# 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

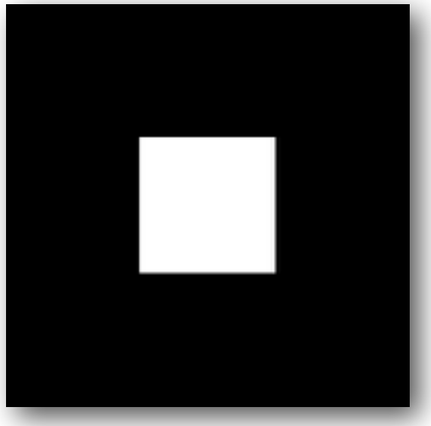
# Image Processing and Filtering, via Convolution and Cross-Correlation

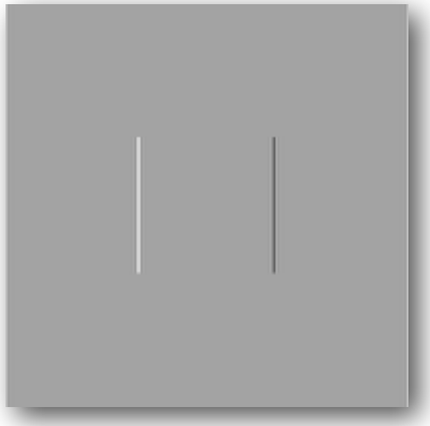
* Towards Edge Detection: Computing Image Gradients

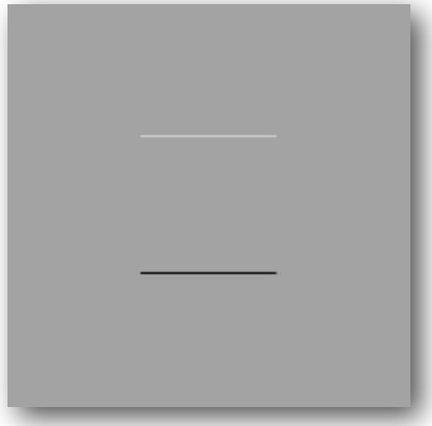

1. Compute Edges
2. Derivatives using kernels and neighborhood operations
3. Three (3) methods for computing edges using kernels
4. Image noise can complicate the computation of gradients
5. The Canny Edge Detector

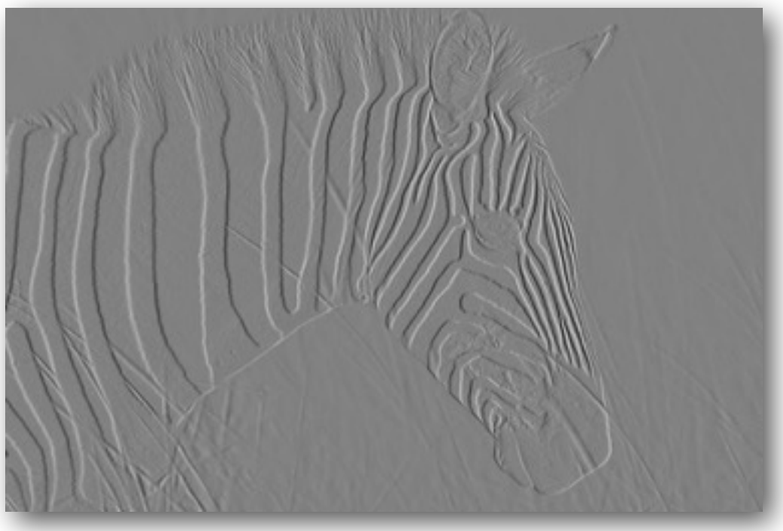
# Recall: Differentiating an Image in X and Y

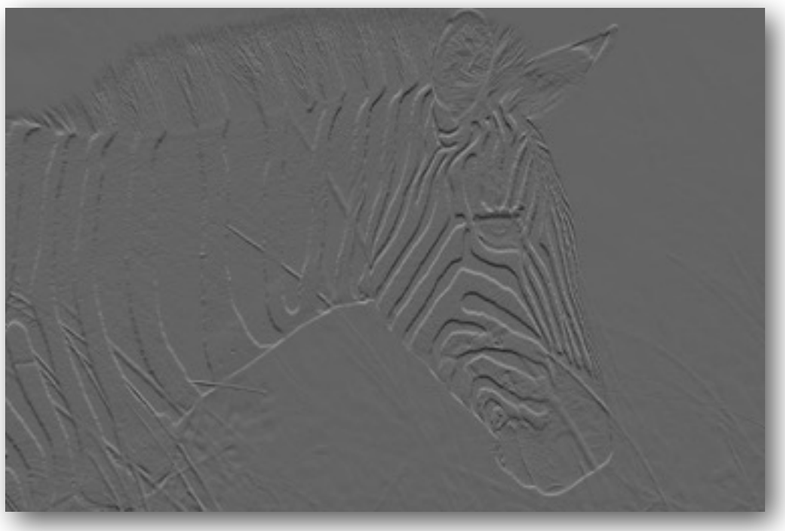
$$\frac{\delta F(x,y)}{\delta x} \approx F(x+1,y) - F(x,y) \qquad \frac{\delta F(x,y)}{\delta y} \approx F(x,y+1) - F(x,y)$$











# 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)

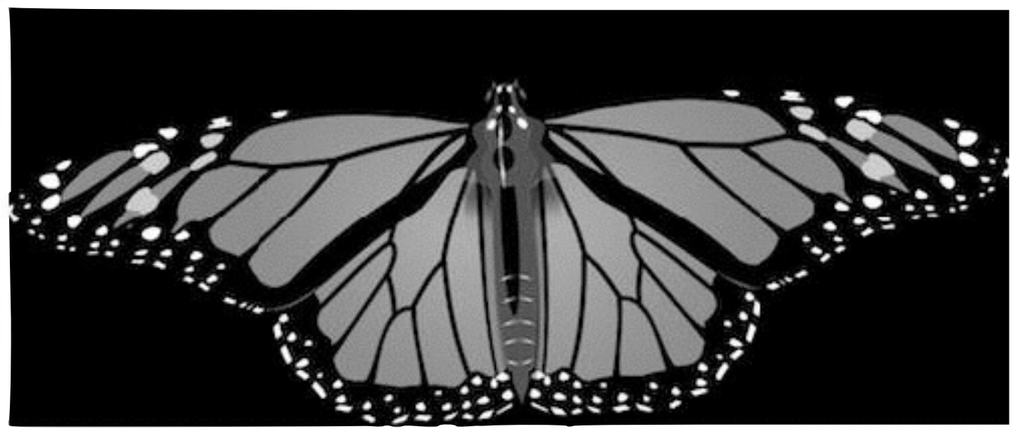
$$= \begin{bmatrix} -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} F(x,y) & F(x+1,y) \end{bmatrix}^T$$

# Derivative using Cross-Correlation

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

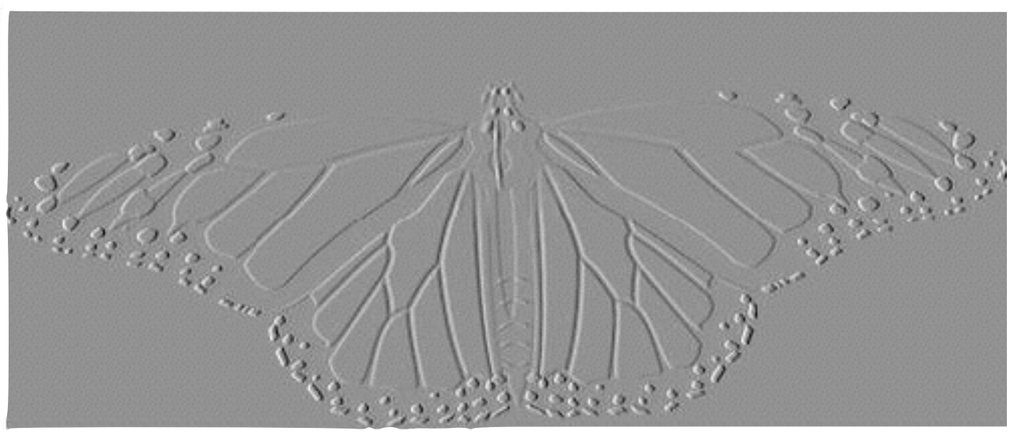
| -1 | 1 |
|---|---|

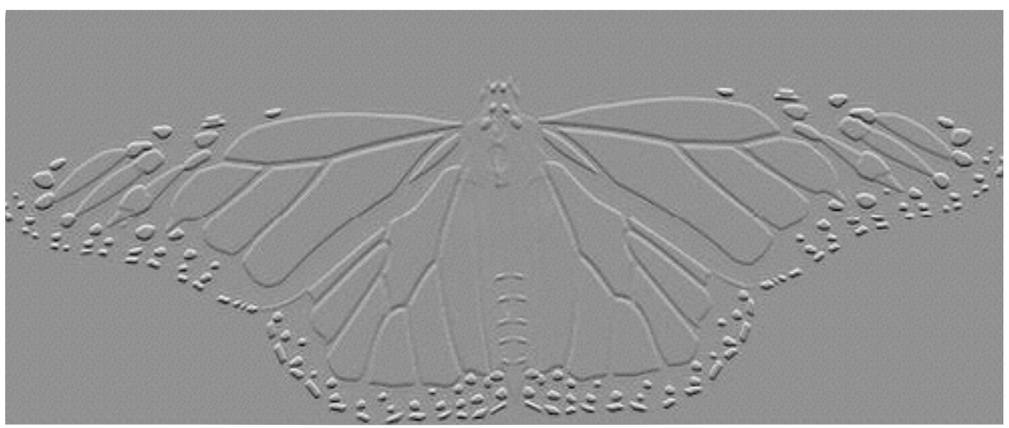
Cross-Correlate with Kernel

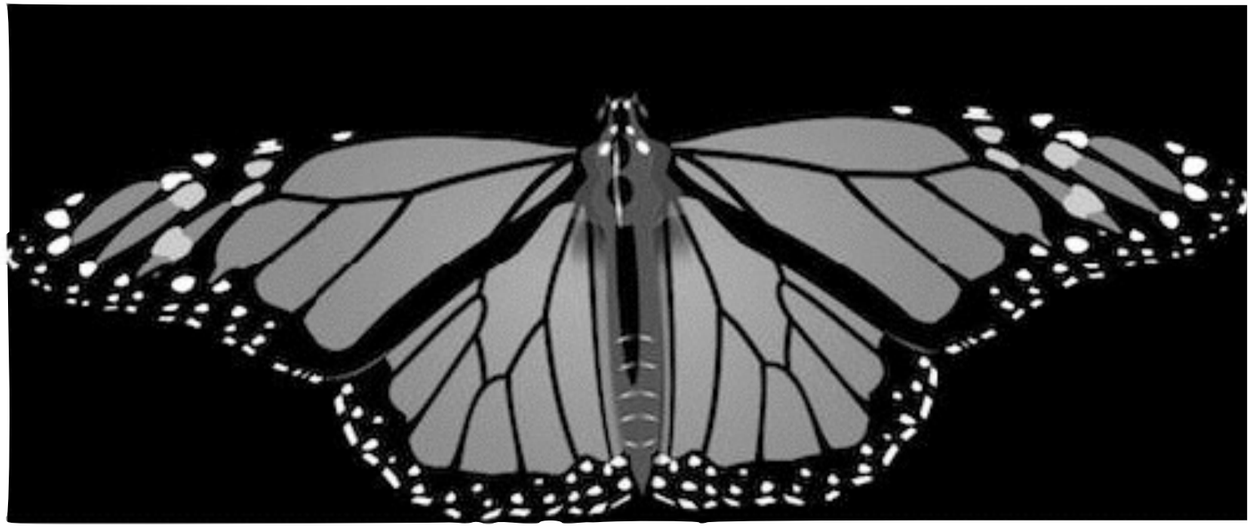


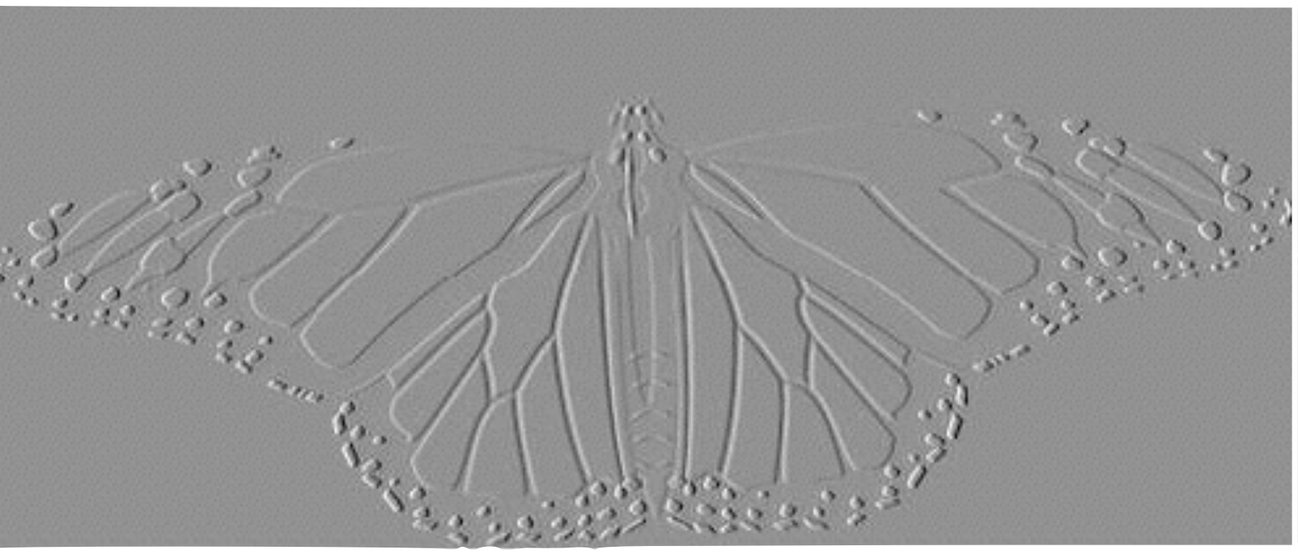
$$\frac{\delta F(x,y)}{\delta y}$$

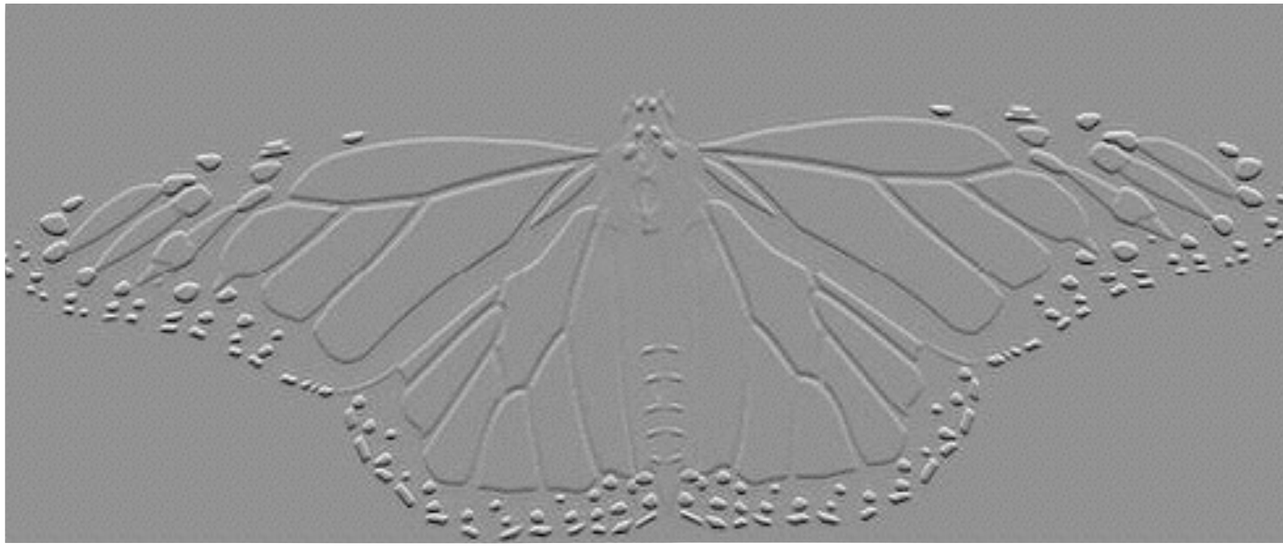
| -1 |
|---|
| 1 |











# 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 |

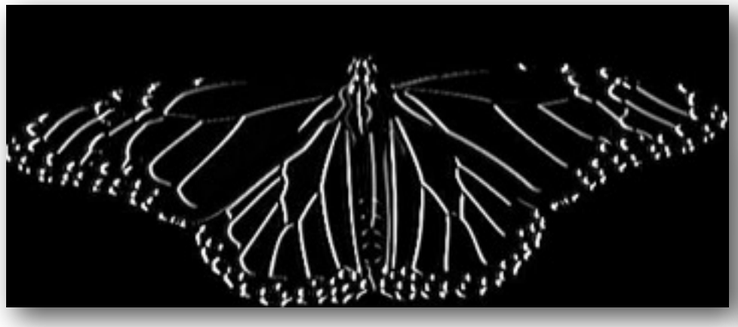
| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

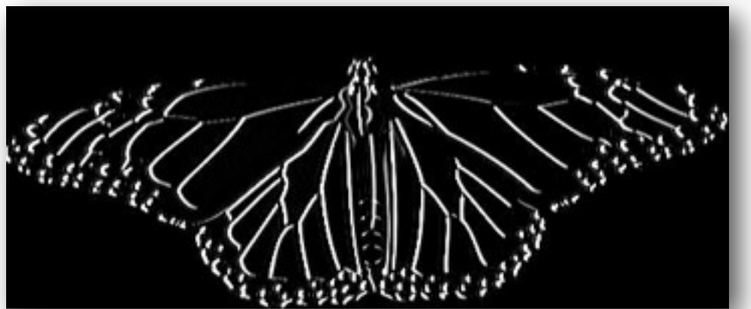
Roberts

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

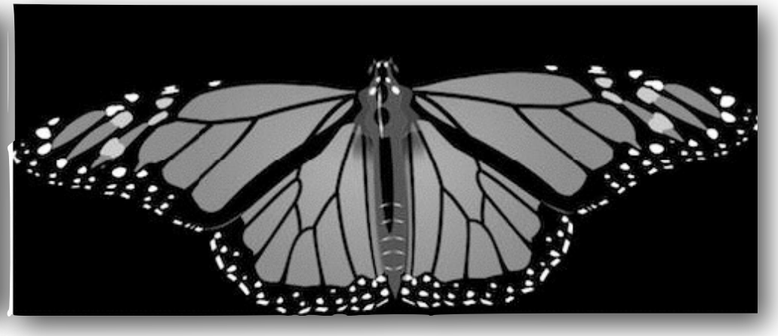
| | |
|---|---|
| 1 | 0 |
| 0 | -1 |

$H_x \otimes F$





$F$



Input Image

Slide adapted from Aaron Bobick

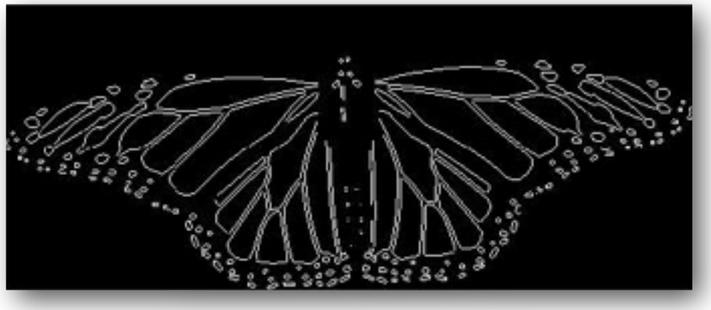
# Various Kernels for Computing Gradients

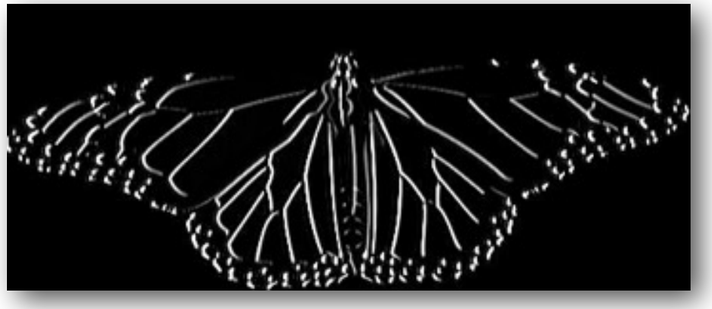
Edges

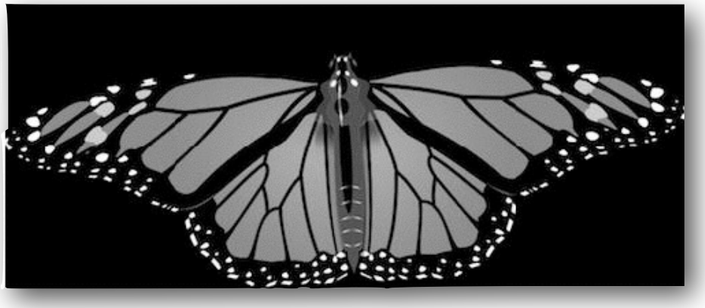
$H_x \otimes F$

$F$

Prewitt

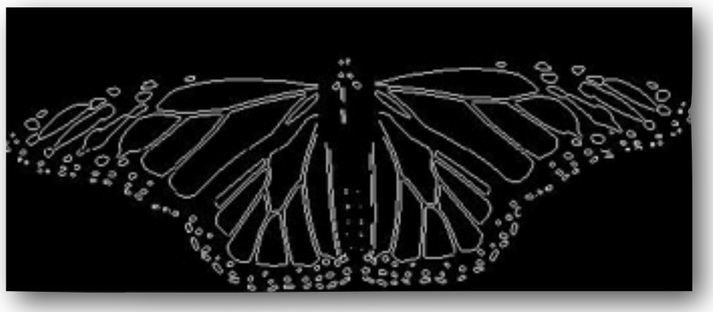


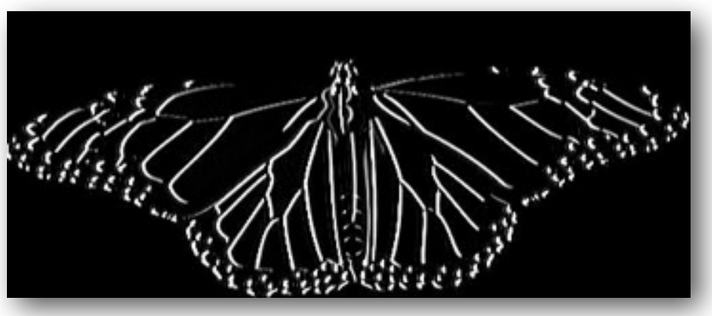




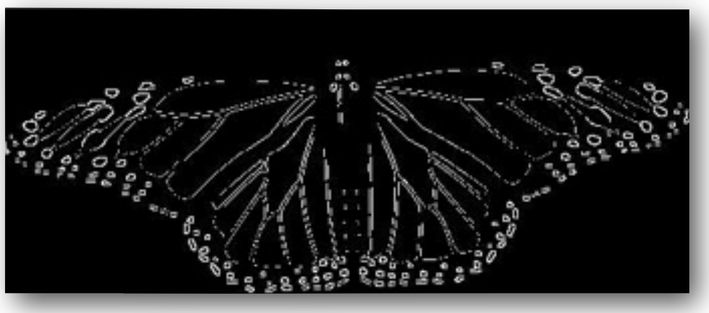
Input Image

Sobel



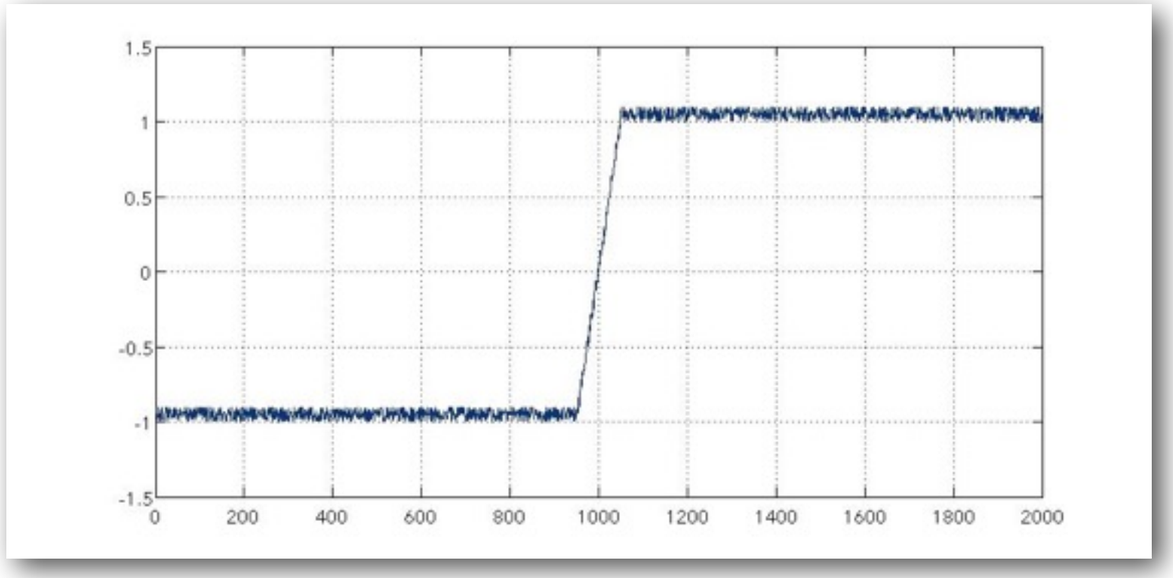


Roberts

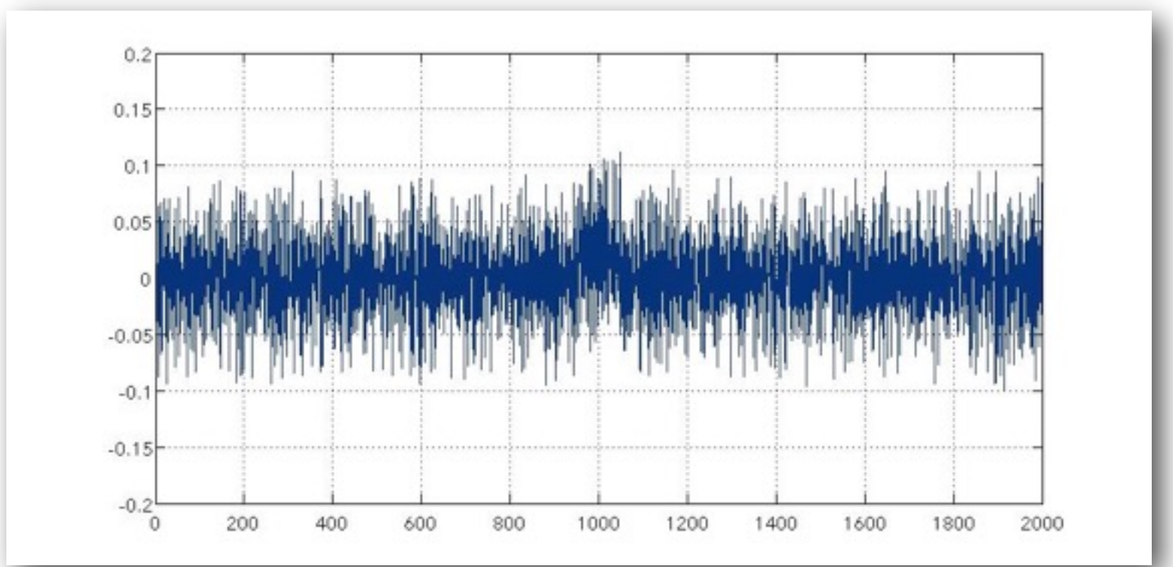


# Impact of Noise on Gradients (1-D Example)

$f(x)$

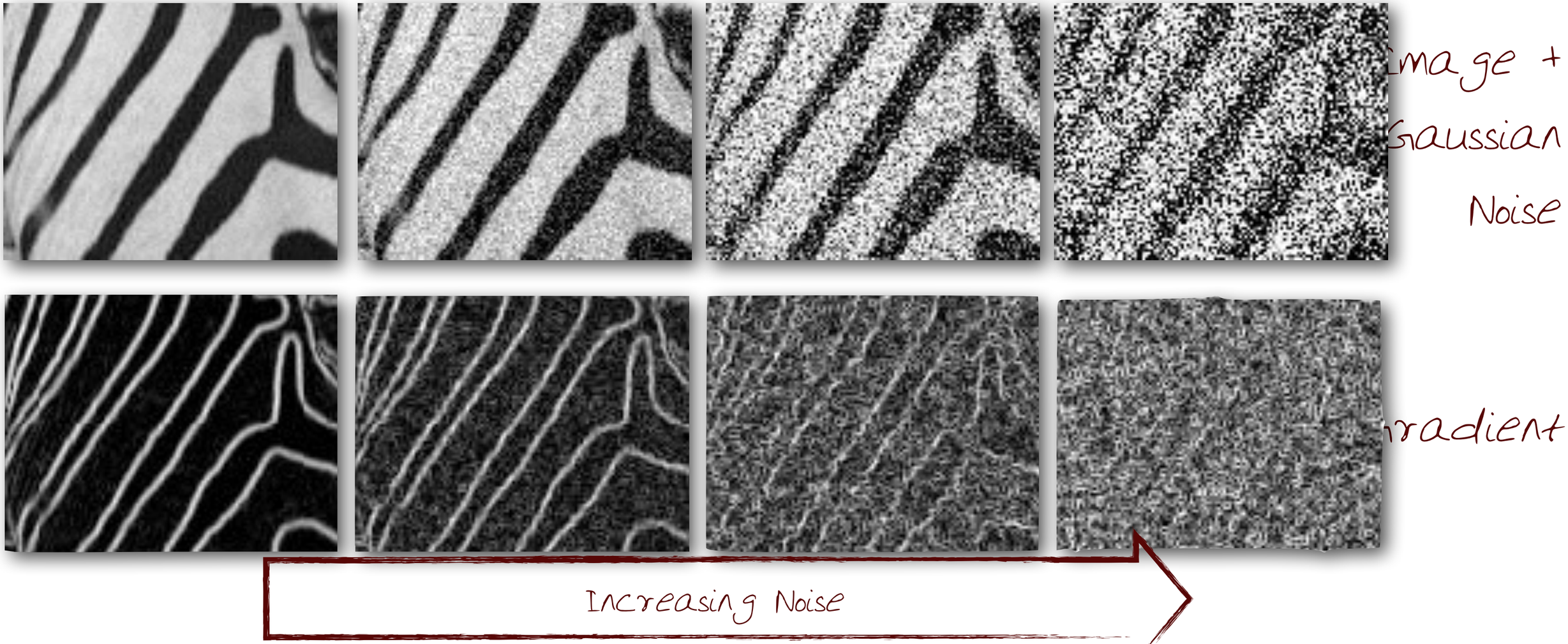



$$\frac{\delta f(x)}{\delta x}$$




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

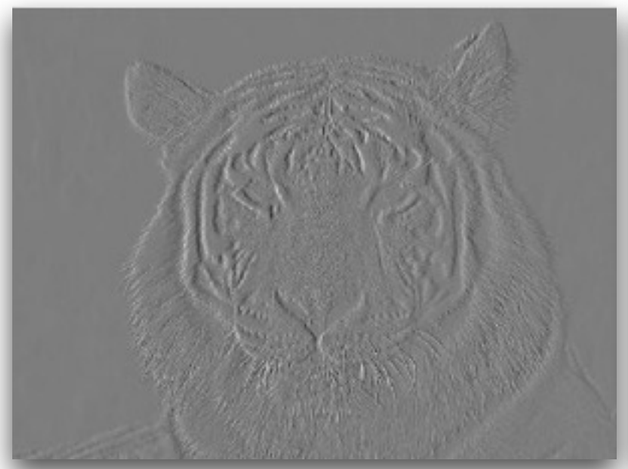
Gradient Image

Edge Image

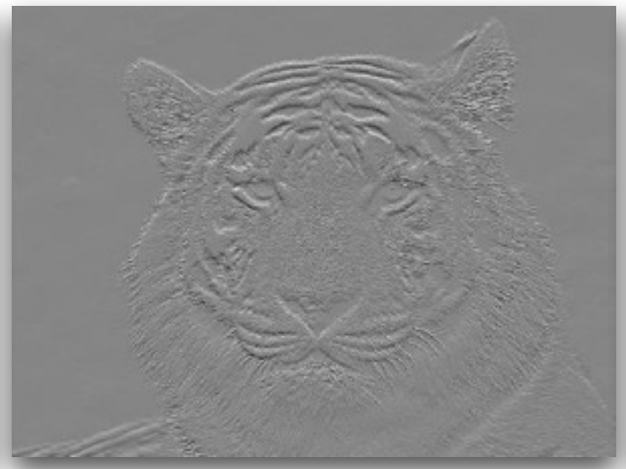
1. Smoothing: suppress noise
2. Compute Gradient
3. Apply Edge "enhancement":
4. Edge localization:
   1. Edges vs. noise
5. Threshold, Thin

# 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/