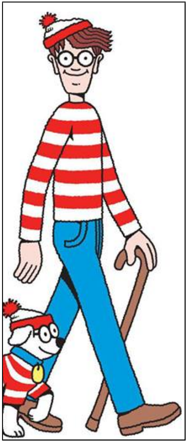
Edge Detection

Finding Waldo

- Let's revisit the problem of finding Waldo
- And let's take a simple example

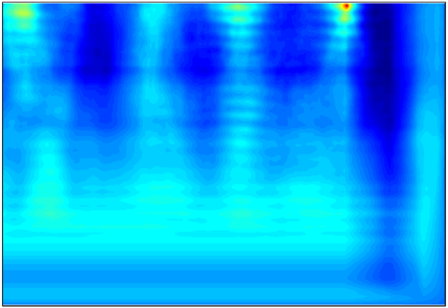
image



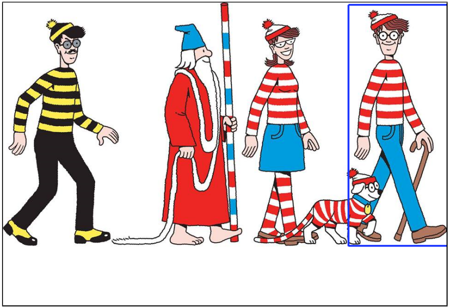
template (filter)

Finding Waldo

- Let's revisit the problem of finding Waldo
- And let's take a simple example



normalized cross-correlation



Waldo detection
(putting box around max response)

Finding Waldo

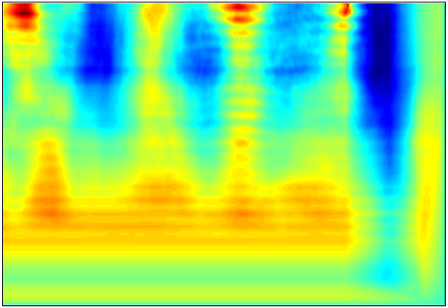
- Now imagine Waldo goes shopping
- ... but our filter **doesn't know that**

image

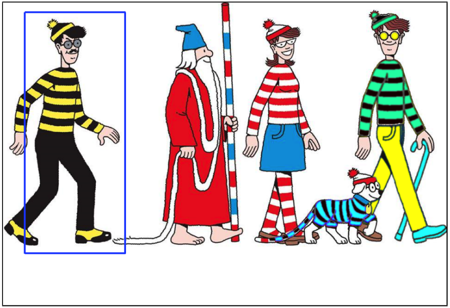
template (filter)

Finding Waldo

- Now imagine Waldo goes shopping (and the dog too)
- ... but our filter **doesn't know that**



normalized cross-correlation



Waldo detection
(putting box around max response)

Finding Waldo (again)

- What can we do to find Waldo again?
-

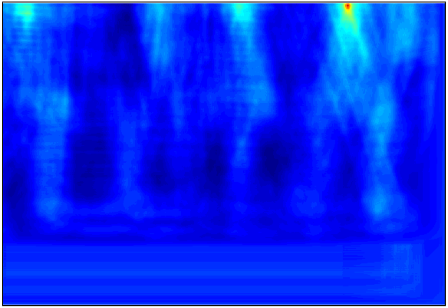
- What can we do to find Waldo again?
- **Edges!!!**

image

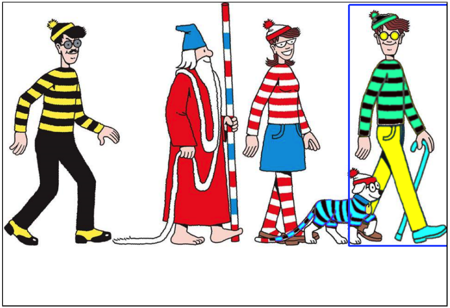
template (filter)

Finding Waldo (again)

- What can we do to find Waldo again?
- **Edges!!!**

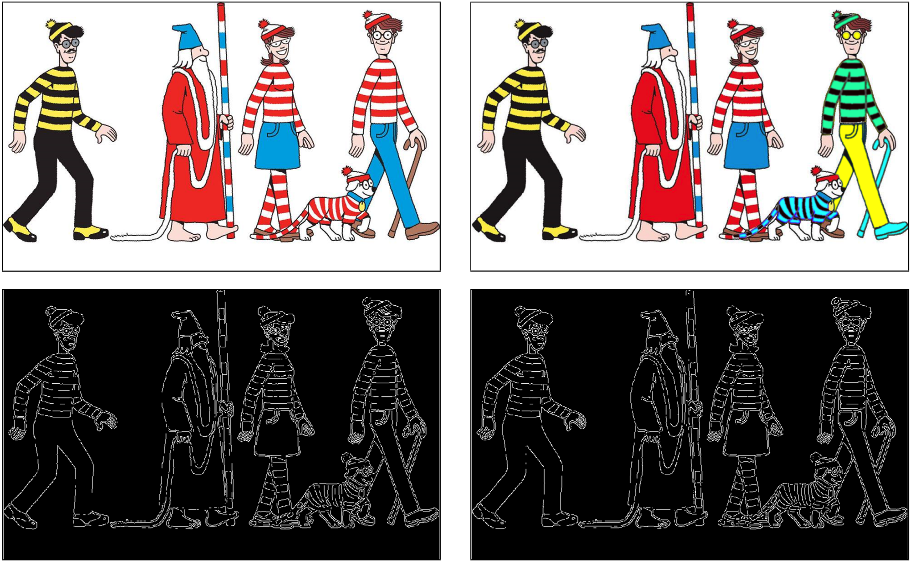


normalized cross-correlation
(using the edge maps)



Waldo detection
(putting box around max response)

Waldo and Edges



Edge detection

- Map image from 2d array of pixels to a set of **curves** or **line segments** or **contours**.
- More compact than pixels.
- Edges are invariant to changes in illumination
- Important for recognition

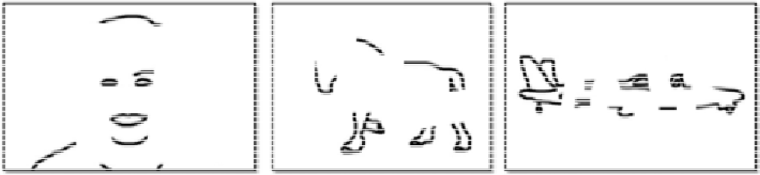
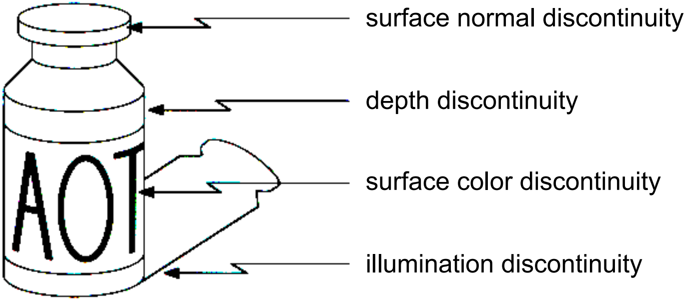


Figure: [Shotton et al. PAMI, 07]

[Source: K. Grauman]

Origin of Edges

- Edges are caused by a variety of factors

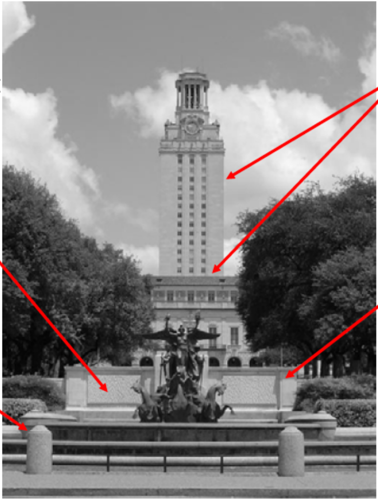


[Source: N. Snavely]

What Causes an Edge?

Reflectance change: appearance information, texture

Change in surface orientation: shape

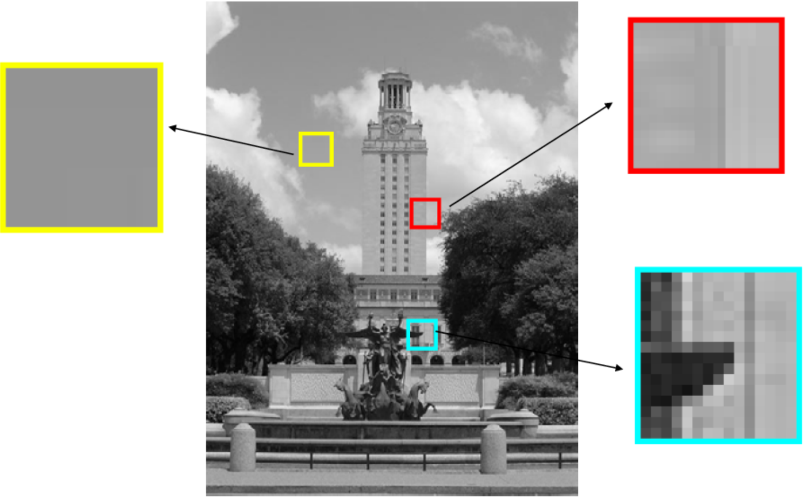


Depth discontinuity: object boundary

Cast shadows

[Source: K. Grauman]

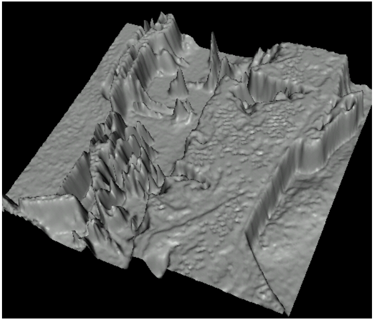
Looking More Locally...



[Source: K. Grauman]

Images as Functions

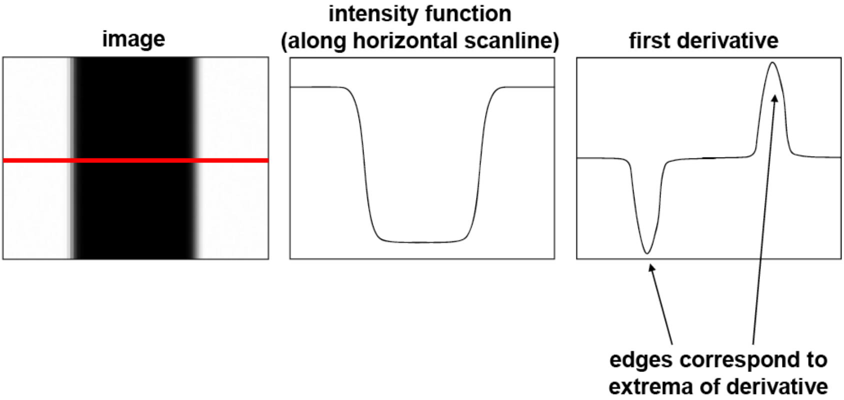
- Edges look like steep cliffs



[Source: N. Snavely]

Characterizing Edges

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



[Source: S. Lazebnik]

How to Implement Derivatives with Convolution

How can we differentiate a digital image $f[x, y]$?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x)}{\epsilon}$$

- Option 2: take discrete derivative (finite difference)

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f[x+1, y] - f[x]}{1}$$

How to Implement Derivatives with Convolution

How can we differentiate a digital image $f[x, y]$?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x)}{\epsilon}$$

- Option 2: take discrete derivative (finite difference)

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f[x + 1, y] - f[x]}{1}$$

- What would be the filter to implement this using convolution?

How to Implement Derivatives with Convolution

How can we differentiate a digital image $f[x, y]$?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x)}{\epsilon}$$

- Option 2: take discrete derivative (finite difference)

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f[x+1, y] - f[x]}{1}$$

- What would be the filter to implement this using convolution?

$\frac{\partial f}{\partial x}$:

H_x

$\frac{\partial f}{\partial y}$:

H_y

[Source: S. Seitz]

How to Implement Derivatives with Convolution

How can we differentiate a digital image $f[x, y]$?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x)}{\epsilon}$$

- Option 2: take discrete derivative (finite difference)

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f[x+1, y] - f[x]}{1}$$

- What would be the filter to implement this using convolution?

$\frac{\partial f}{\partial x}$:

H_x

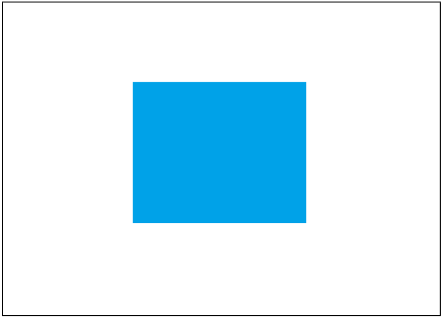
$\frac{\partial f}{\partial y}$:

H_y

[Source: S. Seitz]

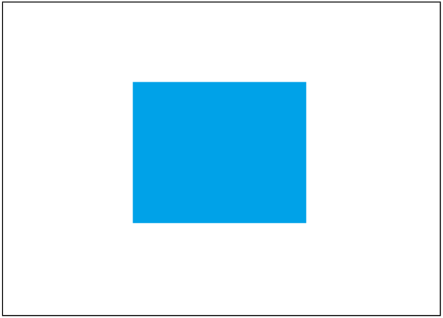
Examples: Partial Derivatives of an Image

- How does the horizontal derivative using the filter $[-1, 1]$ look like?

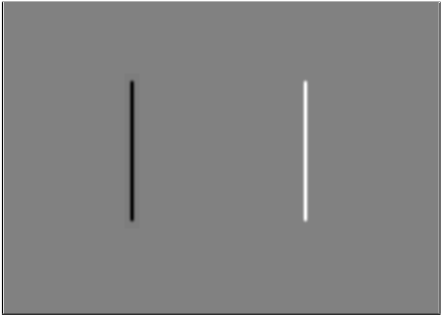


Image

- How does the horizontal derivative using the filter $[-1, 1]$ look like?



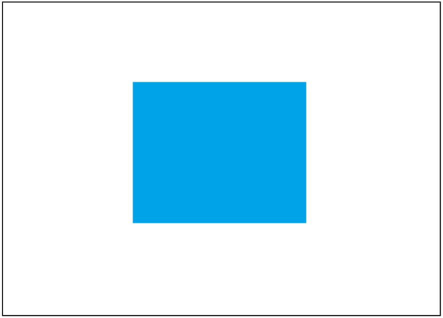
Image



$\frac{\partial f(x,y)}{\partial x}$ with $[-1, 1]$ and correlation

Examples: Partial Derivatives of an Image

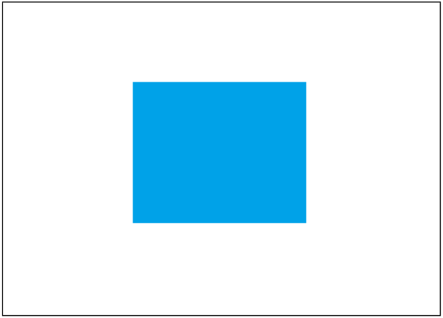
- How about the vertical derivative using filter $[-1, 1]^T$?



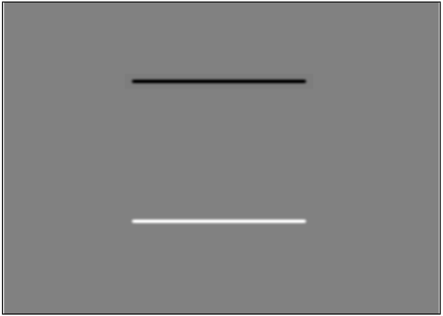
Image

Examples: Partial Derivatives of an Image

- How about the vertical derivative using filter $[-1, 1]^T$?



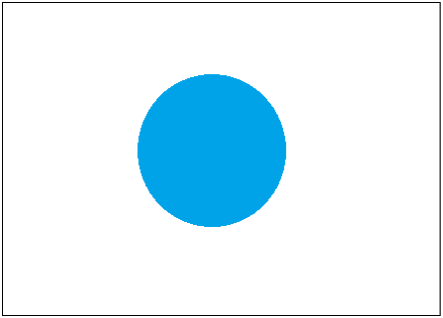
Image



$\frac{\partial f(x,y)}{\partial y}$ with $[-1, 1]^T$ and correlation

Examples: Partial Derivatives of an Image

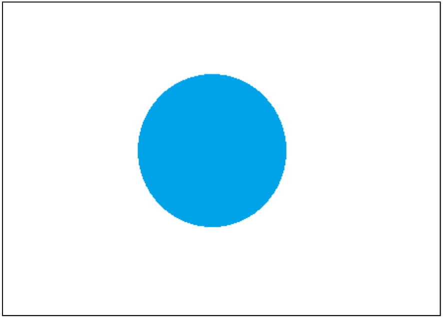
- How does the horizontal derivative using the filter $[-1, 1]$ look like?



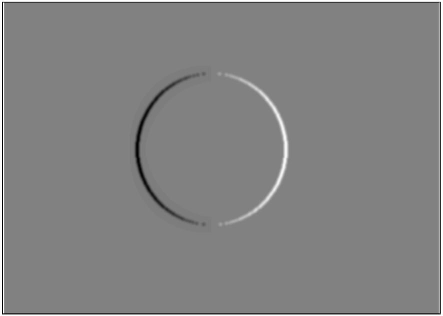
Image

Examples: Partial Derivatives of an Image

- How does the horizontal derivative using the filter $[-1, 1]$ look like?



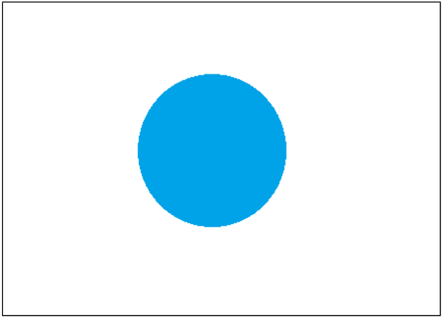
Image



$\frac{\partial f(x,y)}{\partial x}$ with $[-1, 1]$ and correlation

Examples: Partial Derivatives of an Image

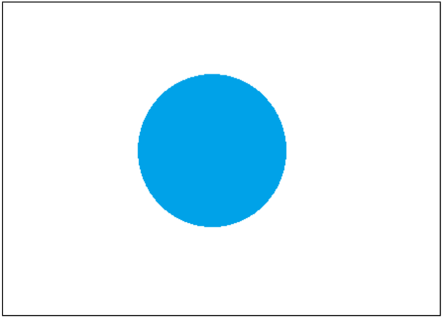
- How about the vertical derivative using filter $[-1, 1]^T$?



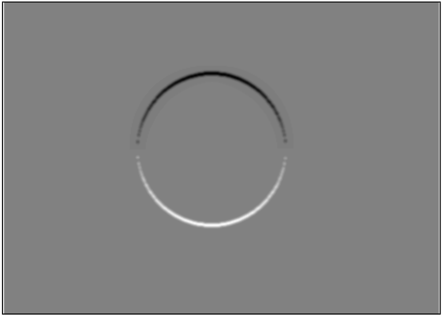
Image

Examples: Partial Derivatives of an Image

- How about the vertical derivative using filter $[-1, 1]^T$?



Image

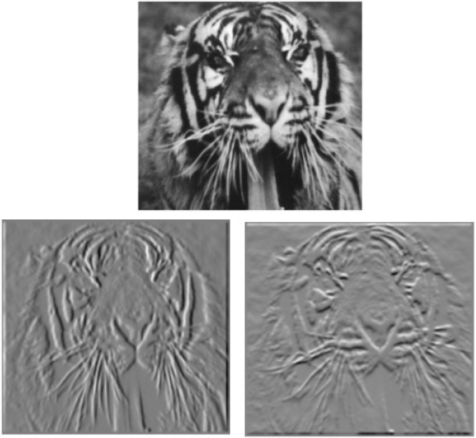


$\frac{\partial f(x,y)}{\partial y}$ with $[-1, 1]^T$ and correlation

Examples: Partial Derivatives of an Image

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

-1	1



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

-1
1

? or

1
-1

Figure: Using correlation filters

[Source: K. Grauman]

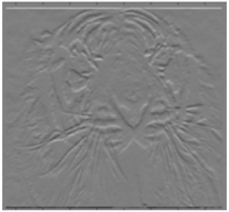
Finite Difference Filters

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$

Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$; $\boxed{M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}}$

Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array}$

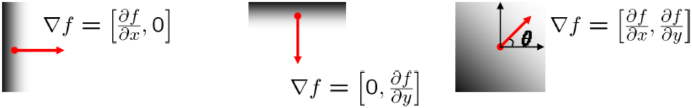
```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```



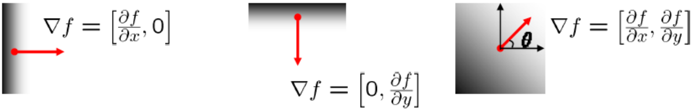
[Source: K. Grauman]

Image Gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity

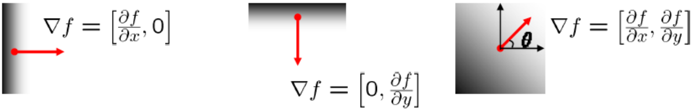


- The **gradient direction** (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

Image Gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



- The **gradient direction** (orientation of edge normal) is given by:

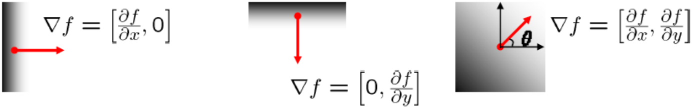
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- The **edge strength** is given by the magnitude $||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

[Source: S. Seitz]

Image Gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



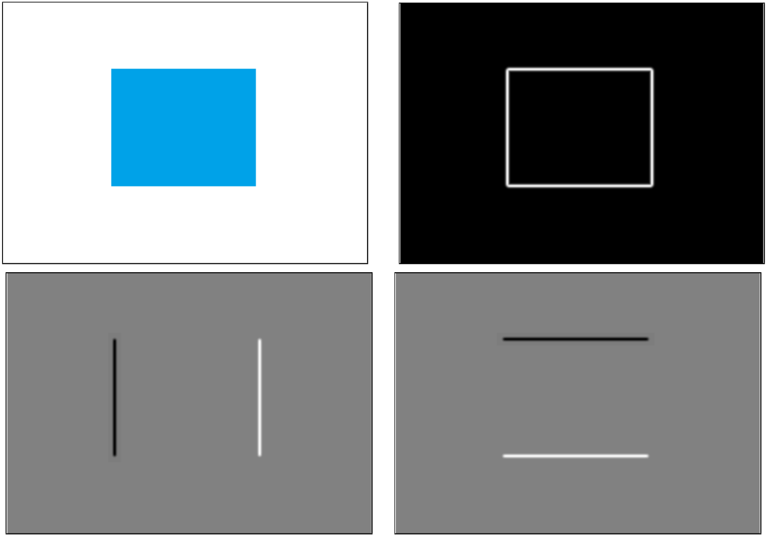
- The **gradient direction** (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

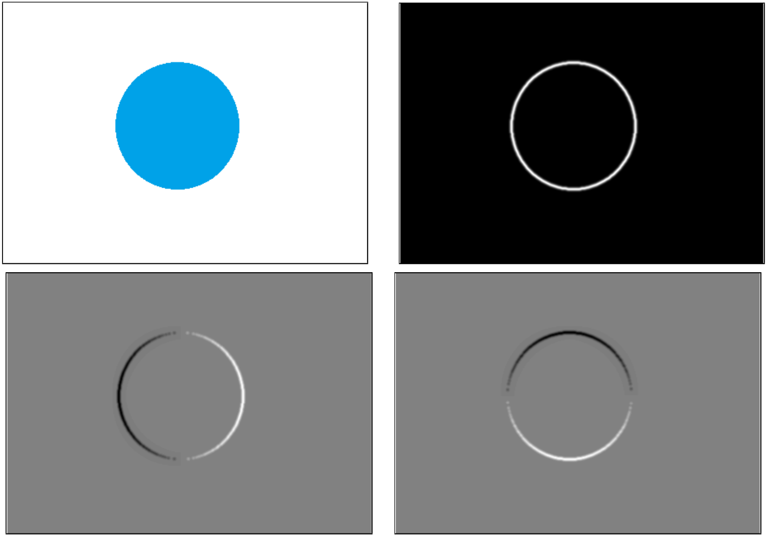
- The **edge strength** is given by the magnitude $||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$

[Source: S. Seitz]

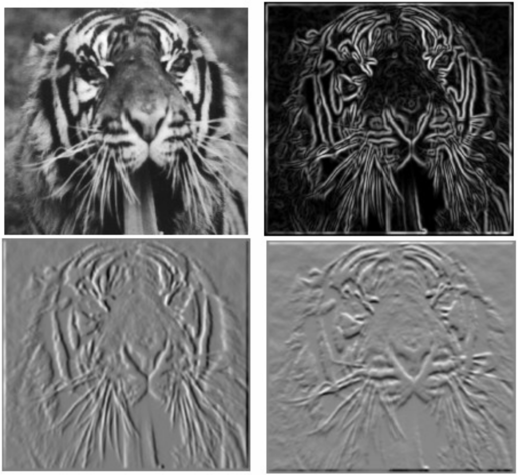
Example: Image Gradient



Example: Image Gradient



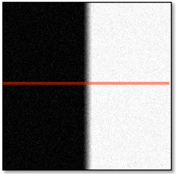
Example: Image Gradient

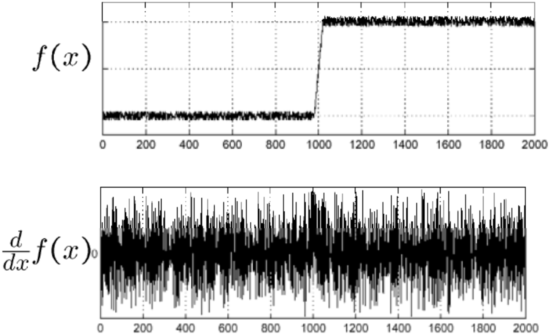


[Source: S. Lazebnik]

Effects of noise

- What if our image is noisy? What can we do?
- Consider a single row or column of the image.
- Plotting intensity as a function of position gives a signal.

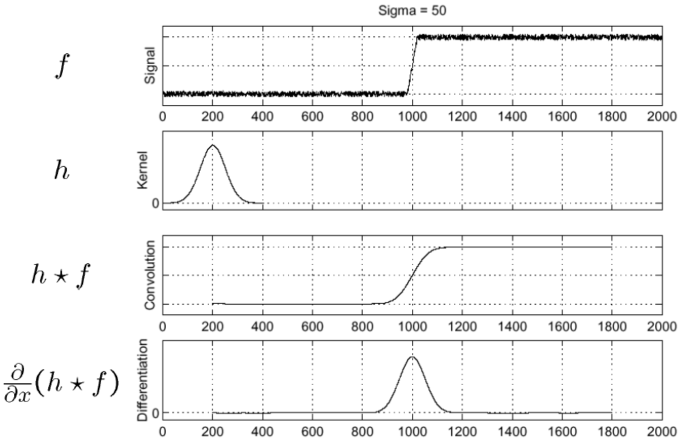

Noisy input image



[Source: S. Seitz]

Effects of noise

- Smooth first with h (e.g. Gaussian), and look for peaks in $\frac{\partial}{\partial x}(h * f)$.



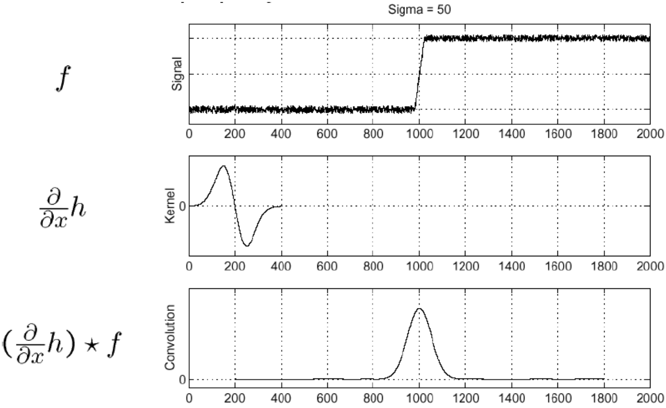
[Source: S. Seitz]

Derivative theorem of convolution

- Differentiation property of convolution

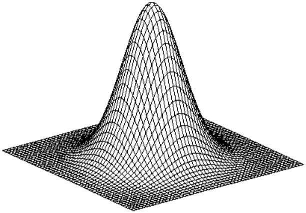
$$\frac{\partial}{\partial x}(h * f) = (\frac{\partial h}{\partial x}) * f = h * (\frac{\partial f}{\partial x})$$

- It saves one operation



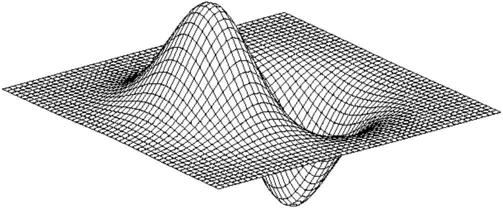
[Source: S. Seitz]

2D Edge Detection Filters



Gaussian

$h_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{u^2+v^2}{2\sigma^2}}$

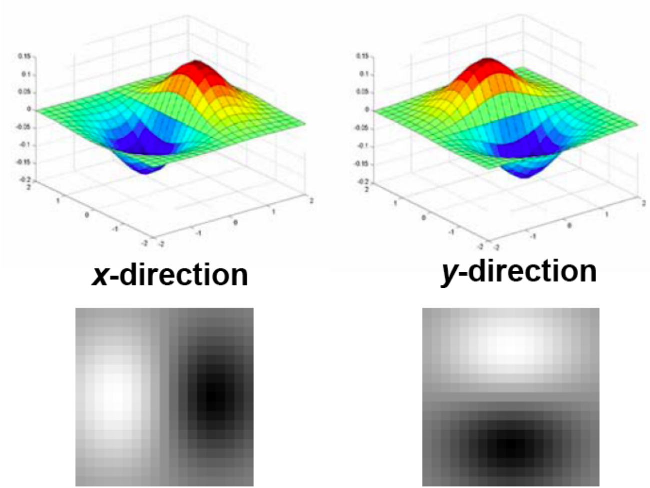


Derivative of Gaussian (x)

$\frac{\partial}{\partial x} h_\sigma(u, v)$

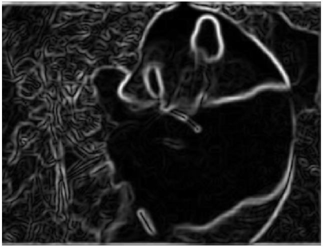
[Source: N. Snavely]

Derivative of Gaussians



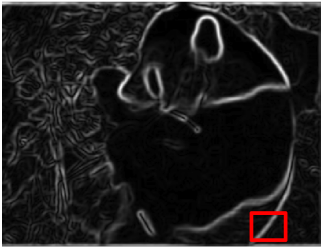
[Source: K. Grauman]

Example



- Applying the Gaussian derivatives to image

Example



- Applying the Gaussian derivatives to image

Properties:

- Zero at a long distance from the edge
- Positive on both sides of the edge
- Highest value at some point in between, on the edge itself

Effect of σ on derivatives

The detected structures differ depending on the **Gaussian's scale parameter**:

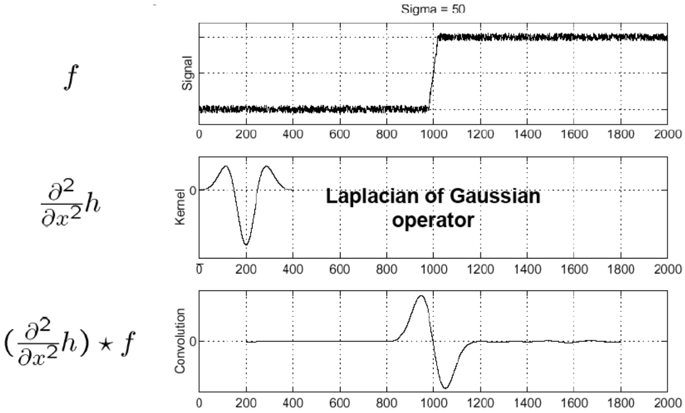
- Larger values: larger scale edges detected
- Smaller values: finer structures detected

σ = 1 pixel σ = 3 pixels

[Source: K. Grauman]

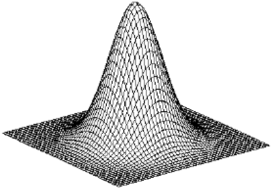
Laplacian of Gaussians

- Edge by detecting **zero-crossings** of bottom graph



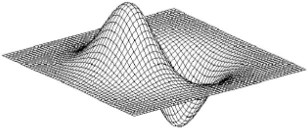
[Source: S. Seitz]

2D Edge Filtering



Gaussian

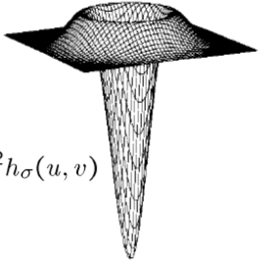
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

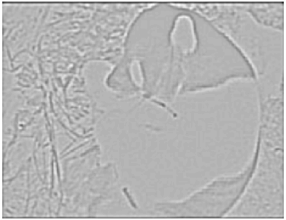


$$\nabla^2 h_\sigma(u,v)$$

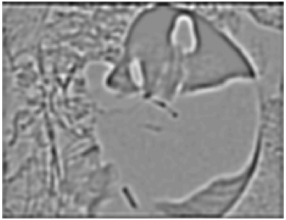
with ∇^2 the Laplacian operator $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

[Source: S. Seitz]

Example



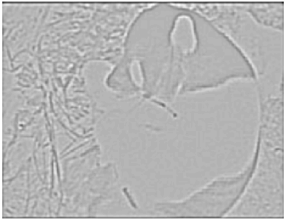
$\sigma = 1$ pixels



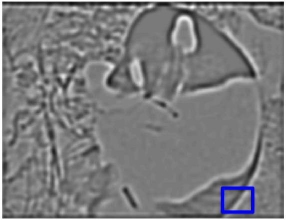
$\sigma = 3$ pixels

- Applying the Laplacian operator to image

Example



$\sigma = 1$ pixels

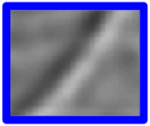


$\sigma = 3$ pixels

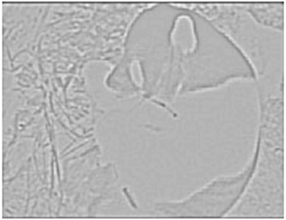
- Applying the Laplacian operator to image

Properties:

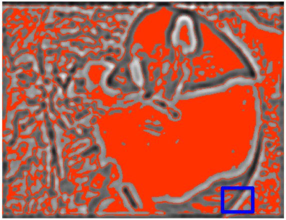
- Zero at a long distance from the edge
- Positive on the darker side of edge
- Negative on the lighter side
- Zero at some point in between, on edge itself



Example



$\sigma = 1$ pixels

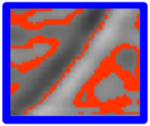


$\sigma = 3$ pixels

- Applying the Laplacian operator to image

Properties:

- Zero at a long distance from the edge
- Positive on the darker side of edge
- Negative on the lighter side
- Zero at some point in between, on edge itself



Locating Edges – Canny's Edge Detector

Let's take the most popular picture in computer vision: Lena (appeared in November 1972 issue of Playboy magazine)

[Source: N. Snavely]

Locating Edges

Figure: Canny's approach takes gradient magnitude

[Source: N. Snavely]

Locating Edges

Figure: Thresholding

[Source: N. Snavely]

Locating Edges

Figure: Gradient magnitude

[Source: N. Snavely]

Non-Maxima Suppression

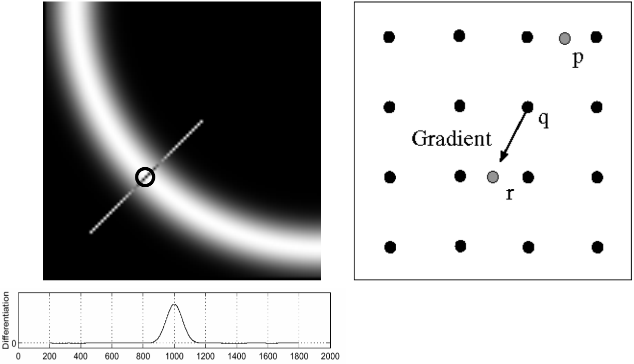


Figure: Gradient magnitude

- Check if pixel is local maximum along gradient direction
- If yes, take it

[Source: N. Snavely]

Finding Edges

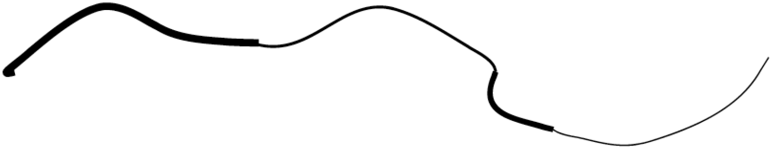
Problem: pixels along this edge didn't survive the thresholding

Figure: Problem with thresholding

[Source: K. Grauman]

Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them



[Source: K. Grauman]

Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

[Source: L. Fei Fei]

Located Edges!

Figure: Thinning: Non-maxima suppression

[Source: N. Snavely]

Canny Edge Detector

Matlab: `edge(image,'canny')`

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
    - Define two thresholds: low and high
    - Use the high threshold to start edge curves and the low threshold to continue them

[Source: D. Lowe and L. Fei-Fei]

Canny Edge Detector

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

original

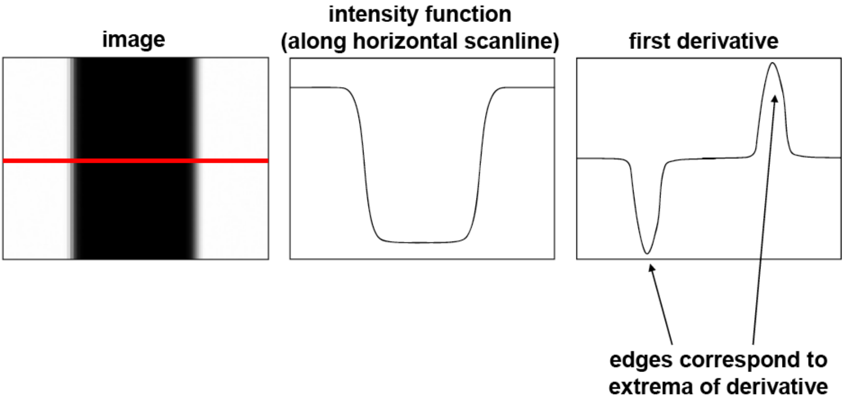
Canny with $\sigma = 1$

Canny with $\sigma = 2$

[Source: S. Seitz]

What Happens Here?

- Remember this?



What Happens Here?

- What happens with an image with the following intensity profile?

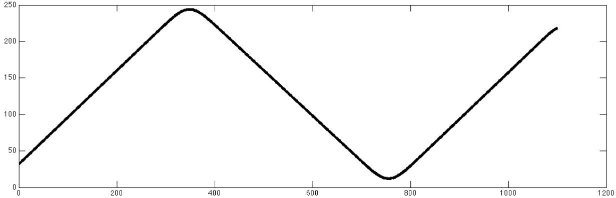


Figure: Intensity of image in one horizontal slice

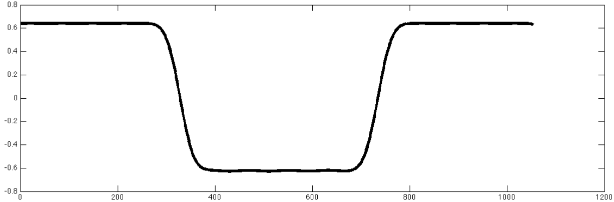


Figure: Horizontal derivative $[-1, 1]$

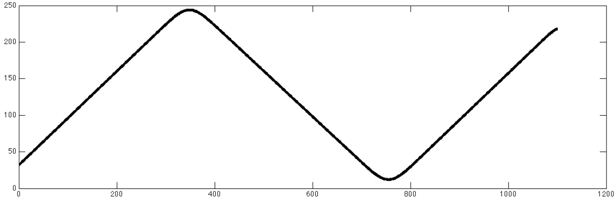


Figure: Intensity of image in one horizontal slice

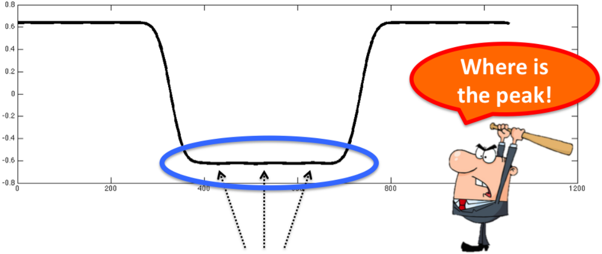


Figure: Horizontal derivative $[-1, 1]$

What Happens Here?

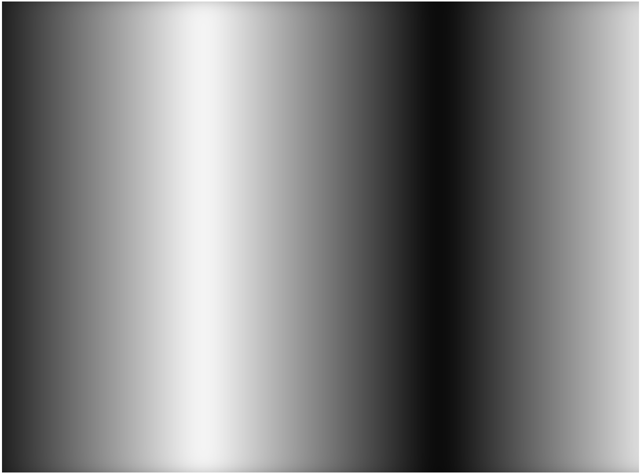


Figure: The image

- Is there really an edge in this image?

What Happens Here?

Figure: Canny's edge detection

- Is there really an edge in this image?

Canny edge detector

- Still one of the most widely used edge detectors in computer vision
- J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.
- Depends on several parameters: σ of the **blur** and the **thresholds**

[Source: R. Urtasun]

Summary – Stuff You Should Know

Not so good:

- **Horizontal image gradient**: Subtract intensity of left neighbor from pixel's intensity (filtering with $[-1, 1]$)
- **Vertical image gradient**: Subtract intensity of bottom neighbor from pixel's intensity (filtering with $[-1, 1]^T$)

Much better (more robust to noise):

- **Horizontal image gradient**: Apply derivative of Gaussian with respect to x to image (filtering!)
- **Vertical image gradient**: Apply derivative of Gaussian with respect to y to image
- **Magnitude of gradient**: compute the horizontal and vertical image gradients, square them, sum them, and $\sqrt{}$ the sum
- **Edges**: Locations in image where magnitude of gradient is high
- Phenomena that **causes** edges: rapid change in surface's normals, depth discontinuity, rapid changes in color, change in illumination

Summary – Stuff You Should Know

- **Properties of gradient's magnitude**:
  - Zero far away from edge
  - Positive on both sides of the edge
  - Highest value directly on the edge
  - Higher σ emphasizes larger structures
- **Canny's edge detector**:
  - Compute gradient's direction and magnitude
  - Non-maxima suppression
  - Thresholding at two levels and linking

Matlab functions:

- FSPECIAL: gives a few gradients filters (PREWITT, SOBEL, ROBERTS)
- SMOOTHGRADIENT: function to compute gradients with derivatives of Gaussians. Find it in Lecture's 3 code (check class webpage)
- EDGE: use EDGE(I,'CANNY') to detect edges with Canny's method, and EDGE(I,'LOG') for Laplacian method

Edge Detection
State of The Art

P. Dollar and C. Zitnick

Structured Forests for Fast Edge Detection

ICCV 2013

Code: `http://research.microsoft.com/en-us/downloads/389109f6-b4e8-404c-84bf-239f7cbf4e3d/default.aspx`

(Time stamp: Sept 15, 2014)

Testing the Canny Edge Detector

- Let's take this image
- Our goal (a few lectures from now) is to detect objects (cows here)

Testing the Canny Edge Detector

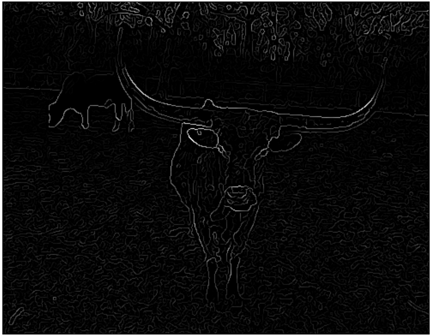


image gradients + NMS

Canny's edges

Testing the Canny Edge Detector

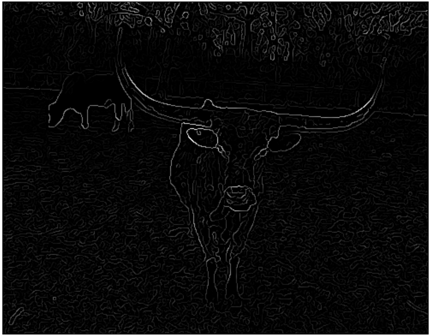


image gradients + NMS

Canny's edges

Testing the Canny Edge Detector

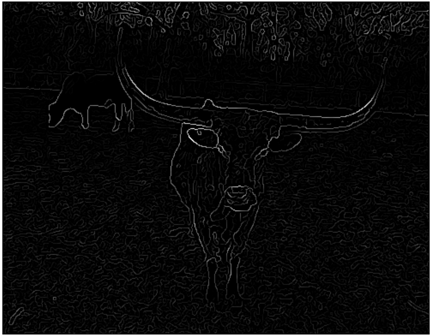


image gradients + NMS

Canny's edges

- Lots of "distractor" and missing edges
- Can we do better?

Annotate...

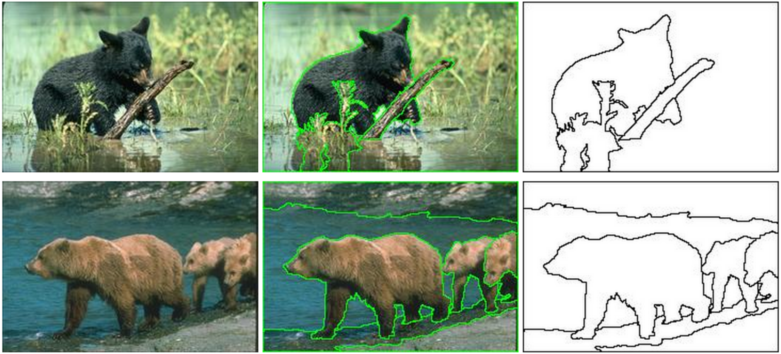
- Imagine someone goes and **annotates** which edges are **correct**
- ... and someone has:

Annotate...

- Imagine someone goes and **annotates** which edges are **correct**
- ... and someone has:

The Berkeley Segmentation Dataset and Benchmark

by D. Martin and C. Fowlkes and D. Tal and J. Malik



... and do Machine Learning

- How can we make use of such data to **improve** our edge detector?

- How can we make use of such data to **improve** our edge detector?
- We can use Machine Learning techniques to:

Train classifiers!

- Please **learn what a classifier /classification is**
- In particular, learn what a **Support Vector Machine** (SVM) is (some links to tutorials are on the class webpage)
- With each week it's going to be more important to know about this
- You don't need to learn all the details / math, but to understand the concept enough to know what's going on

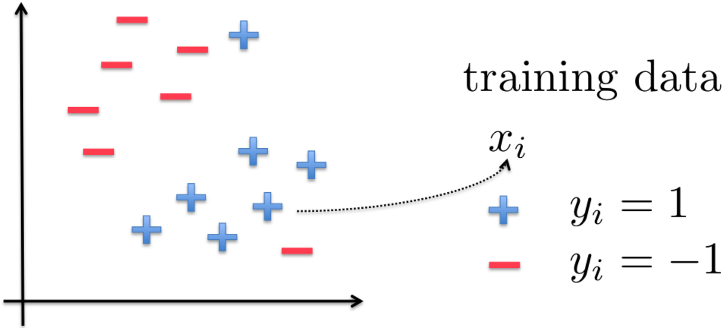
- How can we make use of such data to **improve** our edge detector?
- We can use Machine Learning techniques to:

Train classifiers!

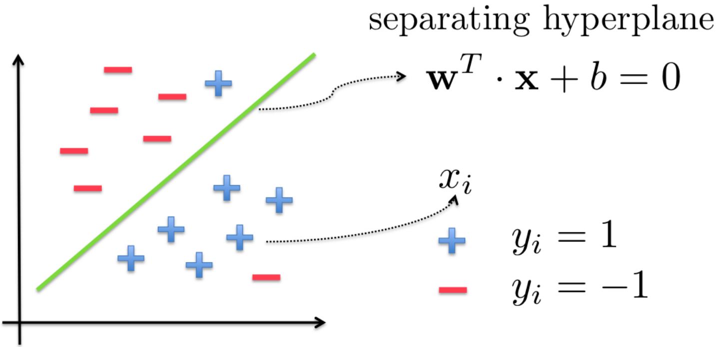
- Please **learn what a classifier /classification is**
- In particular, learn what a **Support Vector Machine** (SVM) is (some links to tutorials are on the class webpage)
- With each week it's going to be more important to know about this
- You don't need to learn all the details / math, but to understand the concept enough to know what's going on

Classification – a Disney edition (pictures only)

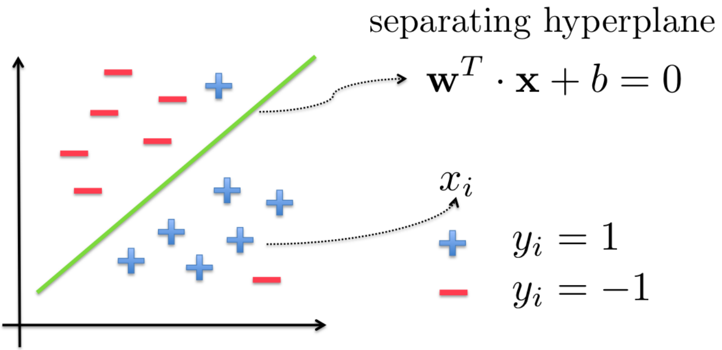
- Each data point $\mathbf{x}$ lives in a n-dimensional space, $x \in \mathbb{R}^n$
- We have a bunch of data points $\mathbf{x}_i$, and for each we have a **label**, y_i
- A label y_i can be either 1 (positive example – correct edge in our case), or -1 (negative example – wrong edge in our case)



Classification – a Disney edition (pictures only)



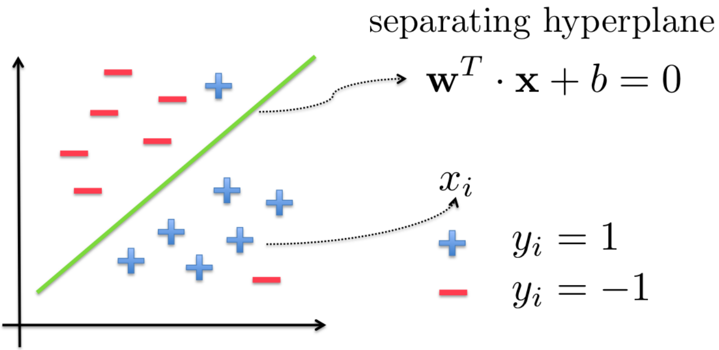
Classification – a Disney edition (pictures only)



At training time:

Finding **weights w** so that positive and negative examples are optimally separated

Classification – a Disney edition (pictures only)



At test time:

$$\mathbf{w}^T \cdot \mathbf{x} + b > 0 \rightarrow \mathbf{x} \text{ is a positive example}$$
$$\mathbf{w}^T \cdot \mathbf{x} + b < 0 \rightarrow \mathbf{x} \text{ is a negative example}$$

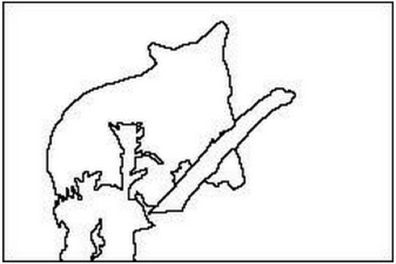
Training an Edge Detector

- How should we do this?

Training an Edge Detector

- How should we do this?

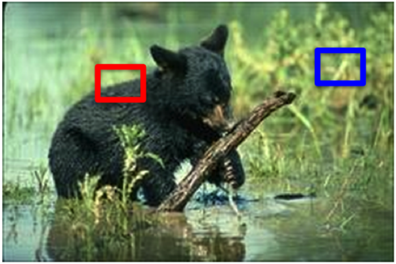
image

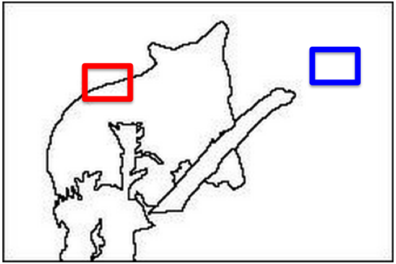


annotation

Training an Edge Detector

- We extract lots of image patches

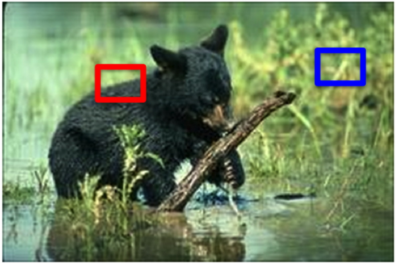


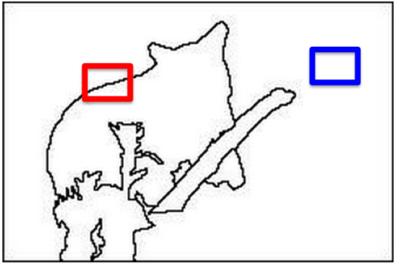


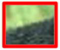
We call each such crop an **image patch**

Training an Edge Detector

- We extract lots of image patches
- These are our training data





 → edge

 → no edge

} our training data

Training an Edge Detector

- We extract lots of image patches
- These are our training data
- We convert each image patch **P** (a matrix) into a vector **x**

matrix $\mathbf{P}$

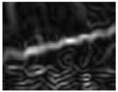
$$\rightarrow \quad \mathbf{x} = \mathbf{P}(:)$$

Training an Edge Detector

- We extract lots of image patches
- These are our training data
- We convert each image patch **P** (a matrix) into a vector **x**
- Well... This works better: Extract **image features** for each patch

matrix $\mathbf{P}$

$\xrightarrow{\text{compute gradients}}$



matrix $\mathbf{G}$

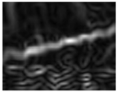
$$\mathbf{x} = \mathbf{G}(:)$$

Training an Edge Detector

- We extract lots of image patches
- These are our training data
- We convert each image patch **P** (a matrix) into a vector **x**
- Well... This works better: Extract **image features** for each patch
- Image features are mappings from images (or patches) to other (vector) meaningful representations. More on this in the next class!

matrix $\mathbf{P}$

compute gradients $\longrightarrow$



matrix $\mathbf{G}$

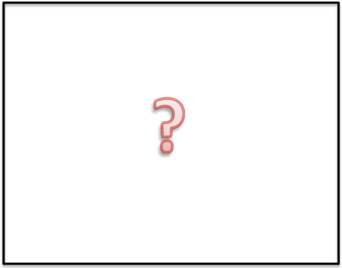
$$\mathbf{x} = \mathbf{G}(:)$$

compute color $\longrightarrow$

Using an Edge Detector

- Once trained, **how can we use** our new edge detector?

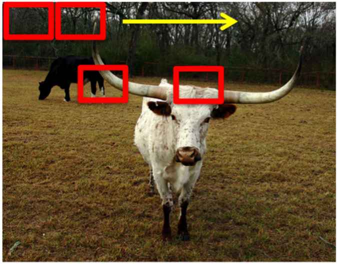
image



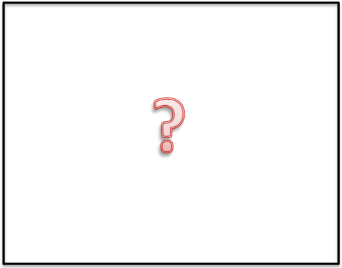
prediction

Using an Edge Detector

- We extract all image patches



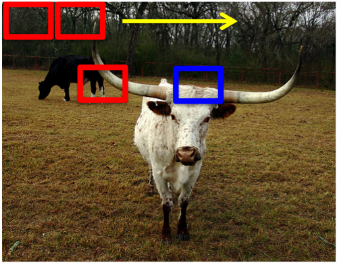
image



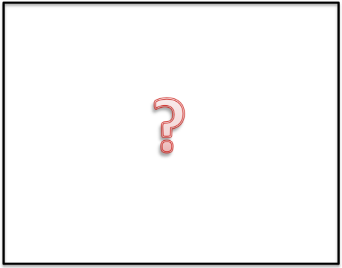
prediction

Using an Edge Detector

- We extract all image patches
- Extract features and use our trained classifier



image

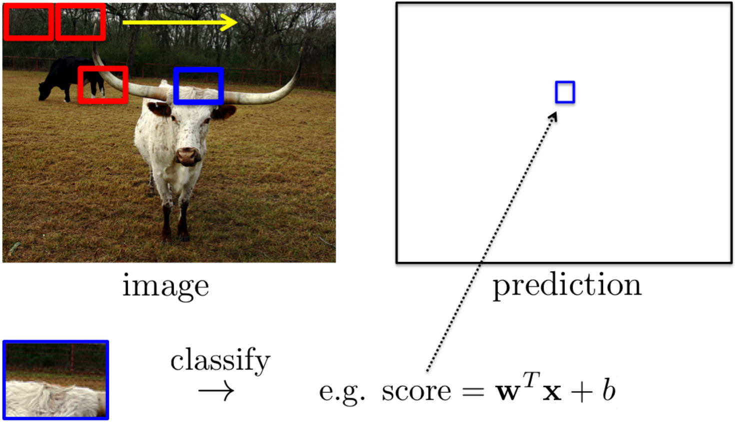


prediction

$\xrightarrow{\text{classify}}$ e.g. $\text{score} = \mathbf{w}^T\mathbf{x} + b$

Using an Edge Detector

- We extract all image patches
- Extract features and use our trained classifier
- Place the predicted value (score) in the output matrix



$\xrightarrow{\text{classify}}$ e.g. $\text{score} = \mathbf{w}^T\mathbf{x} + b$

Comparisons: Canny vs Structured Edge Detector

image

image gradients

gradients + NMS

“edgeness score”

score + NMS

Comparisons: Canny vs Structured Edge Detector


image


image gradients


gradients + NMS


image gradient


"edgeness" score


"edgeness" score


score + NMS

Comparisons: Canny vs Structured Edge Detector

image

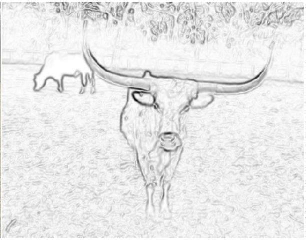
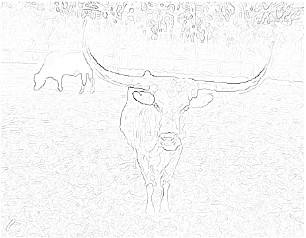
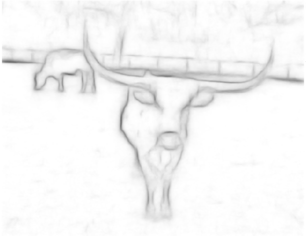


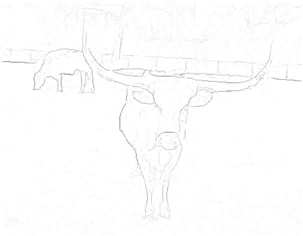
image gradients



gradients + NMS



"edgeness" score



score + NMS

Comparisons: Canny vs Structured Edge Detector

image

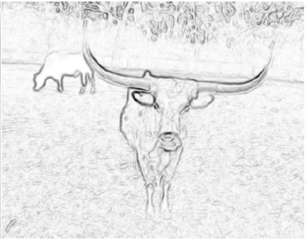
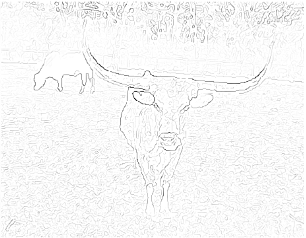
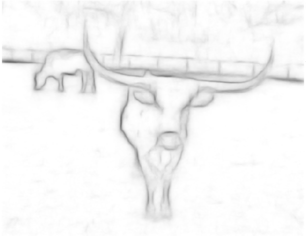


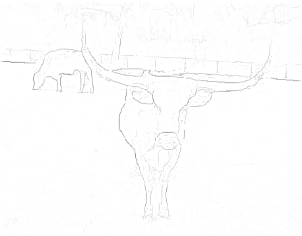
image gradients



gradients + NMS



“edgeness” score



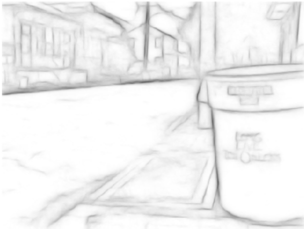
score + NMS

Comparisons: Canny vs Structured Edge Detector

image

image gradients

gradients + NMS



"edgeness" score

score + NMS

Comparisons: Canny vs Structured Edge Detector

image

image gradients

gradients + NMS

image gradient

"edgeness" score

"edgeness" score

score + NMS

Evaluation

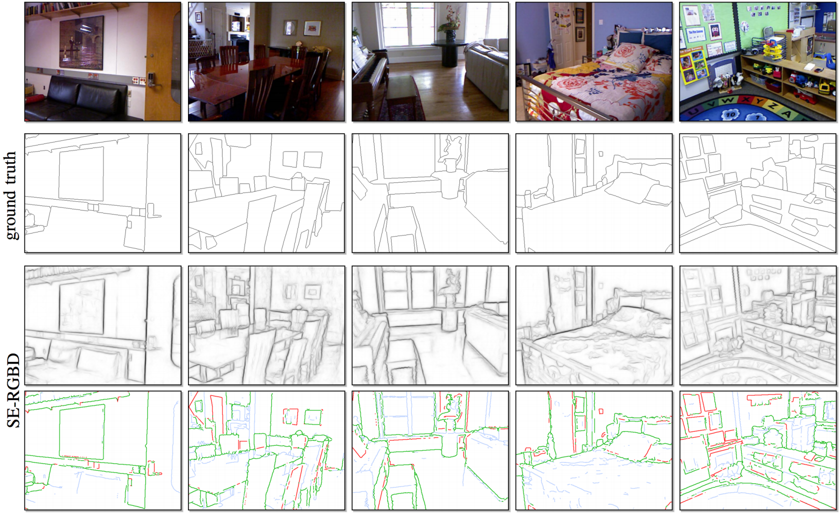
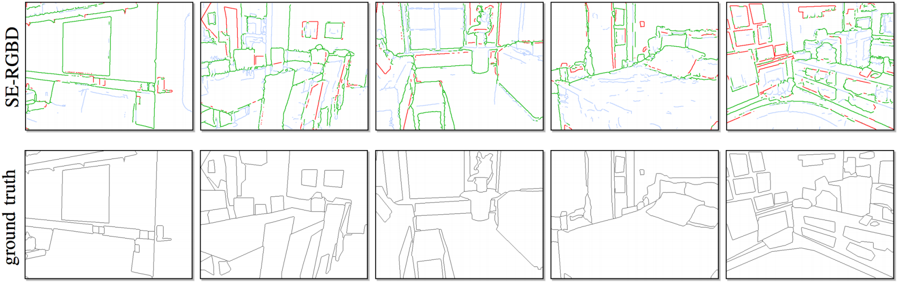


Figure: green=correct, blue=wrong, red=missing, green+blue=output edges

Evaluation

- **Recall:** How many of all **annotated** edges we got correct (best is 1)
- **Precision** How many of all **output** edges we got correct (best is 1)

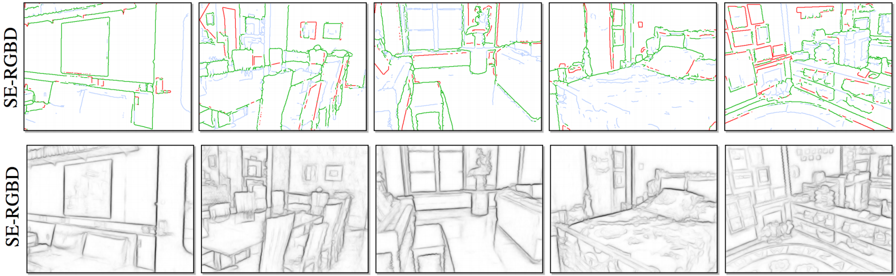
$$\textbf{Recall} = \frac{\#\text{ of green (correct edges)}}{\#\text{ of all edges in ground-truth (first picture)}}$$



Evaluation

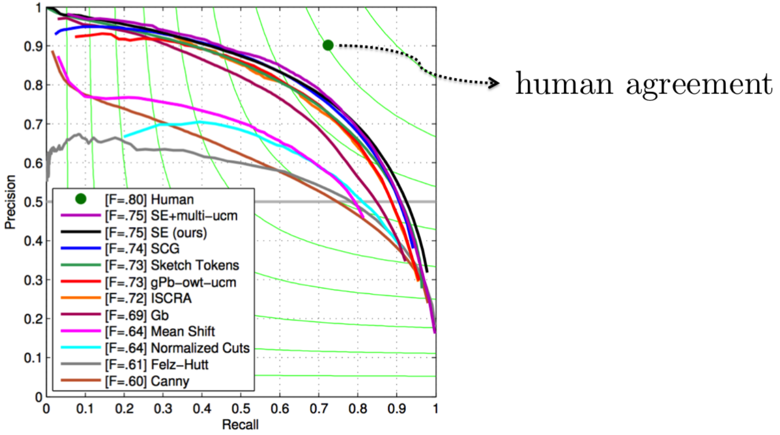
- **Recall:** How many of all **annotated** edges we got correct (best is 1)
- **Precision** How many of all **output** edges we got correct (best is 1)

$$\textbf{Precision} = \frac{\#\text{ of green (correct edges)}}{\#\text{ of all edges in output (first picture)}}$$



Evaluation

- **Recall:** How many of all **annotated** edges we got correct (best is 1)
- **Precision** How many of all **output** edges we got correct (best is 1)



Lesson 1

- **Trained detectors** (typically) perform better (true for all applications)
- In this case, the code seem to work better for finding object boundaries (edges) than finding text boundaries. Any idea **why**?
- What would you do if you wanted to detect text (e.g., licence plates)?
- **Think about your problem**, don't just use code as a black box

So much trouble for just edge computation...

Can we do something cool with it already?

S. Avidan and A. Shamir

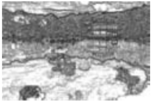
Seam Carving for Content-Aware Image Resizing

SIGGRAPH 2007

Paper: `http://www.win.tue.nl/~wstahw/edu/2IV05/seamcarving.pdf`

Simple Application: Seam Carving

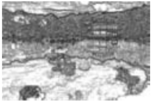
- Content-aware resizing



- Find path from top to bottom row with minimum gradient energy
- Remove (or replicate) those pixels

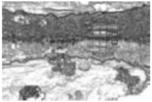
Simple Application: Seam Carving

- Content-aware resizing



Simple Application: Seam Carving

- Content-aware resizing



Seam Carving

- A vertical seam $\mathbf{s}$ is a list of column indices, one for each row, where each subsequent column differs by no more than one slot.
- Let G denote the image gradient magnitude. Optimal 8-connected path:

$$\mathbf{s}^* = \mathrm{argmin}_{\mathbf{s}} E(\mathbf{s}) = \mathrm{argmin}_{\mathbf{s}} \sum_{i=1}^{n} G(s_i)$$

- Can be computed via dynamic programming
- Compute the cumulative minimum energy for all possible connected seams at each entry (i, j):

$$M(i,j) = G(i,j) + \min\left(M(i-1,j-1), M(i-1,j), M(i-1,j+1)\right)$$

- Backtrack from min value in last row of M to pull out optimal seam path.

Seam Carving – Examples

- Implement seam carving for 5% extra credit on first assignment

Next time:

Image Features