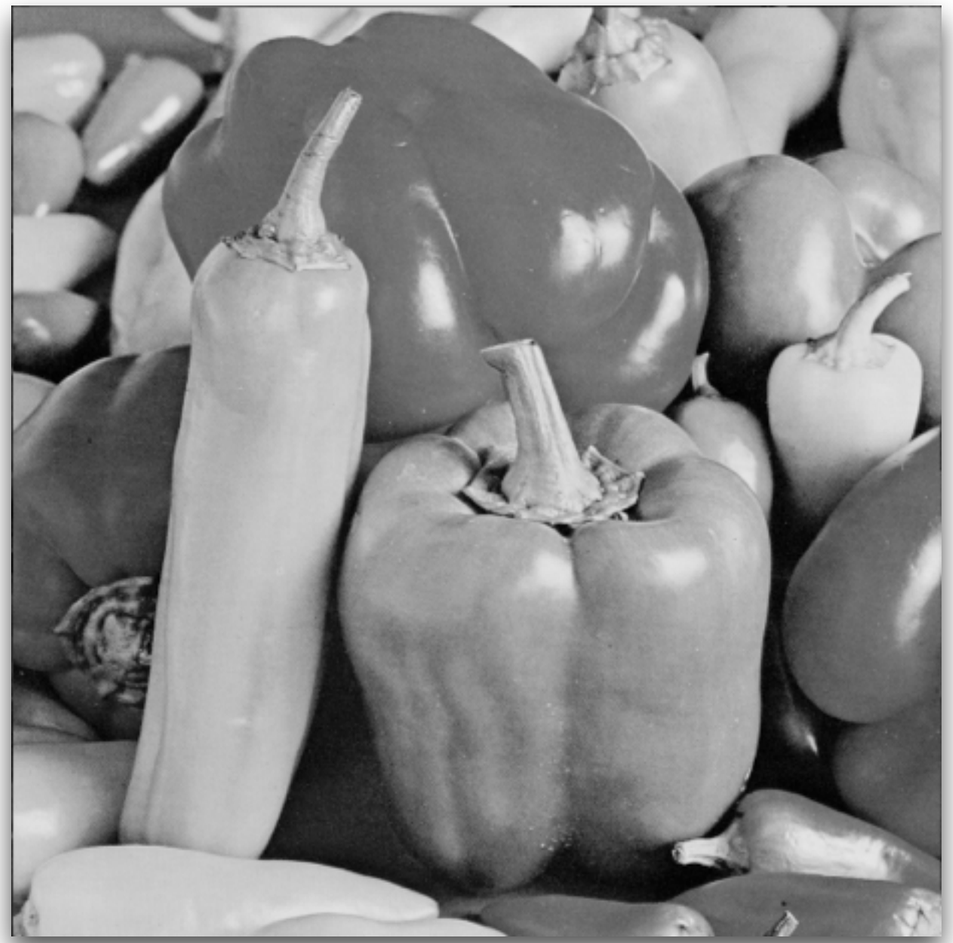
Gaussian Filter

- Size: 21x21
- Values: Gaussian or Normal distribution

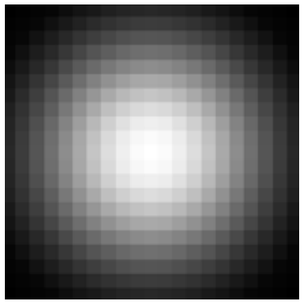




Example: Gaussian Filter



Original



Filter

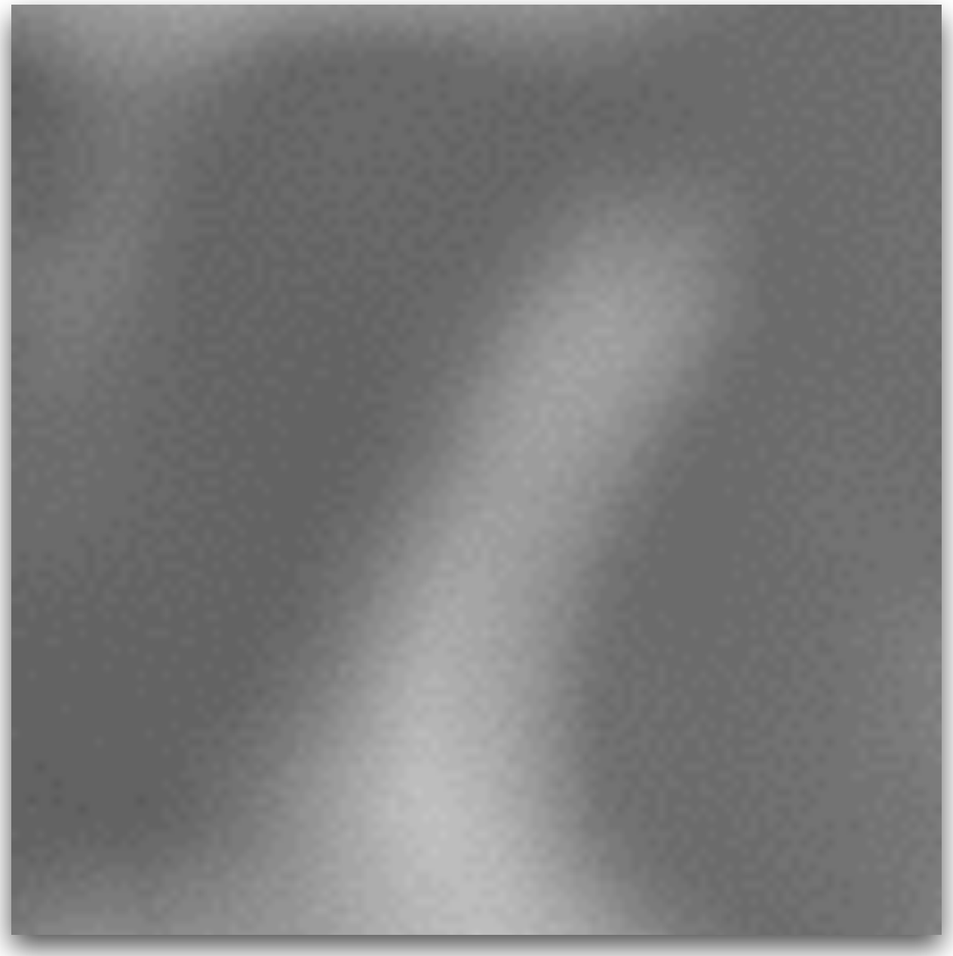
Result

Compare: Average vs. Gaussian

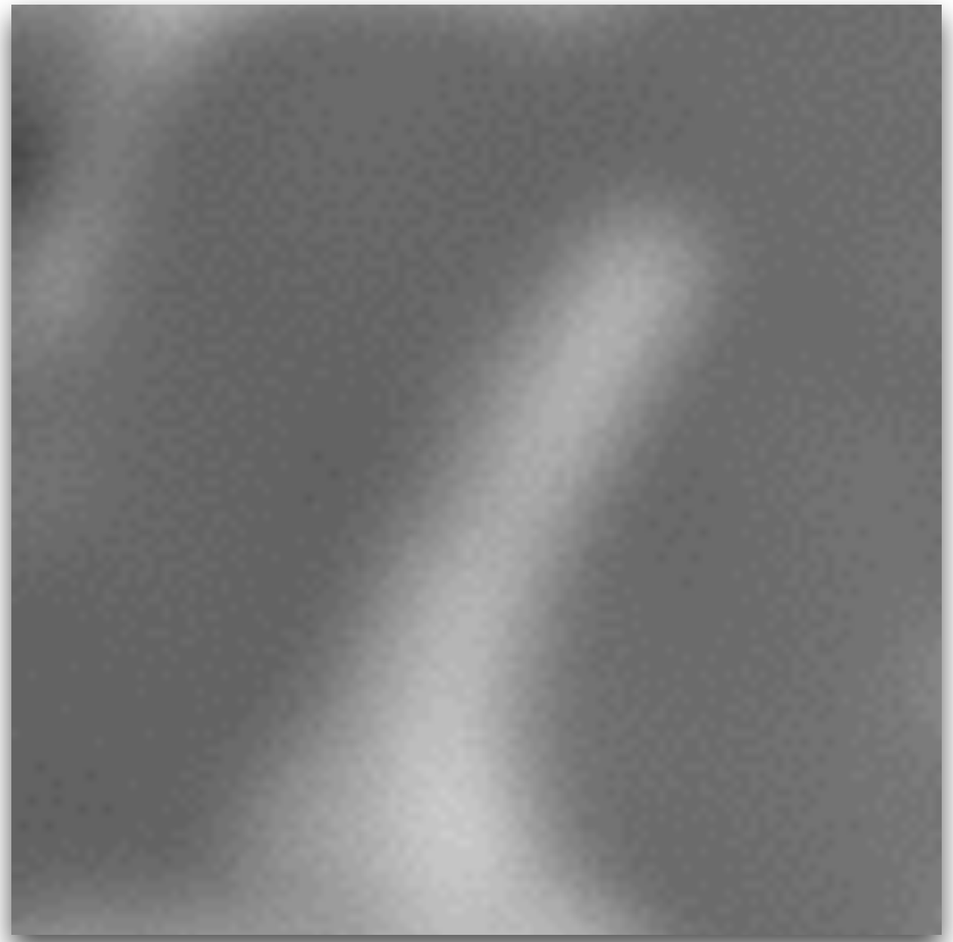
Average filter result

Gaussian filter result

Compare: Average vs. Gaussian



Average filter result

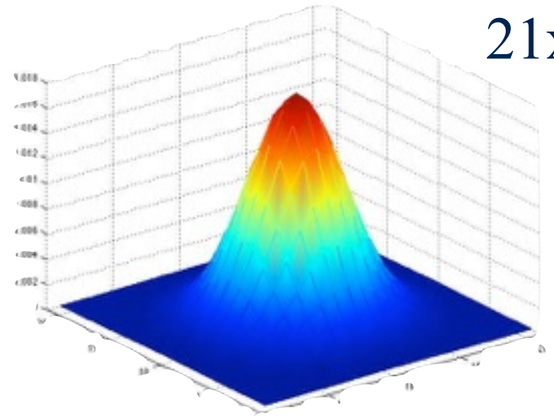


Gaussian filter result

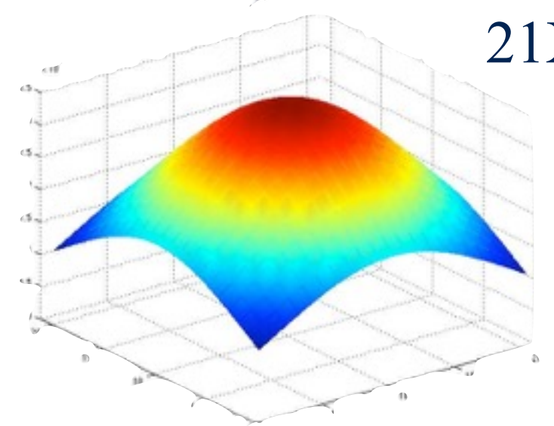
Using Gaussian Filters?

* Square kernels are NOT smooth
* Average filter not eq. to a defocussed lens
  * A single point of light viewed in a defocussed lens looks like a fuzzy blob; the averaging process is square.
* Gaussian function in 2D, with σ as the Variance, centered at (0,0):

21x21, σ=3



21X21, σ=9



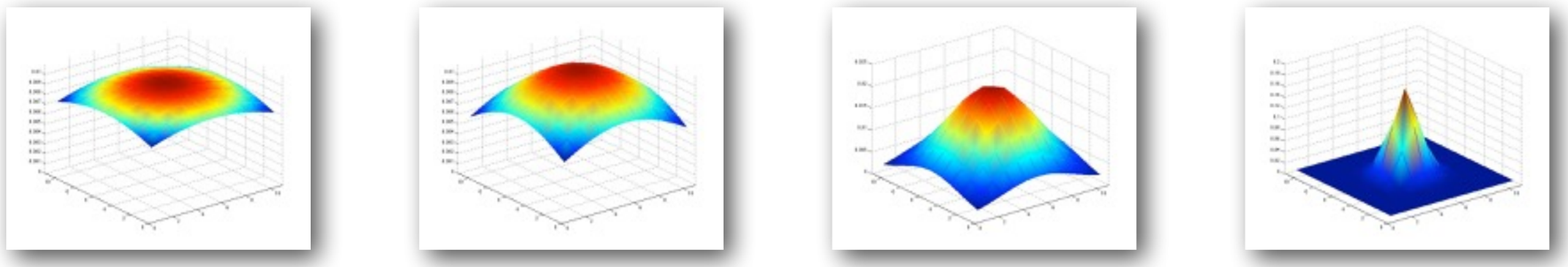
$$h(u,v) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{(u^2+v^2)}{2\sigma^2}}$$

Slide Adapted from Aaron Bobick and David Forsyth

For more information on Gaussian see http://en.wikipedia.org/wiki/Gaussian_function

Using Gaussian Filters for Smoothing

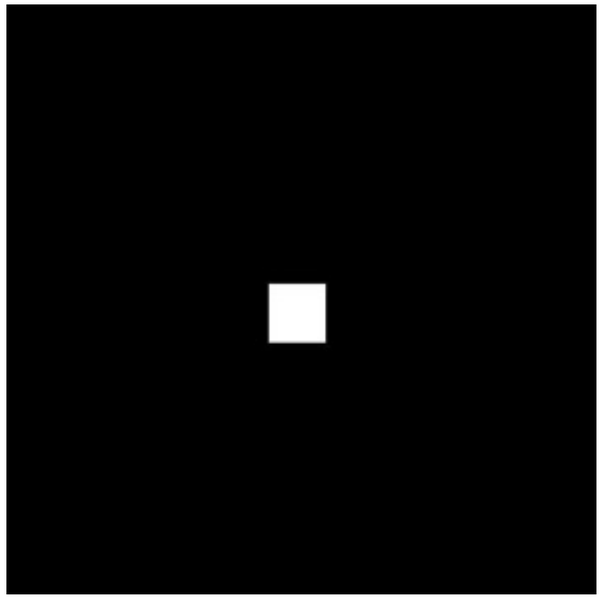
Original, 256 X 256



$\sigma=9$ $\sigma=6$ $\sigma=3$ $\sigma=1$

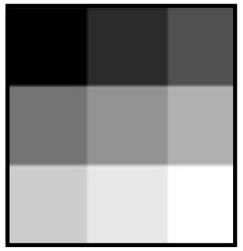
σ determines extent of smoothing

Filtering by a Kernel (defining Convolution)



F[i,j]

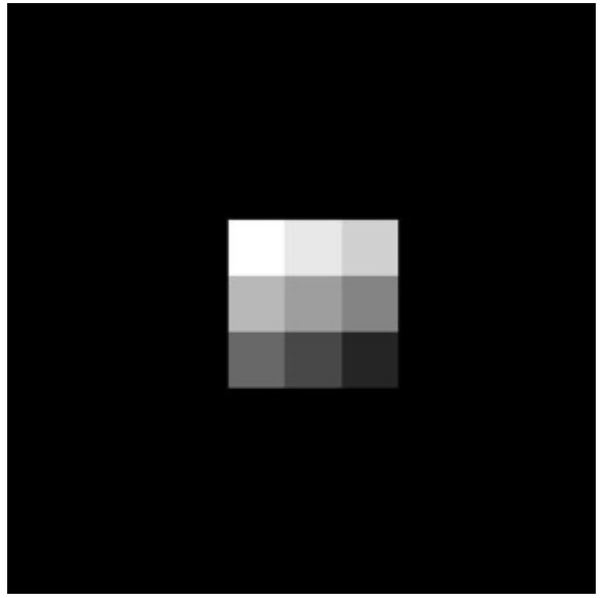
$\otimes$



h[i,j]

$=$

a	b	c
d	e	f
g	h	i



G[i,j]

* 'Filter' means to slide the kernel over the image
* Results in a reversed response

Slide Adapted from Aaron Bobick

Convolution Method

* Convolution is a mathematical operation on two functions **F** and **h**,
* Produces a third function that is typically viewed as a modified version
* Gives the area of overlap between the two functions
* In a form of the amount that one of the original functions is translated.

Convolution Method

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

Denoted by $G = h * F$

* Flip filter in both dimensions:
  * Bottom to top

a	b	c
d	e	f
g	h	i

  * Right to left

g	h	i
d	e	f
a	b	c

i	h	g
f	e	d
c	b	a

* Then apply cross-correlation

Convolution vs. Cross-Correlation

Cross-Correlation:

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

a	b	c
d	e	f
g	h	i

Convolution:

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

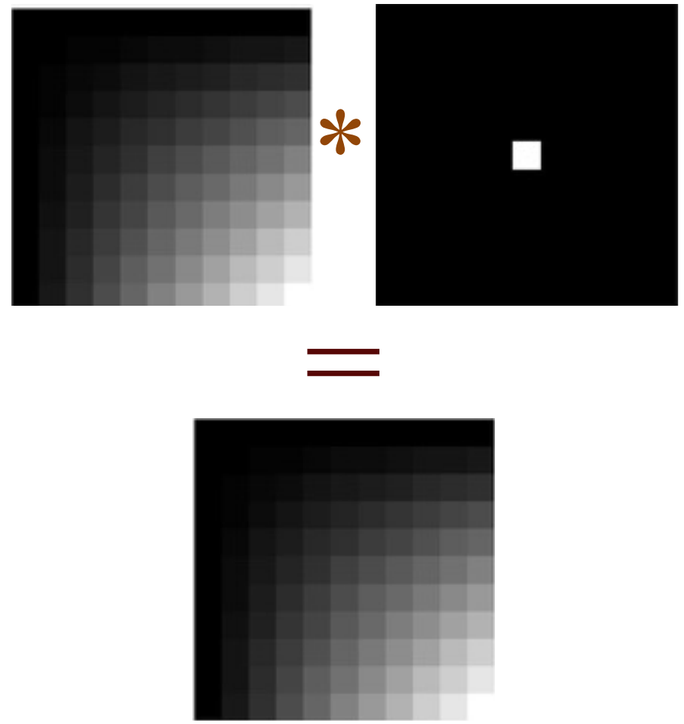
g	h	i
d	e	f
a	b	c

i	h	g
f	e	d
c	b	a

Properties of Convolution

* Linear and Shift Invariants:
  * Behaves the same everywhere (i.e., the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood).
* Commutative: $F * G = G * F$
* Associative: $(F * G) * H = F * (G * H)$

Properties of Convolution

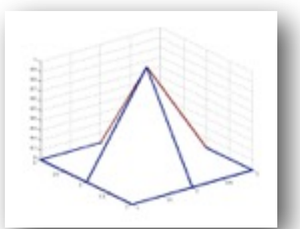


* Identity: Unit Impulse
  * $E = [....0,0,1,0,0....]$,
  * $F * E = F$
  * True of Cross-Correlation?
* Separable:
  * If the filter is separable, convolve all rows, then convolve all columns.

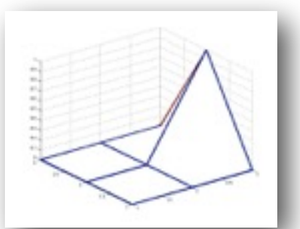
Linear Filters

original, 64 X 64

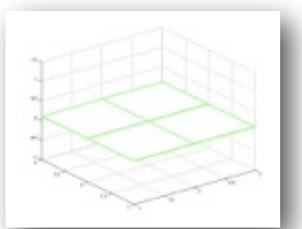
0	0	0
0	1	0
0	0	0



0	0	0
0	0	1
0	0	0



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Linear Filters

0	0	0
0	1	0
0	0	0

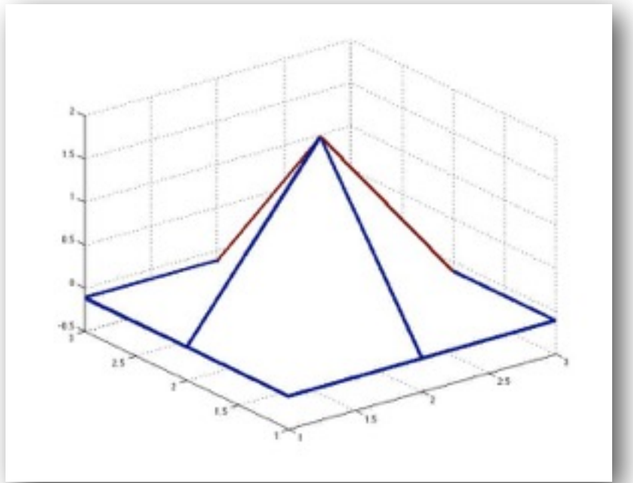
x 2 -

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

original, 64x64

-0.11	-0.11	-0.11
-0.11	1.9	-0.11
-0.11	-0.11	-0.11



Summary

* Cross-Correlation
* Convolution
* Differences between the Cross-Correlation and Convolution methods for filtering images.
* Properties of the Convolution method for filtering images.

Next Class

Image Analysis:

Edge Detection

Credits

* matlab software by Mathworks Inc.
* Some Slides adapted from Aaron Bobick, Steve Seitz, Steve Marschner & David Forsyth.
* Images used from USC's Signal and Image Processing Institute's Image Database

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Image Processing and Filtering, via Convolution and Cross-Correlation

* Towards Edge Detection: Computing Image Gradients

Lesson Objectives

1. Detect Features in an image.
2. Edge of an image from the perspective of Information Theory.
3. Use of an Image Gradient to compute Edges.
4. Image Gradient in continuous form for a function; and in a discrete form for an image.

Recall: Convolution and Cross-Correlation

Cross-Correlation:

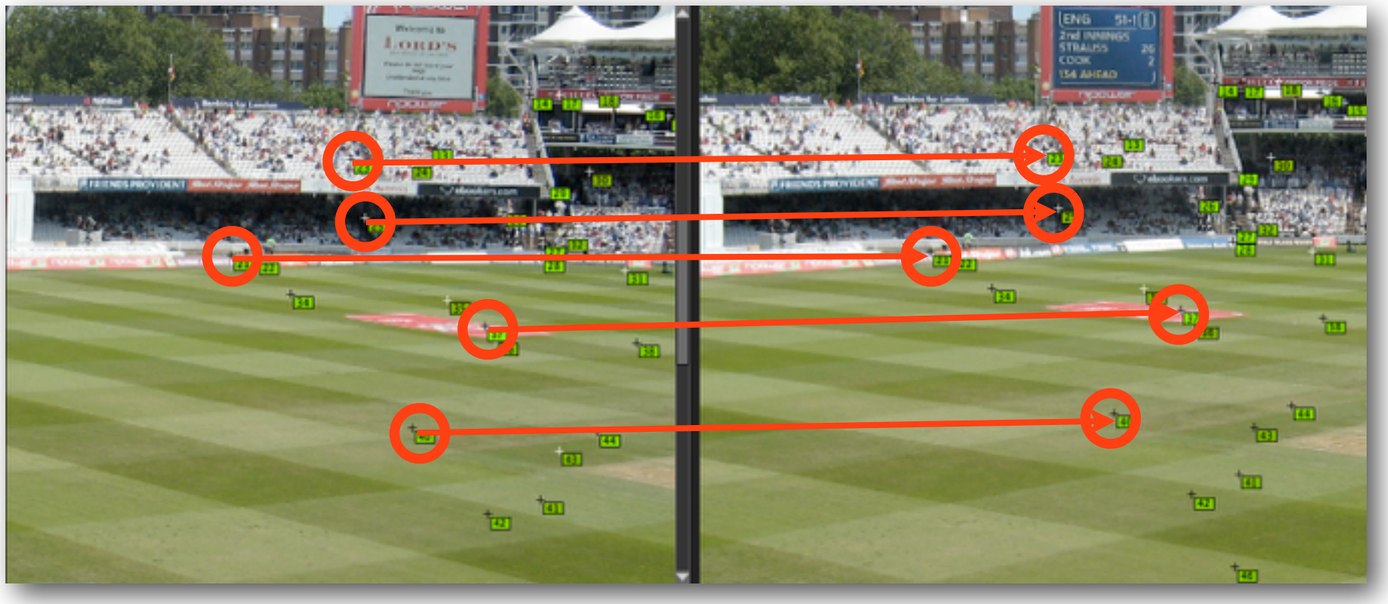
$$G = h \otimes F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i+u, j+v]$$

Convolution:

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

* Filters can be applied to process images
* When $\mathbf{h}$ is symmetric, which one to use?

Using Filters to Find Features



Extract higher-level "features"

* Map raw pixels to an intermediate representation
* Reduce amount of data, preserve useful information

Good Features to Match between Images

Features

* Parts of an image that encode it in a compact form

Edges

* Edges in an image??!!
* Information theory view: Edges encode change, therefore edges efficiently encode an image

Slide adapted from Aaron Bobick

What kind of discontinuities are in a scene?

surface normal

depth

surface color

illumination

Good Features to Match between Images

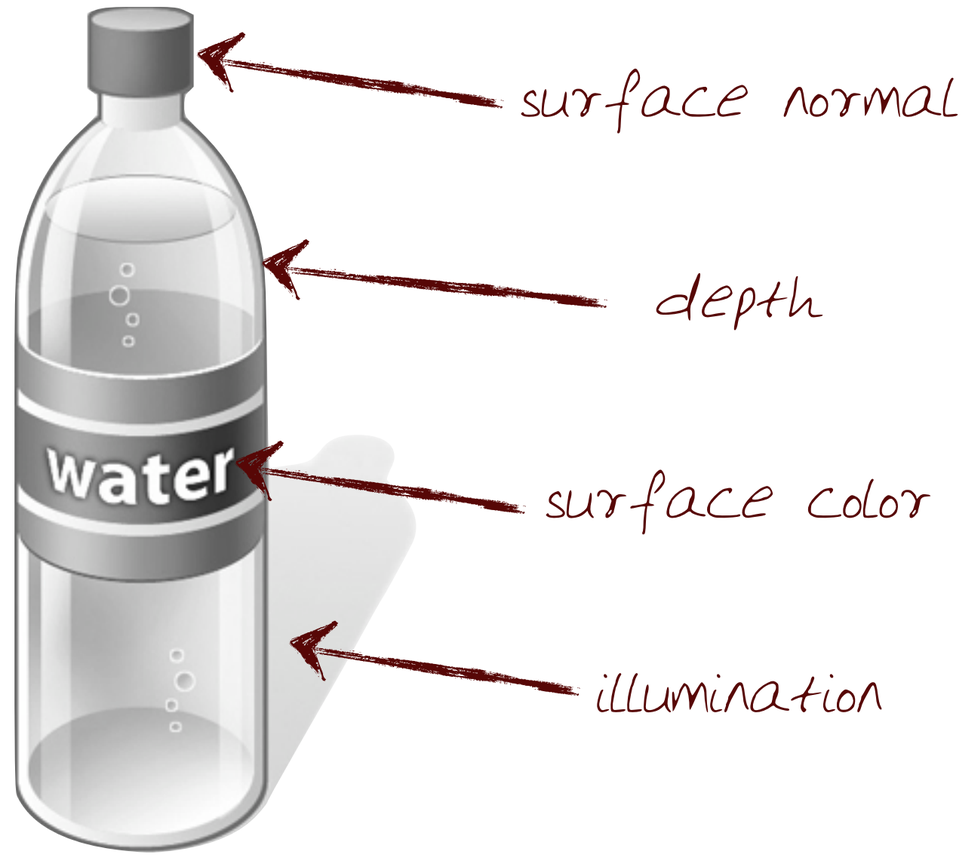
Features

* Parts of an image that encode it in a compact form

Edges

* Edges in an image??!!
* Information theory view: Edges encode change, therefore edges efficiently encode an image

What kind of discontinuities are in a scene?



Slide adapted from Aaron Bobick

Edges in Real Images

depth discontinuity

object boundary

surface color/shape discontinuity

reflectance change

illumination discontinuity (shadows)

surface normal discontinuity

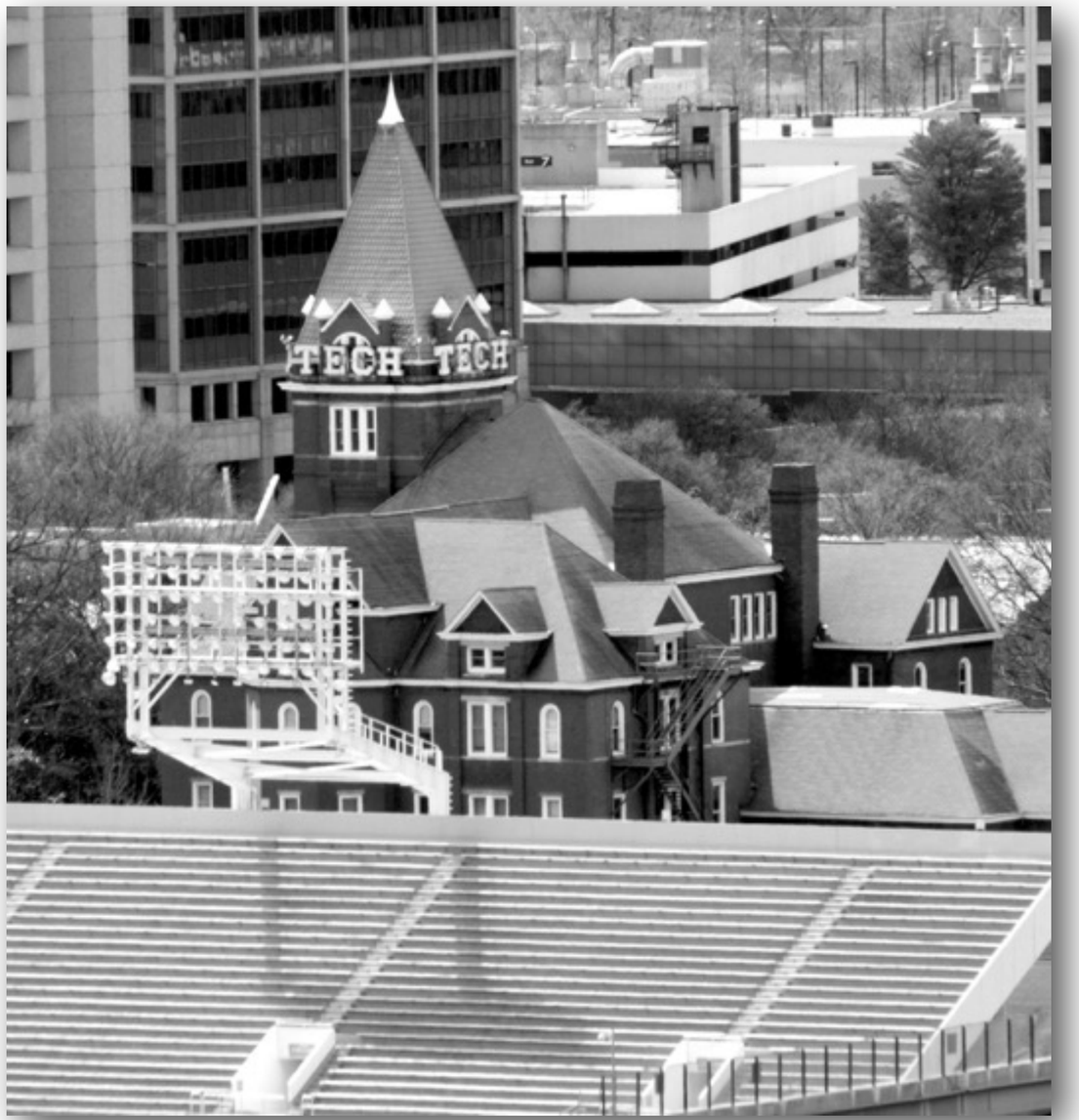
Image Courtesy Professor Henrik Christensen

Edges in Real Images

depth discontinuity

reflectance change

surface normal discontinuity



object boundary

surface color/shape discontinuity

illumination discontinuity (shadows)

Image Courtesy Professor Henrik Christensen

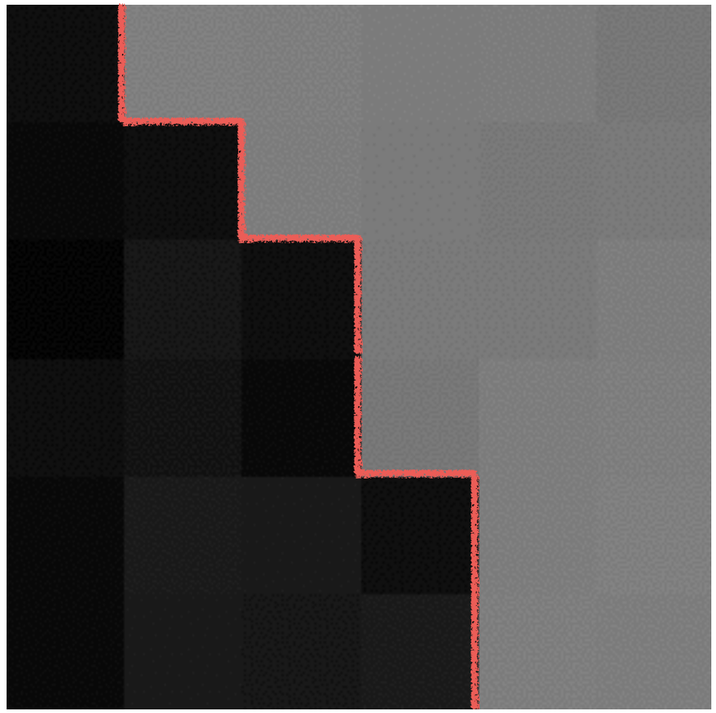
Recall: Images as Functions: F(x,y)

Edges appear as ridges in the 3D height map of an image

Edge Detection

Basic Idea:

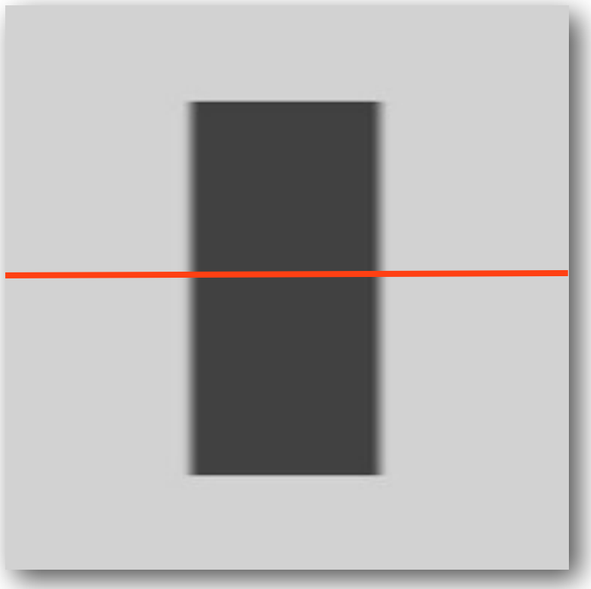
* Look for a neighborhood with strong signs of change



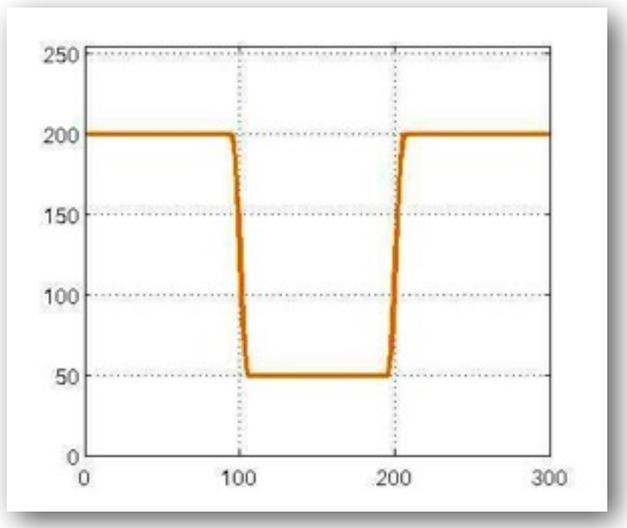
Issues to consider:

* Size of the neighborhood?
* What metrics represent a "change" ?

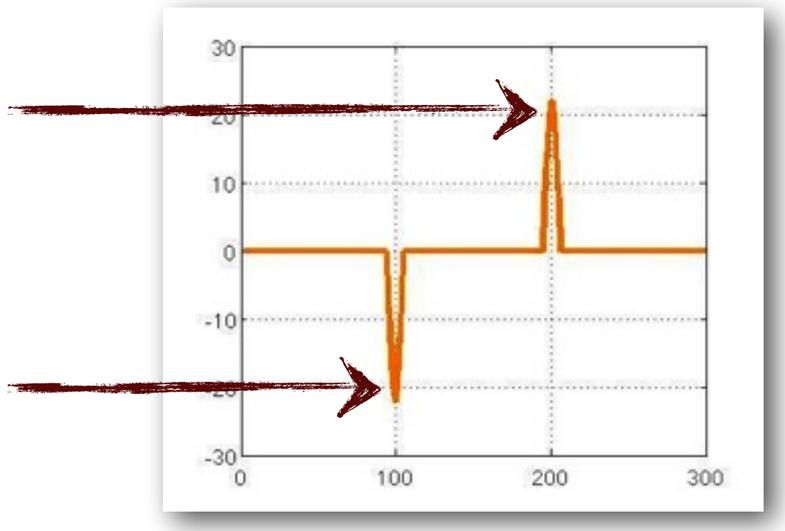
Derivatives of F(x,y) to get Edges



Test Image



Intensity along horizontal scan line (in red)

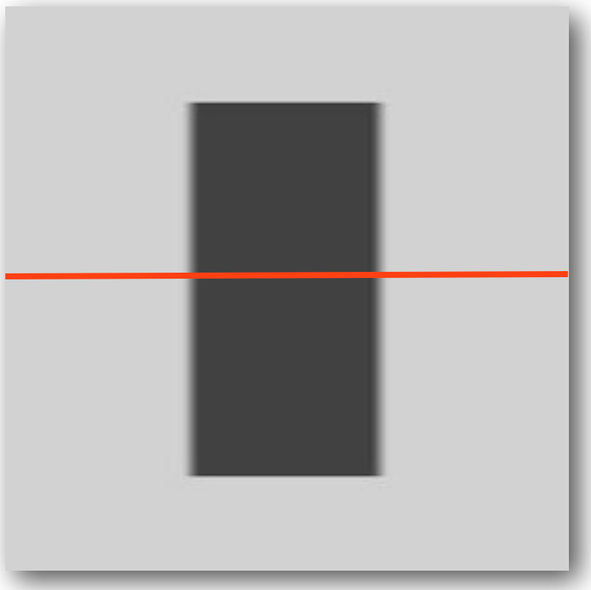


First Derivative

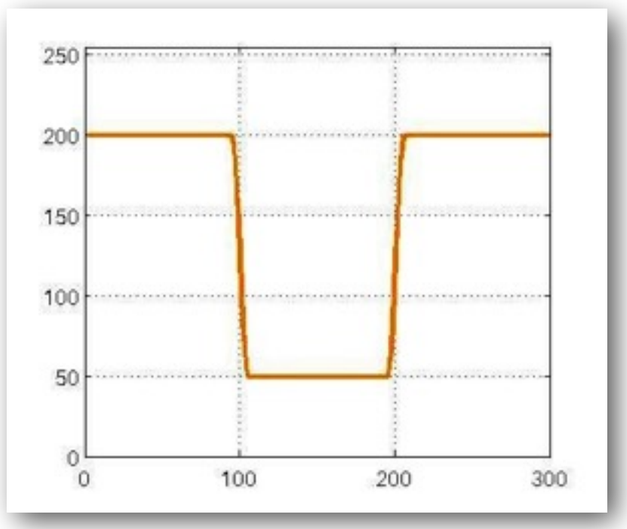
* An edge is where there is rapid change in the image intensity function
* Extrema indicate vertical edges (here we just looked for derivative in x)

Slide adapted from Aaron Bobick and Svetlana Lazebnik

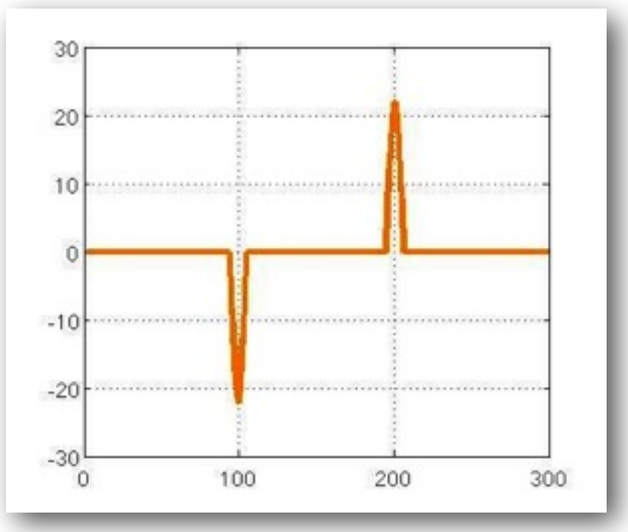
Derivatives of F(x,y) to get Edges



Test Image



Intensity along
horizontal scan line (in red)



First Derivative

* An edge is where there is rapid change in the image intensity function
* Extrema indicate vertical edges (here we just looked for derivative in x)

Slide adapted from Aaron Bobick and Svetlana Lazebnik

Differential Operators for Images

Need an operation that when applied to an image returns its derivatives.

a	b	c
d	e	f
g	h	i

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

* Model these "operators" as masks/ kernels.
  When applied, yields a new function that is the image gradient.
* Then "threshold" this gradient function to select edge pixels.

Image Gradient

Need to define "gradient" !

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

Gradient of an image =

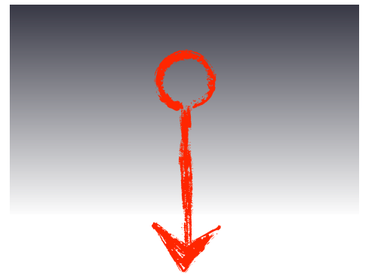
Measure of change in Image function, $\mathbf{F(x,y)}$ in $\mathbf{x}$ (across / columns) and $\mathbf{y}$ (down / rows)

Definition: Image Gradient (Mathematically)

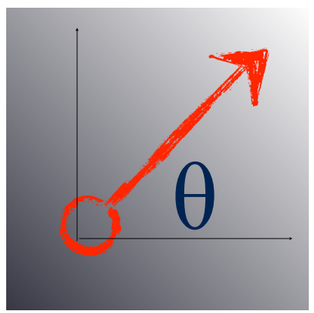
$$\nabla F = \left[\frac{\delta F}{\delta x}, \frac{\delta F}{\delta y}\right]$$

measure of change in Image function (**F**), in **x** and **y**

$$\nabla F = \left[\frac{\delta F}{\delta x}, 0\right]$$



$$\nabla F = \left[0, \frac{\delta F}{\delta y}\right]$$



$$\nabla F = \left[\frac{\delta F}{\delta x}, \frac{\delta F}{\delta y}\right]$$

Note: Gradient points in the direction of most rapid increase in intensity (θ)

Slide adapted from Aaron Bobick

Definition: Image Gradient (Mathematically)

$$\nabla F = \left[\frac{\delta F}{\delta x}, \frac{\delta F}{\delta y}\right]$$

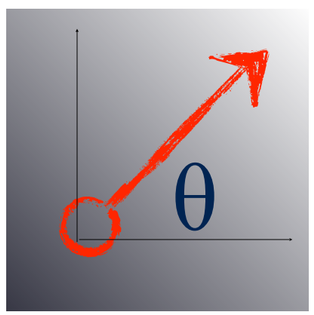
measure of change in Image function (**F**), in **x** and **y**

Gradient Direction is:

$$\theta = \tan^{-1}\left[\frac{\delta F}{\delta y} / \frac{\delta F}{\delta x}\right]$$

Gradient magnitude provides edge strength:

$$\| \nabla F \| = \sqrt{\left(\frac{\delta F}{\delta x}\right)^2 + \left(\frac{\delta F}{\delta y}\right)^2}$$



Q: How does this relate to edge direction?

Slide adapted from Aaron Bobick

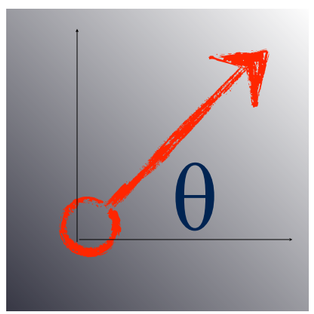
Definition: Image Gradient (Discrete)

For 2D function, F(x,y), the partial derivative is:

$$\frac{\delta F(x,y)}{\delta x} = \lim_{\epsilon \to 0} \frac{F(x+\epsilon, y) - F(x,y)}{\epsilon}$$

For discrete data, we can approximate using finite differences:

$$X: \quad \frac{\delta F(x,y)}{\delta x} \approx \frac{F(x+1,y) - F(x,y)}{1}$$



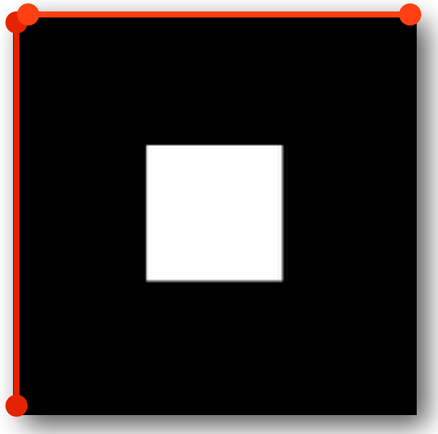
$$Y: \quad \frac{\delta F(x,y)}{\delta y} \approx F(x,y+1) - F(x,y)$$

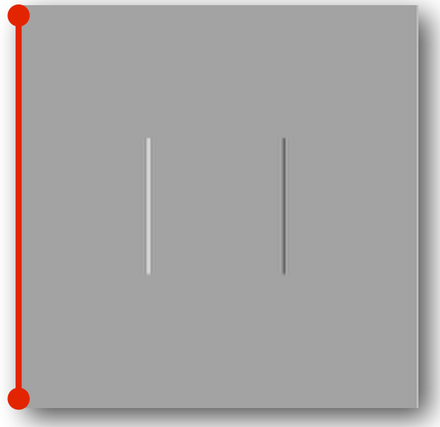
Slide adapted from Aaron Bobick

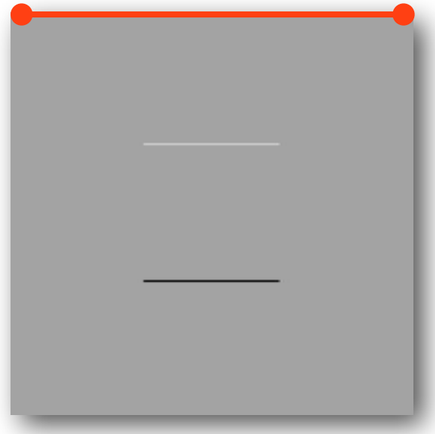
Differentiating an Image in X and Y

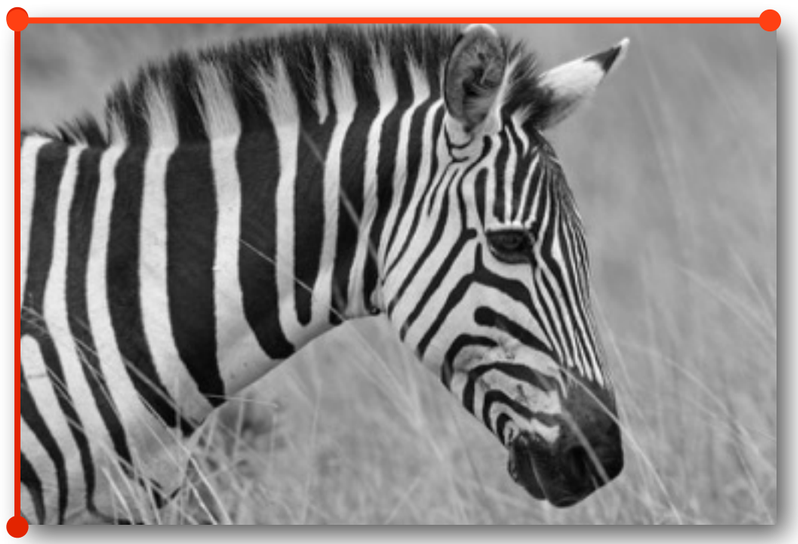
$$\frac{\delta F(x,y)}{\delta x} \approx F(x+1,y) - F(x,y)$$

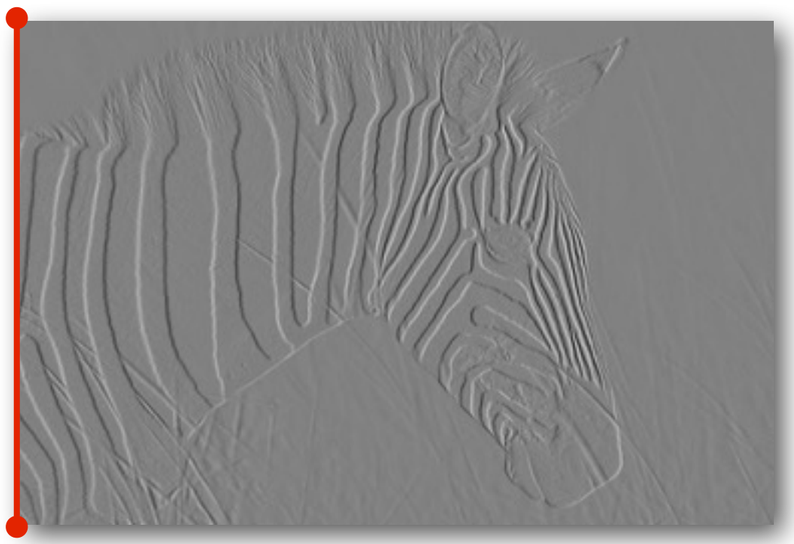
$$\frac{\delta F(x,y)}{\delta y} \approx F(x,y+1) - F(x,y)$$

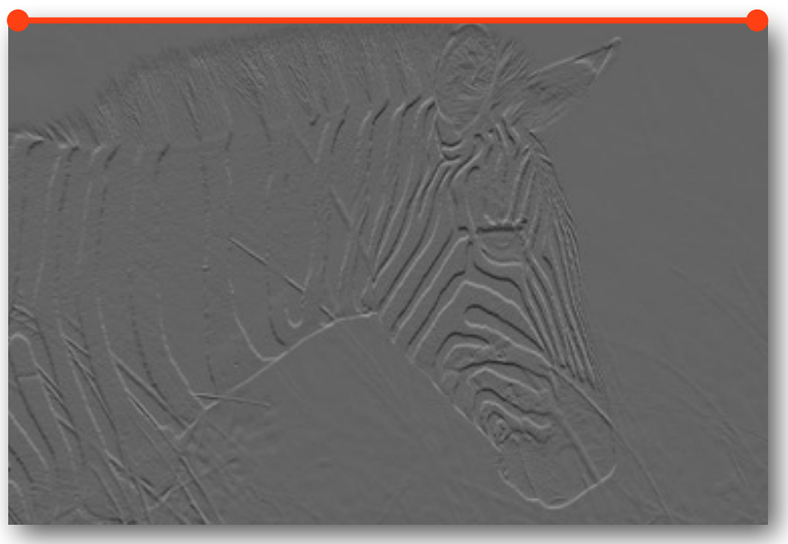




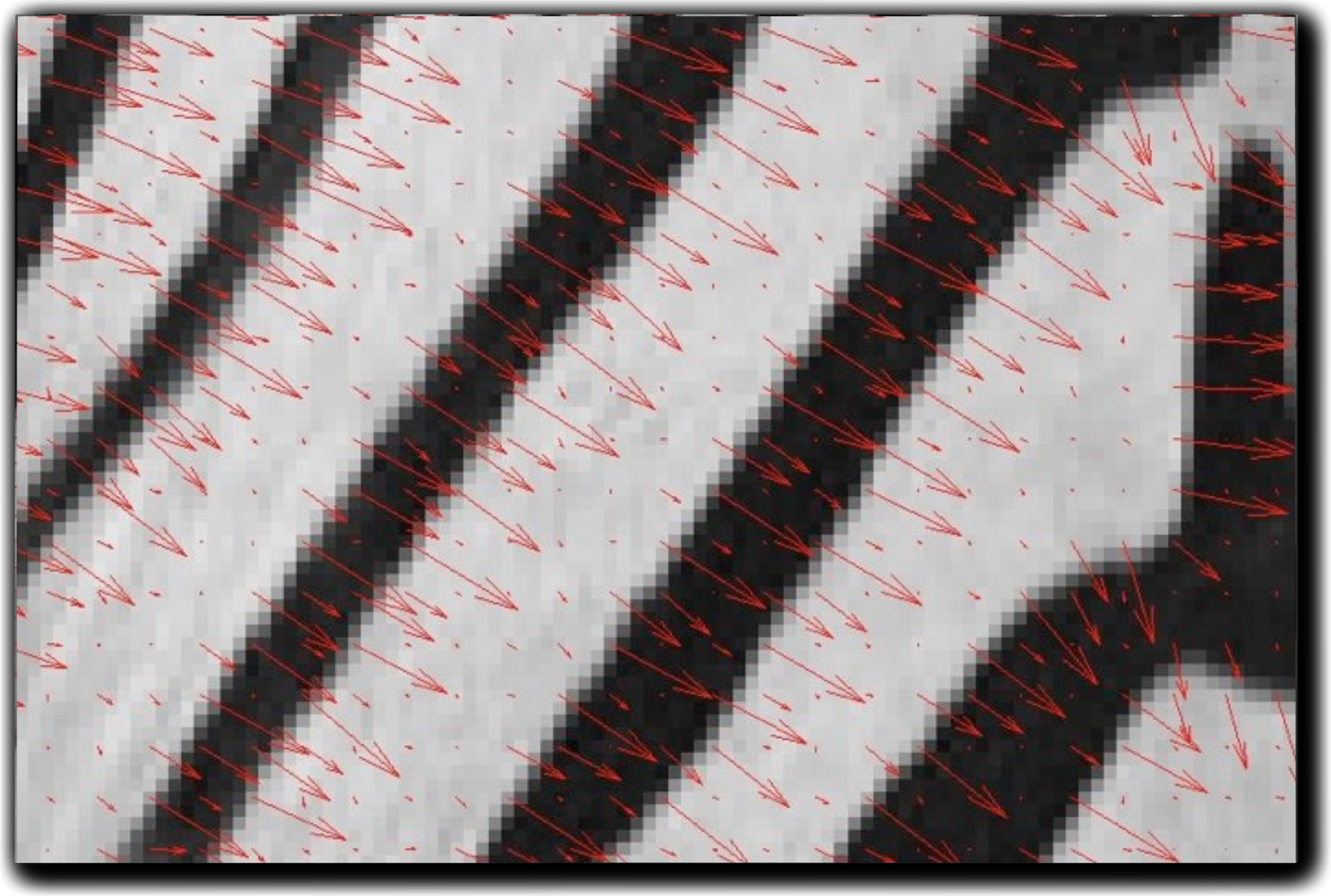








Gradient Images



$$\| \nabla F \| = \sqrt{\left(\frac{\delta F}{\delta x}\right)^2 + \left(\frac{\delta F}{\delta y}\right)^2}$$

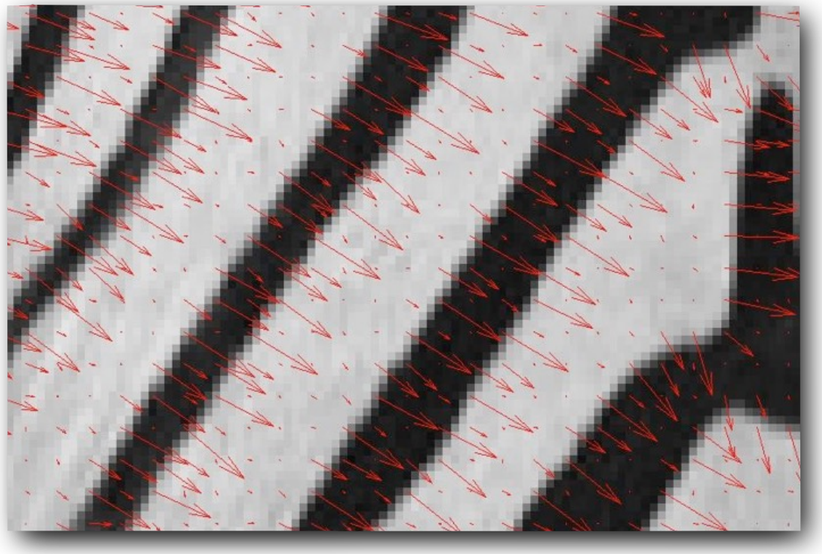
$$\theta = \tan^{-1}\left[\frac{\delta F}{\delta y} / \frac{\delta F}{\delta x}\right]$$

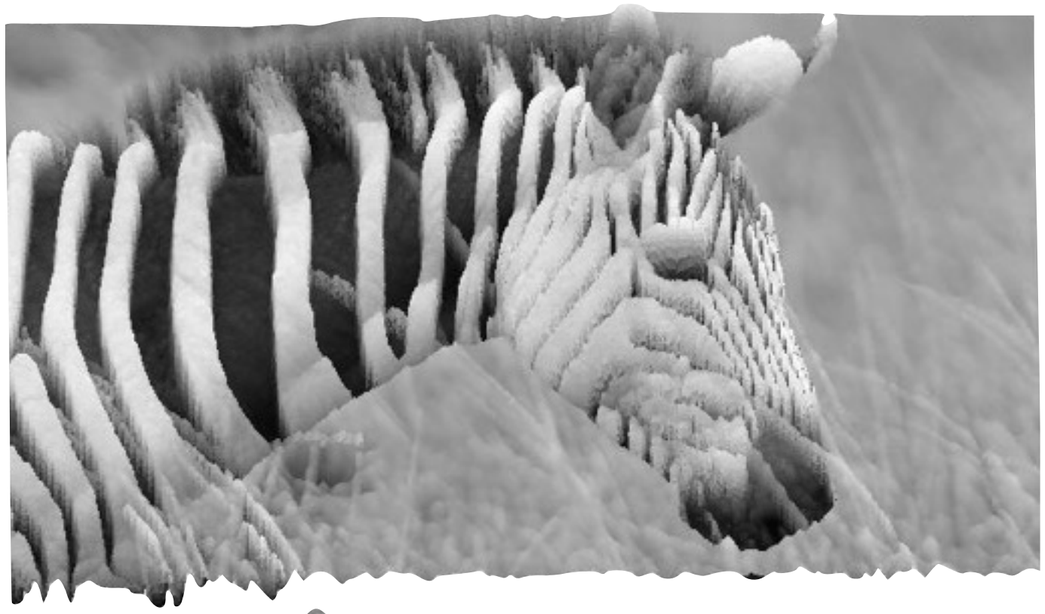
Visualizing Gradients

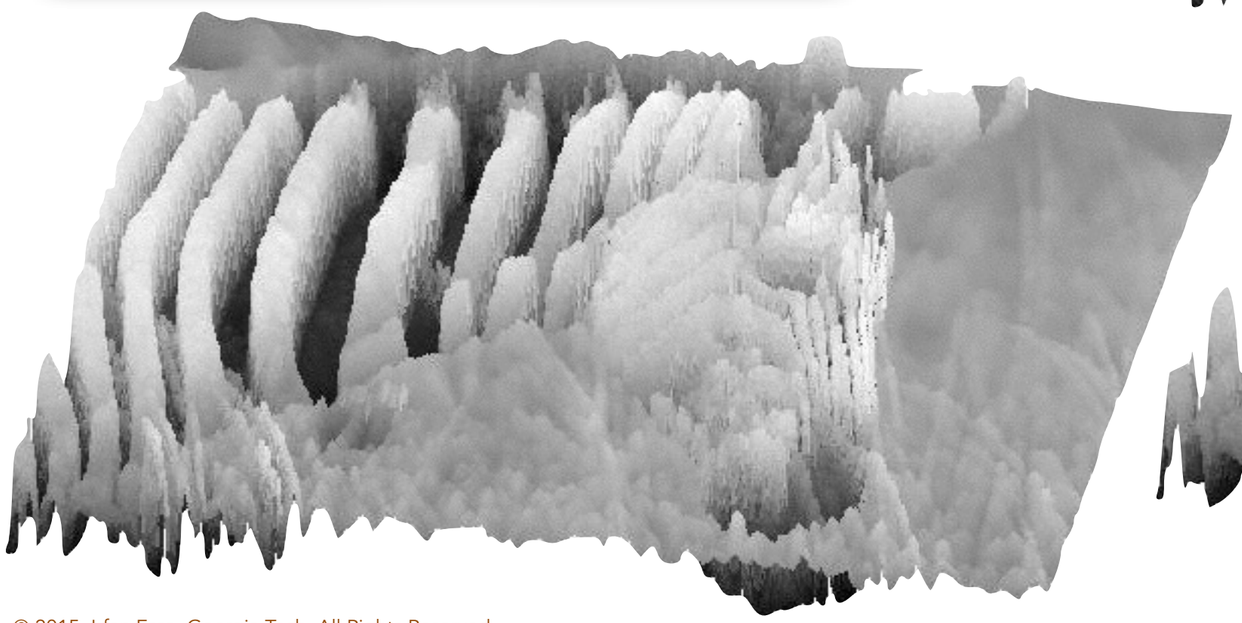
$$\| \nabla F \| = \sqrt{\left(\frac{\delta F}{\delta x}\right)^2 + \left(\frac{\delta F}{\delta y}\right)^2}$$

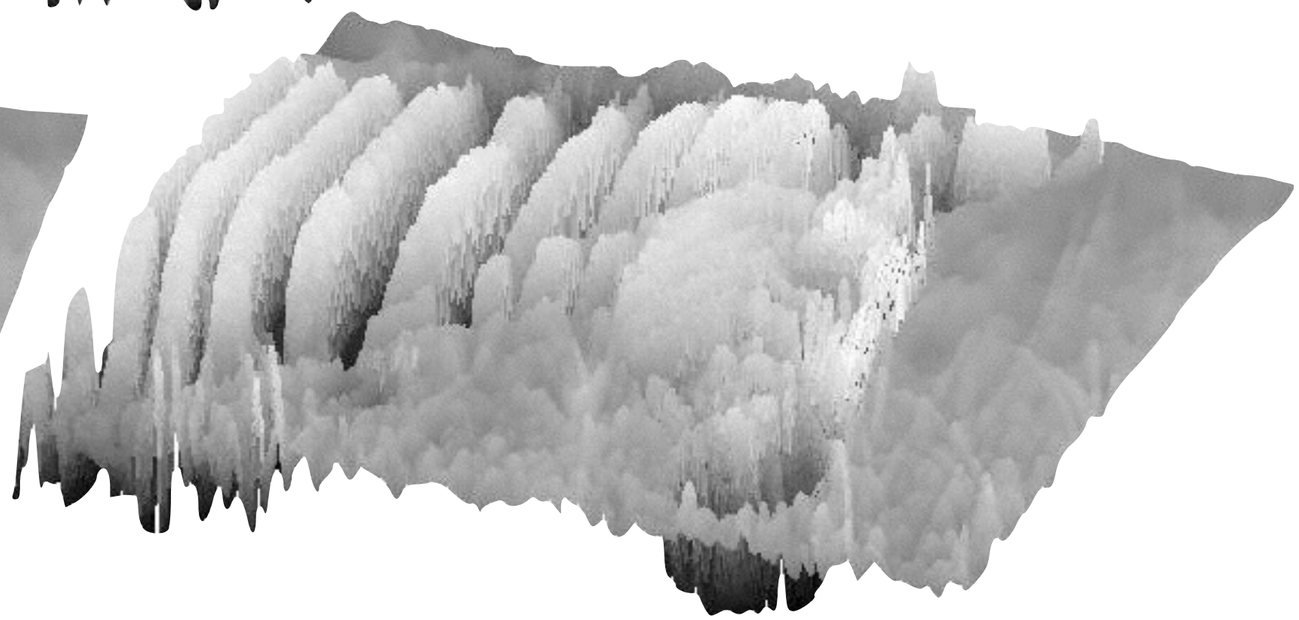
$$\theta = \tan^{-1}\left[\frac{\delta F}{\delta y} / \frac{\delta F}{\delta x}\right]$$

Visualizing Gradients









Summary

* Detecting features for matching
* Compute Edges by measuring changes across an image
* Use of derivatives to compute the gradient over an image

Next Class

Image Analysis: Edge Detection using Gradients

Credits

* Matlab software by Mathworks INC.
* Some Slides adapted from Aaron Bobick
* For more information, see Szeliski OR Forsyth & Ponce Text Book.
* Images
  * Image Courtesy Professor Henrik Christensen
  * Images used from USC's Signal and Image Processing Institute's Image Database
  * Zebra Image by http://www.flickr.com/photos/lipkee/2904603582/

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

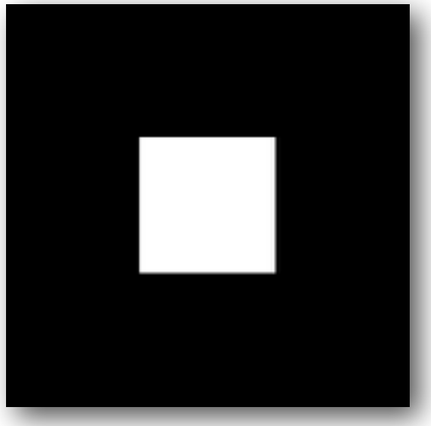
Image Processing and Filtering, via Convolution and Cross-Correlation

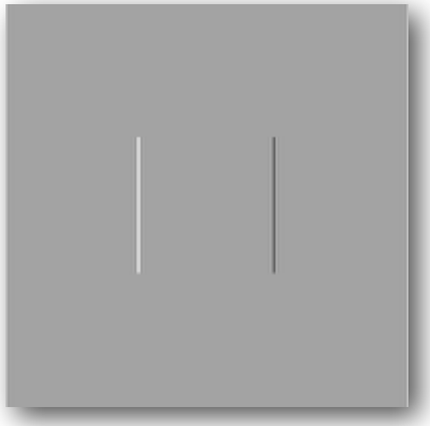
* Towards Edge Detection: Computing Image Gradients

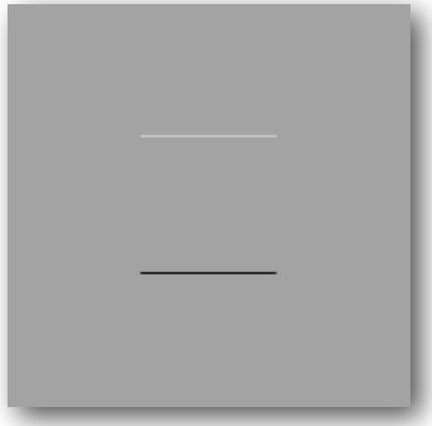
1. Compute Edges
2. Derivatives using kernels and neighborhood operations
3. Three (3) methods for computing edges using kernels
4. Image noise can complicate the computation of gradients
5. The Canny Edge Detector

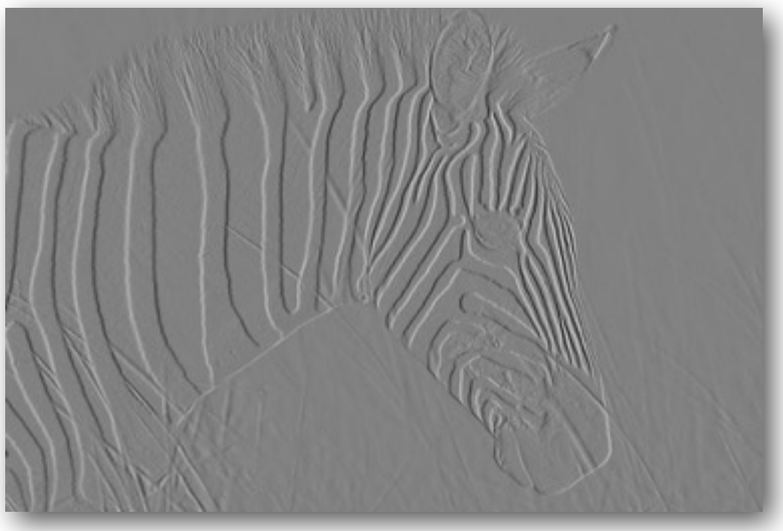
Recall: Differentiating an Image in X and Y

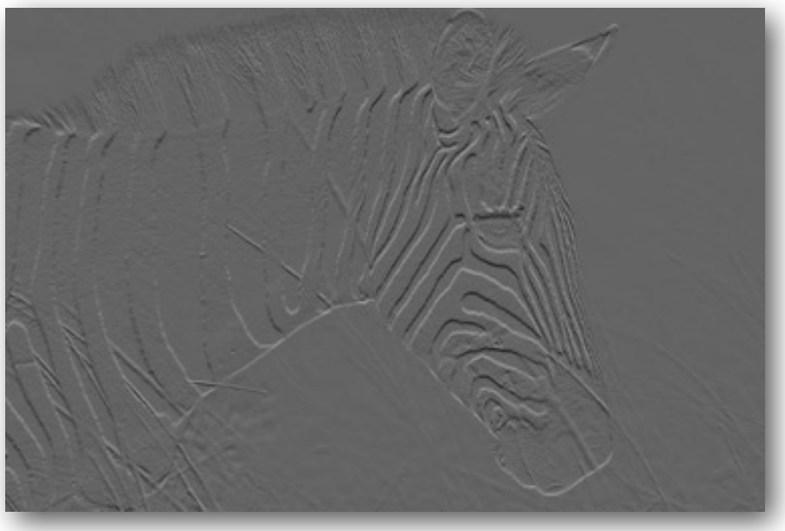
$$\frac{\delta F(x,y)}{\delta x} \approx F(x+1,y) - F(x,y) \qquad \frac{\delta F(x,y)}{\delta y} \approx F(x,y+1) - F(x,y)$$











Derivative as a local product

$$\frac{\delta F(x,y)}{\delta x} = F(x+1,y) - F(x,y)$$

$$= (1) \times F(x+1,y) + (-1) \times F(x,y)$$

(rearranged)

$$= (-1) \times F(x,y) + (1) \times F(x+1,y)$$

(dot product)

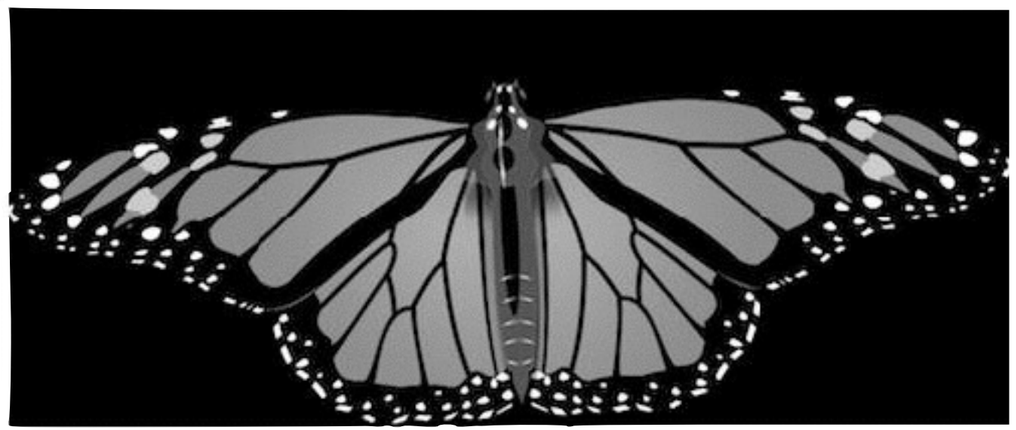
$$= \begin{bmatrix} -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} F(x,y) & F(x+1,y) \end{bmatrix}^T$$

Derivative using Cross-Correlation

$$\frac{\delta F(x,y)}{\delta x}$$

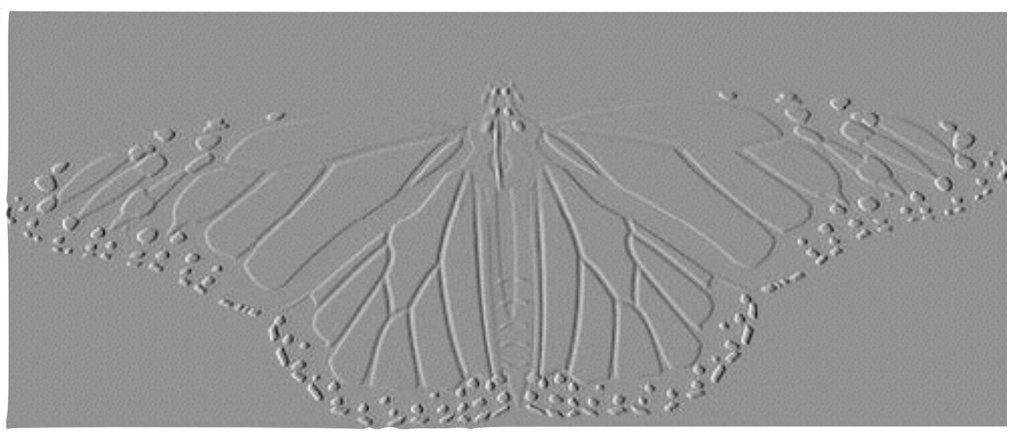
-1	1

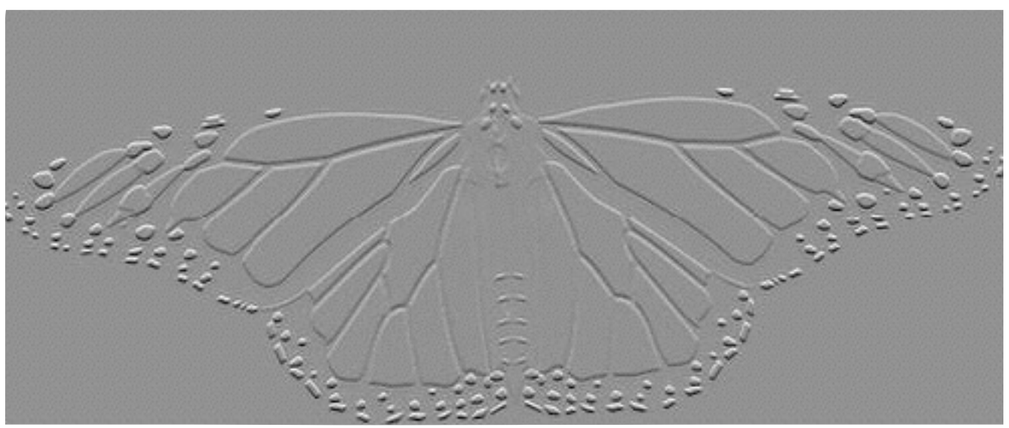
Cross-Correlate with Kernel

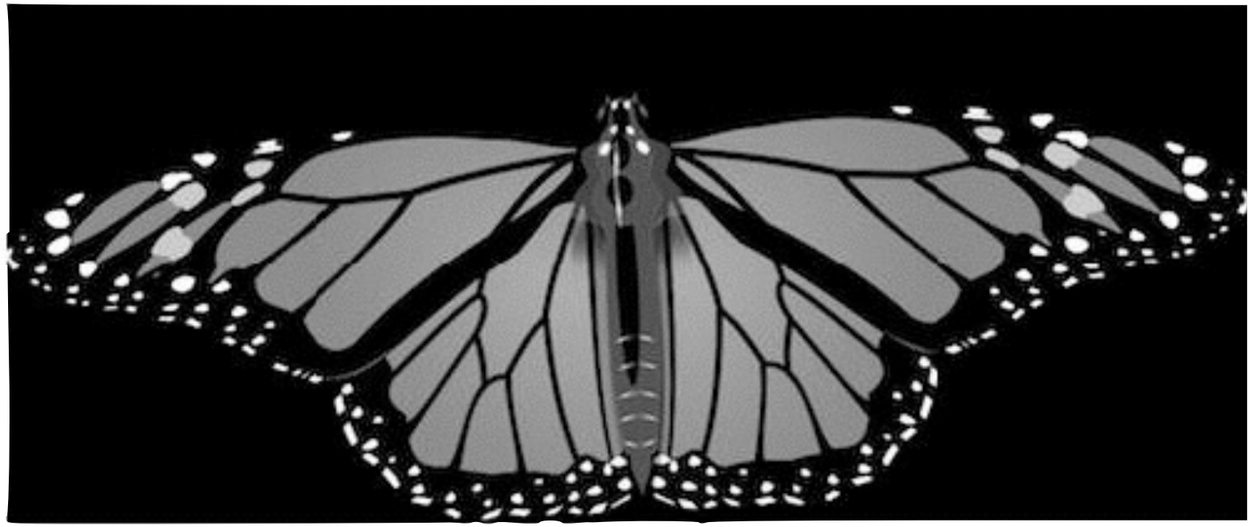


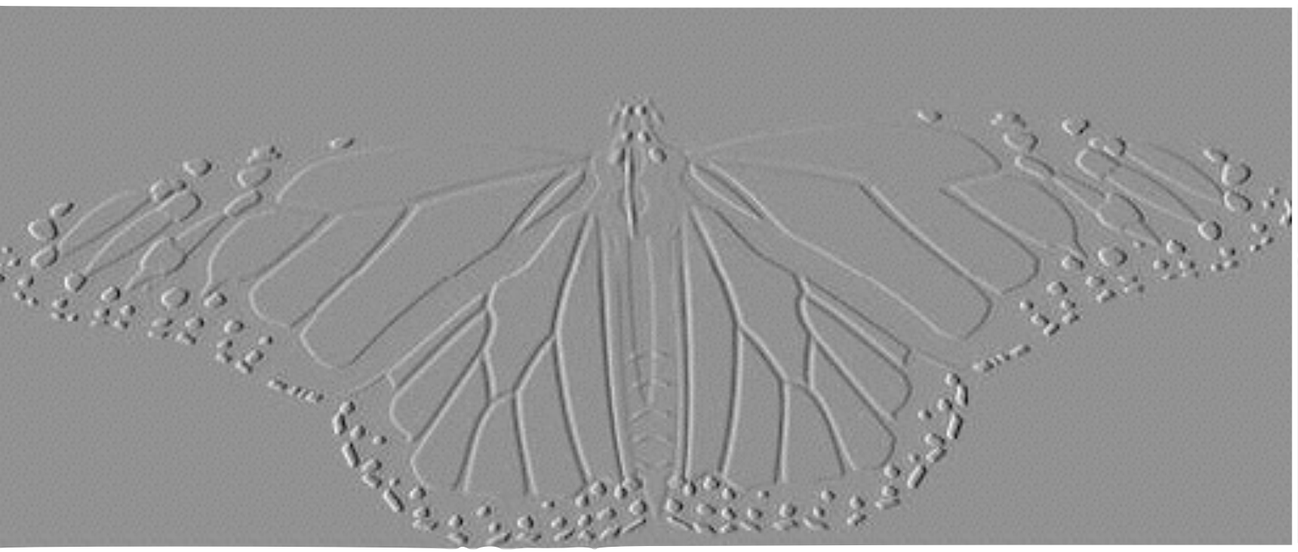
$$\frac{\delta F(x,y)}{\delta y}$$

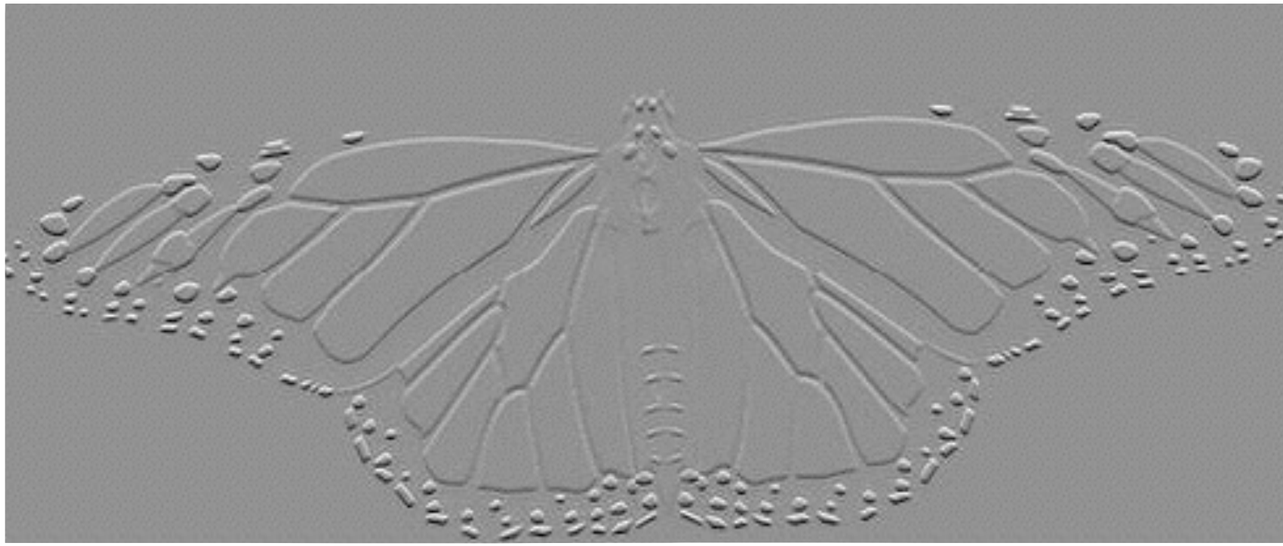
-1
1











Computing discrete gradients

Desired: An "operator" (mask/kernel) that effectively computes discrete derivative values with cross-correlation (i.e. using Finite Differences*)

		a	b	c				
		d	e	f				
		g	h	i				
20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

*Finite Differences provide a numerical solution for differential equations using approximation of derivatives

Computing discrete gradients

Gradient in X and Y directions:

$$\frac{\delta F(x,y)}{\delta x} \approx F(x+1,y) - F(x,y)$$

$$\frac{\delta F(x,y)}{\delta y} \approx F(x,y+1) - F(x,y)$$

Equivalent kernels:

H_x

0	0
-1	1
0	0

H_y

0	-1	0
0	1	0

(transposed kernel for Y)

Ideally, a kernel should be symmetric about an image point

Where is the "middle" point in these kernels?

Computing discrete gradients

H_x

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

Average of "left" and "right" derivatives

Is this a better kernel?

Similarly, for Y:

H_y

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

Various Kernels for Computing Gradients

Prewitt

H_x

-1	0	1
-1	0	1
-1	0	1

H_y

-1	-1	-1
0	0	0
1	1	1

Sobel

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

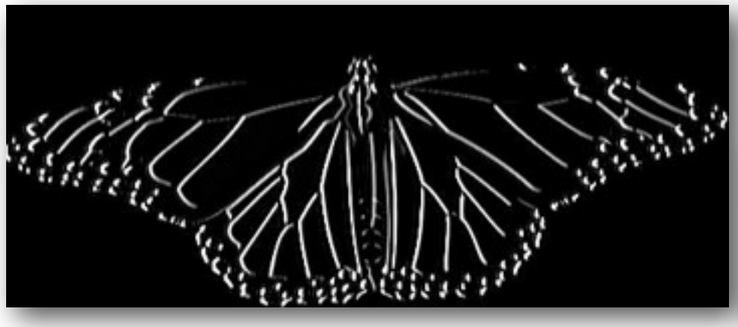
-1	-2	-1
0	0	0
1	2	1

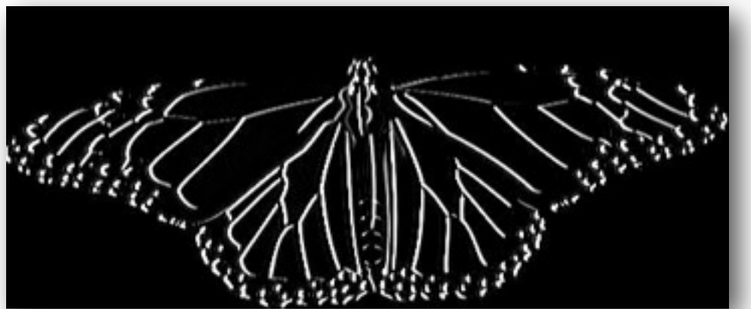
Roberts

0	1
-1	0

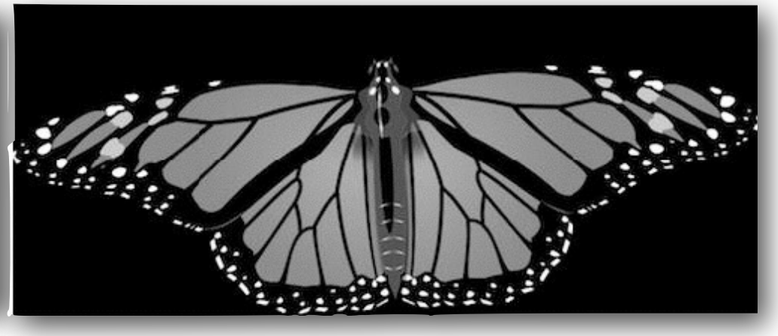
1	0
0	-1

$H_x \otimes F$





F



Input Image

Slide adapted from Aaron Bobick

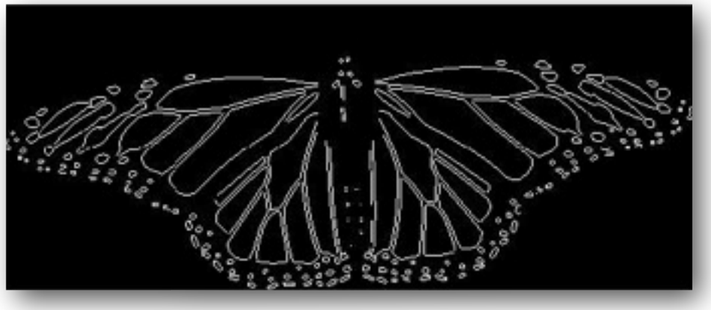
Various Kernels for Computing Gradients

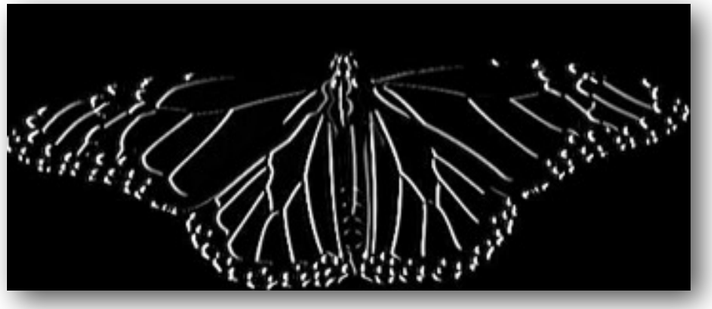
Edges

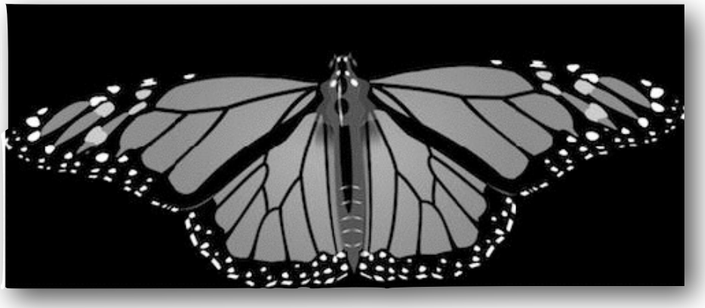
$H_x \otimes F$

F

Prewitt

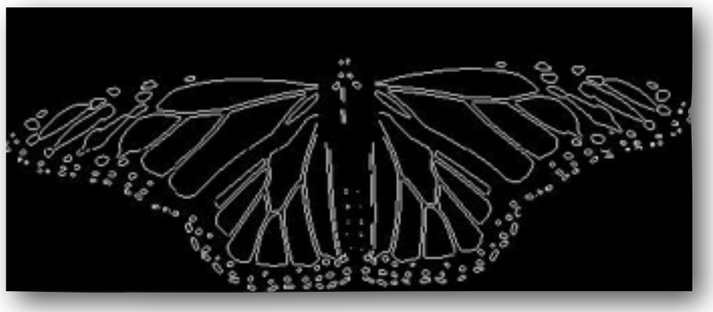


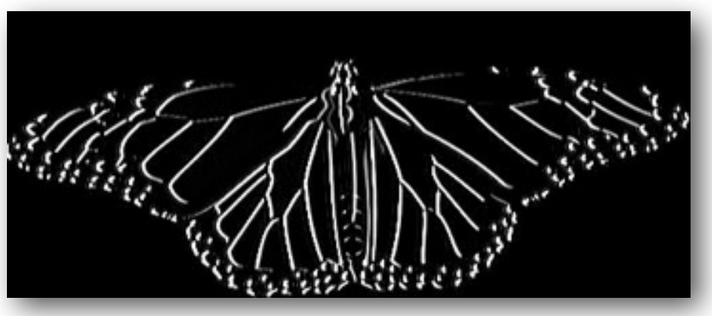




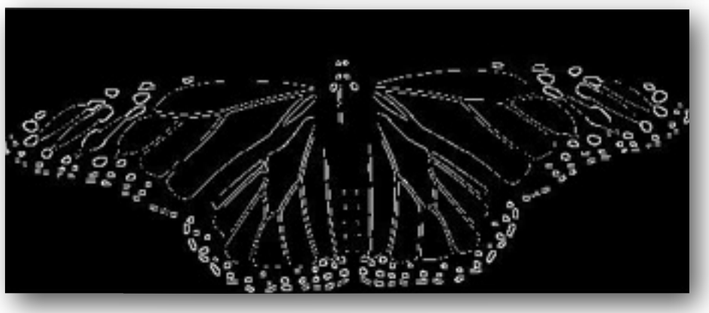
Input Image

Sobel



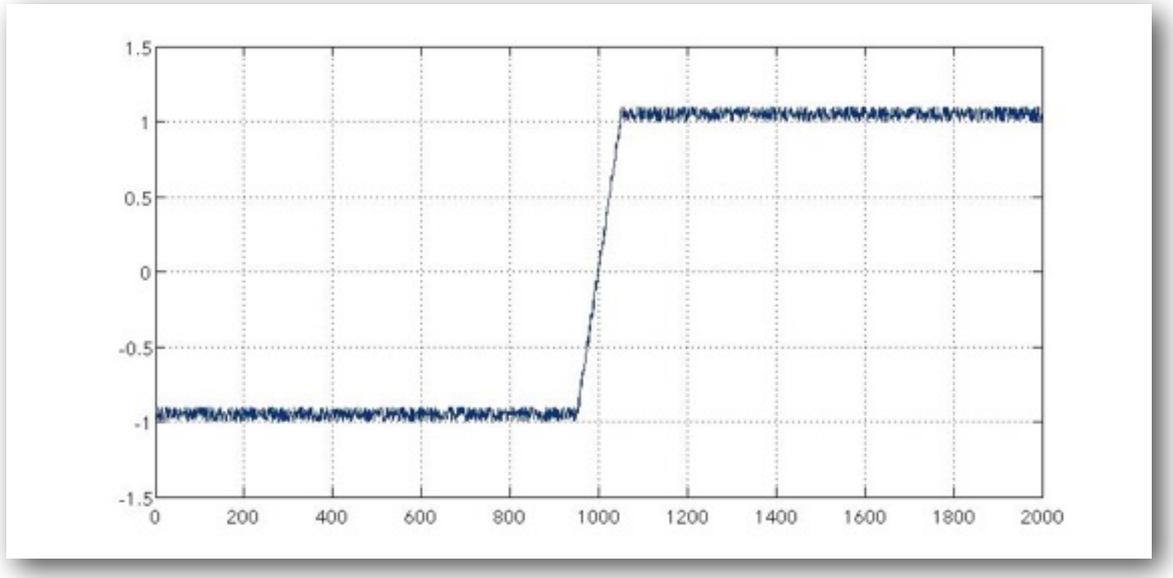


Roberts

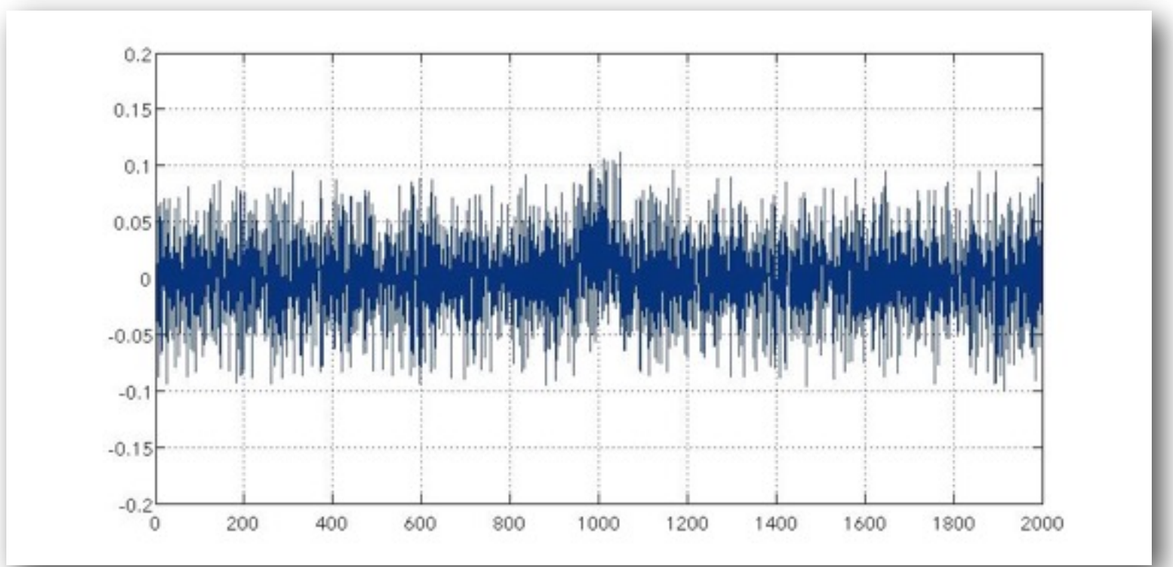


Impact of Noise on Gradients (1-D Example)

$f(x)$

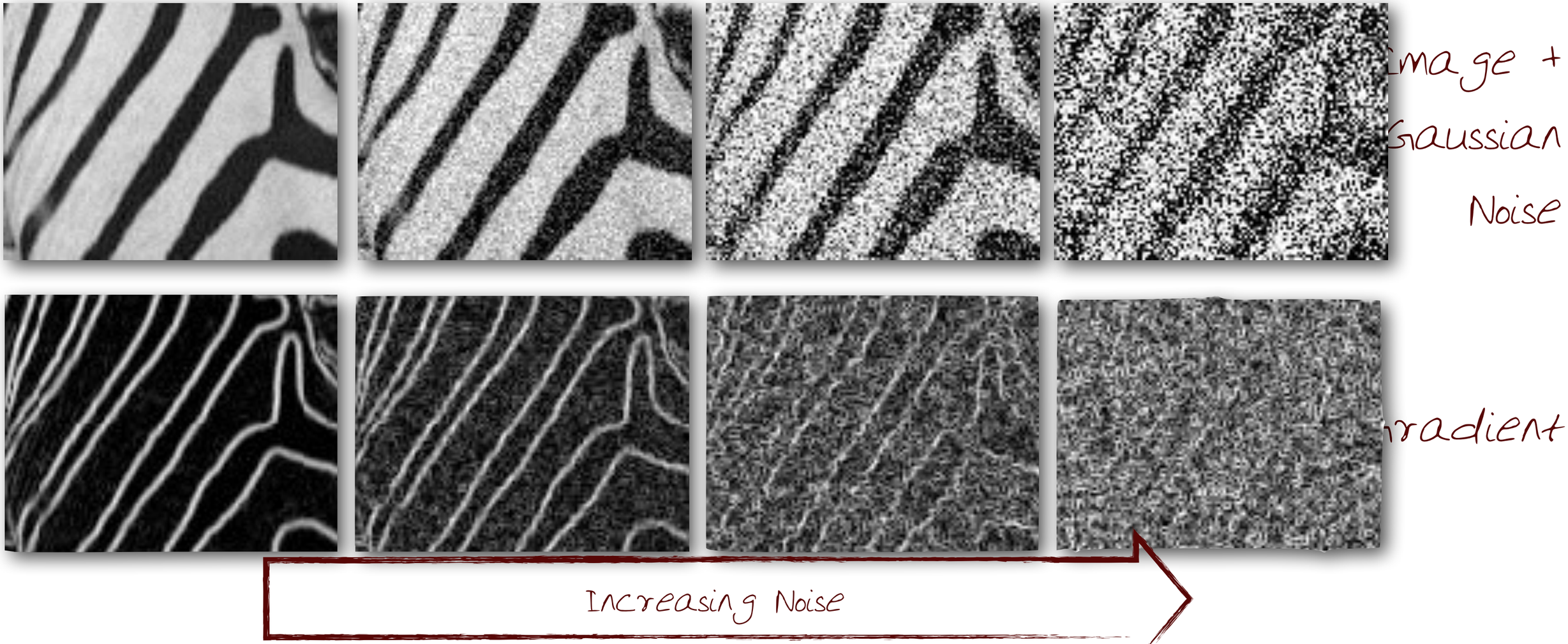


$$\frac{\delta f(x)}{\delta x}$$



It gets harder to detect an edge when there is significant noise in the signal.

Impact of Noise on Gradients



Smooth the Image, before applying the Gradient Operations.

Convolution and Gradients

Recall, Convolution is: $G = h * F$

Derivative of a Convolution: $\frac{\delta G}{\delta x} = \frac{\delta}{\delta x}(h * F)$

If $\boldsymbol{D}$ is a kernel to compute derivatives and $\boldsymbol{H}$ is the kernel for smoothing...

We could define kernels with derivative and smoothing in one: $D * (H * F) = (D * H) * F$

Gradient to Edges

Image

Gradient Image

Edge Image

1. Smoothing: suppress noise
2. Compute Gradient
3. Apply Edge "enhancement": filter for contrast
4. Edge localization: Determine which local maxima from filter output are actually edges vs. noise
5. Threshold, Thin

Gradient to Edges

Image

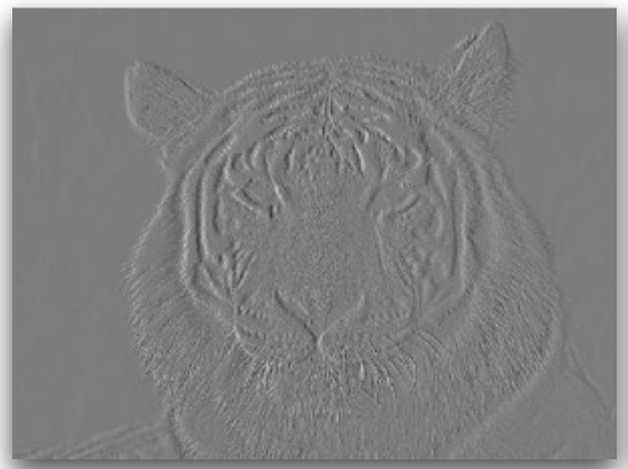
Gradient Image

Edge Image

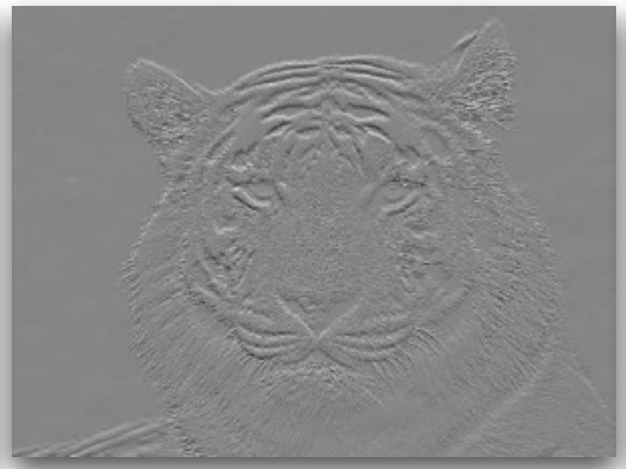
1. Smoothing: suppress noise
2. Compute Gradient
3. Apply Edge "enhancement":
4. Edge localization:
   1. Edges vs. noise
5. Threshold, Thin

Canny Edge Detector (1)

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient



dx



dy

mag of gradient

theta

Note: All images here just for demonstration

Canny Edge Detector (3)

3. Non-maximum suppression:

   1. Thin multi-pixel wide "ridges" down to single pixel width

mag of gradient

Note: All images here just for demonstration

Canny Edge Detector (4)

4. Linking and thresholding (hysteresis):

   1. Define two thresholds: low and high

   2. Use the high threshold to start edge curves and the low threshold to continue them

Note: All images are actual output

mag of gradient

Canny Edge Detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
    1. Thin multi-pixel wide "ridges" down to single pixel width
4. Linking and thresholding (hysteresis):
    1. Define two thresholds: low and high
    2. Use the high threshold to start edge curves and the low threshold to continue them

Slide adapted from Aaron Bobick

Summary

* Brought together the concepts of Convolution and Correlation with Image Gradient computation.
* Computation of Edges
* Smoothing prior to Gradient computation.
* Four (4) different methods for Edge computation.

Next!

* Camera, Optics, Lenses, etc.
* We will return to using the concepts of Edge detection when we go back to the topic of Feature matching.

Credits

* Matlab software by Mathworks Inc.
* Some slides adapted from Aaron Bobick
* For more information, see Szeliski OR Forsyth & Ponce Text Book.
* Images
  * Images used from USC's Signal and Image Processing Institute's Image Database
  * Monarch Image by Bugboy52.40 (Own work) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], via Wikimedia Commons
  * Tiger Image by http://en.wikipedia.org/wiki/File:Siberischer_tiger_de_edit02.jpg
  * Zebra Image by http://www.flickr.com/photos/lipkee/2904603582/