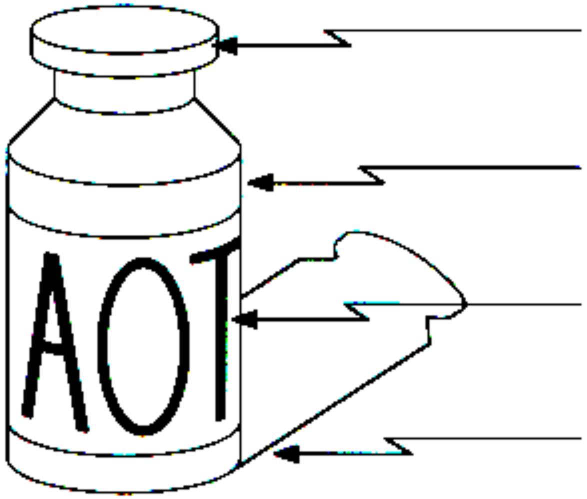
Edge Detection

Edge Detection

The process of identifying parts of a digital image with sharp changes (discontinuities) in image intensity.

Edges are caused by a variety of factors



surface orientation discontinuity

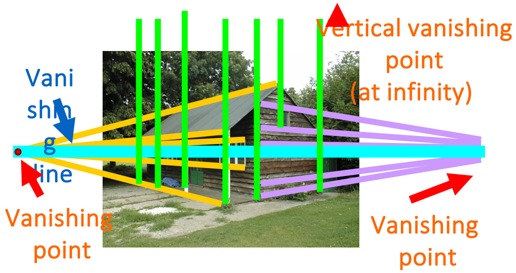
depth discontinuity

surface color discontinuity

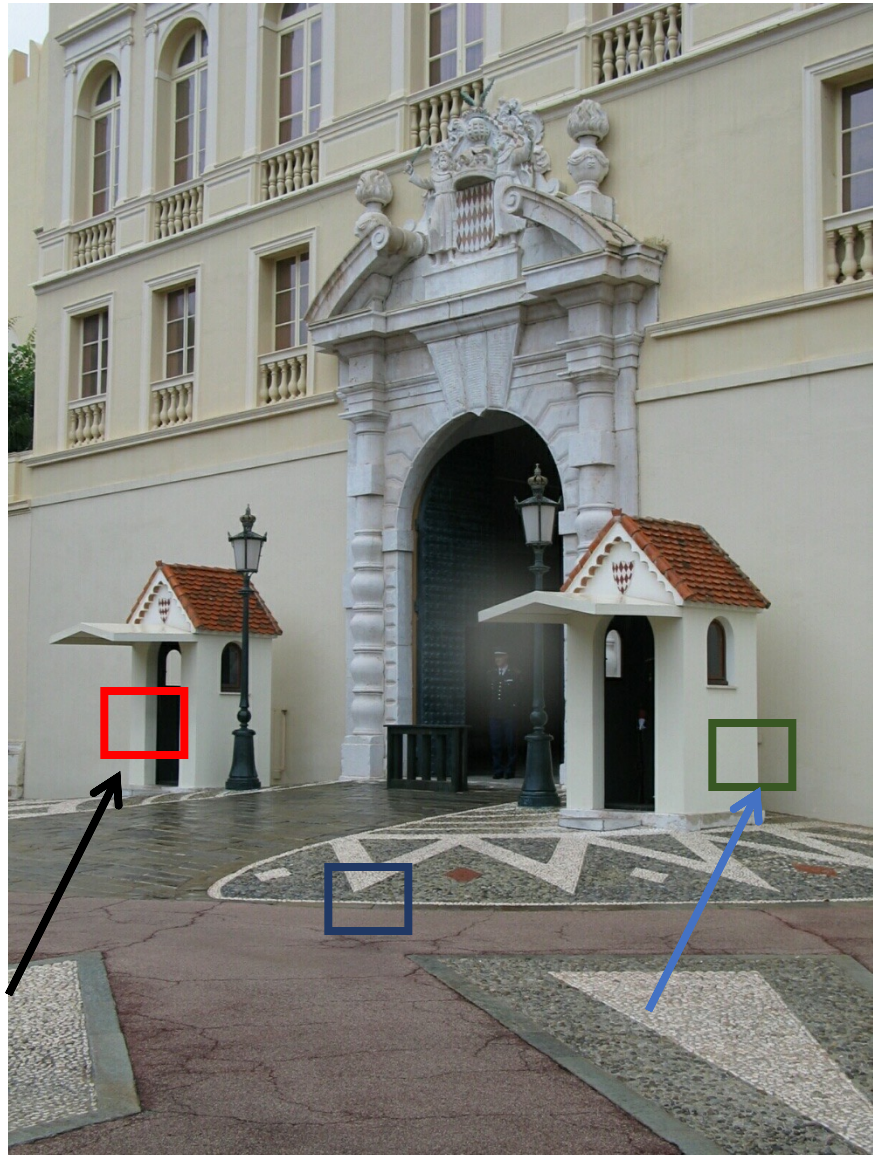
illumination discontinuity

Use of Edges

- Extract information
  - Recognize objects
  - Reconstruct scenes
- Help recover geometry and viewpoint
- Shape matching
- And so on…



Closer Look at Edges



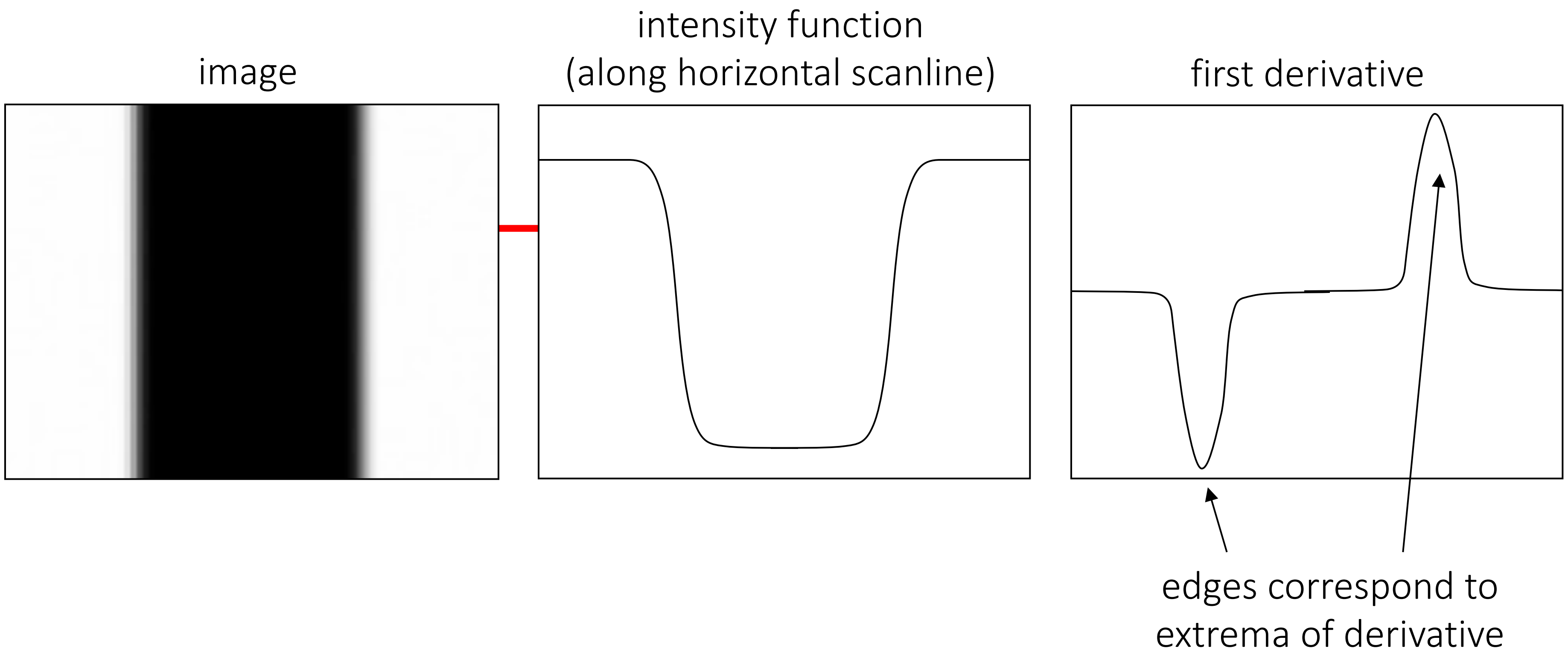
Derek Hoiem

- We can zoom into a small patch in the image.
- Try to interpret the changes between the vertical bars as edges; think about what type of discontinuity each is caused by.

Characterizing Edges

An edge is a place of sharp change (discontinuity) in the image intensity function.

- Simple algorithm for edge detection: find gradient.



- The first derivative is strongest (i.e. has the highest magnitude) where the intensity changes most rapidly.
- The sign of the derivative depends on whether the intensity falls from high to low or rises from low to high.

Adapted from: James Hays

Derivatives in Practice

Digital images are discrete signals

Derivative $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$

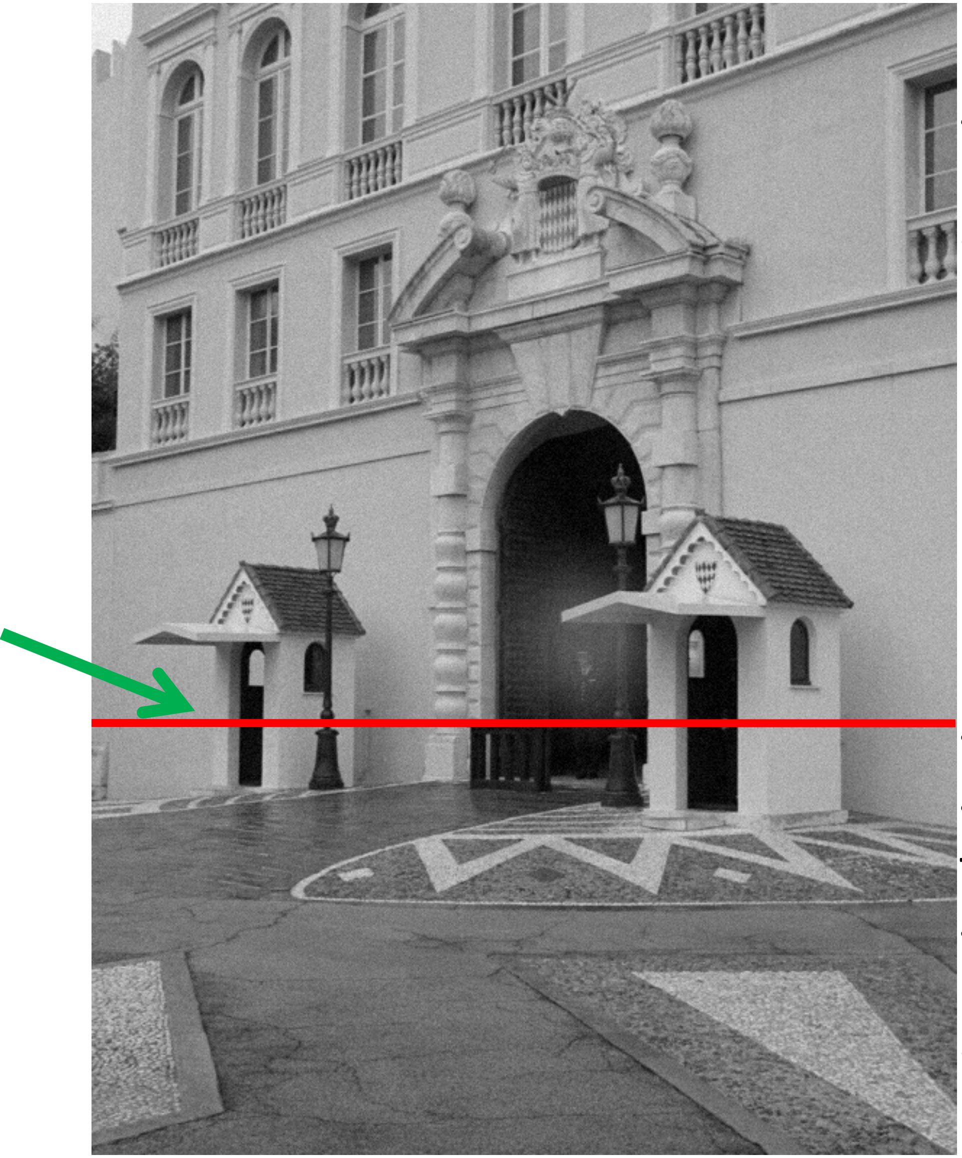
We approximate the true derivative with finite differences.
The smallest sampled ‘h’ in images is 1 (one pixel, smallest step you can take).

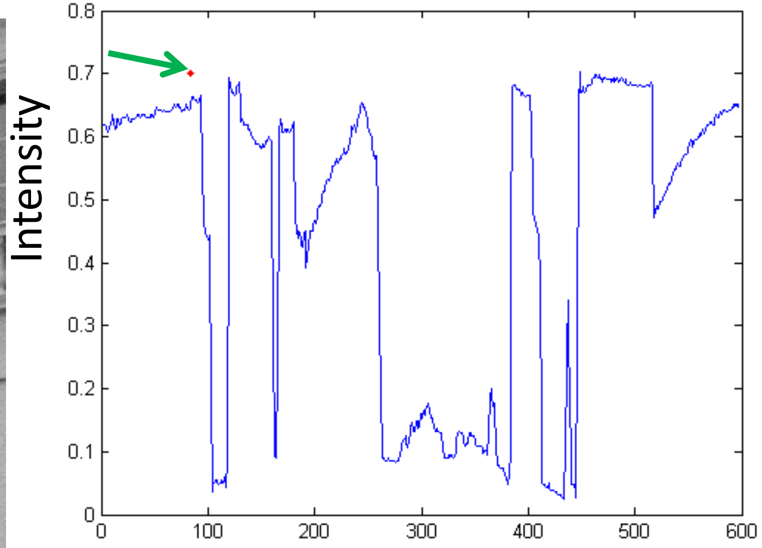
$$f'(x) \approx f(x+1) - f(x).$$

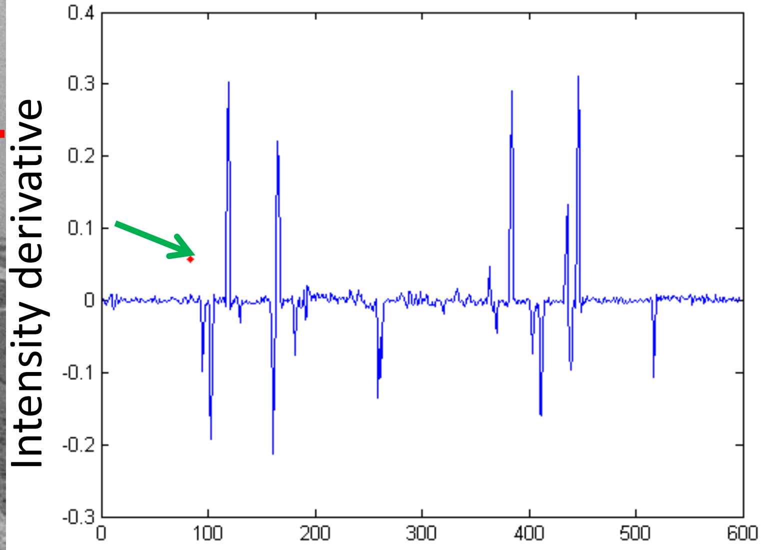
$$f'(x) \approx f(x-1) - f(x+1).$$

Symmetric form produces derivative estimate at x exactly, rather than at the border between (x+1) and x as above.

Intensity Profile



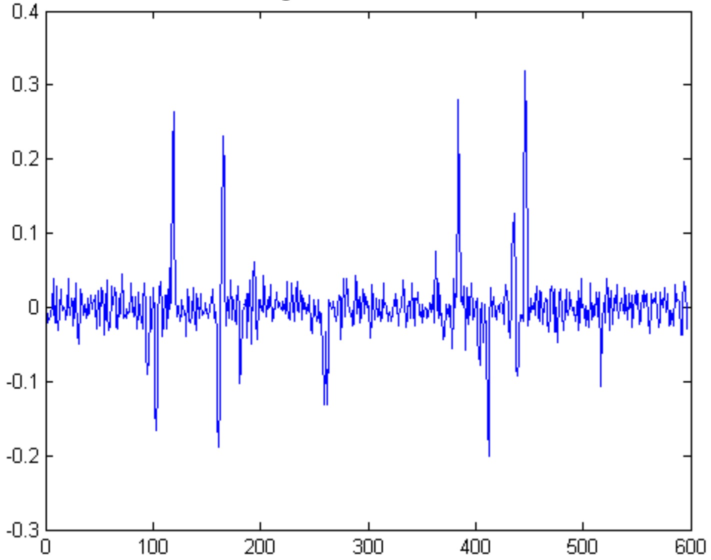




Adapted from: Derek Hoiem

Real images are noisy, resulting in lots of noise in the first derivative and starts to overwhelm the peaks.

Adding Gaussian noise

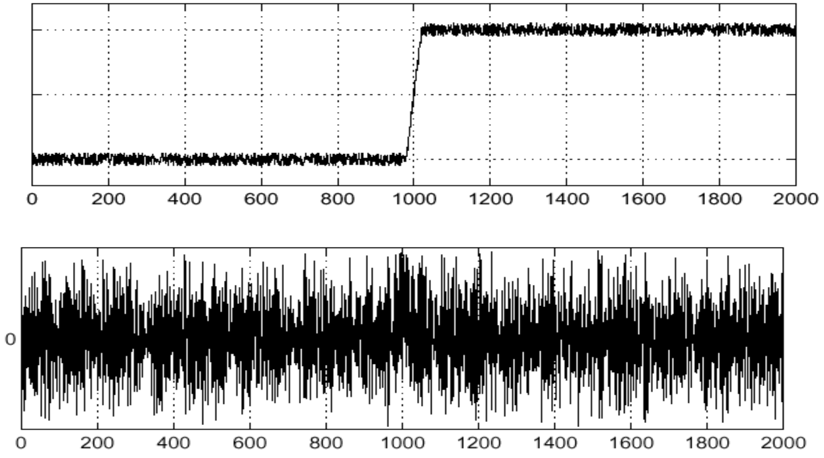


Effects of Noise

Consider a noisy signal f(x)

$f(x)$

$\frac{d}{dx}f(x)$



Unable to detect edges by inspecting the derivative graph with all that noise.

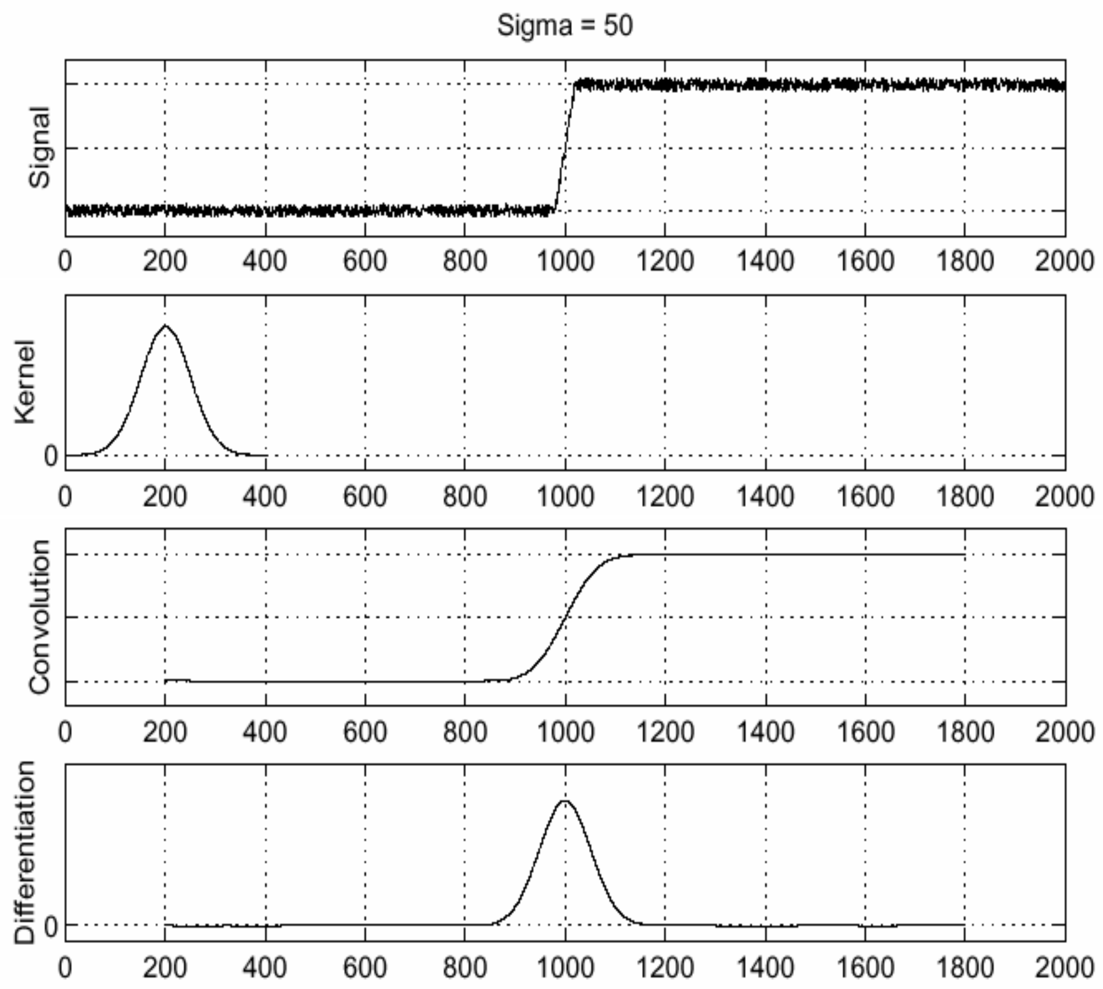
<u>Solution</u>: Apply smoothening filtering to mitigate the noise **before** computing the derivative! A gaussian filter is common.

f

g

$f * g$

$\frac{d}{dx}(f * g)$



To find edges, look for peaks in $\frac{d}{dx}(f * g)$

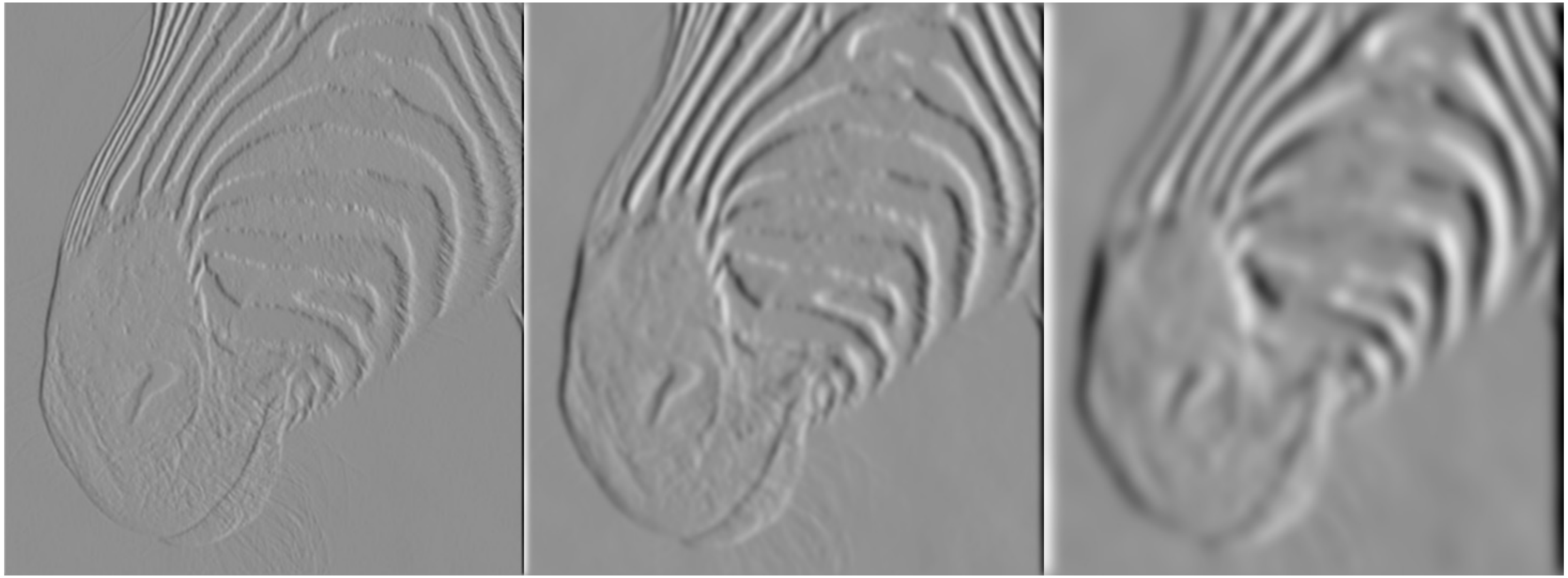
<u>Think, how it is useful here and in DOG filter?</u>

Also, $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$

Adapted from: Steve Seitz

Smoothing-Localization Tradeoff

Smoothed derivative removes noise, but blurs edge.
Also finds edges at different frequencies.



1 pixel | 3 pixels | 7 pixels

Designing an Edge Detector

Criteria for a good edge detector:

- **Good detection:** the optimal detector should find all real edges, ignoring noise or other artifacts
- **Good localization**
  - the edges detected must be as close as possible to the true edges
  - the detector must return one point only for each true edge point

Cues of edge detection

- Differences in color, intensity, or texture across the boundary
- Continuity and closure
- High-level knowledge

Where do humans see boundaries?

image | human segmentation | gradient magnitude

James Hays. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

A Better Edge Detector

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

David Forsyth

Canny Edge Detector

Algorithm

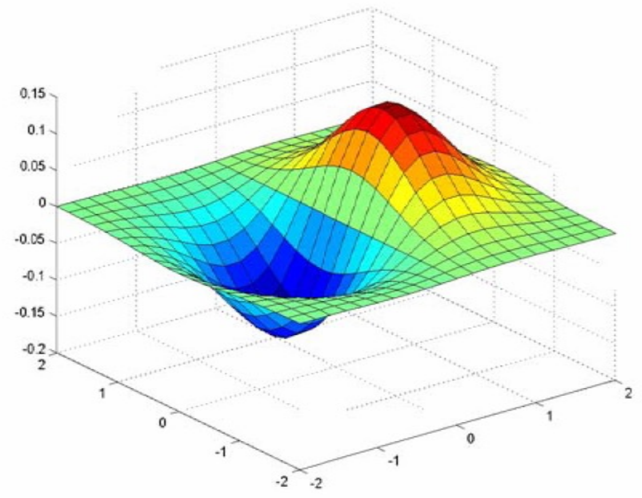
1. Filter image with x, y derivatives of Gaussian

Input Image

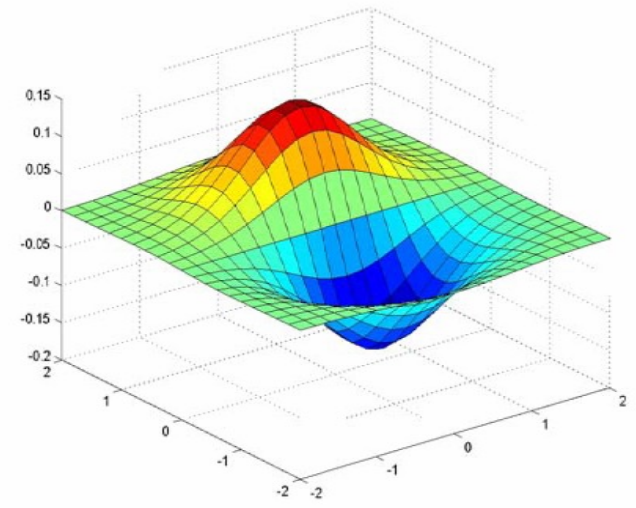
rgb2gray(‘img.png’)

Derivative of Gaussian filter

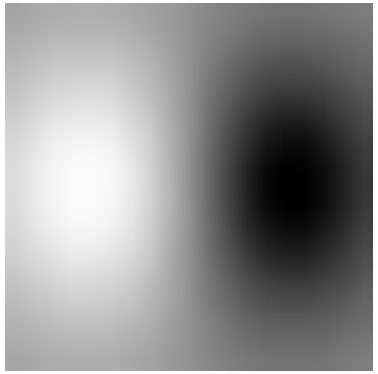
Why?
Already
discussed…

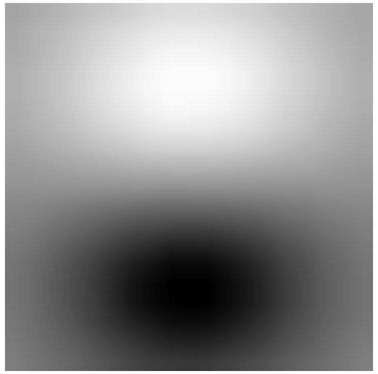


x-direction

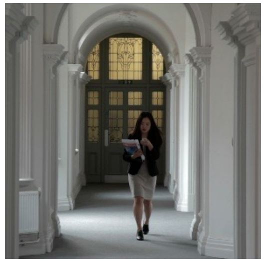


y-direction



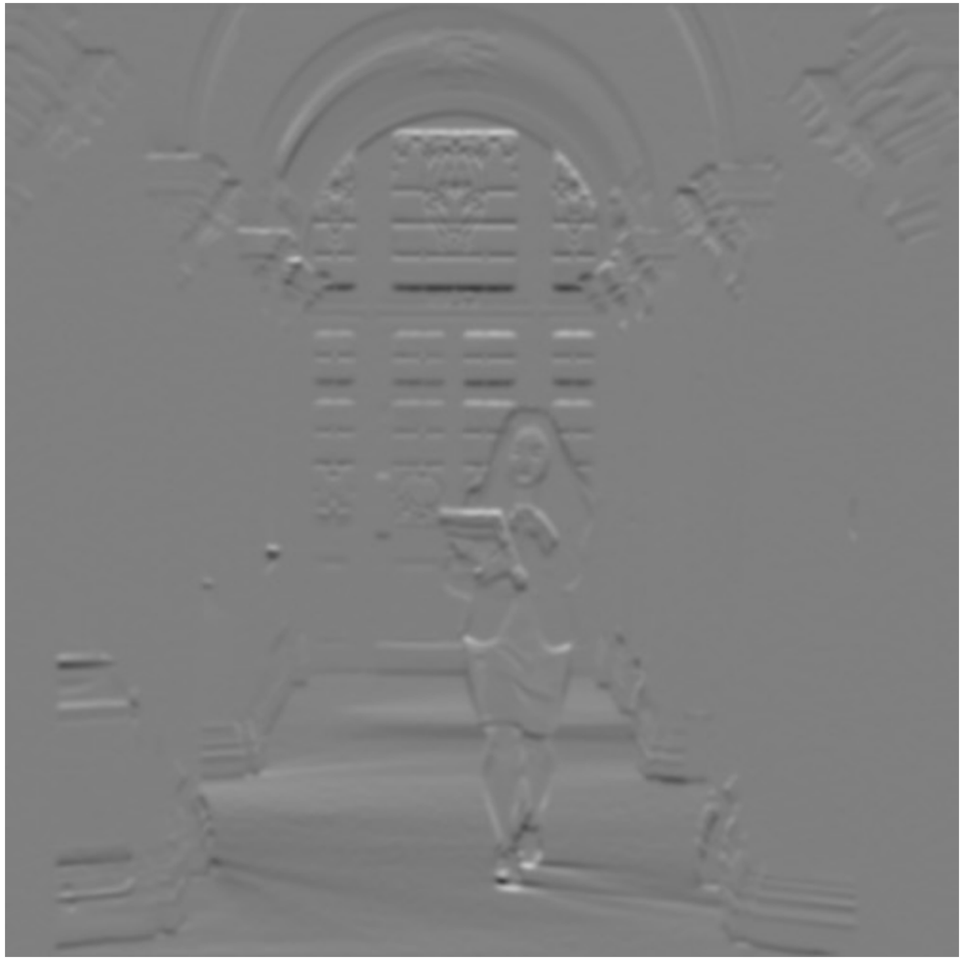


Compute Gradients



X’ Derivative of Gaussian

Y Derivative of Gaussian



More vertical edges are present in the X derivative version

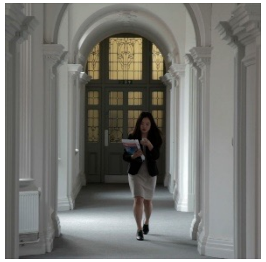
(add 0.5 for visualization)

Canny Edge Detector

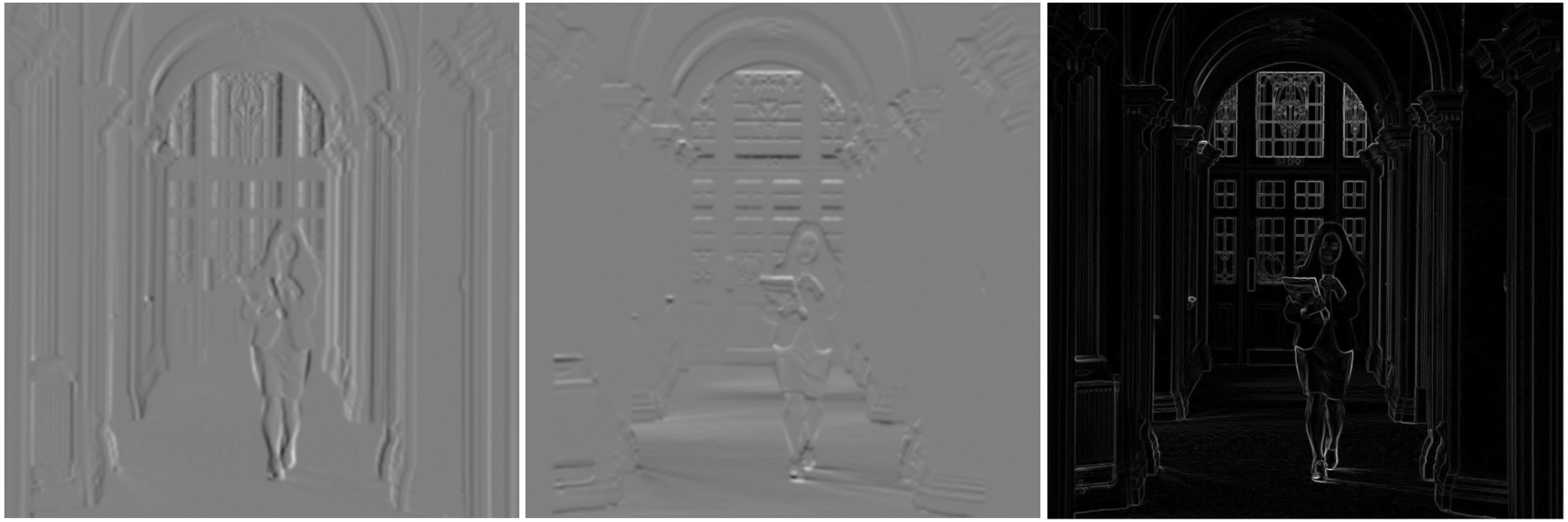
Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient

Compute Gradient Magnitude



sqrt(XDerivOfGaussian .^2 + YDerivOfGaussian .^2) = gradient magnitude



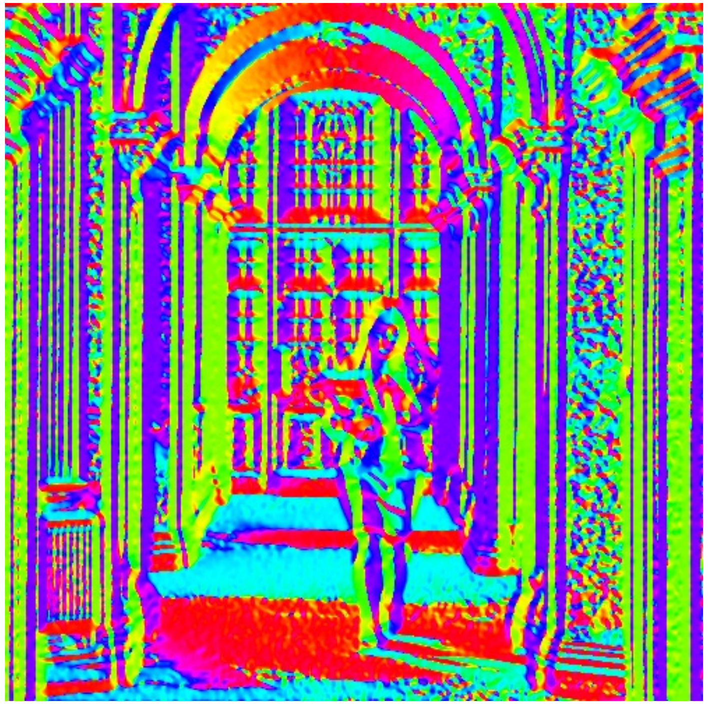
Compute Gradient Orientation

- Threshold magnitude at minimum level
- Get orientation via

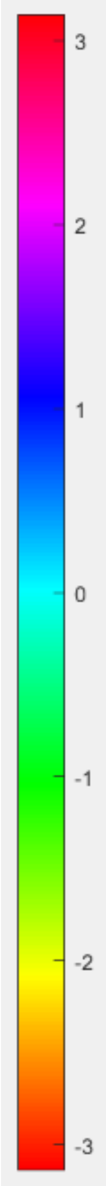
```
theta = atan2(yDeriv, xDeriv)
```

Visualizing the orientation using colors that vary in [-pi, pi]

Unthresholded



Thresholded

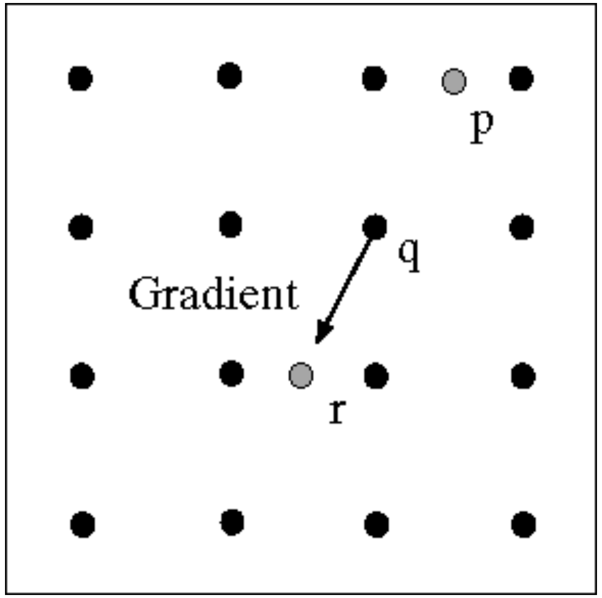


Canny Edge Detector

Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide “ridges” to single pixel width

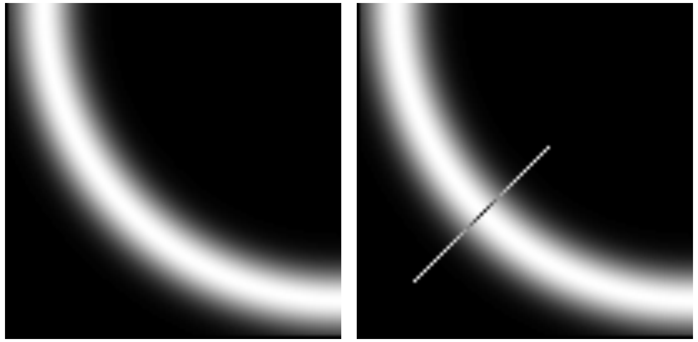
Non-Maximum Suppression for Each Orientation



At pixel q:

We have a maximum if the value is larger than those at both p and at r.

Interpolate along gradient direction to get these values.



Before Non-max Suppression

Gradient magnitude (x4 for visualization)

After Non-max Suppression

Gradient magnitude (x4 for visualization)

Canny Edge Detector

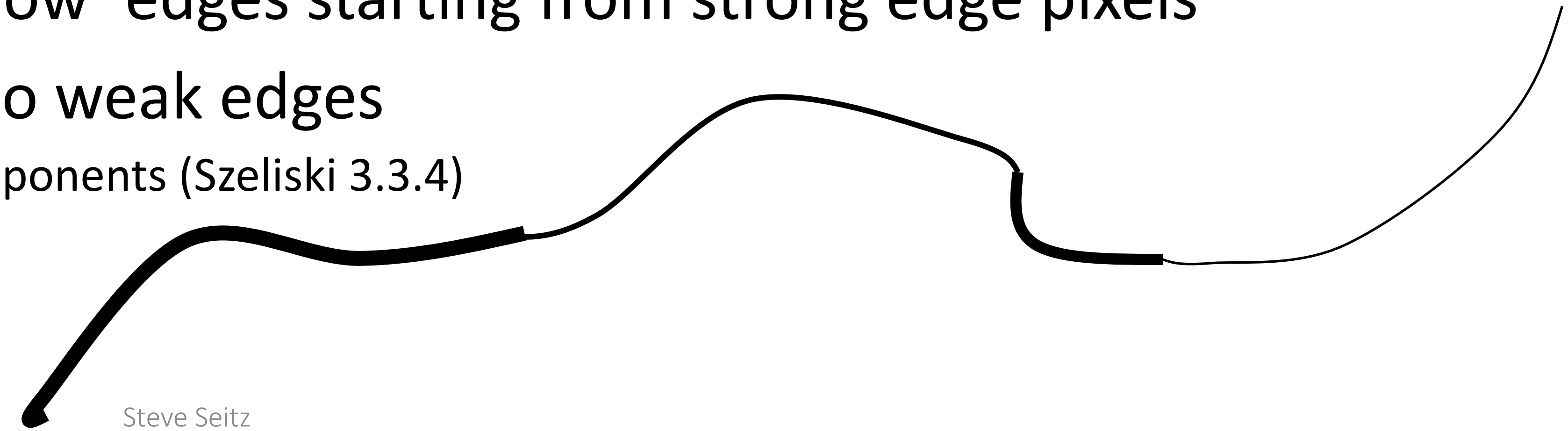
Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide “ridges” to single pixel width
4. ‘Hysteresis’ Thresholding

‘Hysteresis’ Thresholding

- Two thresholds – high and low
- Grad. mag. > high threshold? = strong edge
- Grad. mag. < low threshold? noise
- In between = weak edge

- Edge linking: ‘Follow’ edges starting from strong edge pixels
- Continue them into weak edges
    - Connected components (Szeliski 3.3.4)



Final Canny Edges

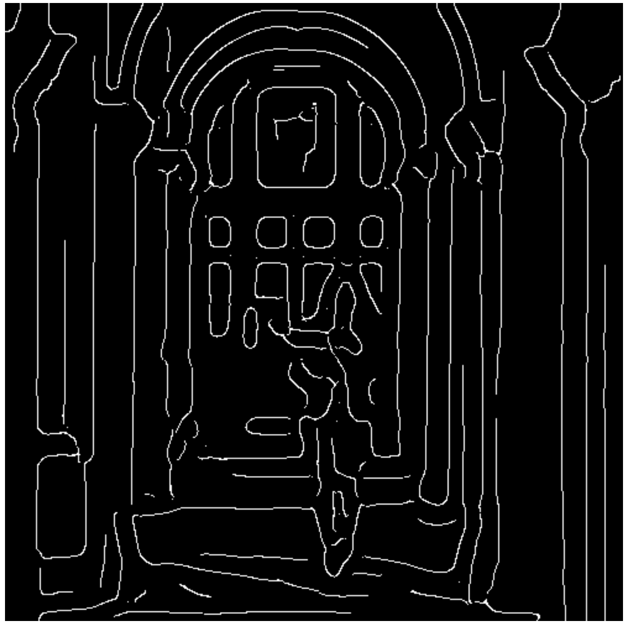
$$\sigma = \sqrt{2}, t_{low} = 0.05, t_{high} = 0.1$$

Steve Seitz

Effect of σ (Gaussian kernel spread/size)

Original

$\sigma = \sqrt{2}$



$\sigma = 4\sqrt{2}$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Steve Seitz

Canny Edge Detector

Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide “ridges” to single pixel width
4. ‘Hysteresis’ Thresholding:
   a. Define two thresholds: low and high
   b. Use the high threshold to start edge curves and the low threshold to continue them
   c. ‘Follow’ edges starting from strong edge pixels
      i. Connected components (Szeliski 3.3.4)

Demo: https://bigwww.epfl.ch/demo/ip/demos/edgeDetector/