Edges

Semester 2, 2021

Kris Ehinger

Review: Filters

Outline

- Basics of edge detection
- Canny algorithm
- Edges for image recognition

Learning outcomes

- Explain the causes of edges in images
- Explain the steps involved in Canny edge detection
- Evaluate computer vision features in terms of their invariance (or tolerance) to image variation

Edge detection

Causes of edges

- Edges are caused by a variety of factors:

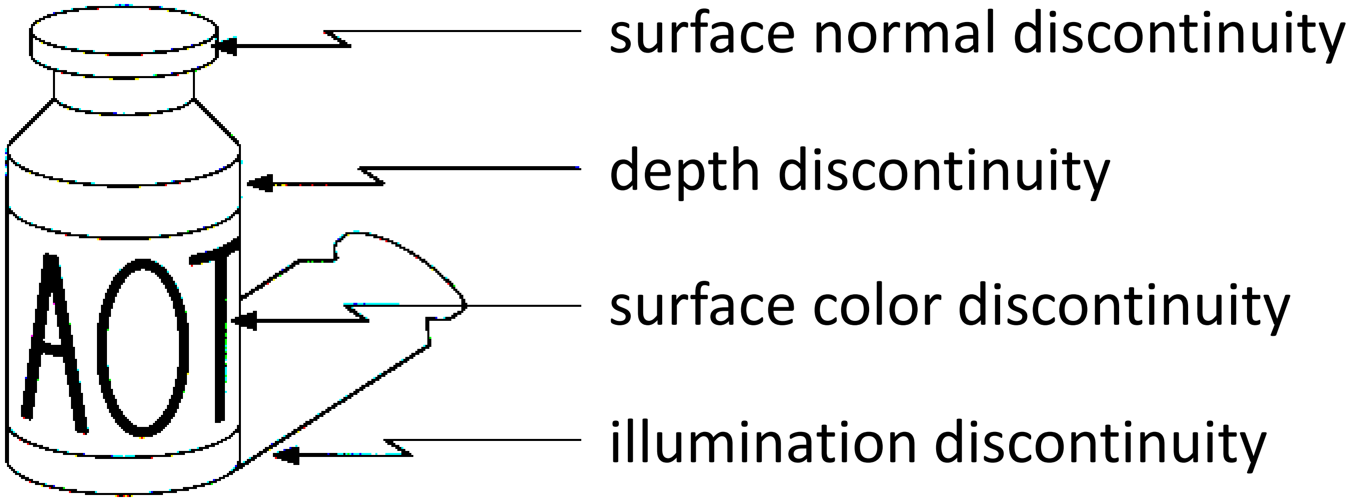
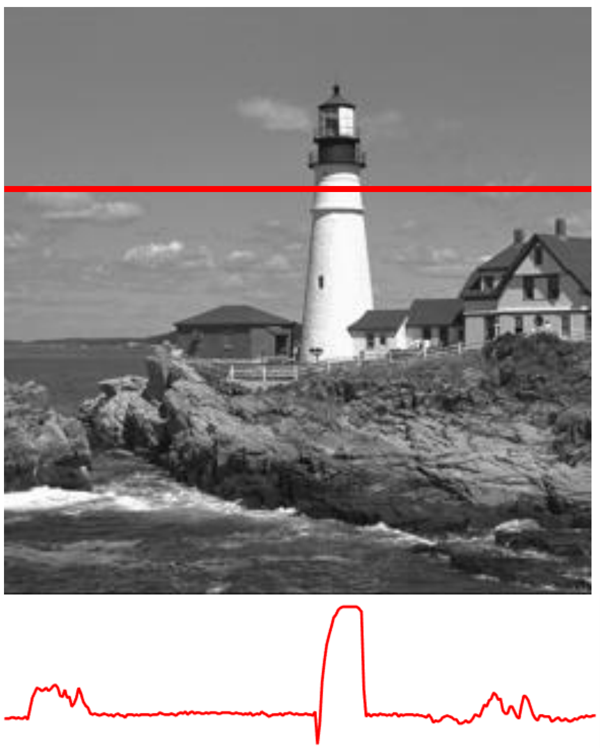


Image: S. Seitz

Photo: A. Adams (1968)

Characterising edges



Characterising edges

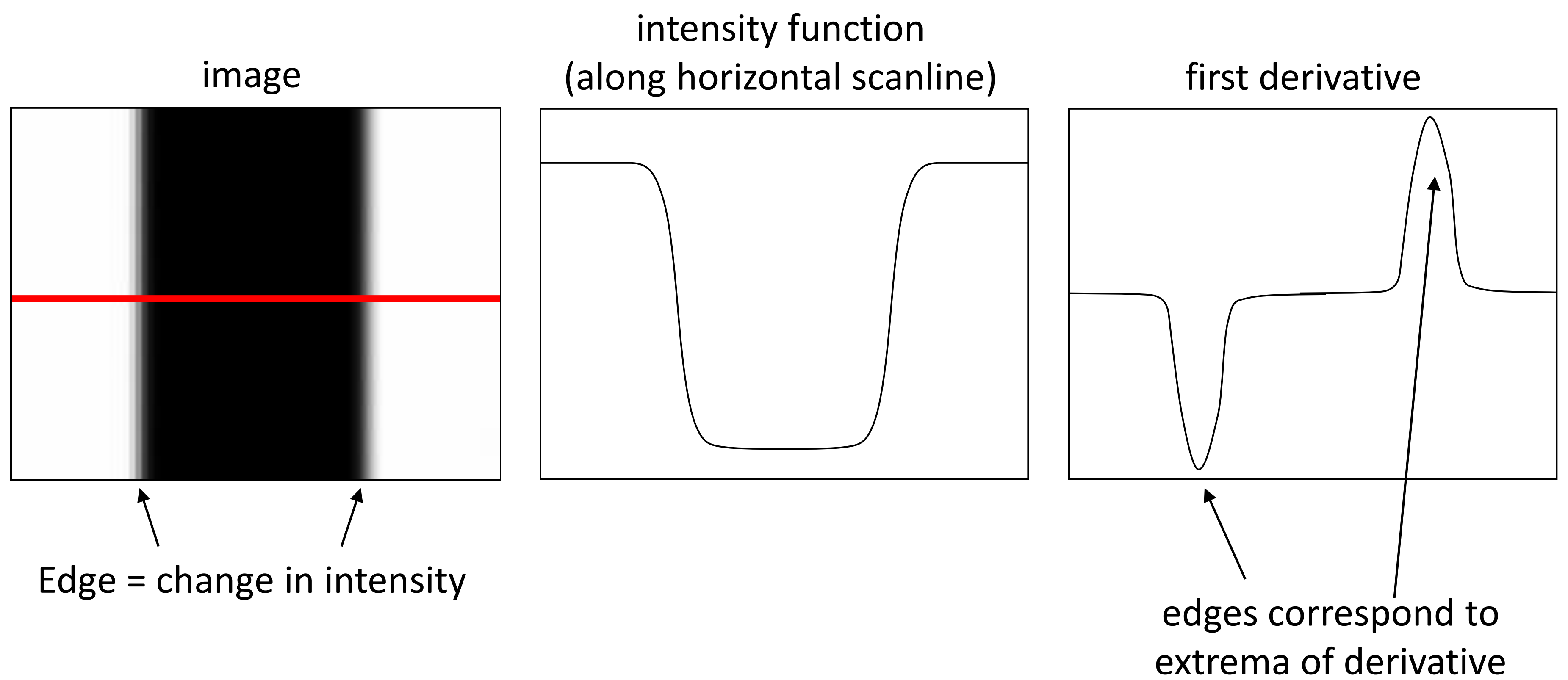
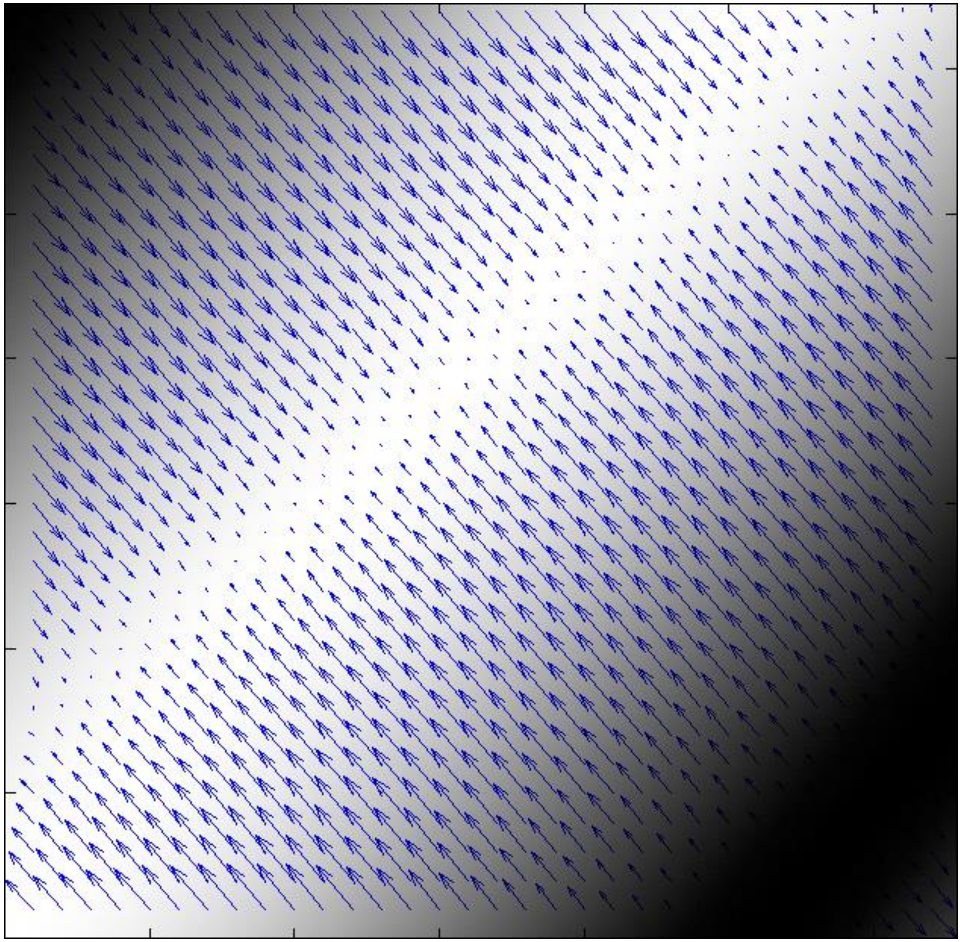


Figure: D. Hoiem

Gradient



Gradient

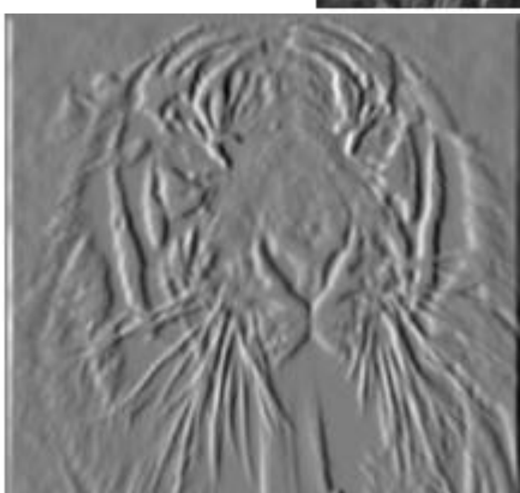
- Gradient of a function over x,y:
- $\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j}$

$\mathbf{i}$ = unit vector in the x direction
$\mathbf{j}$ = unit vector in the y direction

- Gradient at a single point (x,y) is a vector:
  - Direction is the direction of maximum slope:
  - $\theta = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$
  - Length is the magnitude (steepness) of the slope
  - $\|\nabla f\| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$

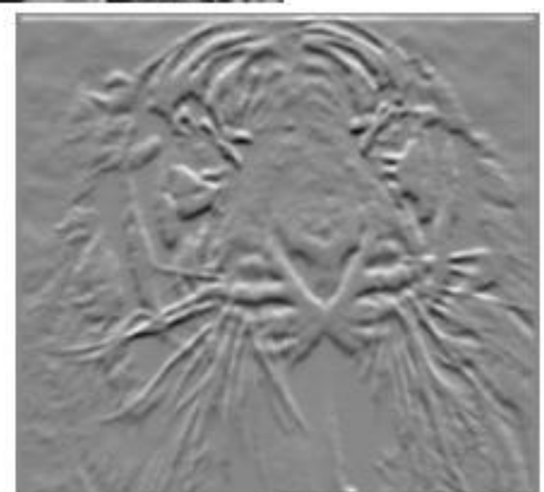
Partial derivatives in x, y

$$\frac{\partial f(x,y)}{\partial x}$$



-1	1

$$\frac{\partial f(x,y)}{\partial y}$$



-1
1

or

1
-1

Figure: D. Hoiem

Issue: noise

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

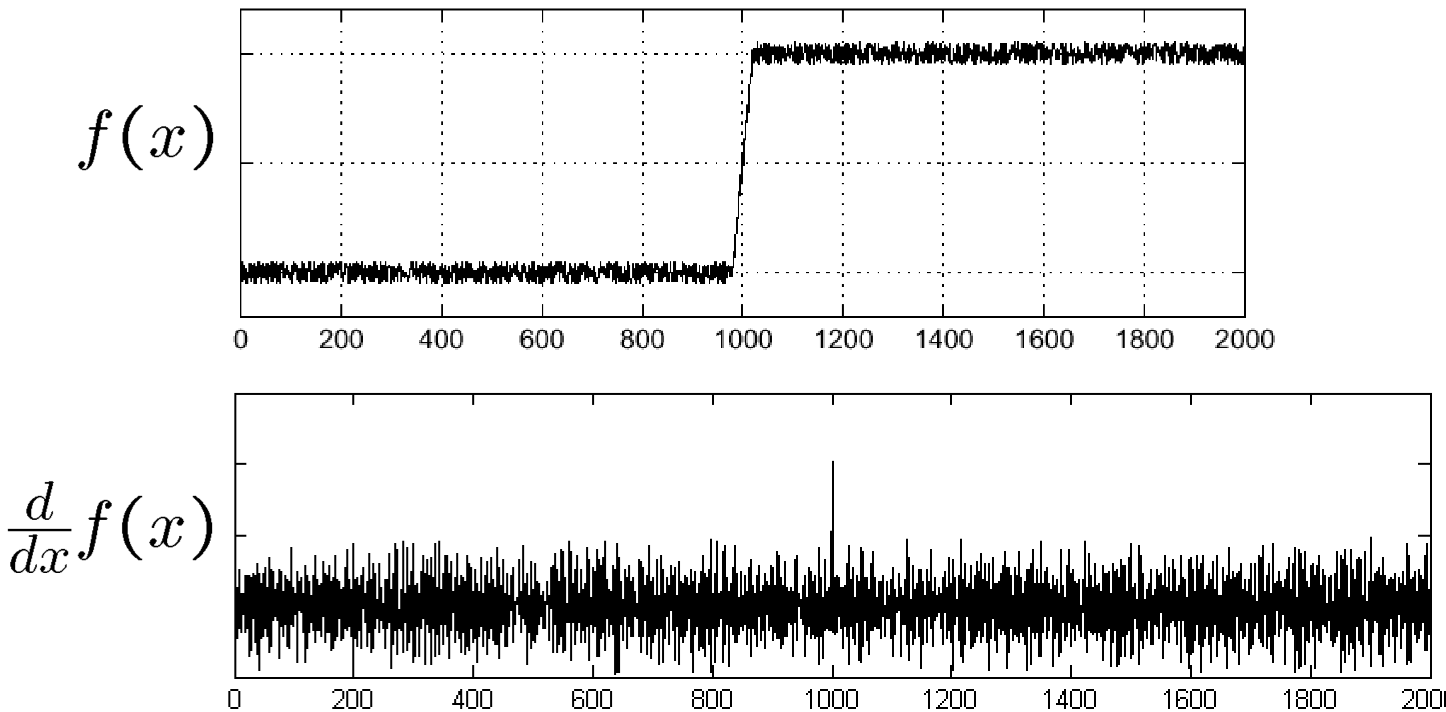


Figure: S. Seitz

Solution: smooth (blur) first

f

g

$f * g$

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

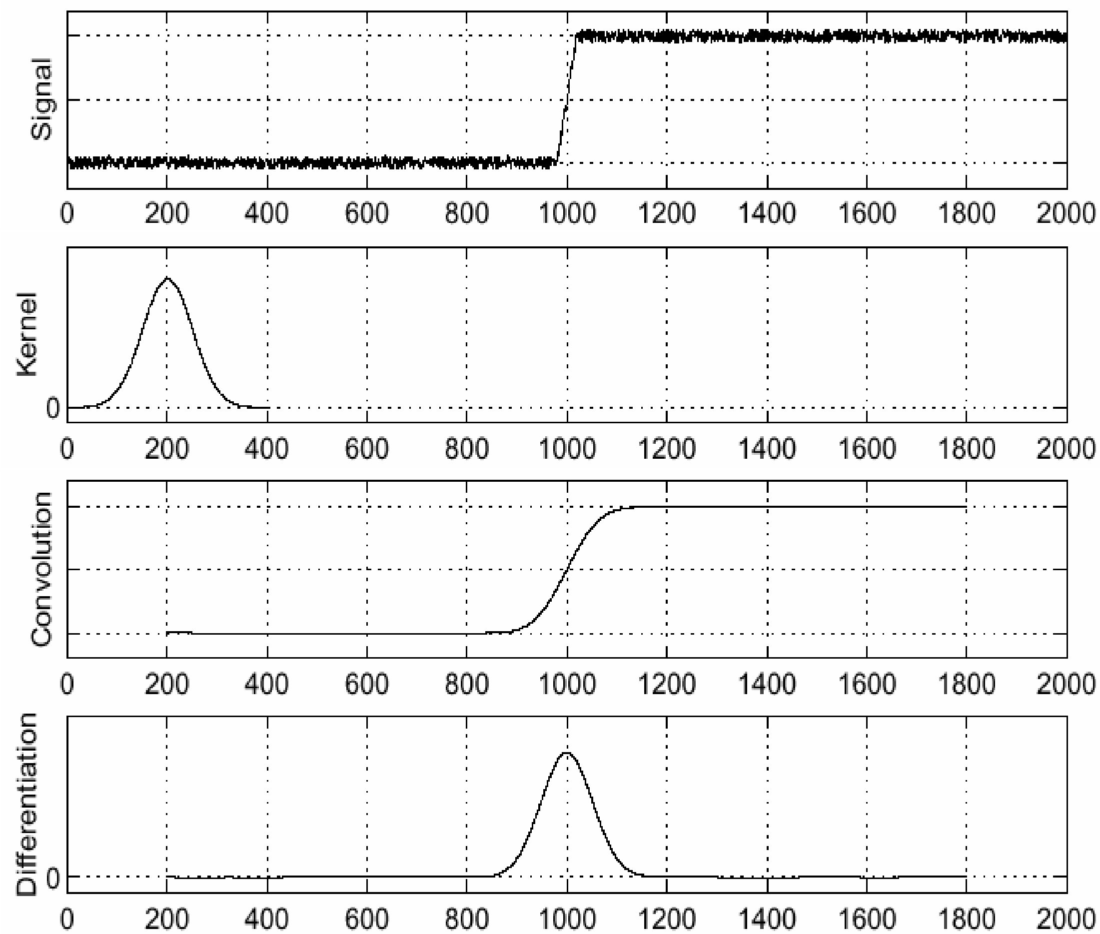


Figure: S. Seitz

More efficient solution

Associative property of convolution: $$\frac{\partial}{\partial x}(f * g) = f * \frac{\partial}{\partial x} g$$

f

$\frac{d}{dx} g$

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

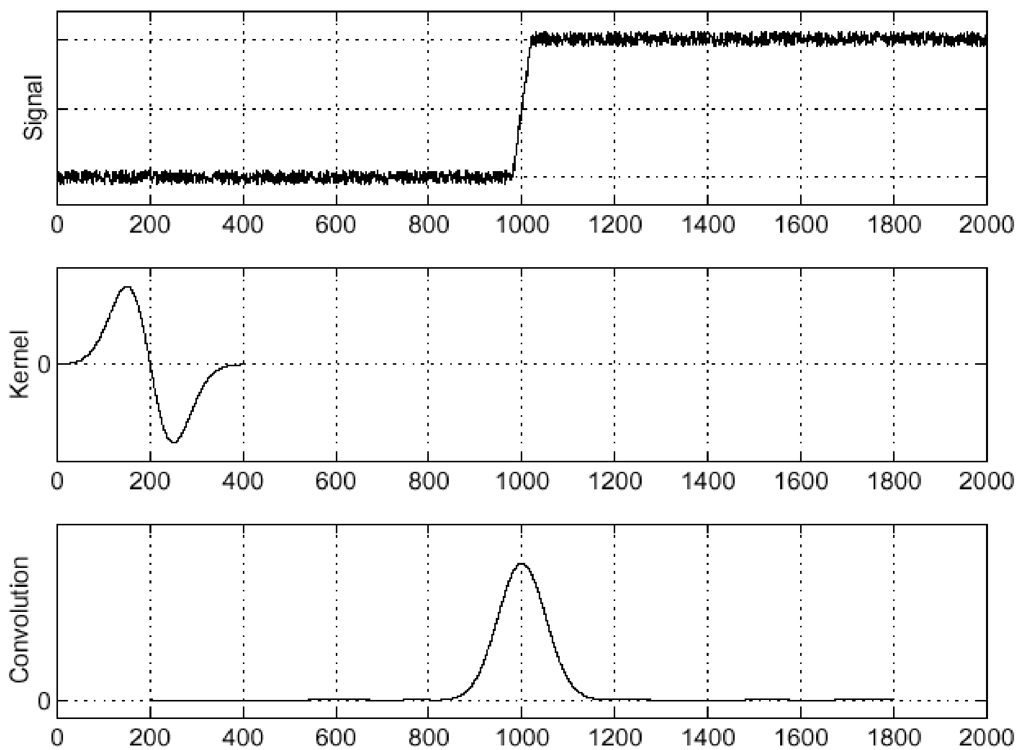
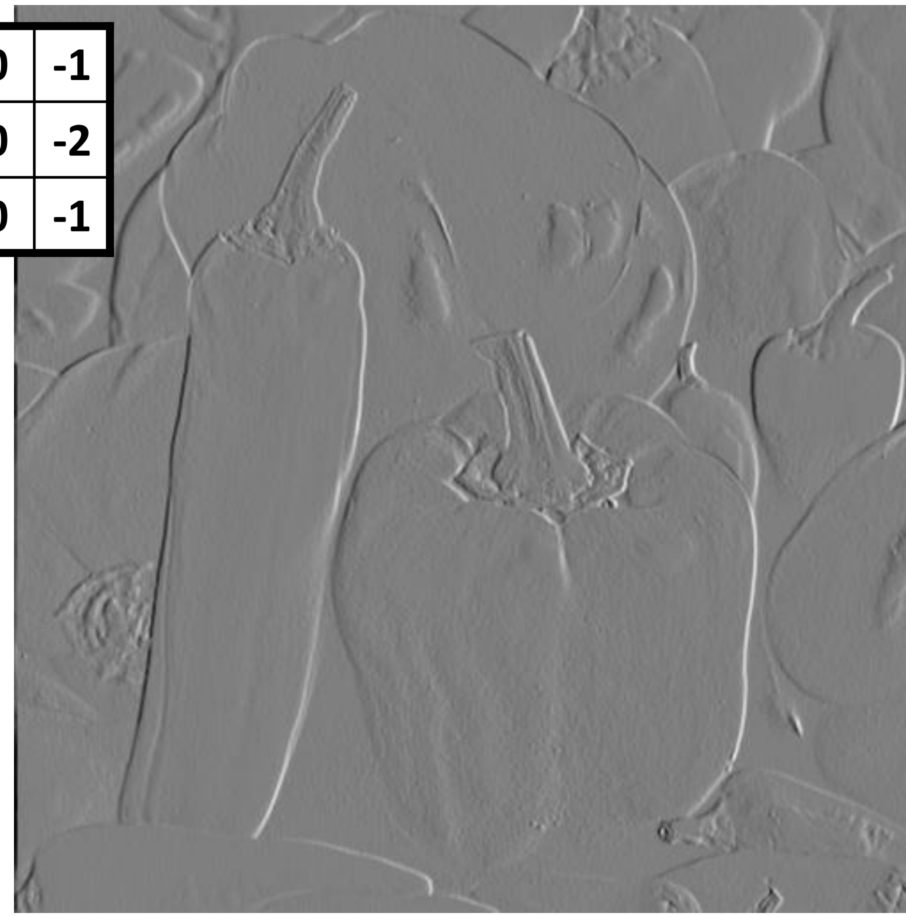


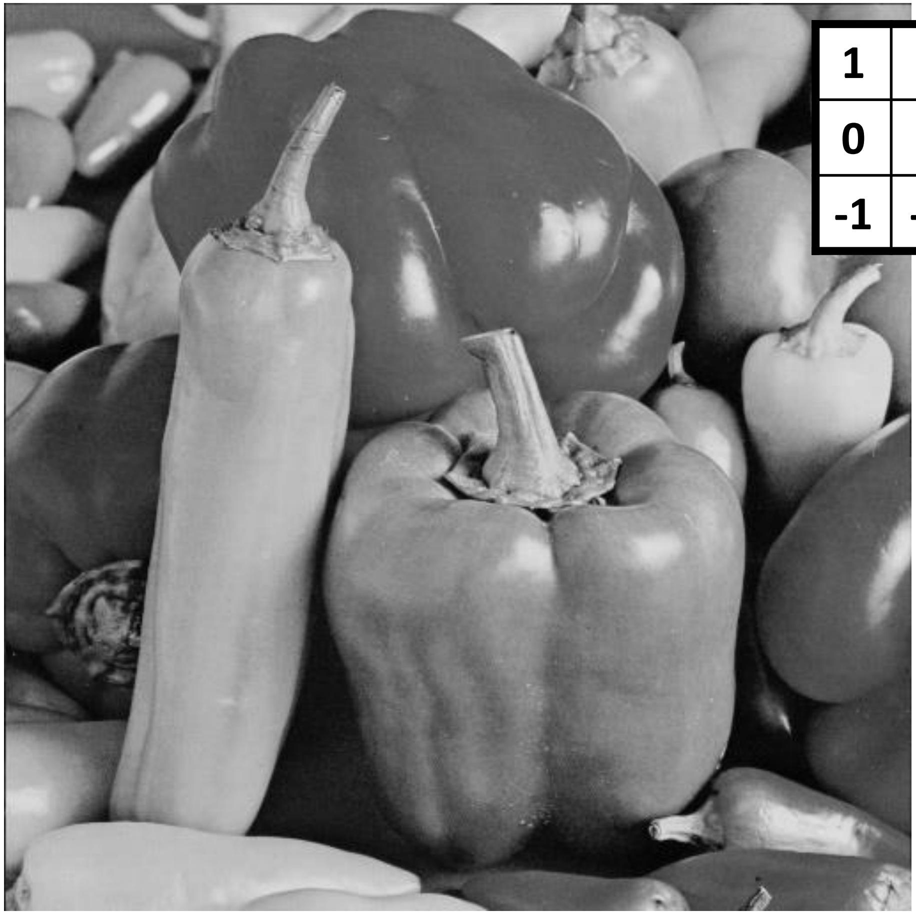
Figure: S. Seitz

Edge filter: Sobel

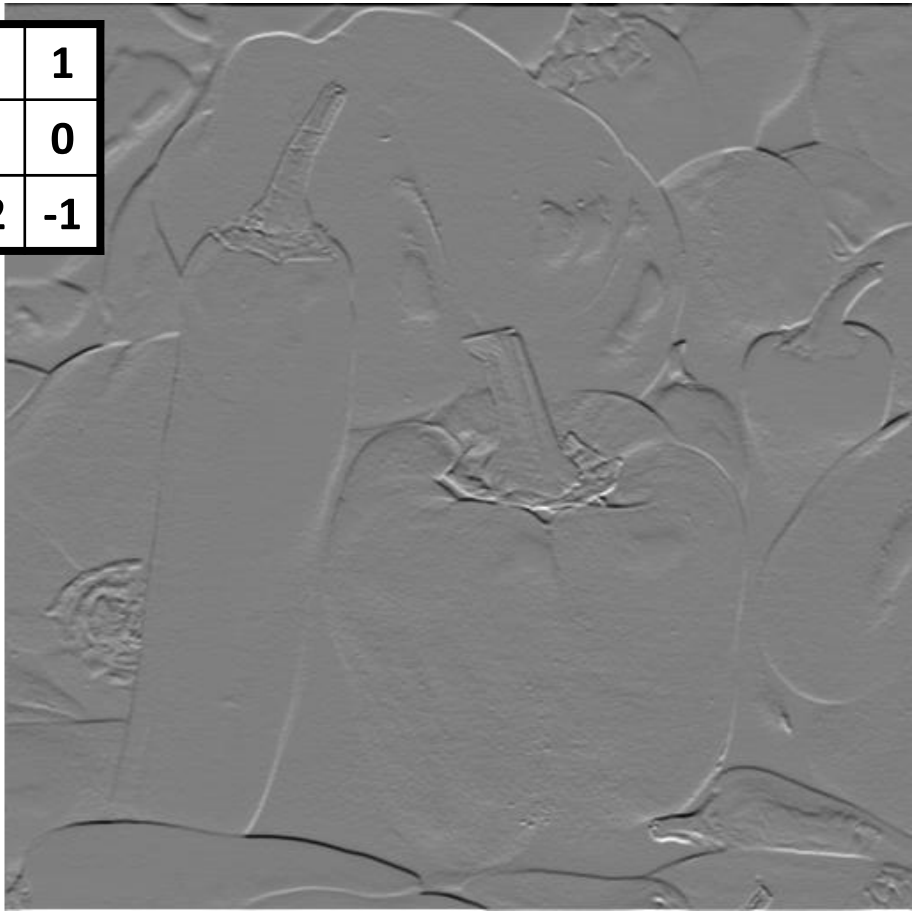
1	0	-1
2	0	-2
1	0	-1



Edge filter: Sobel



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



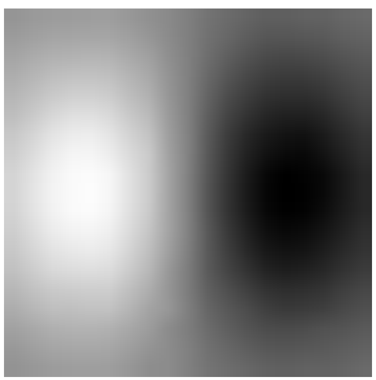
Edge filters

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

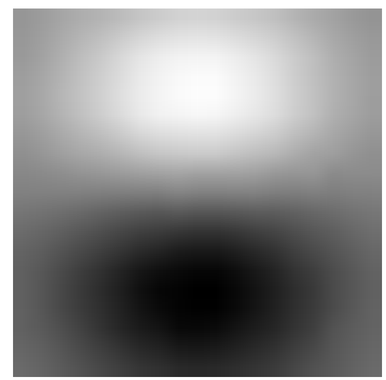
Sobel (x)

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

Sobel (y)



Derivative of Gaussian (x)



Derivative of Gaussian (y)

Summary

- Edges are points in the image with a high change in intensity = high change in gradient
- Accurate edge detection requires smoothing image noise
- Edge detector = derivative of Gaussian filter, combines smoothing and gradient response

Example: Canny edge detection

Canny edge detection

- Canny, *TPAMI* 1986: A Computational Approach to Edge Detection
- Foundational approach to edge detection
- Detect edges based on image gradient, then do additional processing to improve the edge map

Edge detection

- Filter with derivative-of-Gaussian filters
- Get magnitude, orientation of all the edges

- You really only need two oriented filters (dx and dy)

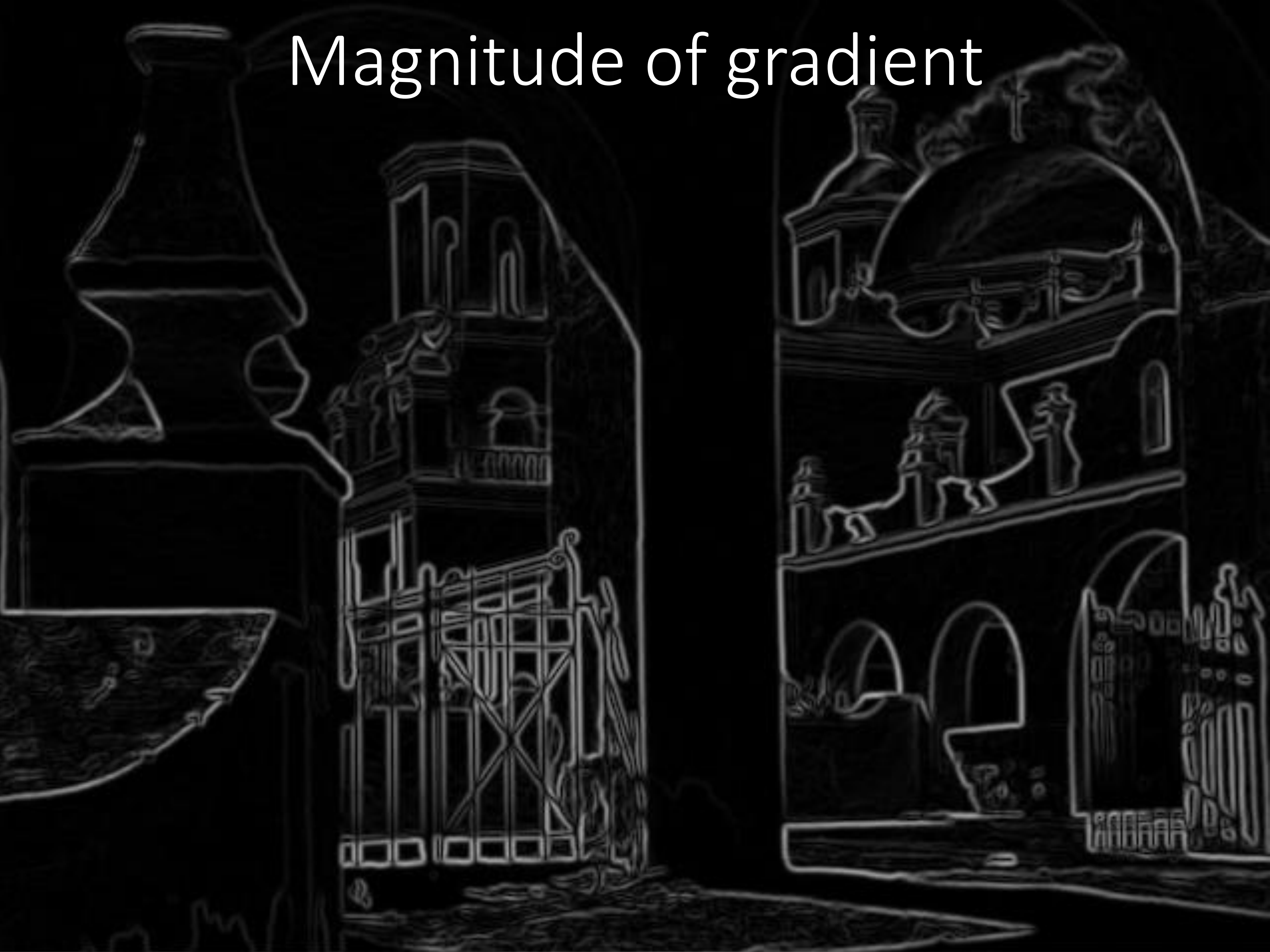
$$\frac{\partial}{\partial\theta} = \cos(\theta)\frac{\partial}{\partial x} + \sin(\theta)\frac{\partial}{\partial y}$$

Photo: A. Adams (1968)

Derivative in x

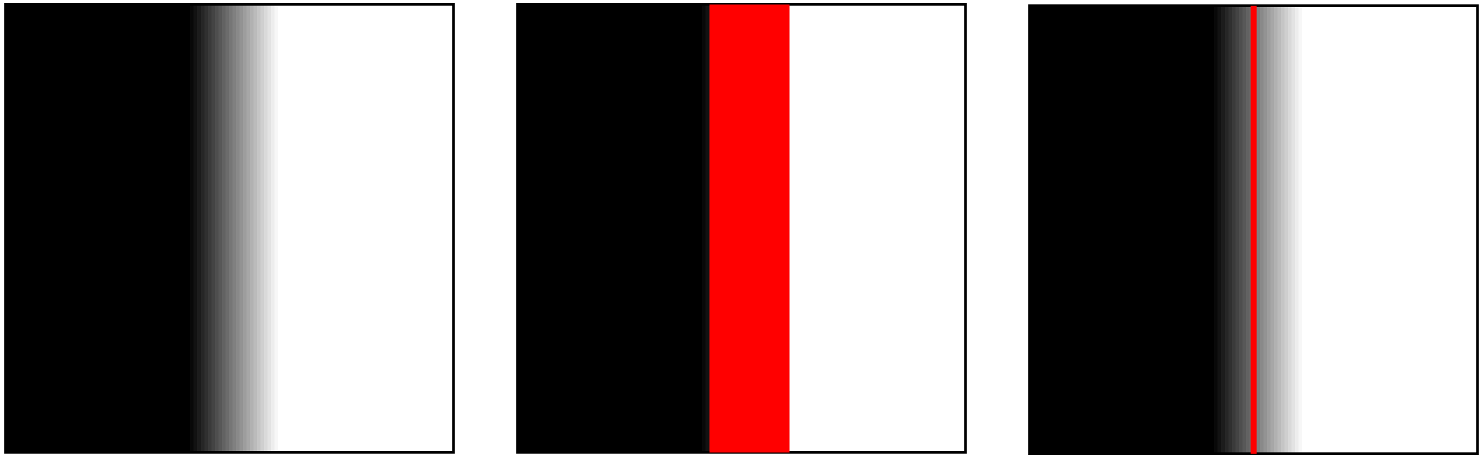
Derivative in y

Magnitude of gradient



Problem: multiple edges

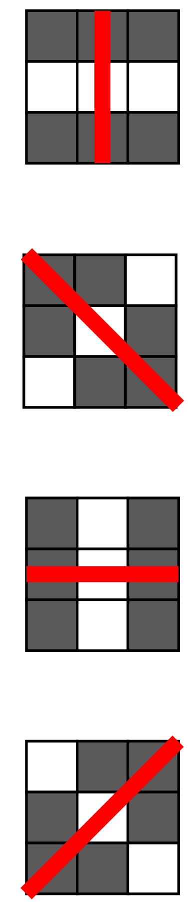
- How many vertical edges?



- Non-maximum suppression: If nearby pixels claim to be part of the same edge, only keep the one with maximum gradient

Non-maximum suppression

- Bin edges by orientation
- For each edge pixel,
  - Check the two neighbour pixels orthogonal to this edge pixel
  - If either neighbour has same edge orientation AND higher magnitude, this pixel is not an edge

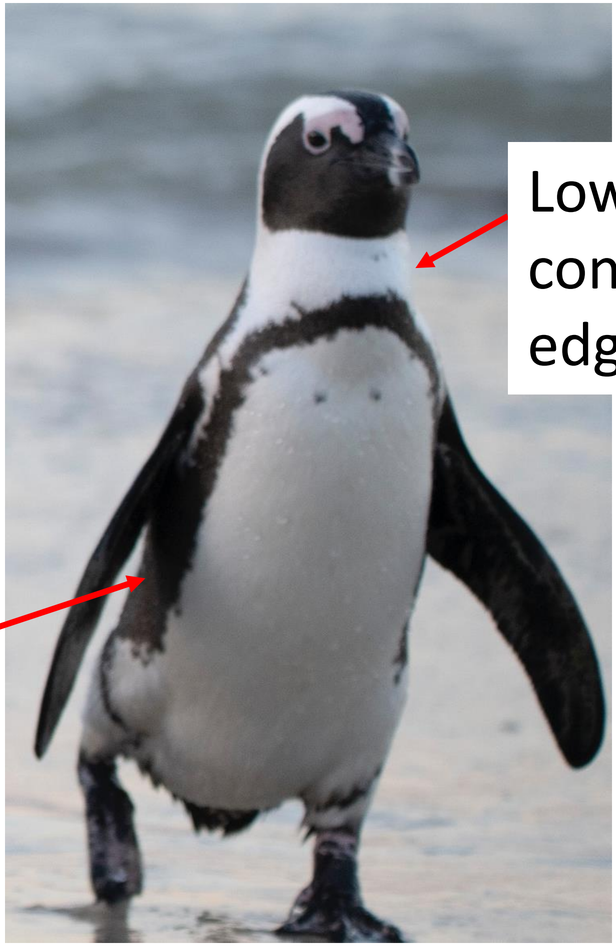


After non-max suppression

Edge orientation

Thresholding with hysteresis

- Magnitude threshold?
- No single threshold will work: use hysteriesis



Low-contrast edge

Low-contrast shadow

Thresholding with hysteresis

- Two thresholds T_1, T_2 with $T_1 > T_2$
- Strong edges: magnitude > T_1
- Weak edges: T_1 > magnitude > T_2
- For each weak edge:
  - Check the 8-pixel neighbourhood around this pixel
  - If any neighbour is a strong edge, relabel the weak edge pixel as a strong edge
- Final edge map = strong edges

After thresholding

Summary

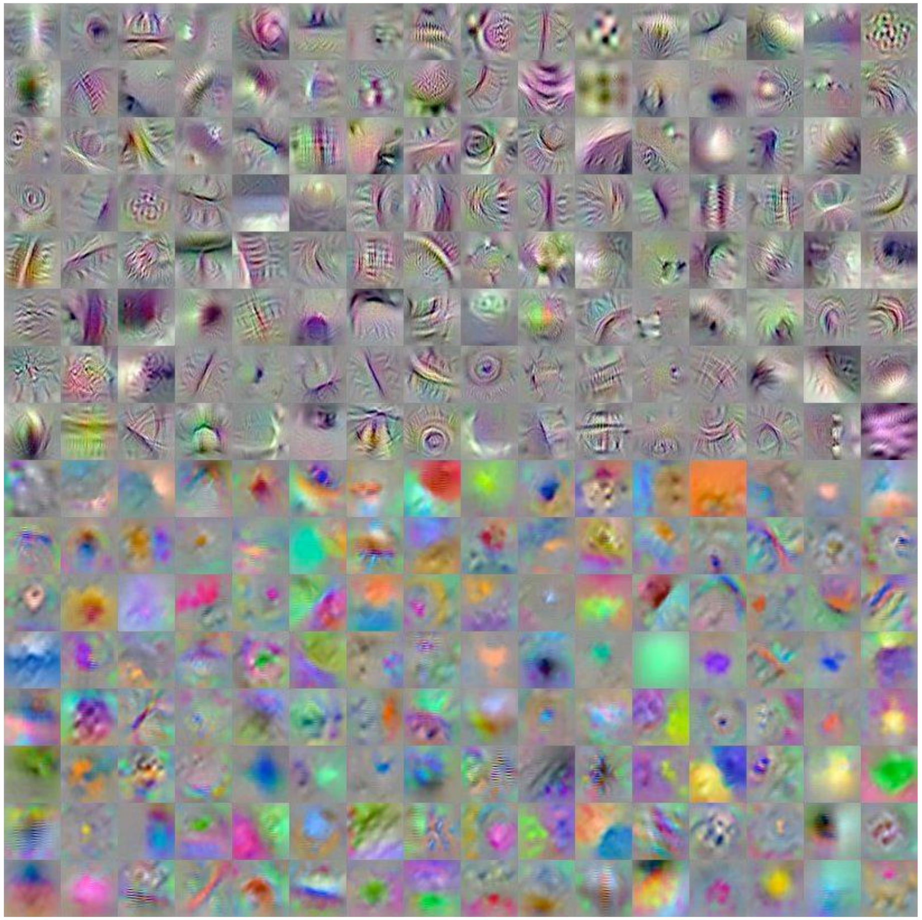
- Canny edge detector: commonly-used algorithm to detect edges in images
- Defines edges based on image gradient
- Post-processing of gradient to better localise edges (non-maximum suppression) and preserve faint/broken edges (thresholding with hysteresis)

Edges for image recognition

Edge features in neural networks

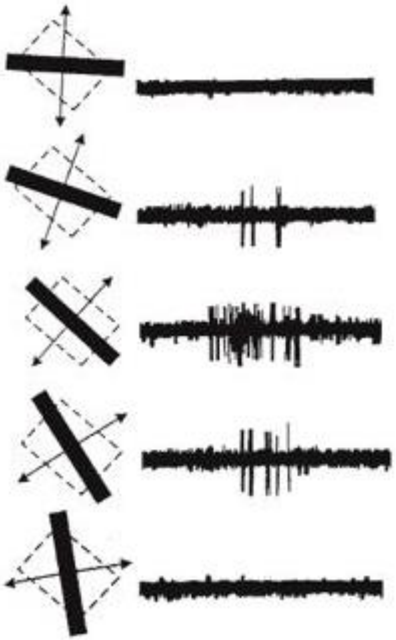
Convolutional layer 1

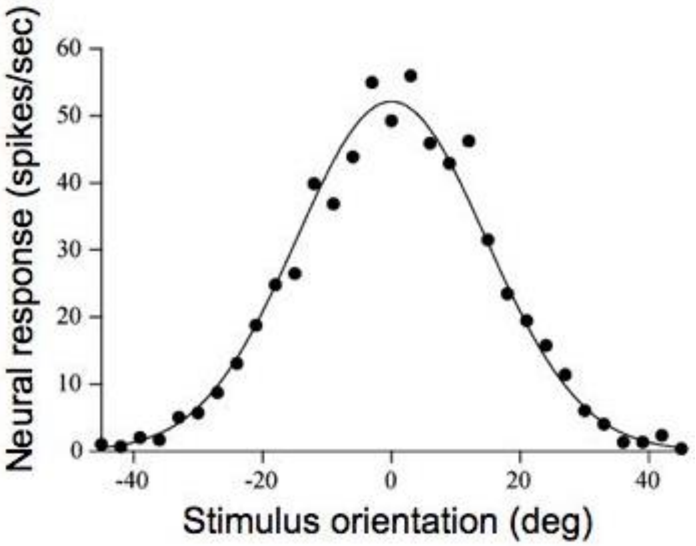
Convolutional layer 2 →



Yosinski, Clune, Nguyen, Fuchs, & Lipson (2015)

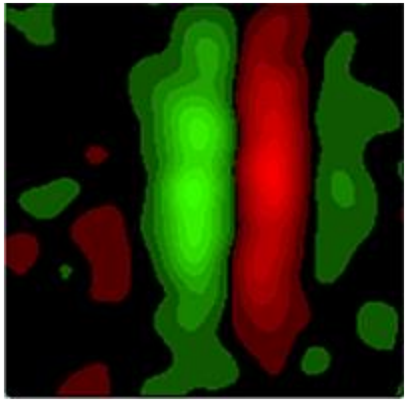
Why edges?





Hubel & Wiesel (1968)

Receptive field
V1 simple cell



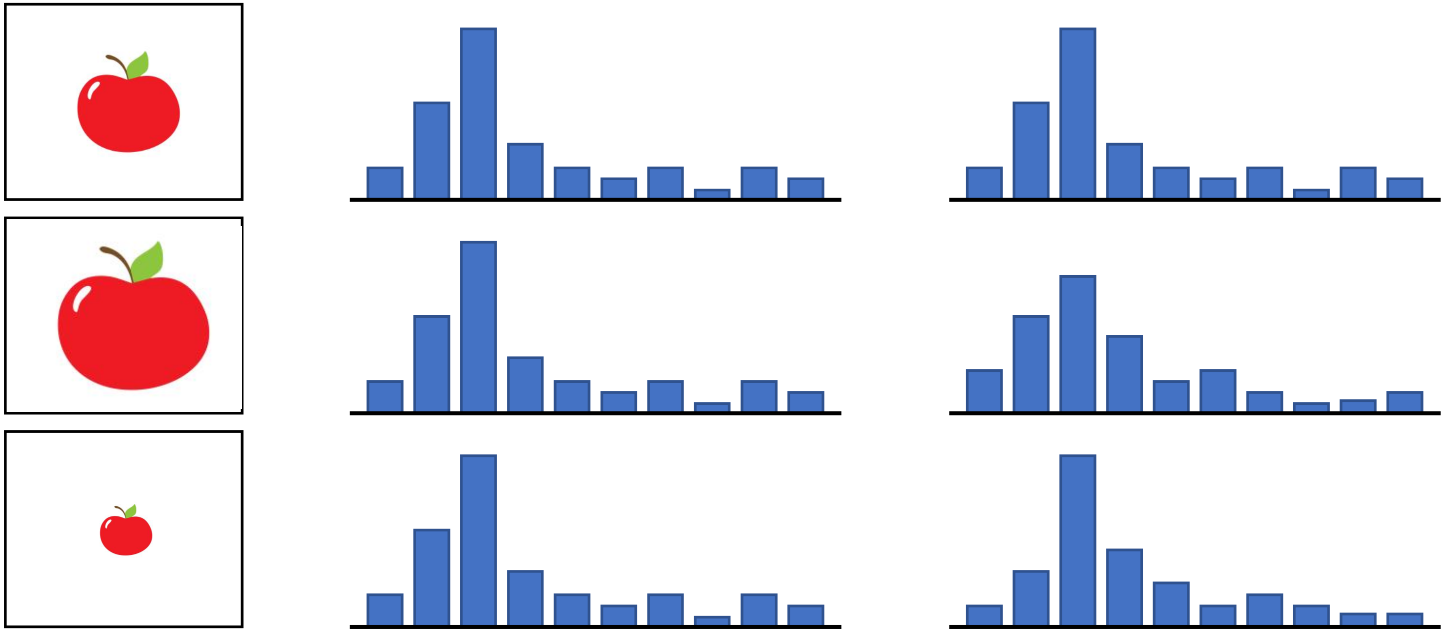
DeAngelis, Ohzawa, & Freeman (1995)

Why edges?

- Compression
  - Edge = discontinuity
  - Efficient way to represent images: only represent points where the signal changes (e.g., Elder & Zucker, 1996)
- Invariance
  - Edge-based features are invariant or tolerant to many irrelevant image changes

Definitions

- **Invariant** to X = response/representation does not vary with X, is insensitive to changes in X
- **Tolerant** to X = response is mostly insensitive to X



Invariant to light intensity?

- Image derivative is invariant to intensity shift ($\mathrm{I}_{\mathrm{new}} = \mathrm{I} + \mathrm{b}$)
- Tolerant to contrast change ($\mathrm{I}_{\mathrm{new}} = \mathrm{aI}$), but depends on thresholds

Invariant to light direction?

Photo: Heshan Jayakody

Invariant to translation?

Invariant to rotation?

Invariant to scale?

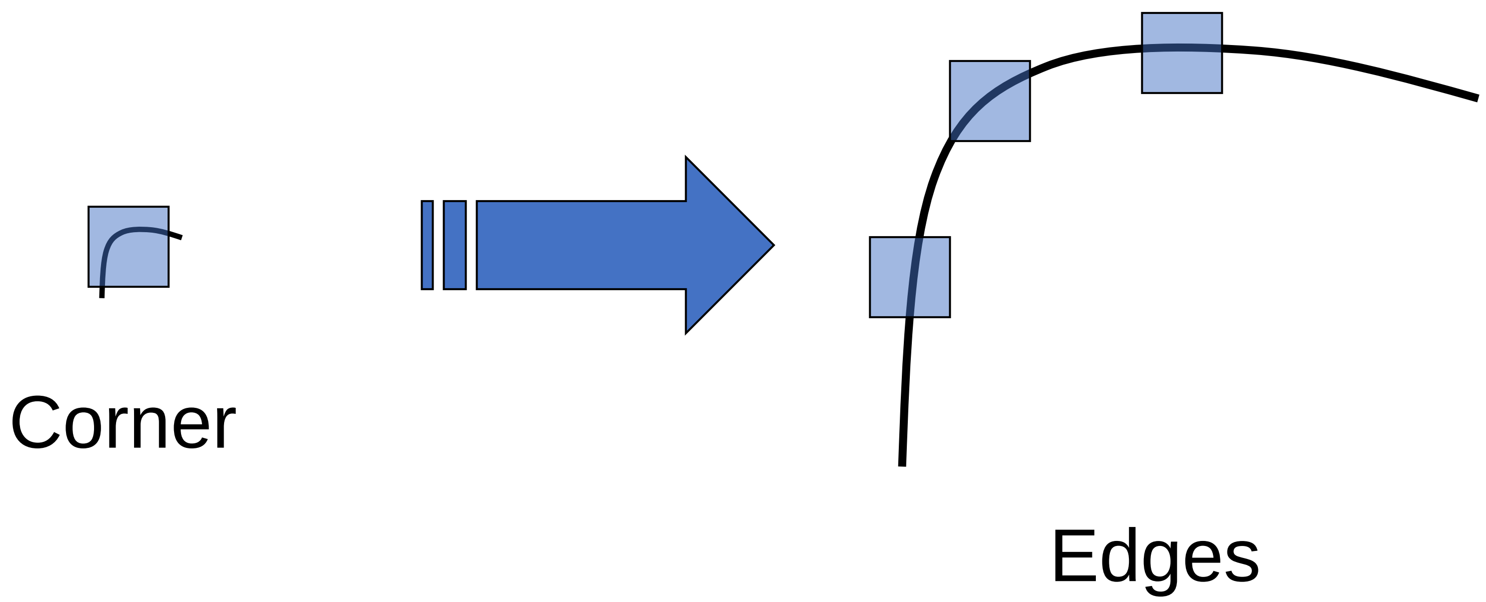


Image: K. Grauman

Invariant to 3D rotation / pose?

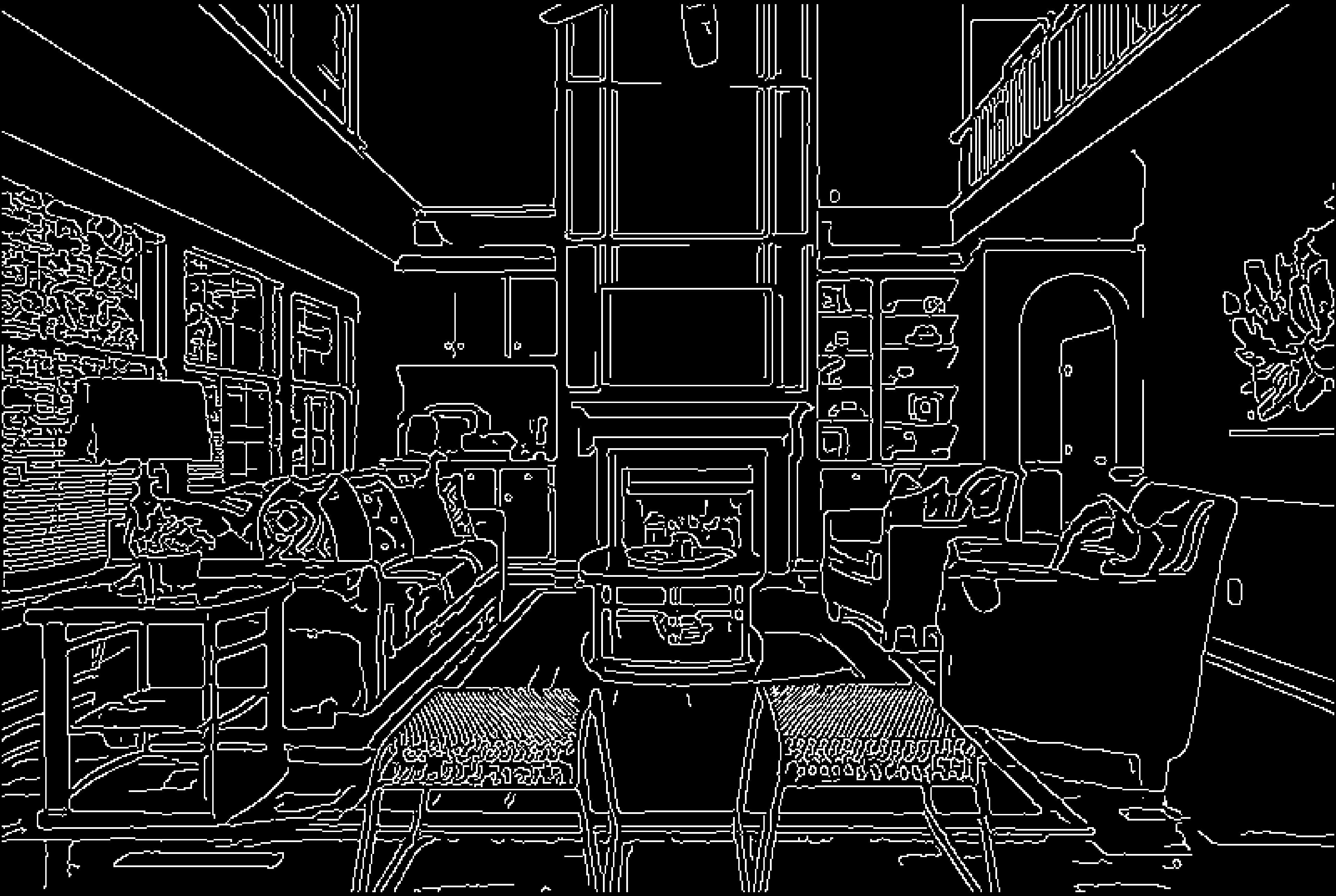
Image recognition

- To recognize objects across variations in lighting, position, size, pose, etc.:
  - Learn invariant features and compare them to image
  - Learn a separate set of features for each variation (e.g., 8 different rotations) and compare each one to image
- Recognition algorithms often use a mixture of both strategies

Edges: What next?

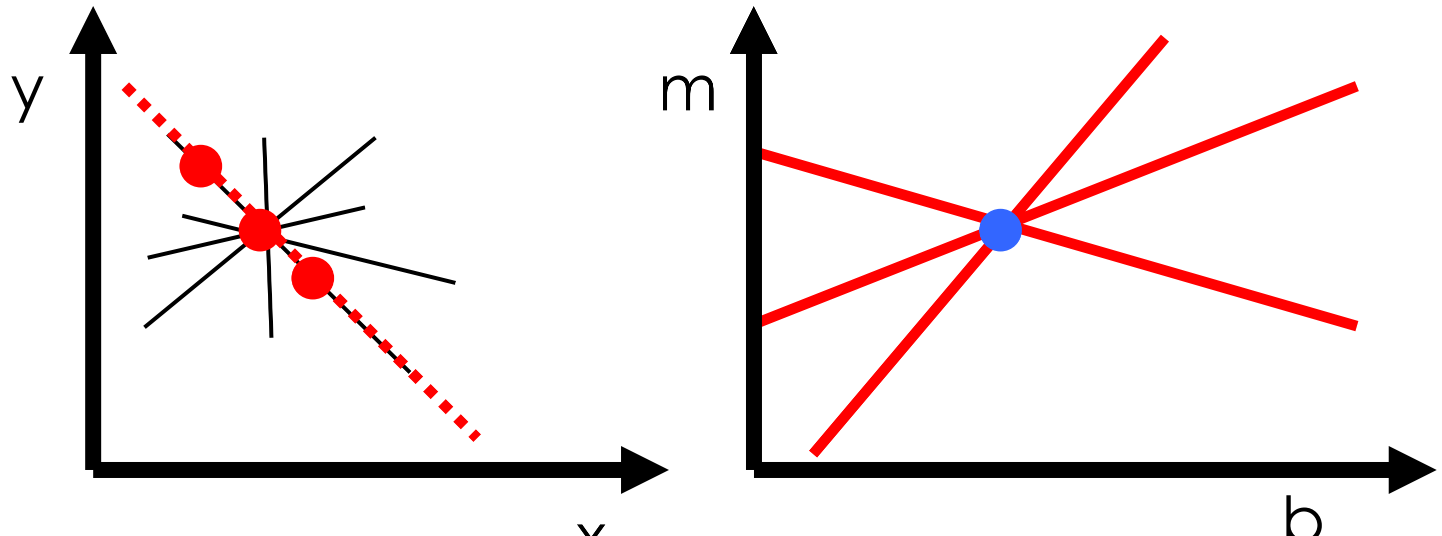
- Build more complex features for recognition
  - Learn local features (combinations of edges) that predict object/scene/etc. categories
  - Weeks 4-5
- Match points across multiple views to track objects or build 3D representations
  - Weeks 6-7
- Detect shapes (lines, circles, etc.) in edge maps

Canny edge map



Hough transform

- Each edge point "votes" for the lines that pass through that point
- Identify lines with the most "votes"

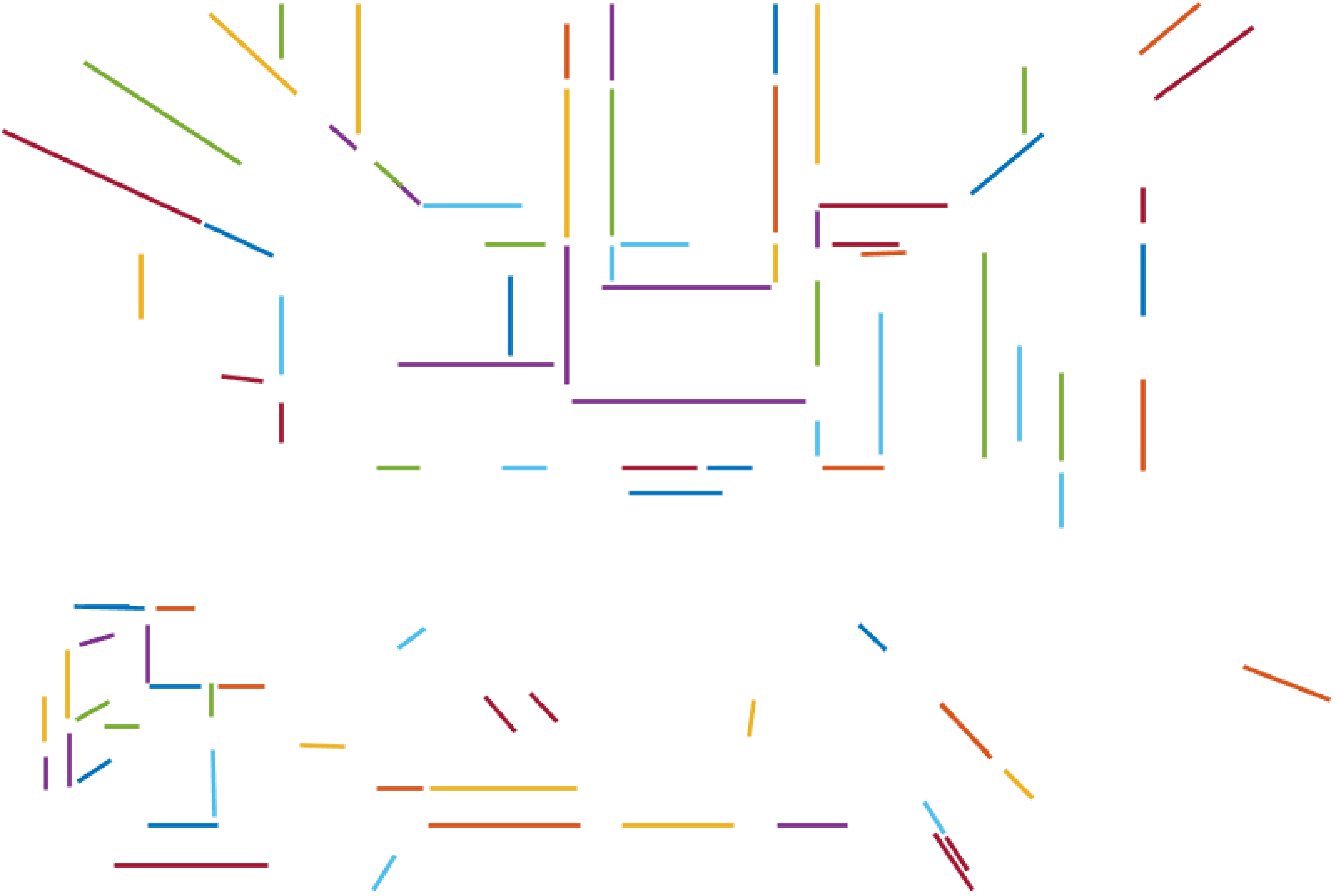


$y = m\,x + b$

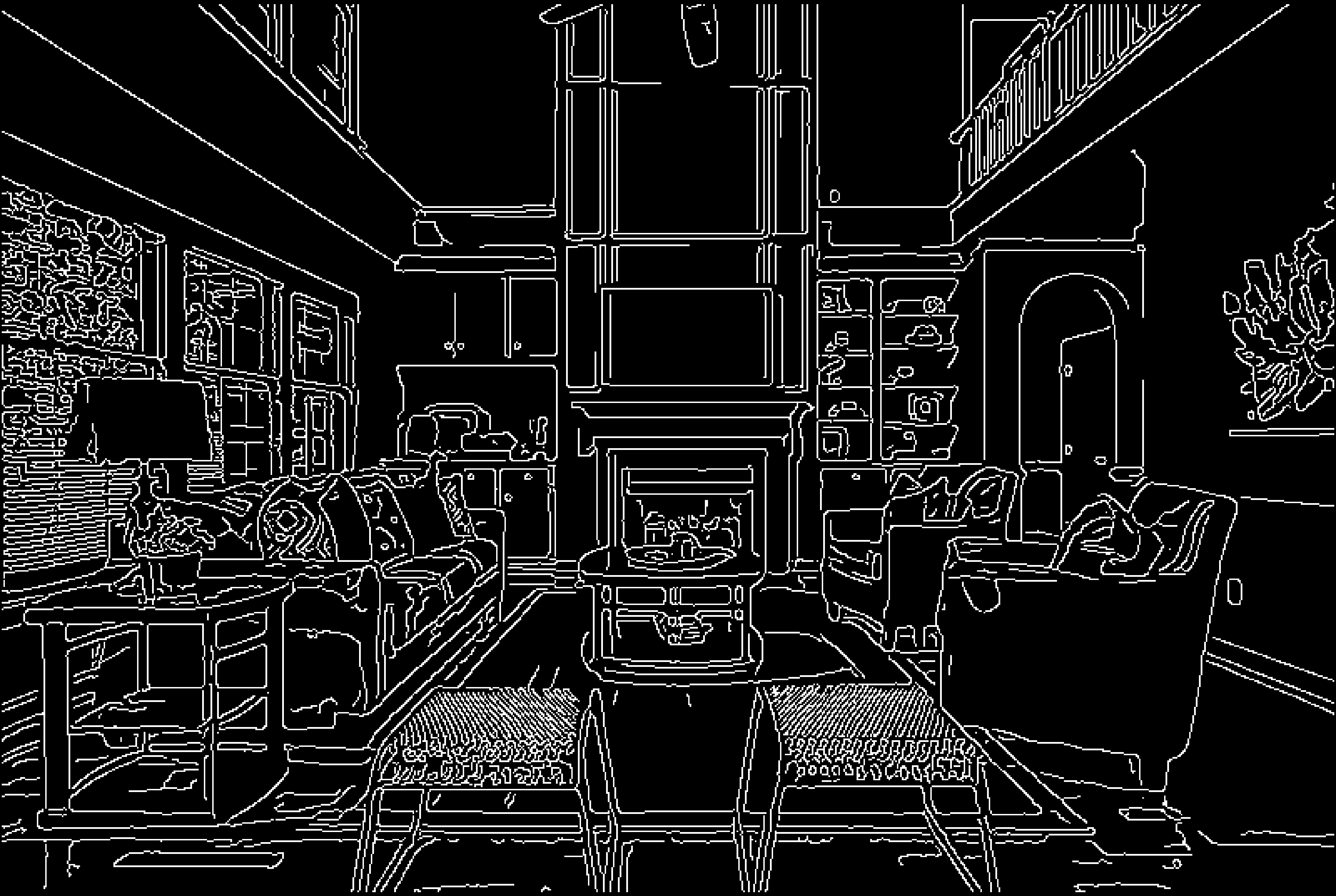
Hough space

Figure: S. Savarese

Hough lines



Canny edge map



Hough lines and estimated vanishing point

Summary

- Edge detection is the first step for most visual processing systems
- Edge-based features have desirable properties for visual recognition
  - Compress information
  - Invariant or tolerant to irrelevant changes in the images