Computer Vision

(Summer Semester 2020)

Lecture 4, Part 1

Edge Detection

Edges

- What types of edges do exist?
- Edges in noisy images
- Canny edge detector

- Note: The core of these slides stems from the class CSCI 1430: “Introduction to Computer Vision” by James Tompkin, Fall 2017, Brown University.

Low Level vs. *High* Level

Sensing

Data Representation

Edges

Corners

Descriptors

Camera Calibration

Alignment of Multi-view stereo

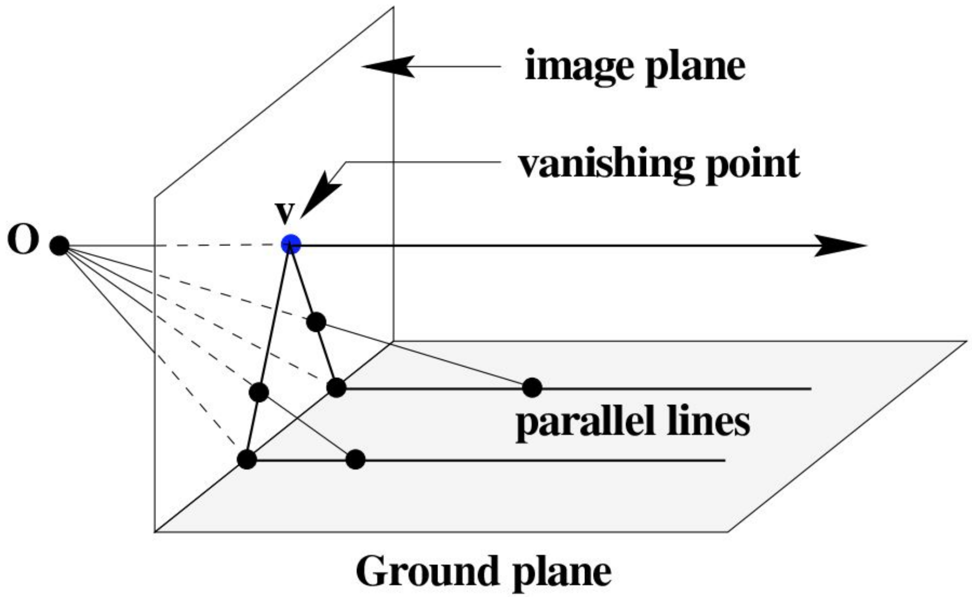
3D Reconstruction

Low

High

Vanishing Point

- Any 2 parallel lines (real world/ground plane) have same **vanishing point** in the image plane
- Ray OV, is parallel to the parallel lines
- There can be multiple vanishing points

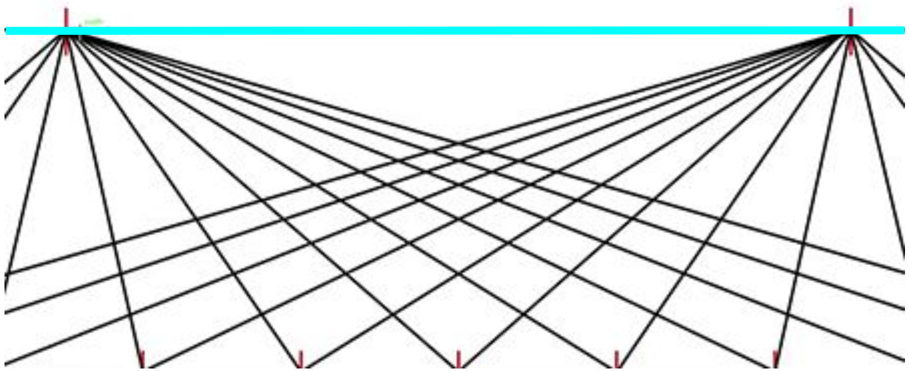


Vanishing Point

- Any 2 parallel lines (real world/ground plane) have same **vanishing point** in the image plane
- Ray OV, is parallel to the parallel lines
- There can be multiple vanishing points

Vanishing Lines

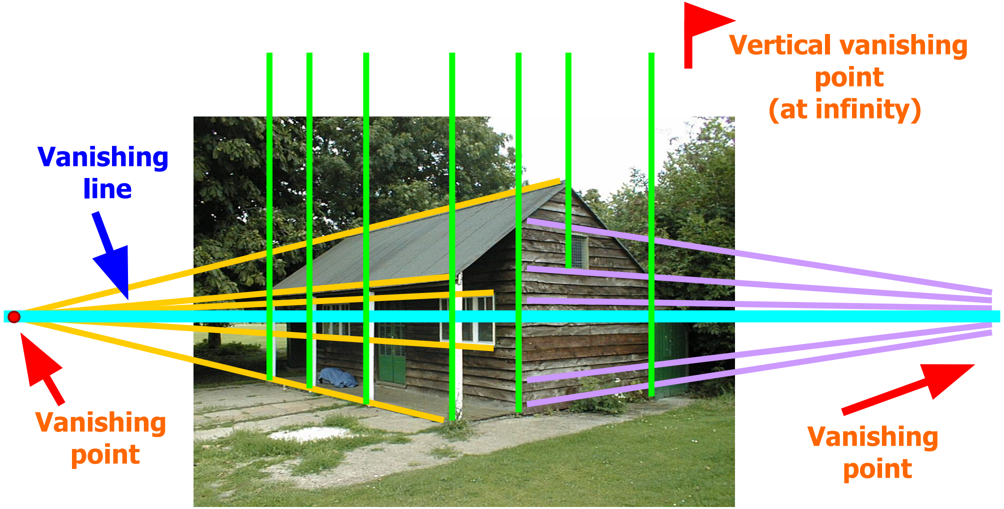
- Any set of parallel lines (ground plane) meet at a vanishing point
- Set of all the vanishing points form the horizon line OR the vanishing line
- Different planes define different vanishing lines



Edge detection (Szeliski 4.2)

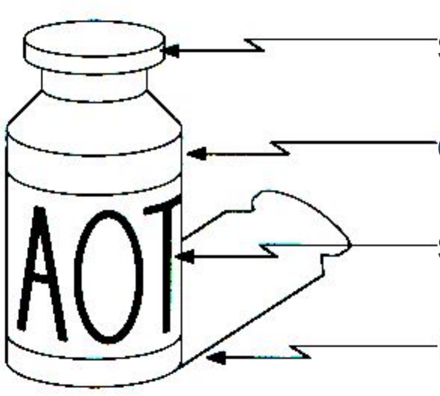
- **Goal:** Identify visual changes (discontinuities) in an image.

- **Why we care about edges?**
  - Recover viewpoint and geometry
  - Higher level vision tasks, e.g. for recognition



Origin of Edges

Edges are caused by a variety of factors



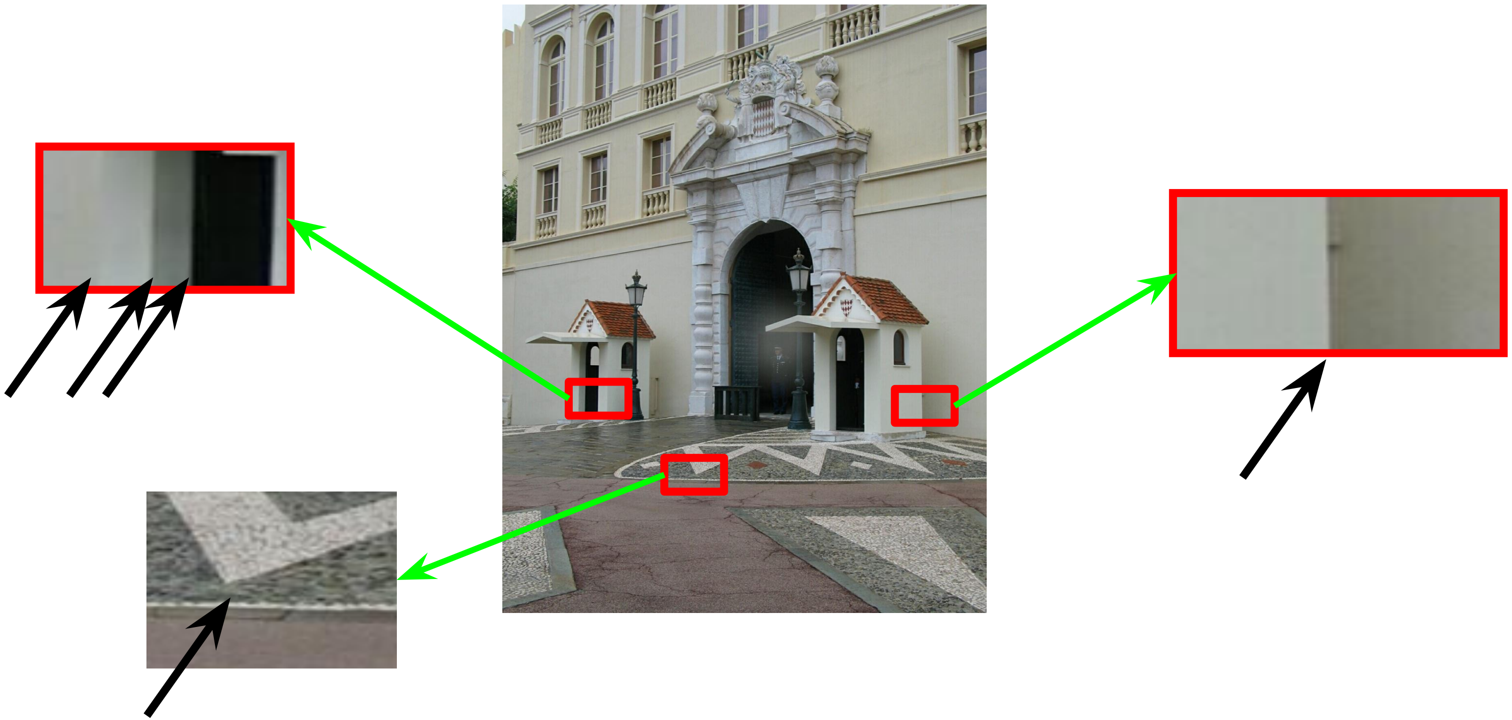
surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

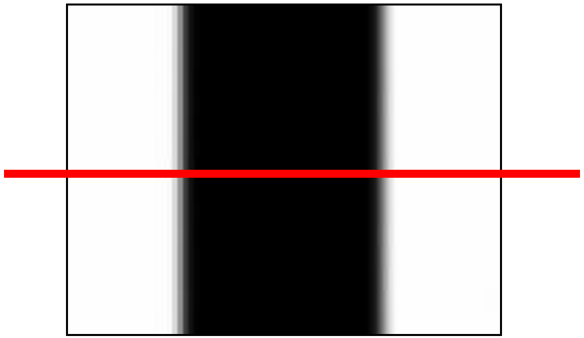
Closeup of edges



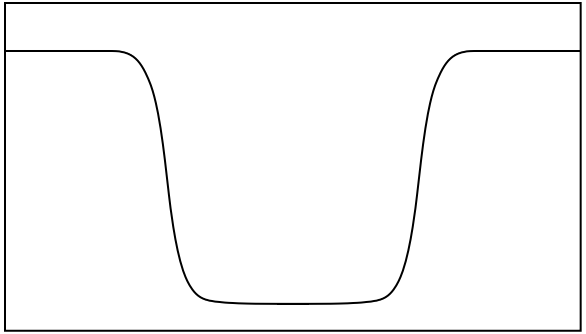
Characterizing edges

An edge is a place of rapid change in the image intensity function

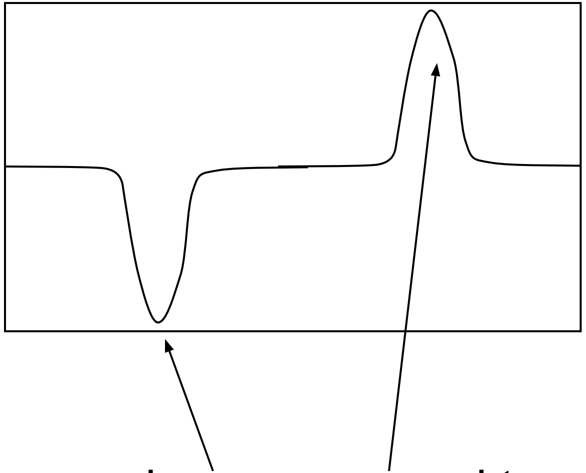
image



intensity function
(along horizontal scanline)

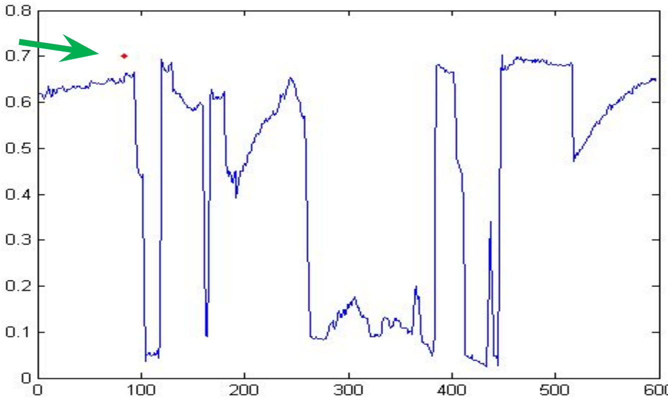


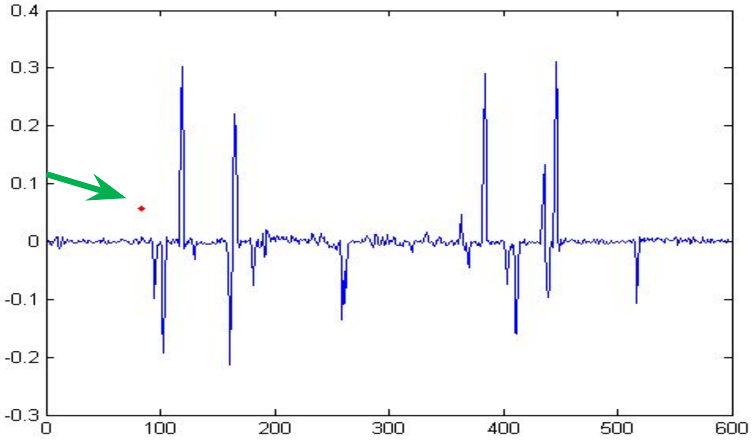
first derivative



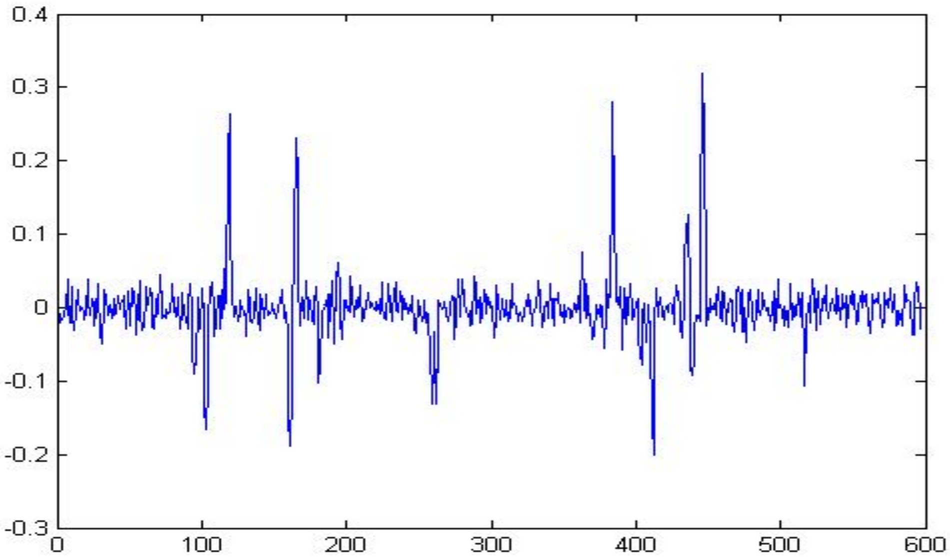
edges correspond to
extrema of derivative

Intensity profile





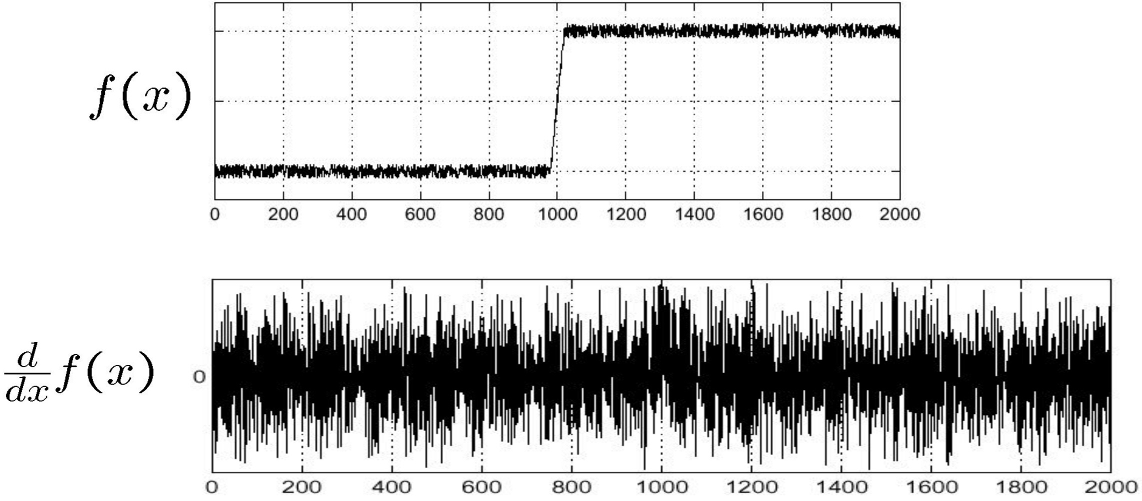
With a little Gaussian noise



Gradient

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



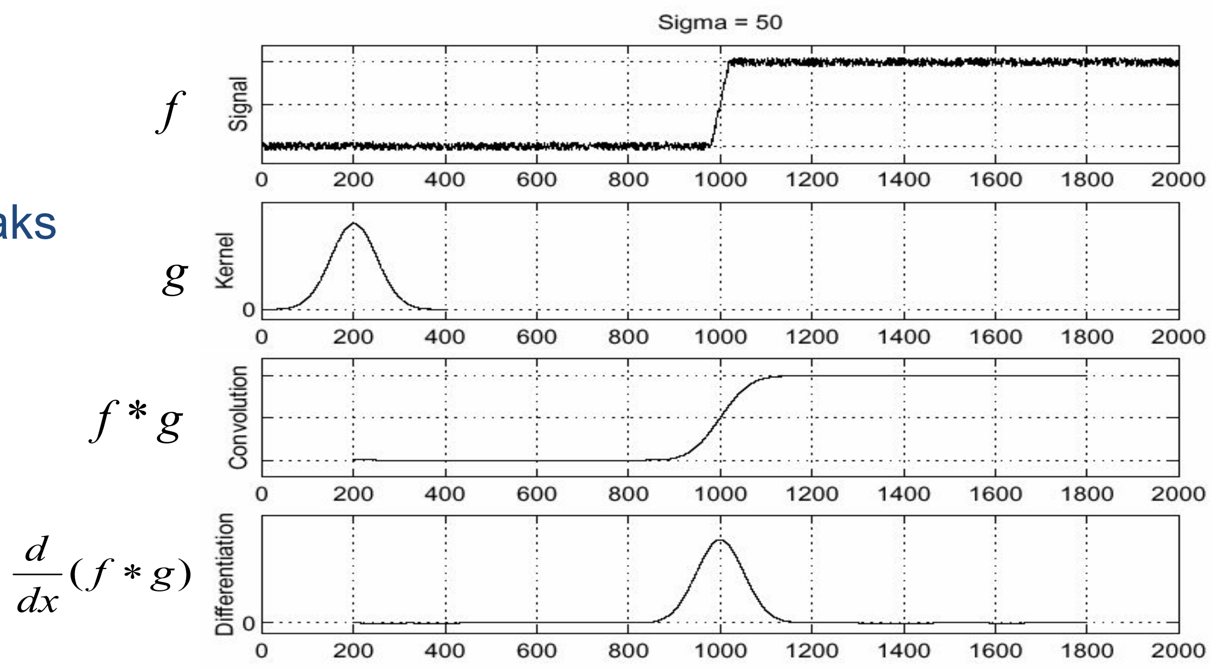
Where is the edge?

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Solution: smooth first

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



Derivative theorem of convolution

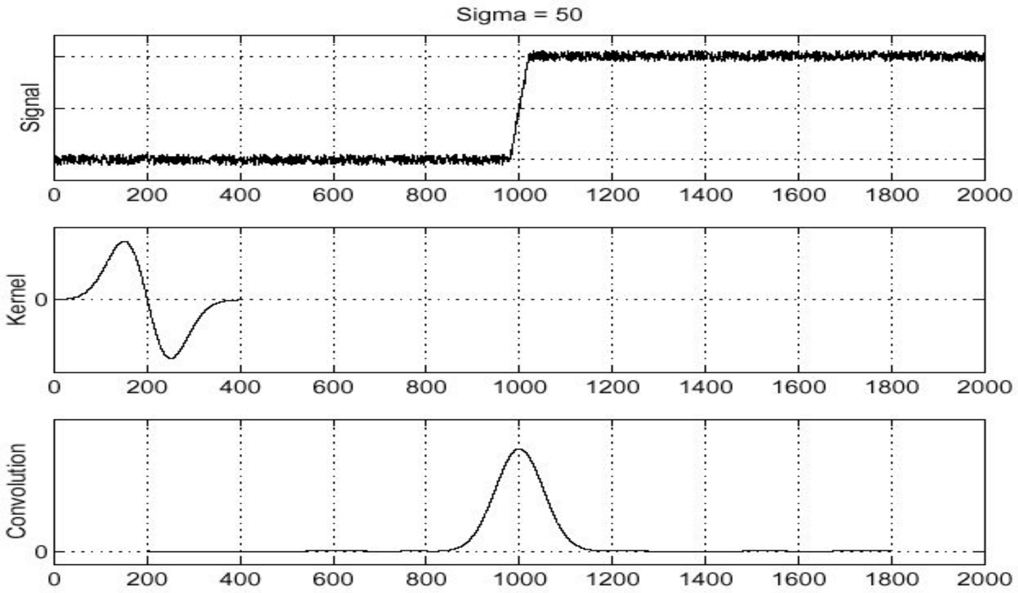
- Convolution is differentiable: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

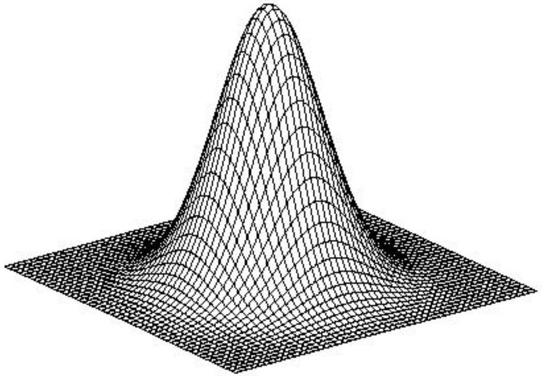
f

$\frac{d}{dx} g$

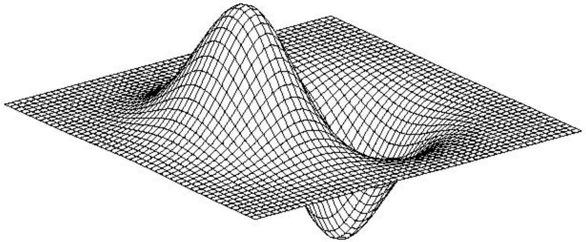
$f * \frac{d}{dx} g$



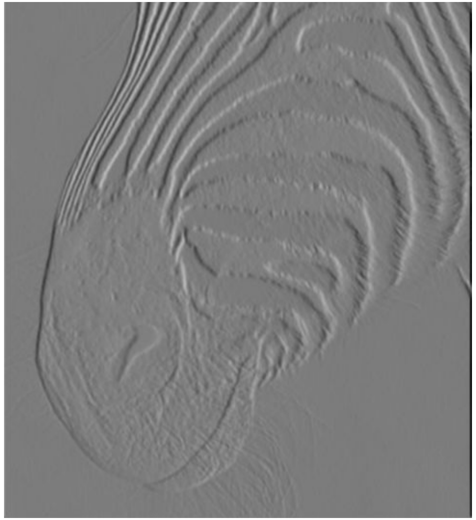
Derivative of 2D Gaussian filter



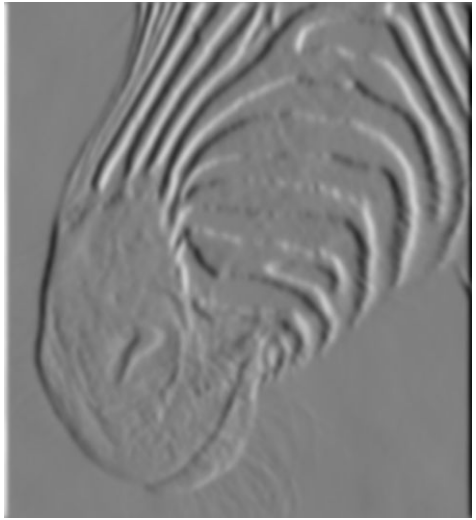
* [1 -1] =



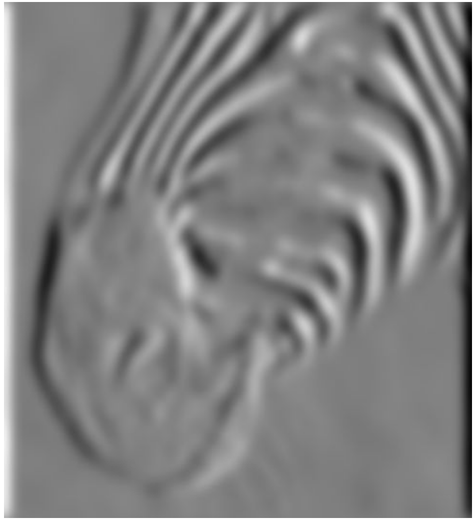
Tradeoff between smoothing and localization



1 pixel



3 pixels



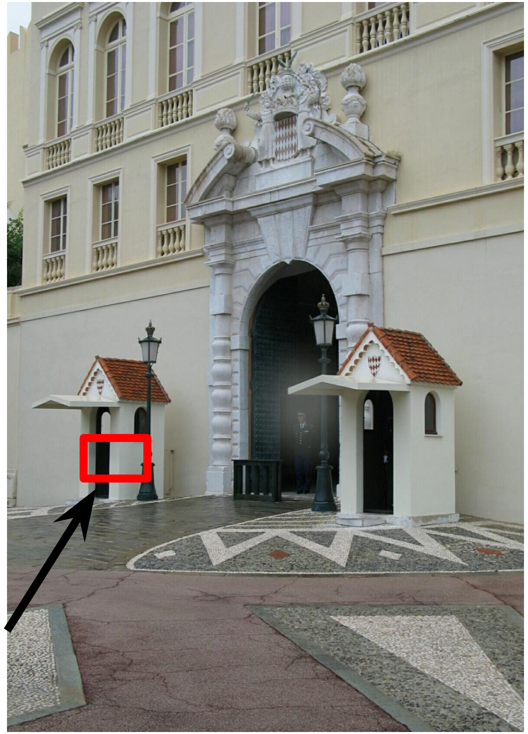
7 pixels

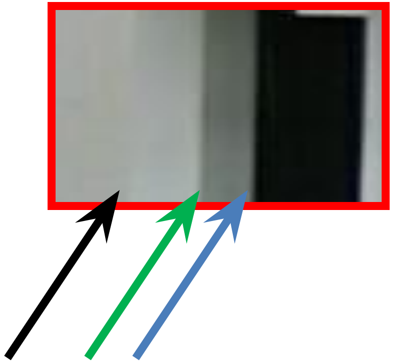
- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

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

Closeup of edges





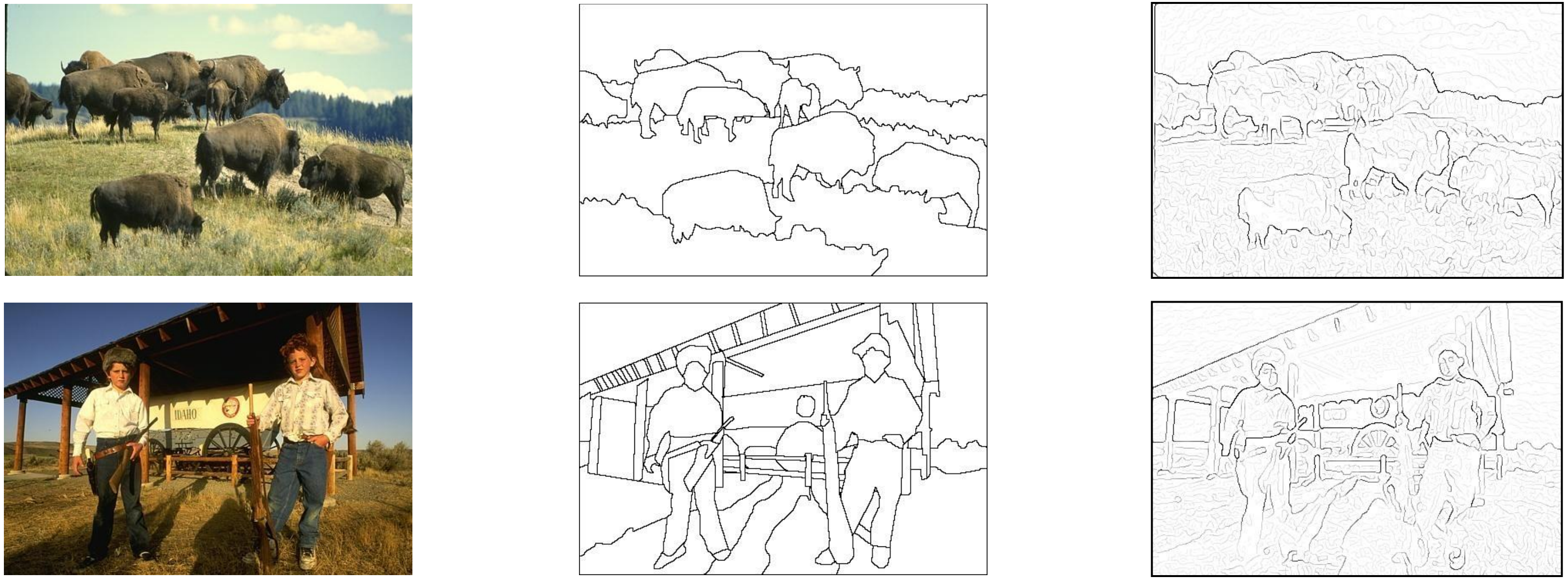
Designing an edge detector

- “All real edges”
  - We can aim to differentiate later on which edges are ‘useful’ for our applications.
  - If we can’t find all things which *could* be called an edge, we don’t have that choice.

- Is this possible?

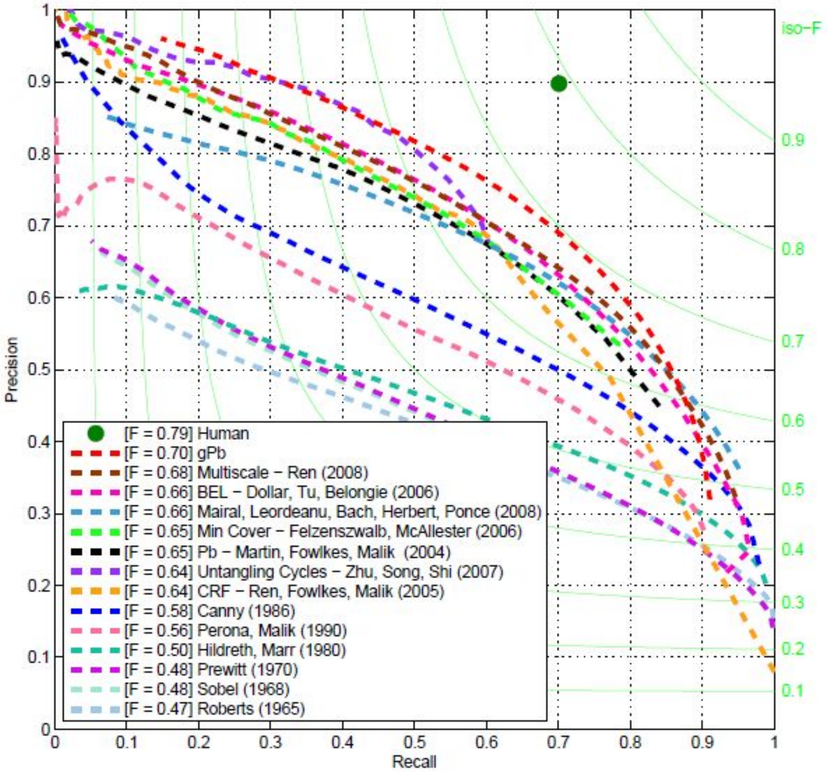
Where do humans see boundaries?

image | human segmentation | gradient magnitude



- Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

45 years of boundary detection



Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011

State of edge detection

- Local edge detection works well
  - ‘False positives’ from illumination and texture edges (depends on our application).
- Some methods to take into account longer contours
- Modern methods that actually “learn” from data. *[not in this class]*
- Poor use of object and high-level information.

Canny edge detector

- Probably the most widely used edge detector in computer vision.
- Theoretical model: step-edges corrupted by additive Gaussian noise.
- Canny showed that first derivative of Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization.

*J. Canny, **A Computational Approach To Edge Detection**, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.*

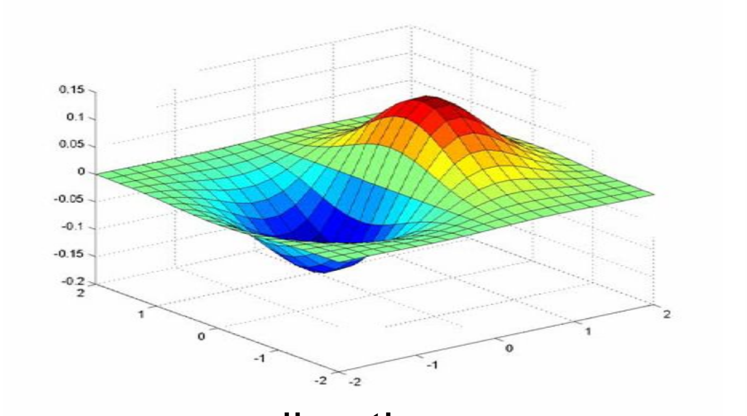
Demonstrator Image

```
rgb2gray('img.png')
```

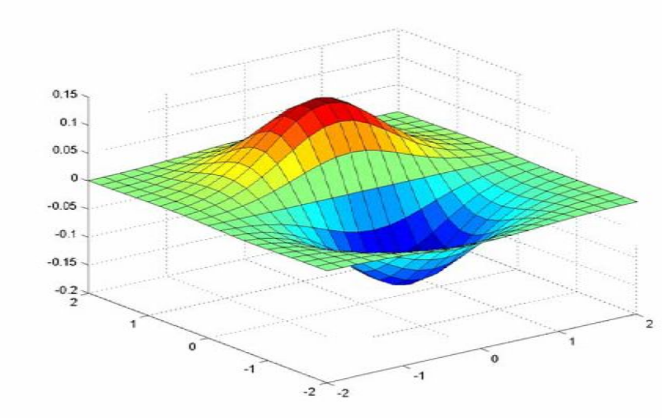
Canny edge detector

1. Filter image with x, y derivatives of Gaussian

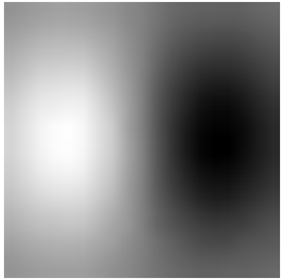
Derivative of Gaussian filter

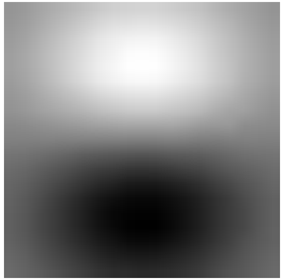


x-direction



y-direction

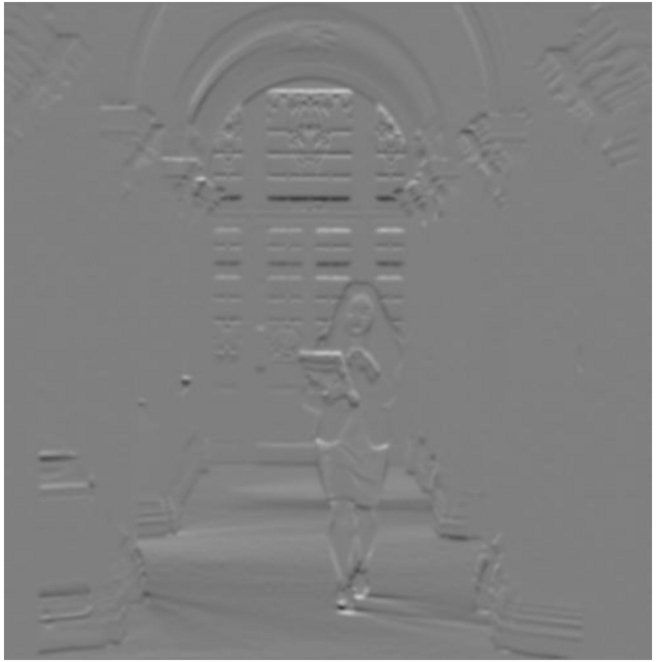




Compute Gradients

X Derivative of Gaussian

Y Derivative of Gaussian



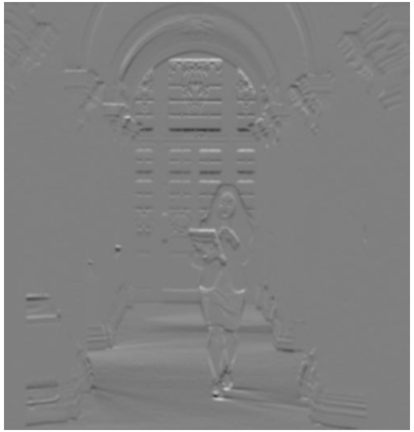
(x2 + 0.5 for visualization)

Canny edge detector

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

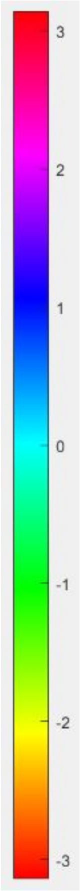


(x4 for visualization)

Compute Gradient Orientation

- Threshold magnitude at minimum level
- Get orientation via **theta = atan2(gy, gx)**

Thresholded:



Compute Gradient Orientation

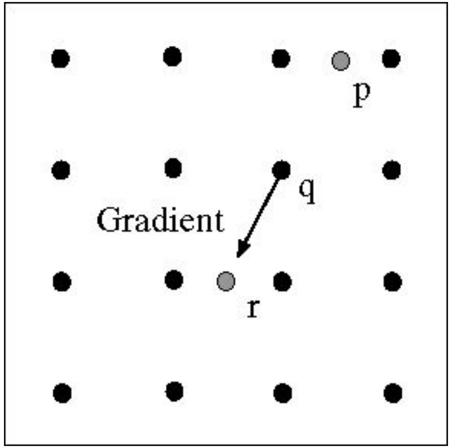
- Threshold magnitude at minimum level
- Get orientation via **theta = atan2(gy, gx)**

Unthresholded:

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - 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)

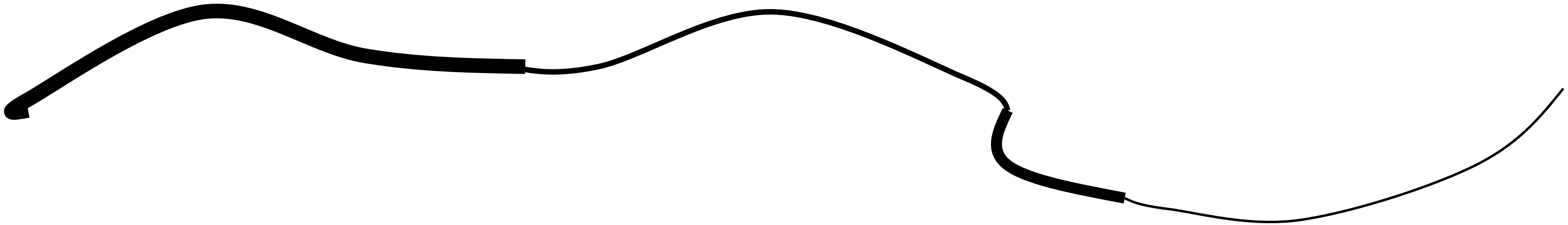
Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - 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

- 'Follow' edges starting from strong edge pixels
- Continue them into weak edges
  - Connected components (Szeliski 3.3.4)



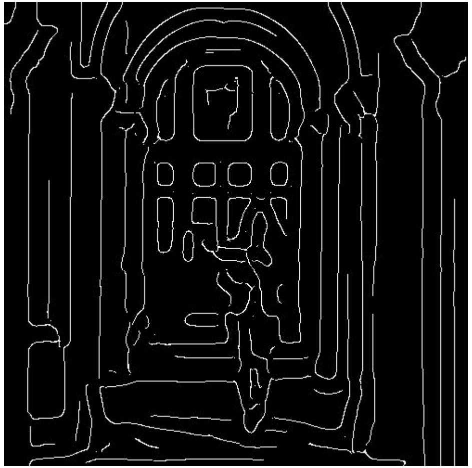
Final Canny Edges

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

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

Canny edge detector

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

MATLAB: edge(image, ‘canny’)