Georgia Institute of Technology

Computational Photography

Dr. Irfan Essa
Professor
School of Interactive Computing

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

Georgia Institute of Technology

Image Processing and Filtering, via Convolution and Cross-Correlation

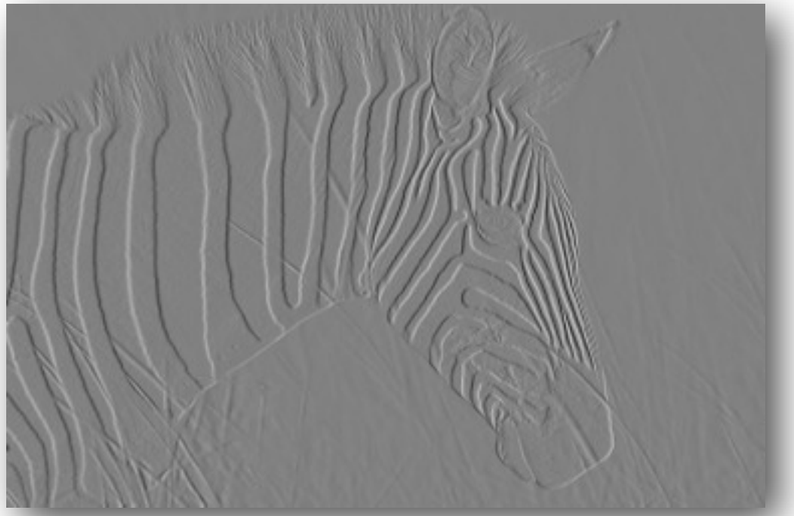
Dr. Irfan Essa
Professor
School of Interactive Computing

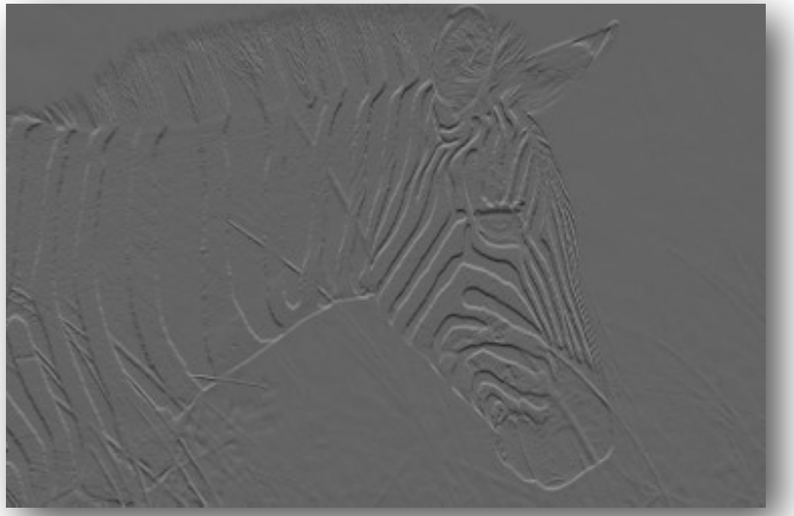
Towards Edge Detection: Computing Image Gradients

Lesson Objectives

- ★ Describe in your own terms how derivatives can be computed for an image using kernels and neighborhood operations.
- ★ Describe the three (3) methods for computing edges using kernels.
- ★ Recall why image noise can complicate the computation of gradients.
- ★ Explain the Canny Edge Detector extends the three (3) Edge detection methods.

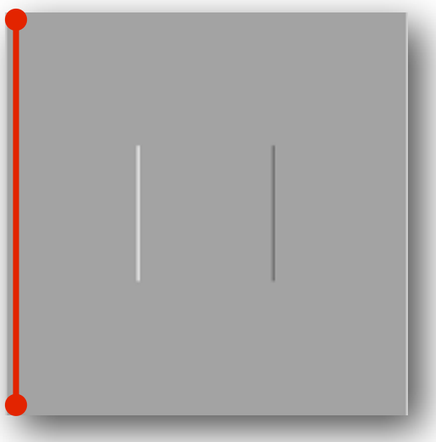
Review: Differentiating an Image in X and Y

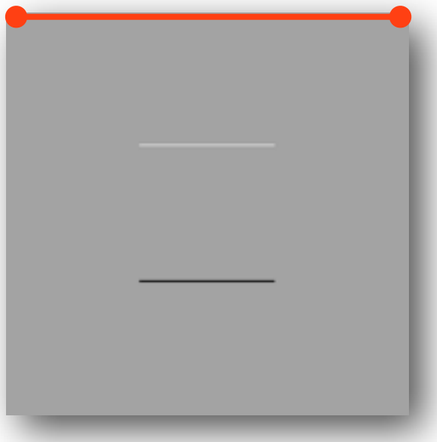


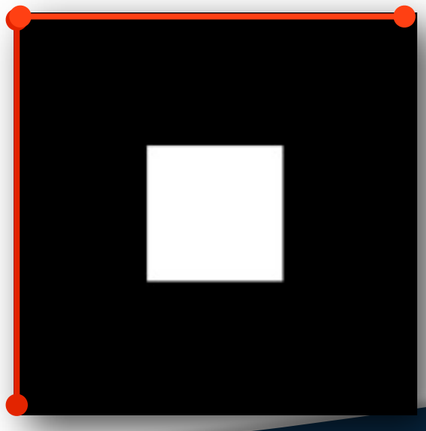


$$\frac{\delta F(x,y)}{\delta x} \approx F(x+1,y) - F(x,y)$$

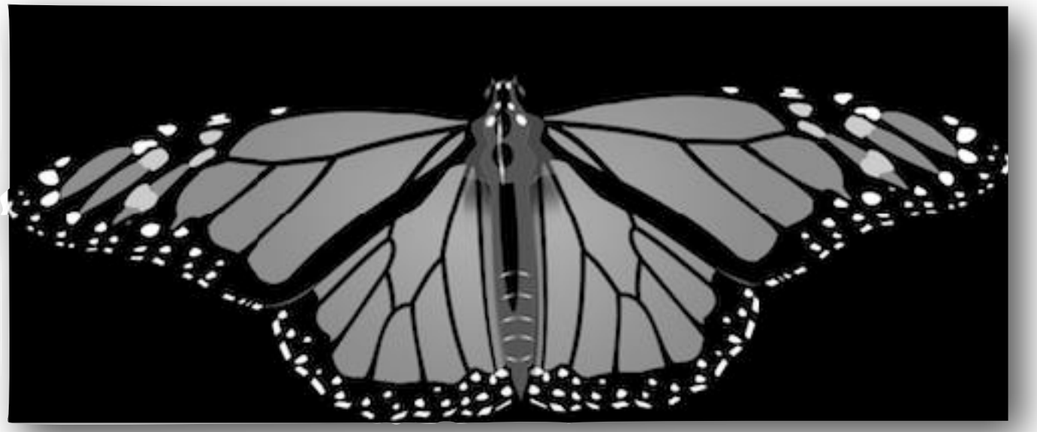
$$\frac{\delta F(x,y)}{\delta y} \approx F(x,y+1) - F(x,y)$$







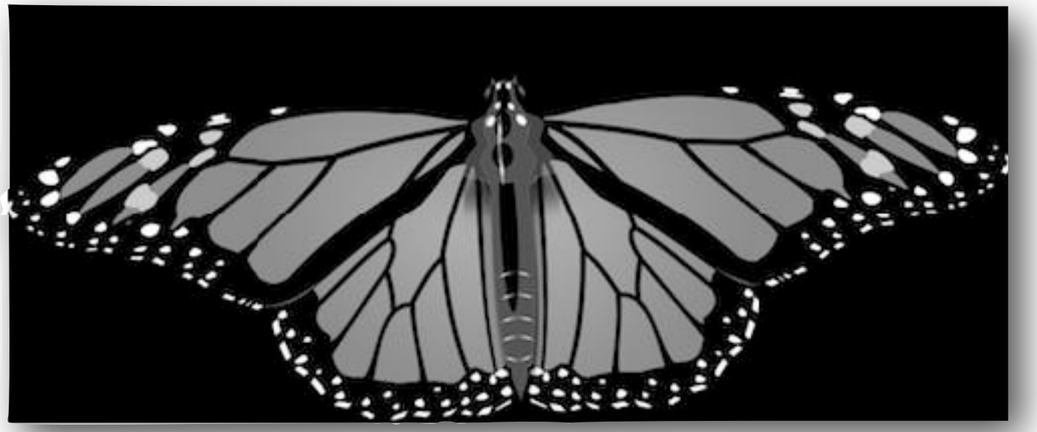
Derivatives of an Image using Correlation



Original Image

Derivatives of an Image using Correlation

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

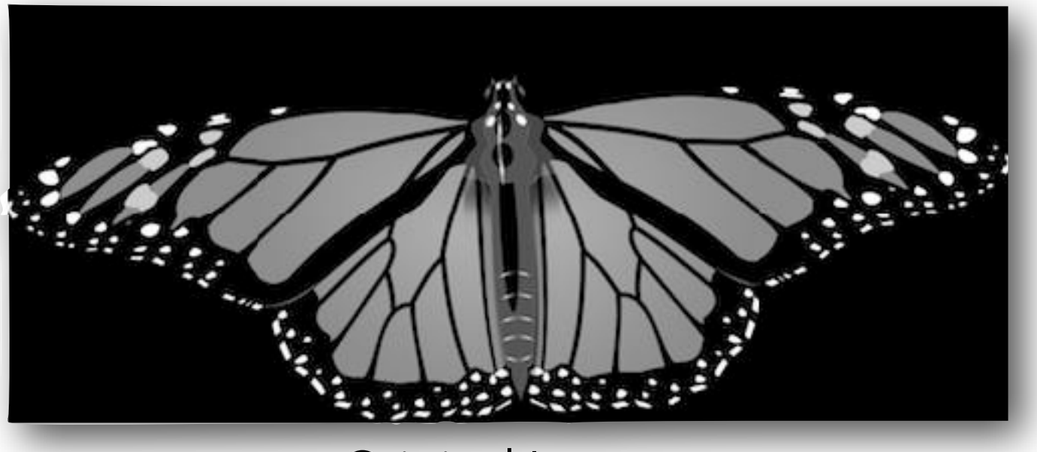


Original Image

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

Derivatives of an Image using Correlation

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



Original Image

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

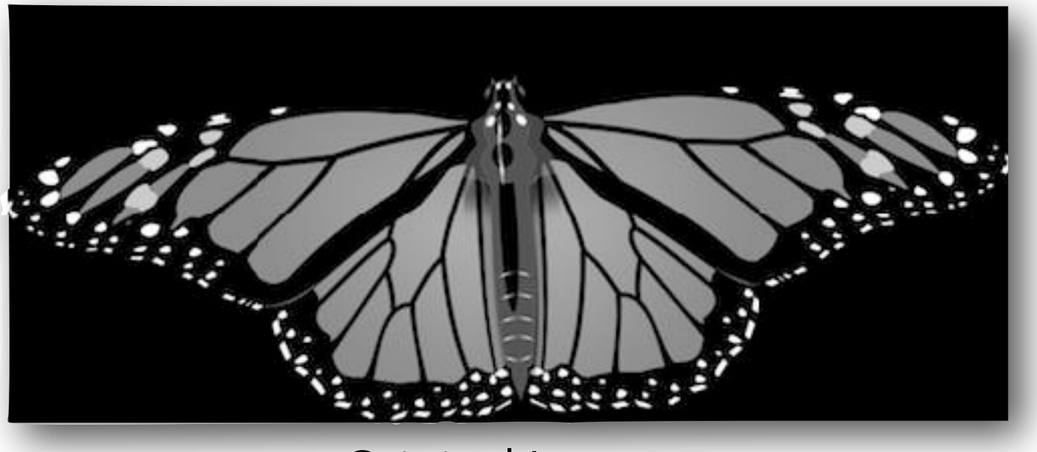
-1	1

X-Correlate with Kernel

-1
1

Derivatives of an Image using Correlation

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



Original Image

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

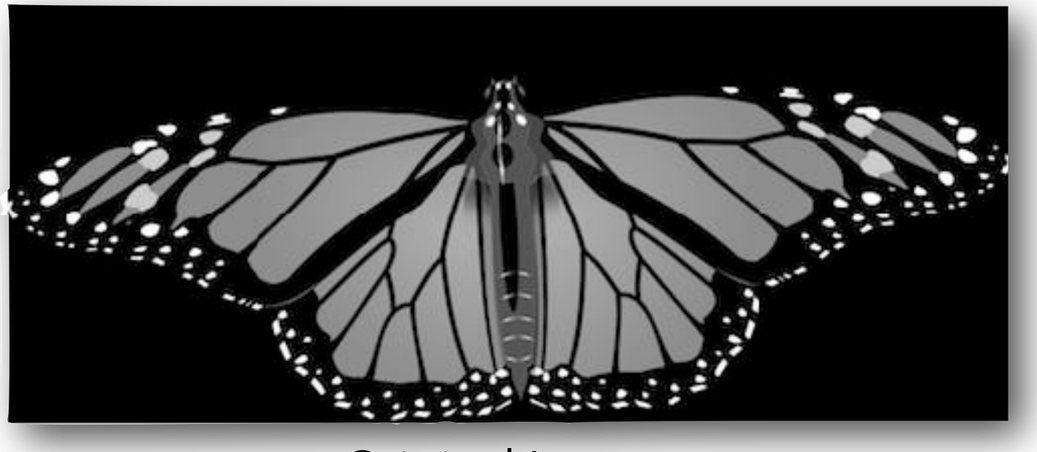
-1	1

X-Correlate with Kernel

-1
1

Derivatives of an Image using Correlation

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



Original Image

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

-1	1

X-Correlate with Kernel

-1
1

Discrete Gradient, using Correlation

We want an "operator" (mask/kernel) that we can apply to the image as a cross-correlation that implements the discretized (using Finite Differences) derivative computations:

Discrete Gradient, using Correlation

We want an "operator" (mask/kernel) that we can apply to the image as a cross-correlation that implements the discretized (using Finite Differences) derivative computations:

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

Gradient in x and y

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

Discrete Gradient, using Correlation

We want an "operator" (mask/kernel) that we can apply to the image as a cross-correlation that implements the discretized (using Finite Differences) derivative computations:

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

Gradient in x and y

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

H_x

0	0
-1	1
0	0

Symmetric about Image Point.
Which is the "middle" point?

Discrete Gradient, using Correlation

We want an "operator" (mask/kernel) that we can apply to the image as a cross-correlation that implements the discretized (using Finite Differences) derivative computations:

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

Gradient in x and y

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

H_x

0	0
-1	1
0	0

Symmetric about Image Point.
Which is the "middle" point?

H_x

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

Average of "left" and "right" derivatives

Discrete Gradient, using Correlation

We want an "operator" (mask/kernel) that we can apply to the image as a cross-correlation that implements the discretized (using Finite Differences) derivative computations:

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

Gradient in x and y

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

Rotated Kernel for y: H_y

H_x

0	0
-1	1
0	0

Symmetric about Image Point.
Which is the "middle" point?

H_x

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

Average of "left" and "right" derivatives

Various Kernels for Computing Gradients

H_x H_y

Prewitt:

Sobel:

Roberts:

Slide adapted from Aaron Bobick

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:

Roberts:

Slide adapted from Aaron Bobick

Various Kernels for Computing Gradients

	H_x	H_y
Prewitt:	-1 0 1 -1 0 1 -1 0 1	-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:		

Slide adapted from Aaron Bobick

Various Kernels for Computing Gradients

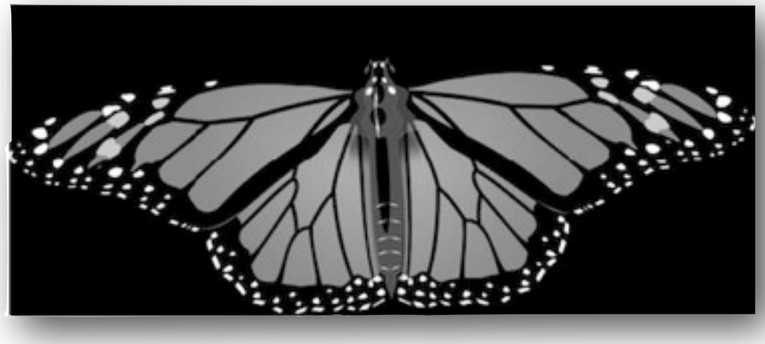
	H_x	H_y
Prewitt:	-1 0 1 -1 0 1 -1 0 1	-1 -1 -1 0 0 0 1 1 1
Sobel:	-1 0 1 -2 0 2 -1 0 1	-1 -2 -1 0 0 0 1 2 1
Roberts:	0 1 -1 0	1 0 0 -1

Slide adapted from Aaron Bobick

Various Kernels for Computing Gradients

	H_x	H_y
Prewitt:	-1 0 1 -1 0 1 -1 0 1	-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

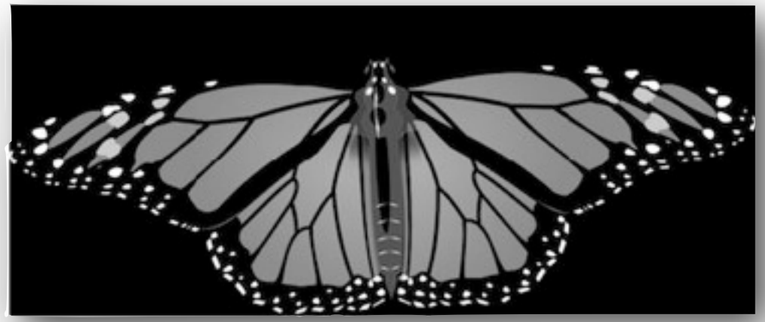
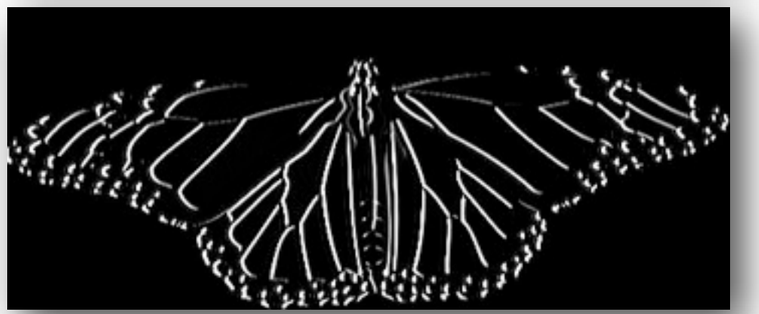
F



Input Image

Slide adapted from Aaron Bobick

Various Kernels for Computing Gradients

	H_x	H_y	$H_x \otimes F$	F
Prewitt:	-1 0 1 -1 0 1 -1 0 1	-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		Input Image
Roberts:	0 1 -1 0	1 0 0 -1		

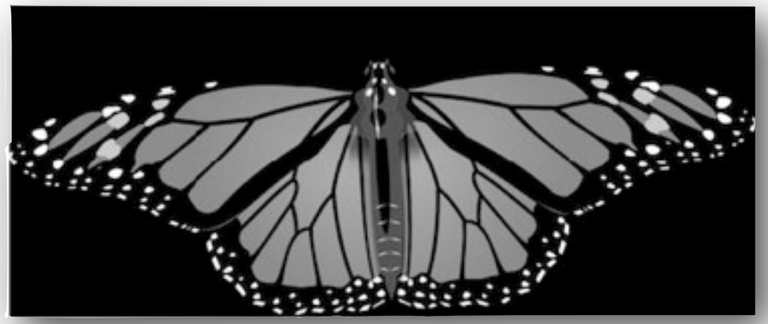
Slide adapted from Aaron Bobick

Various Kernels for Computing Gradients

Edges

$H_x \otimes F$

F



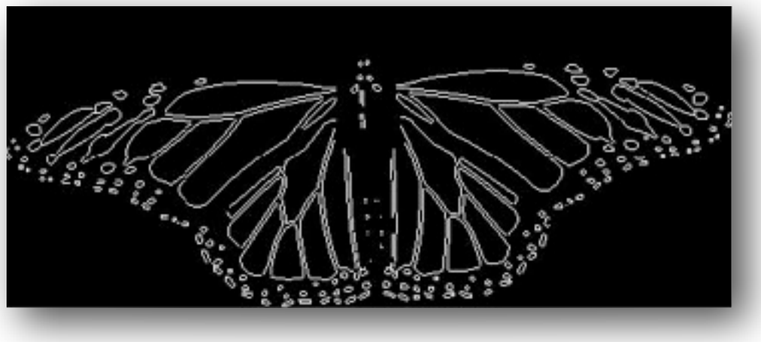
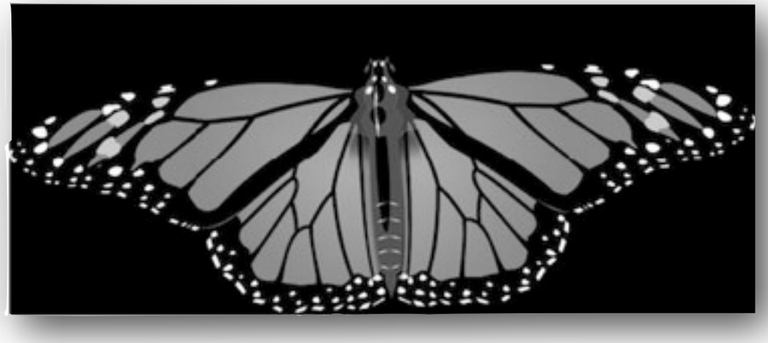
Input Image

Prewitt:

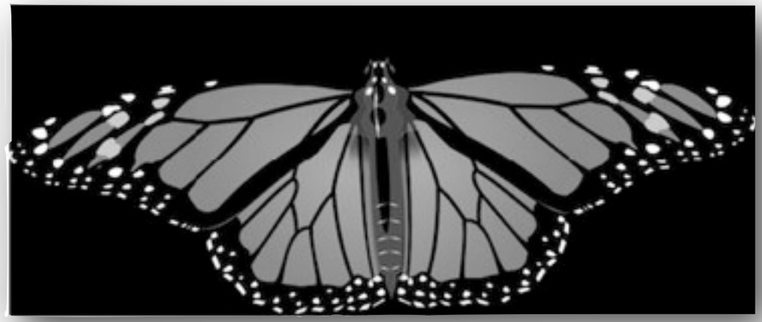
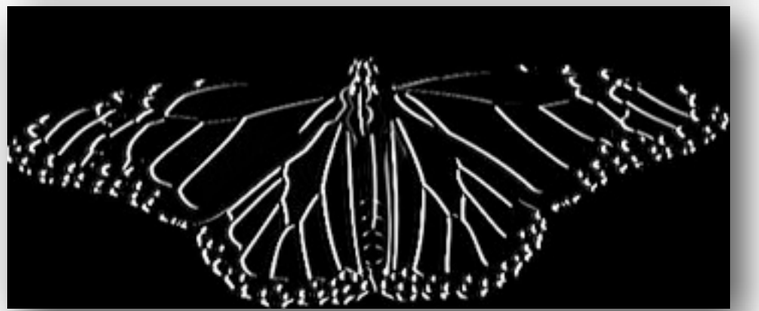
Sobel:

Roberts:

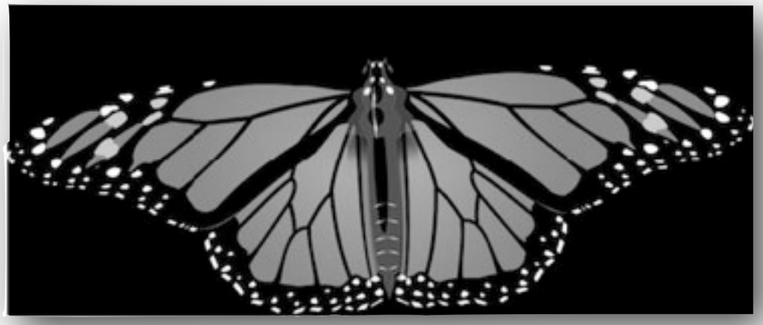
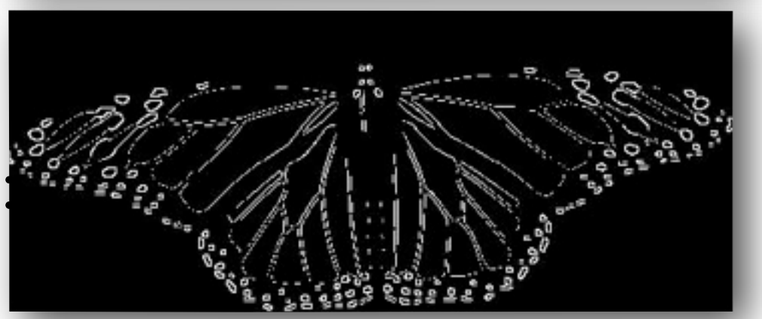
Various Kernels for Computing Gradients

Edges	$H_x \otimes F$	F
Prewitt:			
Sobel:			Input Image
Roberts:			

Various Kernels for Computing Gradients

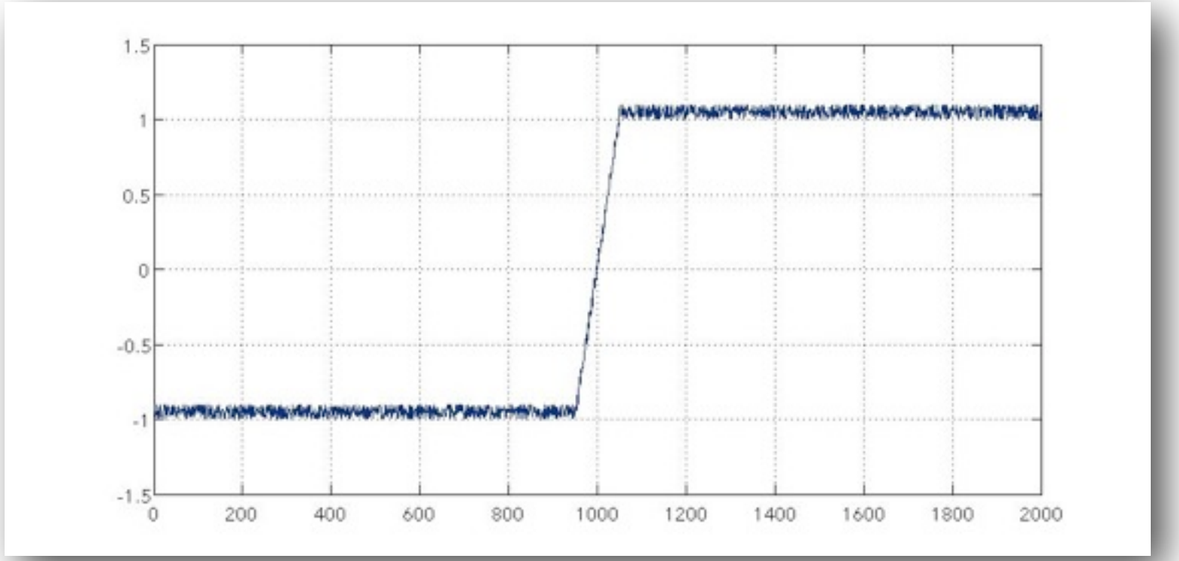
Edges	$H_x \otimes F$	F
Prewitt:			
Sobel:			Input Image
Roberts:			

Various Kernels for Computing Gradients

	Edges	$H_x \otimes F$	F
Prewitt:			
Sobel:			Input Image
Roberts:			

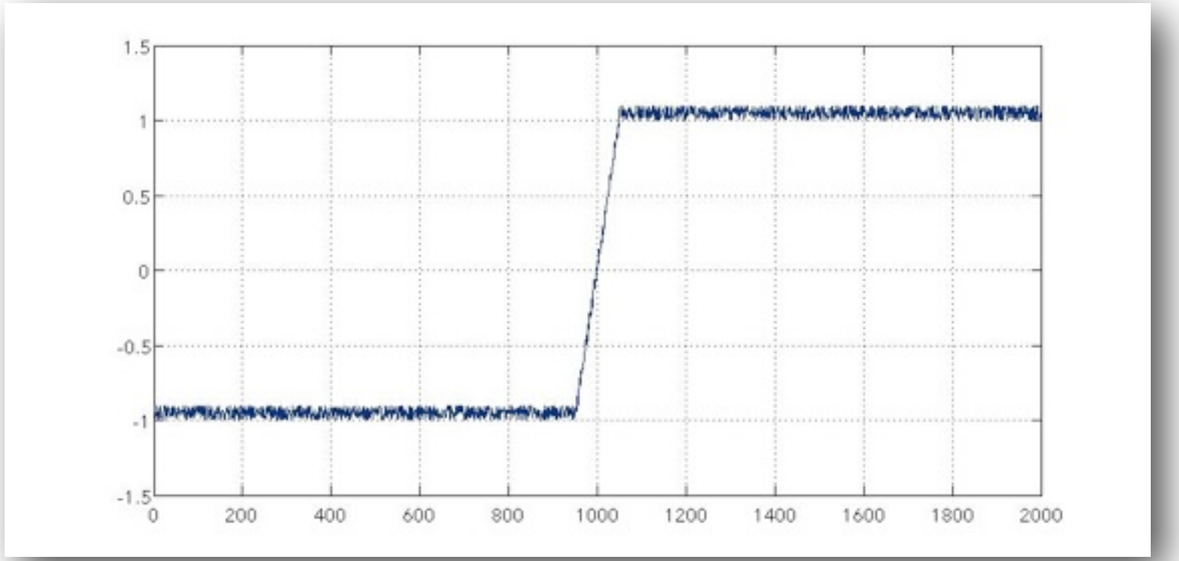
Impact of Noise on Gradients (1D Example)

$f(x)$

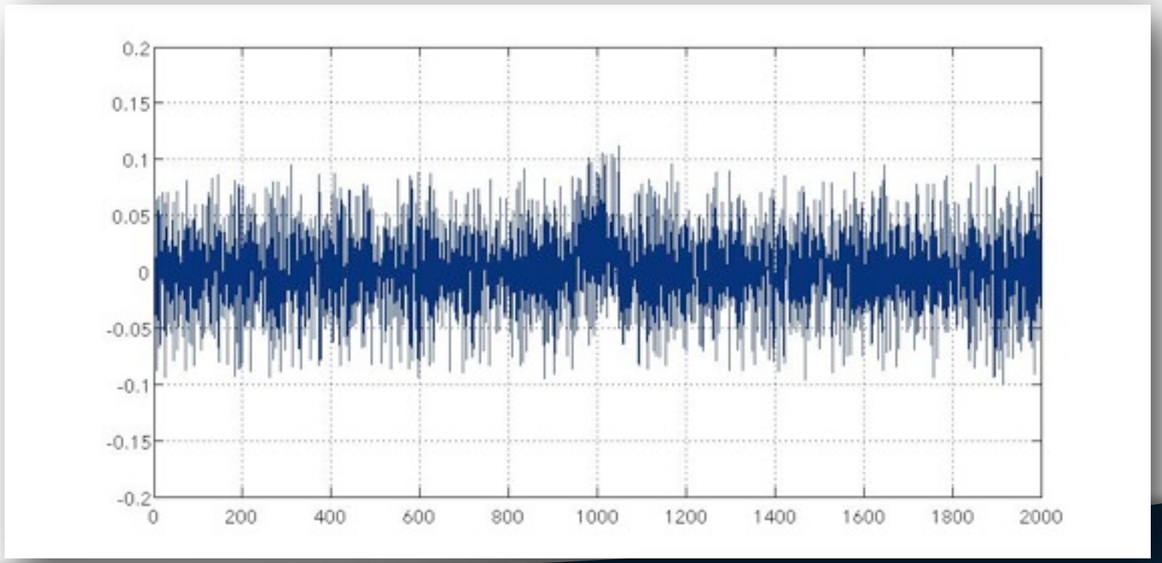


Impact of Noise on Gradients (1D Example)

$f(x)$



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



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

Impact of Noise on Gradients

Original Image +
Increasing Gaussian
Noise

Gradient

Increasing Noise

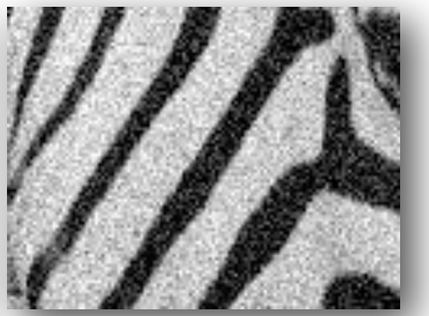
Impact of Noise on Gradients

Original Image + Increasing Gaussian Noise

Gradient

Increasing Noise

Impact of Noise on Gradients



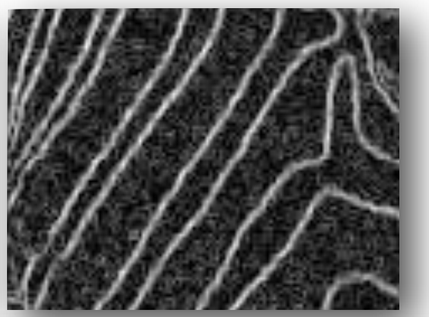
Original Image + Increasing Gaussian Noise

Gradient

Increasing Noise

Impact of Noise on Gradients

Original Image + Increasing Gaussian Noise



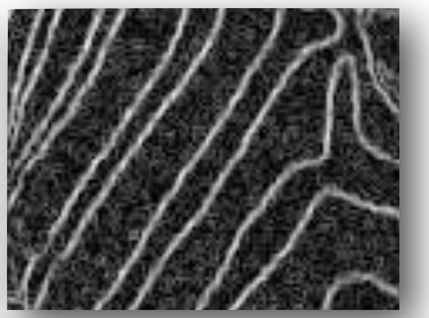
Gradient

Increasing Noise

Impact of Noise on Gradients



Original Image + Increasing Gaussian Noise

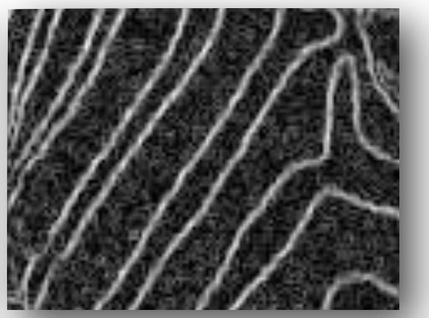
Gradient

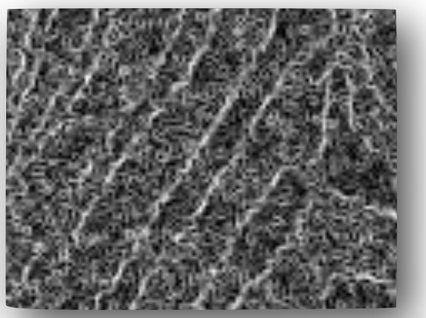
Increasing Noise

10

Impact of Noise on Gradients

Original Image + Increasing Gaussian Noise





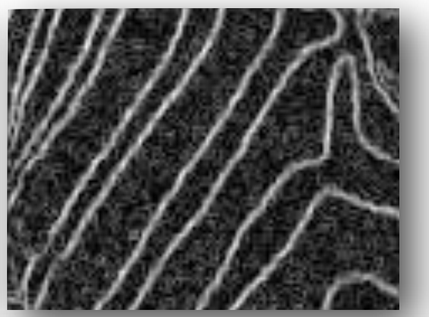
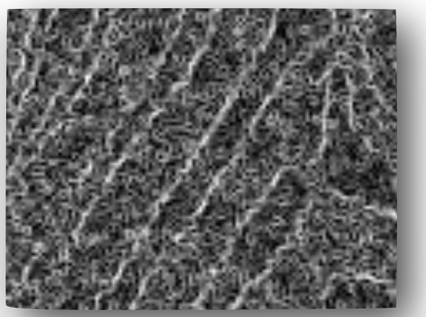
Gradient

Increasing Noise

Impact of Noise on Gradients




Original Image + Increasing Gaussian Noise

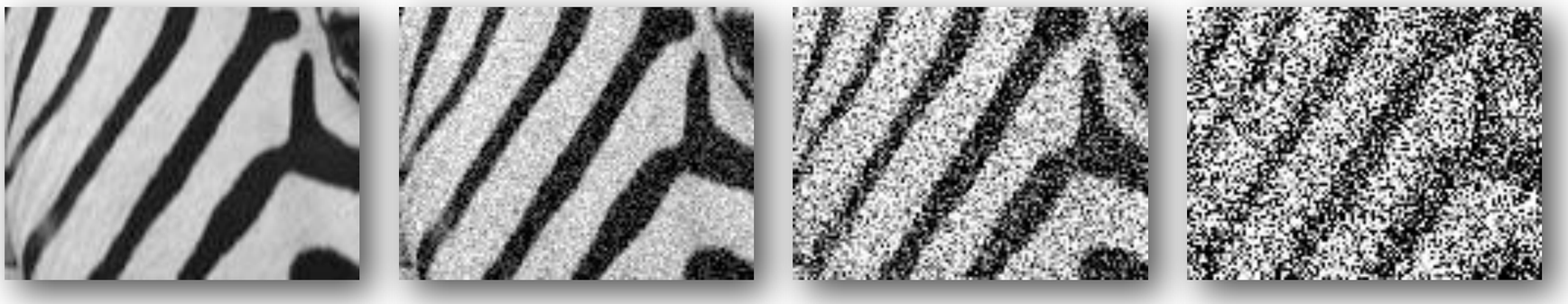




Gradient

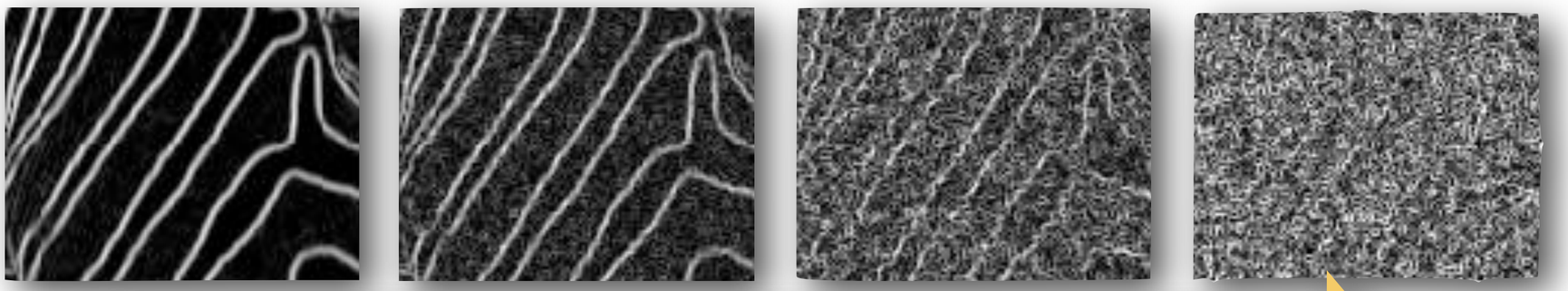
Increasing Noise

10

Impact of Noise on Gradients



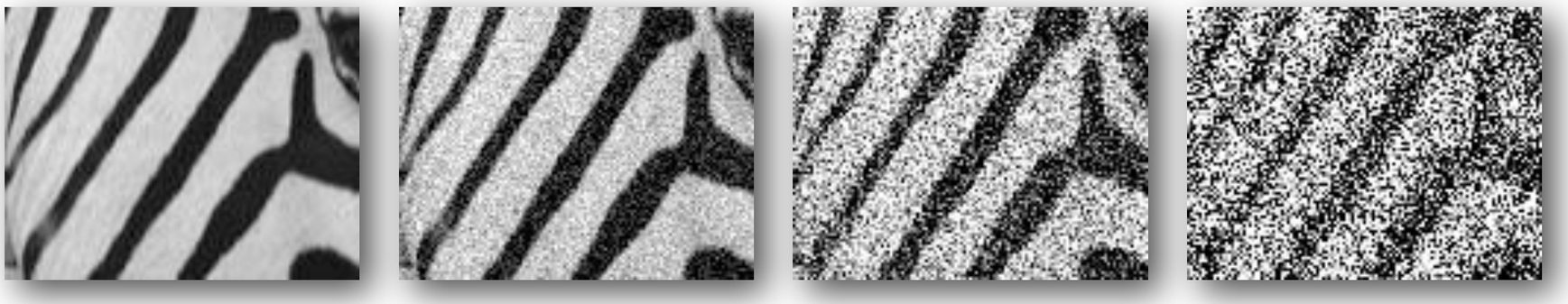
Original Image + Increasing Gaussian Noise



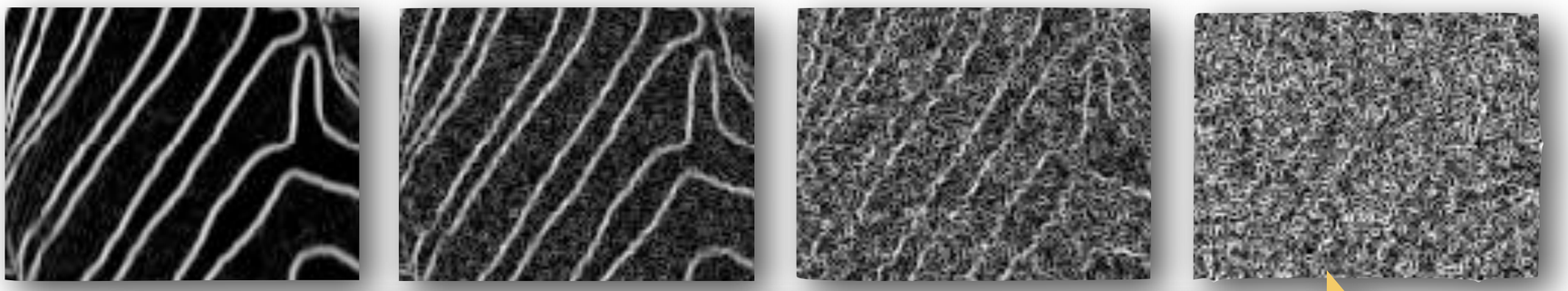
Gradient

Increasing Noise

Impact of Noise on Gradients



Original Image + Increasing Gaussian Noise



Gradient

Increasing Noise

Smooth the Image, before applying the Gradient Operations.

Convolution and Gradients

Convolution and Gradients

- Recall, Convolution is:

$$G = h * F$$

Convolution and Gradients

- Recall, Convolution is:

$$G = h * F$$

- Derivative of a Convolution is

$$\frac{\delta G}{\delta x} = \frac{\delta}{\delta x}(h * F)$$

Convolution and Gradients

- Recall, Convolution is:

$$G = h * F$$

- Derivative of a Convolution is

$$\frac{\delta G}{\delta x} = \frac{\delta}{\delta x}(h * F)$$

- In notation of Convolution, using properties of Convolution, if D is kernel for derivatives and H is the Kernel for smoothing.

$$D * (H * F) = (D * H) * F$$

Convolution and Gradients

- Recall, Convolution is:

$$G = h * F$$

- Derivative of a Convolution is

$$\frac{\delta G}{\delta x} = \frac{\delta}{\delta x}(h * F)$$

- In notation of Convolution, using properties of Convolution, if D is kernel for derivatives and H is the Kernel for smoothing.

$$D * (H * F) = (D * H) * F$$

- So we could define kernels with derivative and smoothing in one

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

Canny Edge Detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide “ridges” down to single pixel width
4. Linking and thresholding (hysteresis):
   a. Define two thresholds: low and high
   b. 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.
- ★ Discussed how Edges are computed.
- ★ Presented the process of smoothing prior to Gradient computation.
- ★ Presented four (4) different methods for Edge computation.

Next Class

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

15

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
  - Creative Commons Search
  - 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/