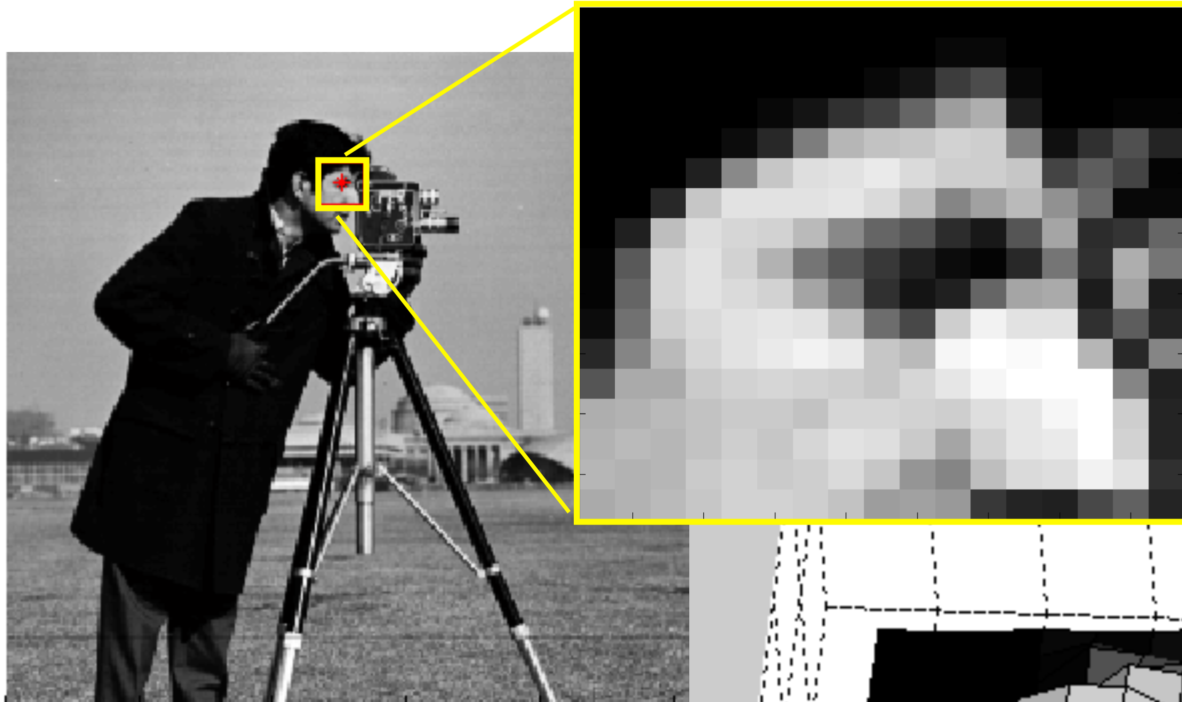
Robert Collins
CMPEN454

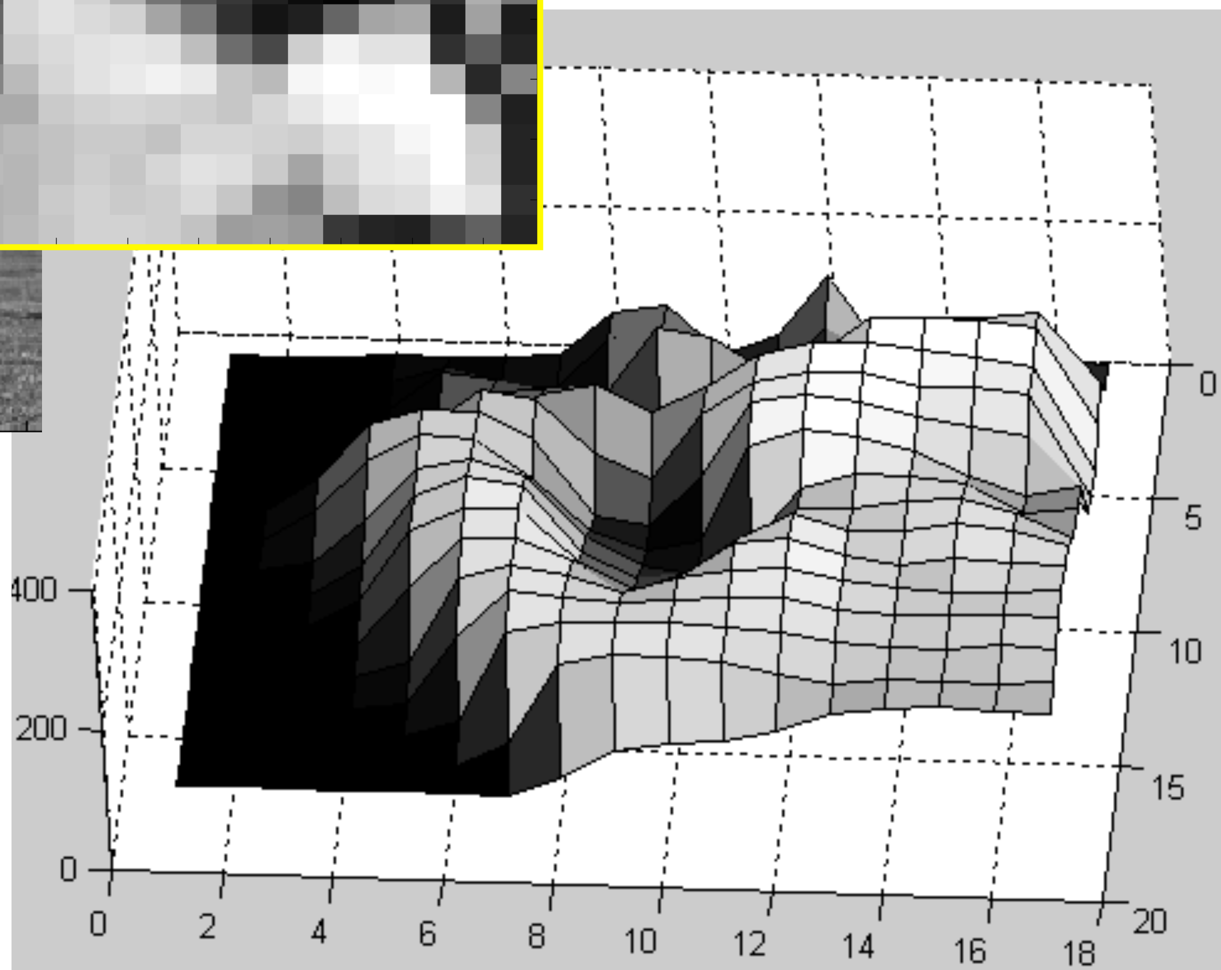
Lecture 3:
Linear Filters and Convolution

Background Reading: Section 3.2 in the Szeliski book or any image processing text.

Robert Collins
CMPEN454

Recall: Images as Surfaces





Surface height proportional to pixel grey value

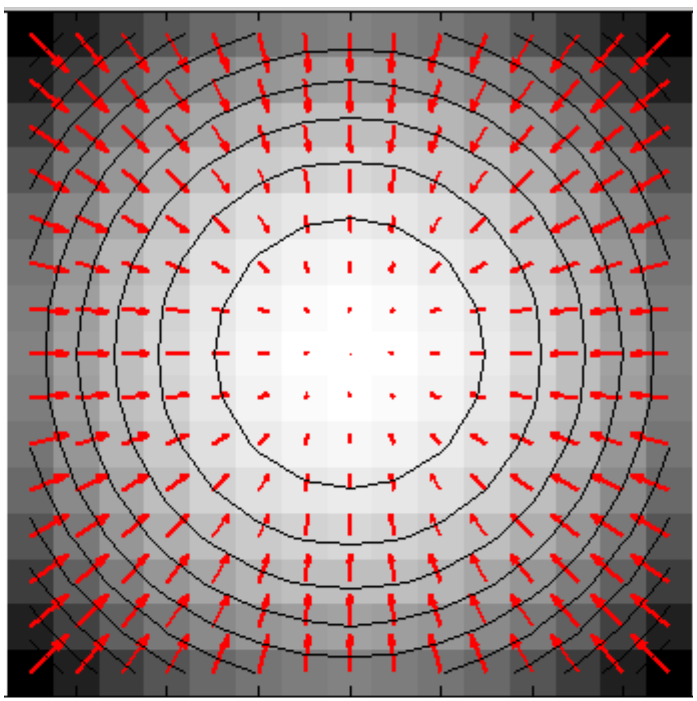
(dark=low, light=high)

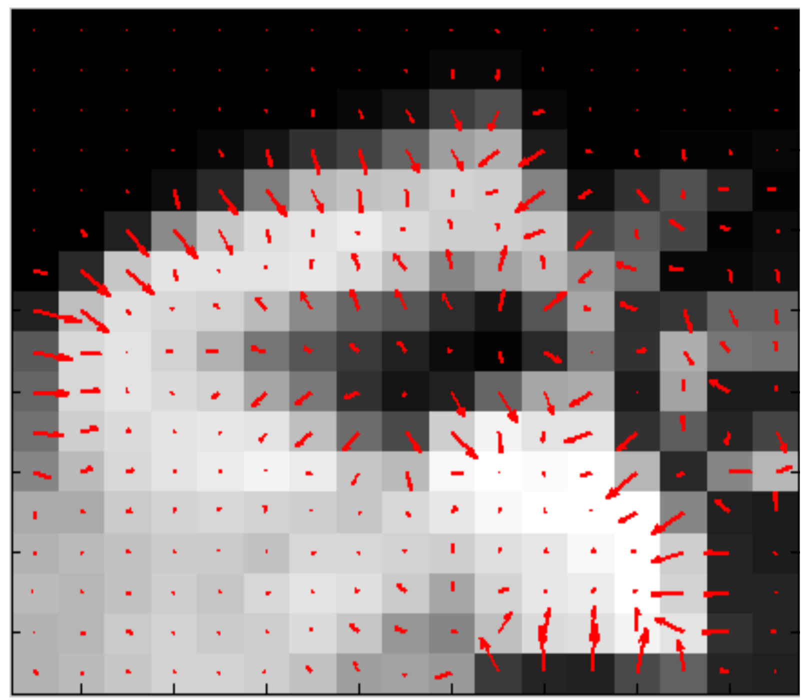
Robert Collins
CMPEN454

Recall: 2D Gradient

Gradient = vector of partial derivatives of image I(x,y)
= [dI(x,y)/dx , dI(x,y)/dy]

Gradient vector field indicates the direction and slope of steepest ascent (when considering image pixel values as a surface / height map).





Numerical Derivatives

See also T&V, Appendix A.2

Finite forward difference

$$\frac{f(x+h)-f(x)}{h} = f'(x) + O(h)$$

Finite backward difference

$$\frac{f(x)-f(x-h)}{h} = f'(x) + O(h)$$

Finite central difference

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + O(h^2)$$

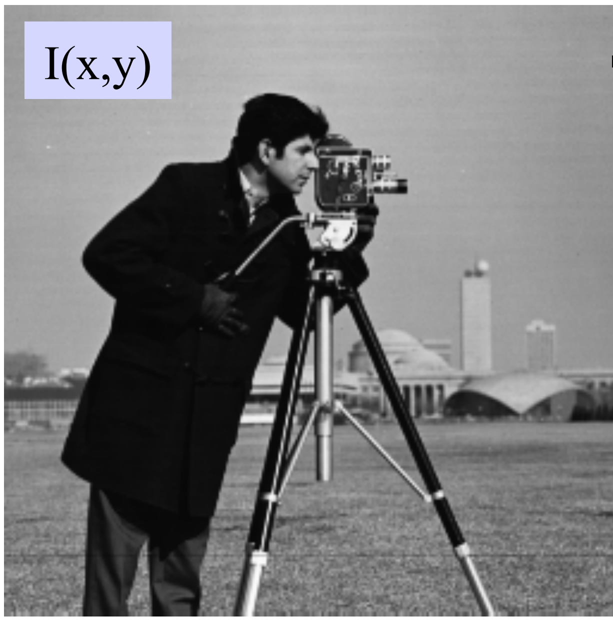
We are using this today.

Robert Collins
CMPEN454

Example: Spatial Image Gradients

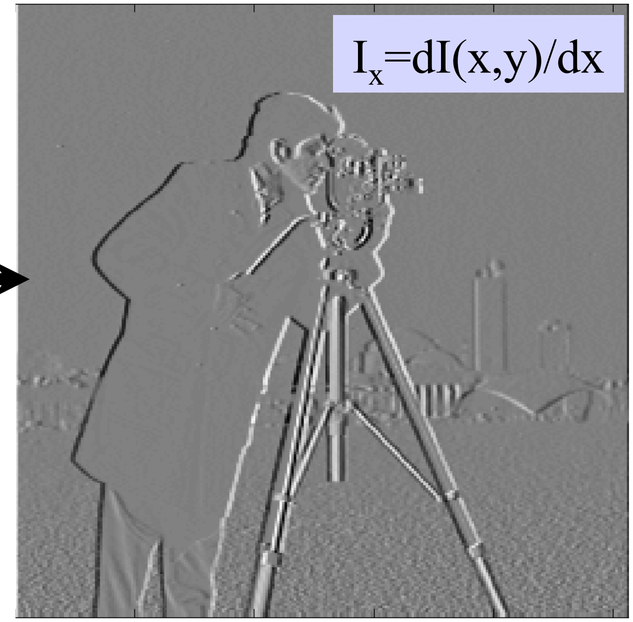
x

y



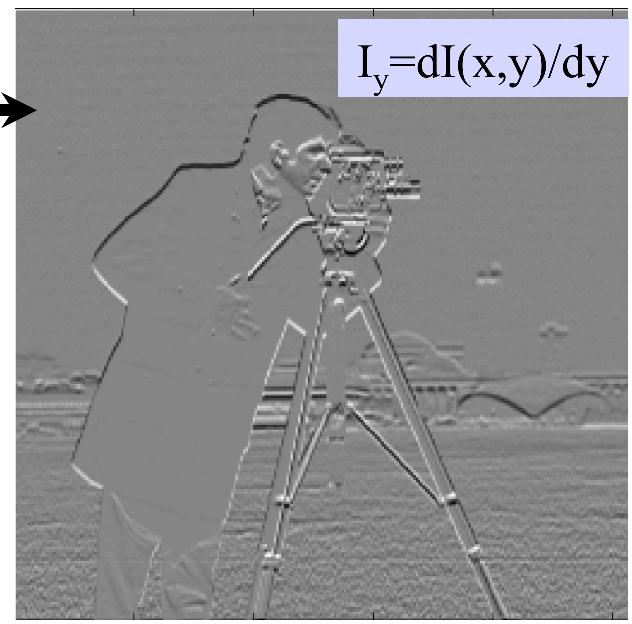
$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

Partial derivative wrt x



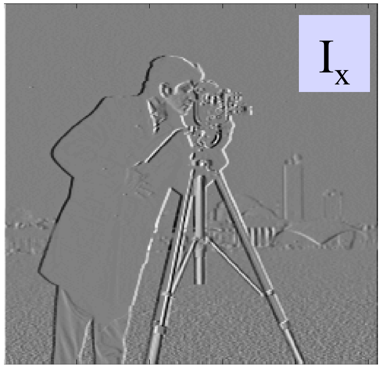
$$\frac{I(x,y+1) - I(x,y-1)}{2}$$

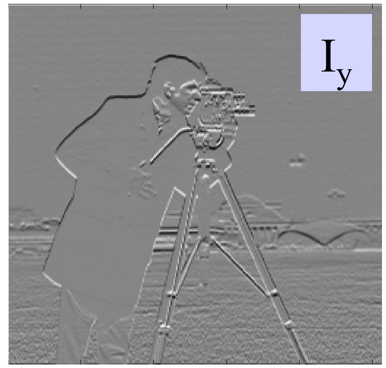
Partial derivative wrt y



finite central difference
approx to first derivative

Functions of Gradients



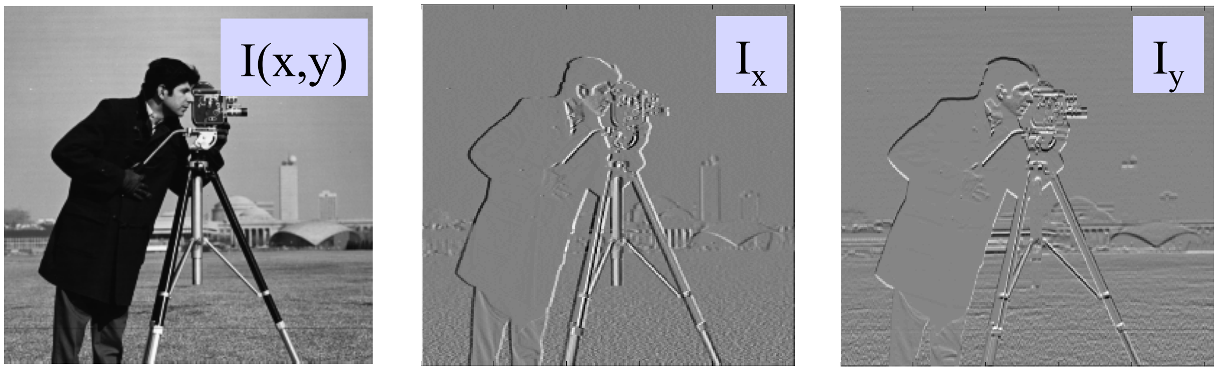


Magnitude of gradient
sqrt(Ix.^2 + Iy.^2)

Measures steepness of slope at each pixel

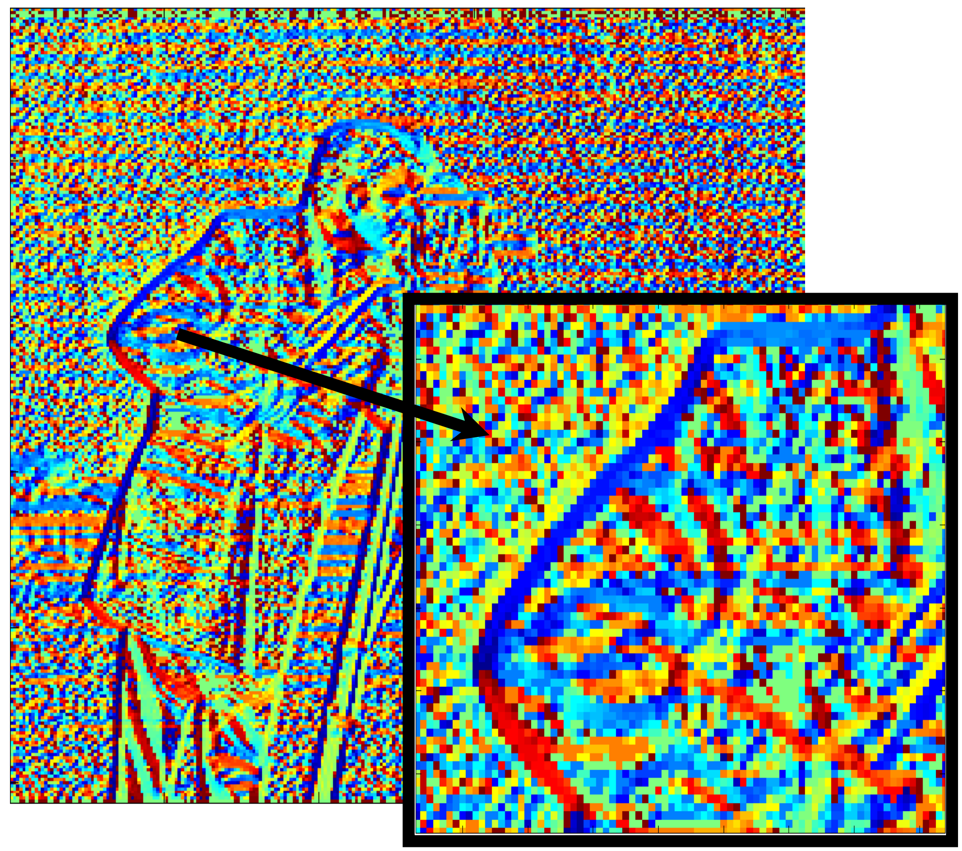
Good for indicating contours of objects / surface markings.

Functions of Gradients



Angle of gradient
atan2(Iy, Ix)

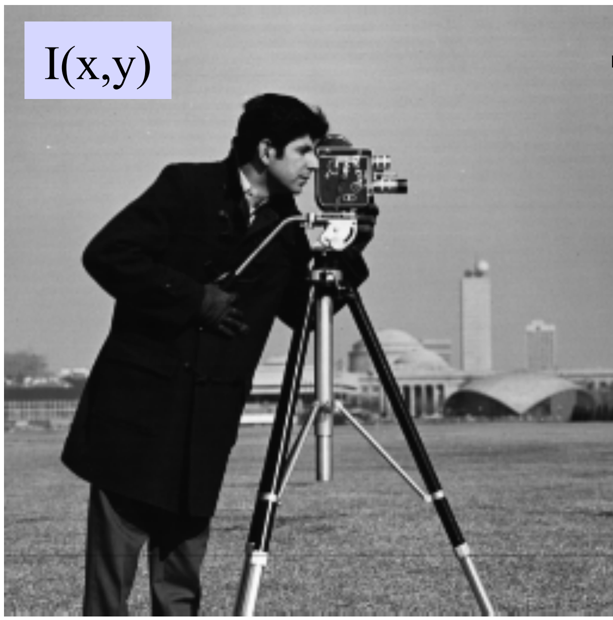
Denotes direction of slope



Linear Filters

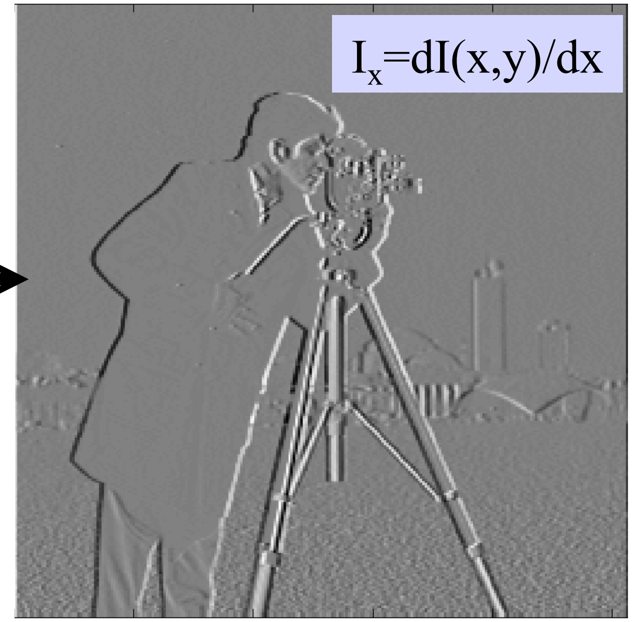
Gradients are an example of linear filters, i.e. the numeric value at a pixel is computed as a linear combination of values of neighboring pixels.

Example: Spatial Image Gradients



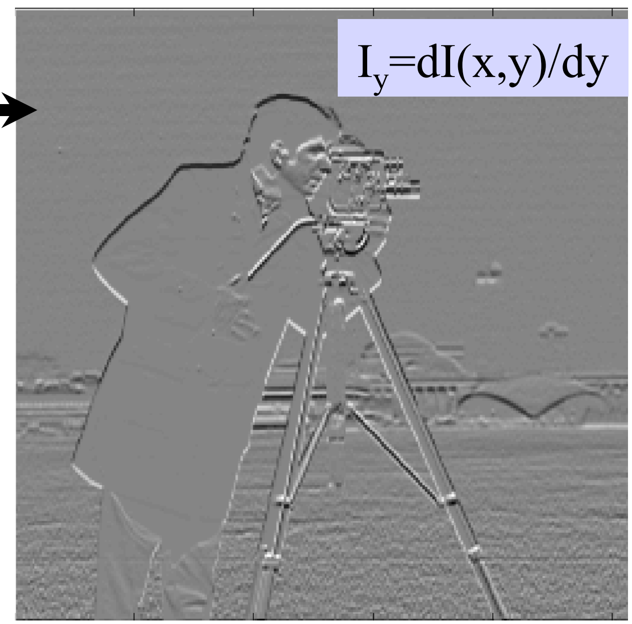
$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

Partial derivative wrt x



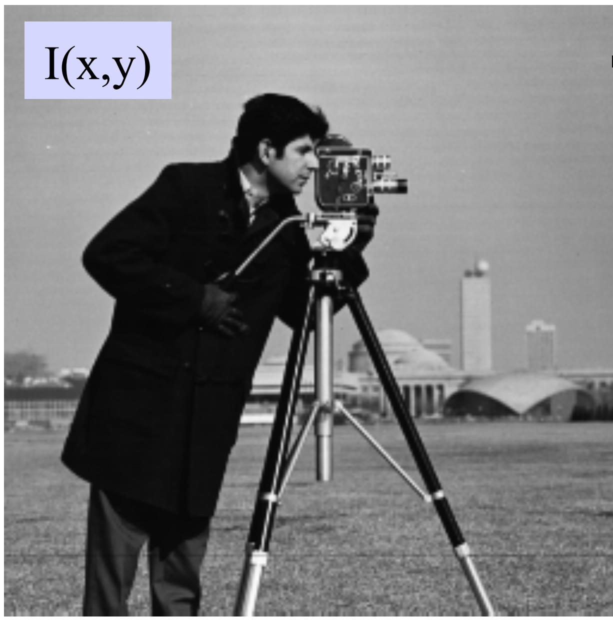
$$\frac{I(x,y+1) - I(x,y-1)}{2}$$

Partial derivative wrt y



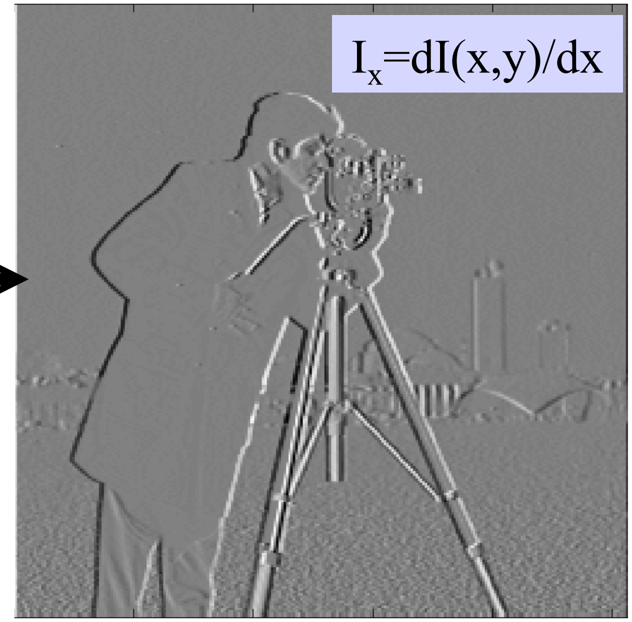
Robert Collins
CMPEN454

Example: Spatial Image Gradients



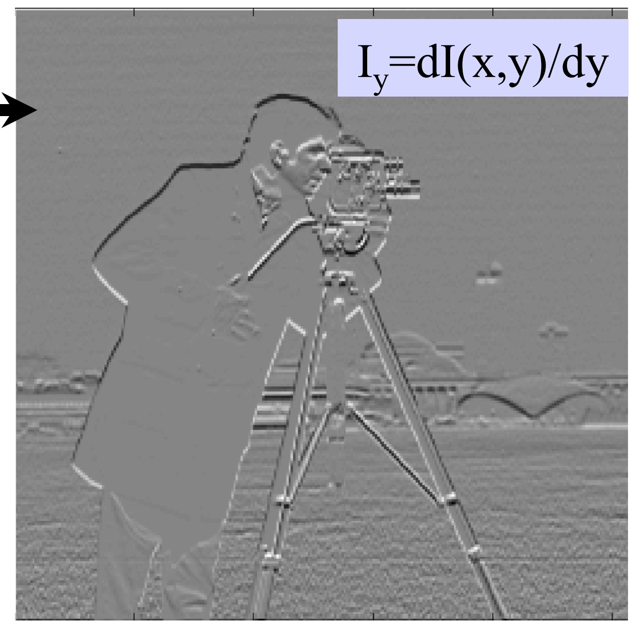
$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

Partial derivative wrt x



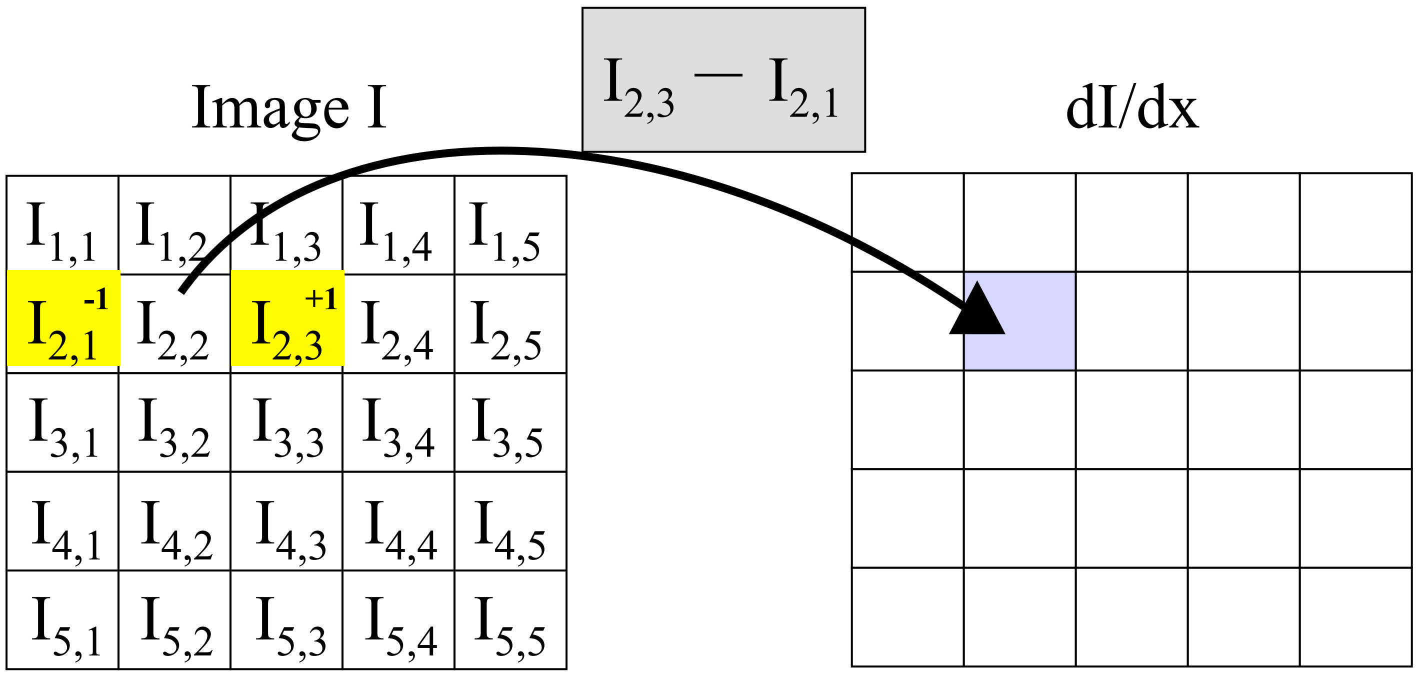
$$\frac{I(x,y+1) - I(x,y-1)}{2}$$

Partial derivative wrt y

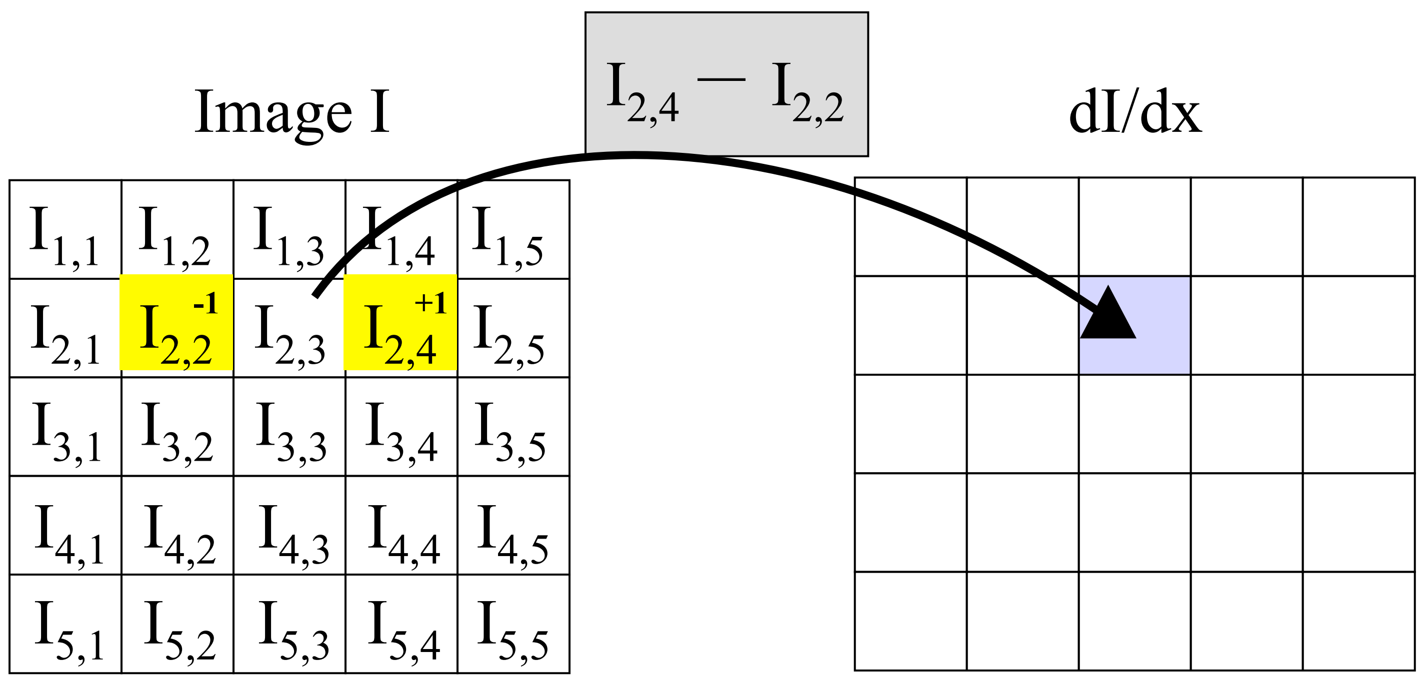


Note: From now on we will drop the constant factor 1/2. We can divide by it later.

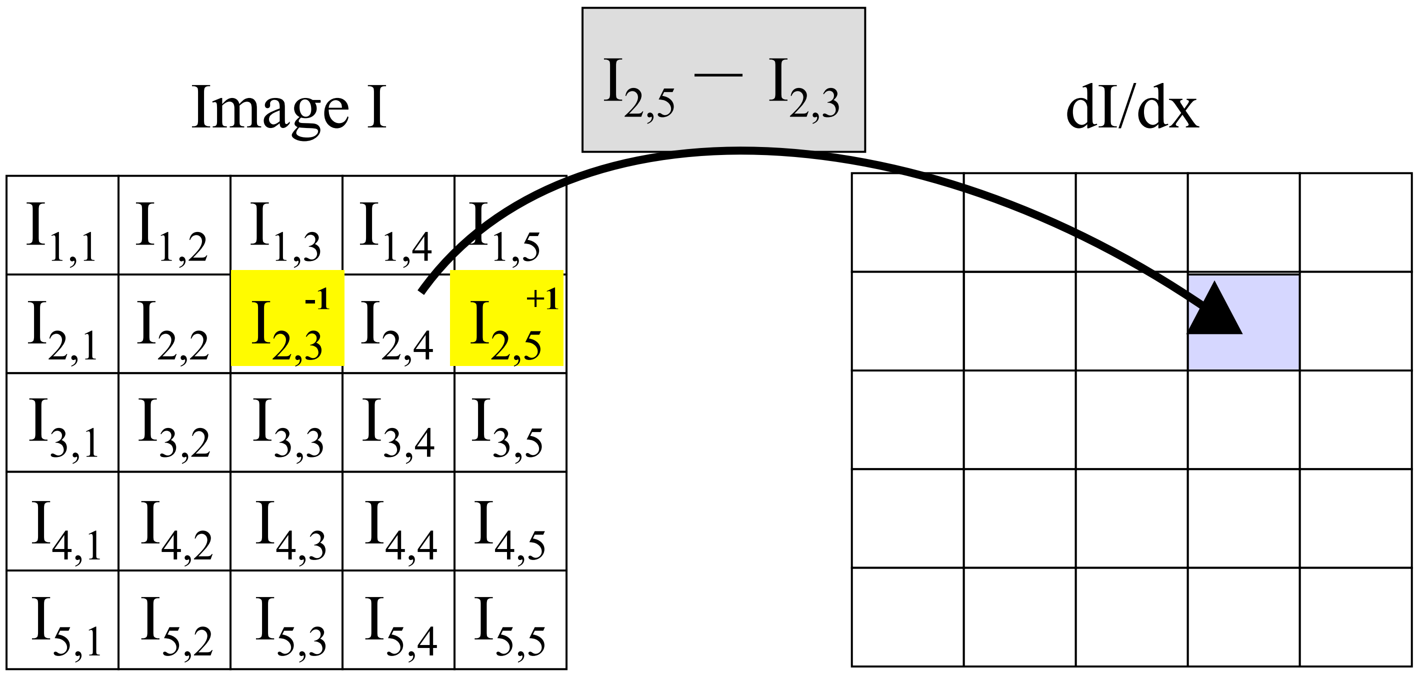
More Specifically



More Specifically

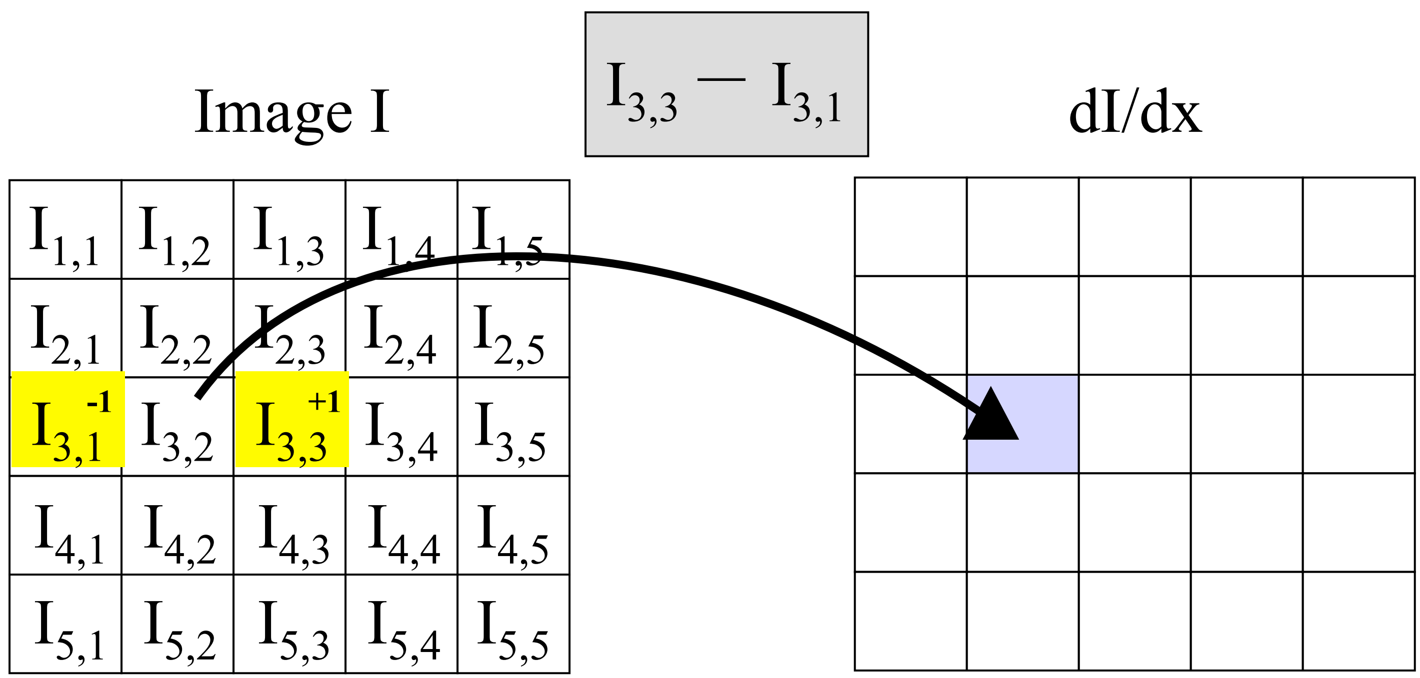


More Specifically



Robert Collins
CMPEN454

More Specifically



And so on …

So, how do we do this as a mathematical operator?

Convolution (2D)

Given a kernel (aka filter; template) f and an image h, the convolution h*f is defined as

$$h(x,y) * f \stackrel{C}{=} \int_v \int_u h(x-u, y-v) f(u,v) du dv$$
$$\stackrel{D}{=} \sum_i \sum_j h[x-i, y-j] f[i,j]$$

1) if f is of size (2m+1)x(2n+1), this formula indexes into f with i ranging from -m to m and j ranging from -n to n.
2) Note minus signs when indexing into neighborhood of h(x-i,y-j). As a result, f behaves as if rotated by 180 degrees before combining with h.
3) That doesn't matter if f has 180 degree symmetry
4) If it *does* matter, use cross correlation instead.

Simple Example (on the board)

Template f

Index	-1	0	1
	a	b	c

i.e. f(-1)=a ; f(0)=b ; f(1)=c

Image h

Index	1	2	3	4	5
	I_1	I_2	I_3	I_4	I_5

i.e. $h(i)= I_i$

Question: What is (h*f)(3) ?

Convolution Example

f

1	-1	-1
1	2	-1
1	1	1

Rotate

1	1	1
-1	2	1
-1	-1	1

Apply

h

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

f * h

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

2D example worked through by hand…

adapted from C. Rasmussen, U. of Delaware

Convolution Example

f

1	-1	-1
1	2	-1
1	1	1

Rotate

1	1	1
-1	2	1
-1	-1	1

Apply

h

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

f * h

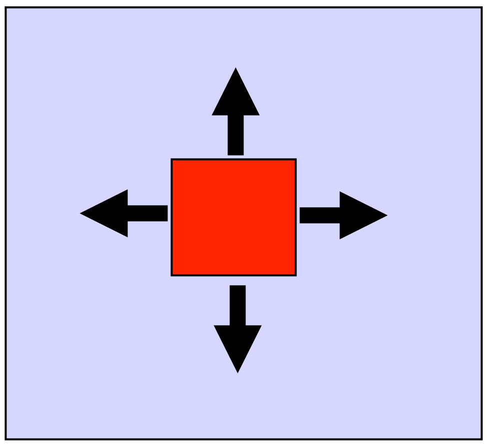
5	?	?	?
9	6	14	?
?	7	?	?
?	?	?	?

Can you fill in the rest?

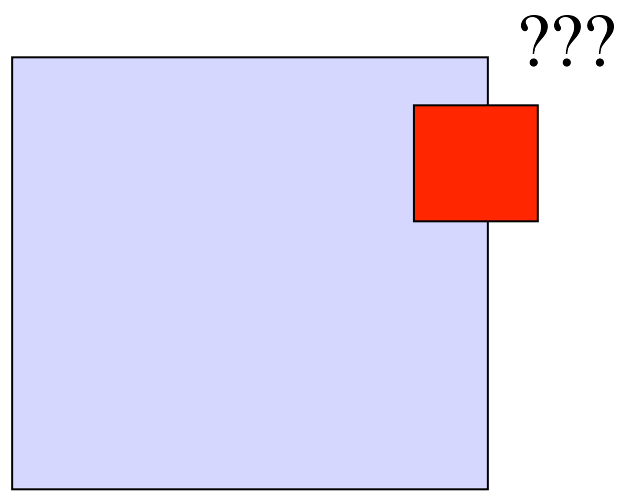
adapted from C. Rasmussen, U. of Delaware

Practical Issue: Border Handling

- Problem: what do we do for border pixels where the kernel does not completely overlap the image?



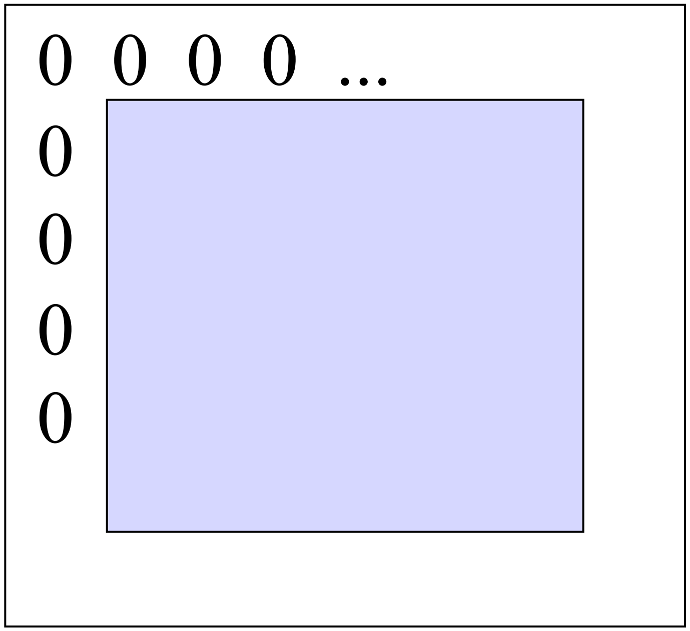
for interior pixels where there is full overlap, we know what to do.



but what values do we use for pixels that are "off the image" ?

Practical Issue: Border Handling

- Different border handling methods specify different ways of defining values for pixels that are off the image.
- One of the simplest methods is **zero-padding**, which we used by default in the earlier example.



Practical Issue: Border Handling

- Other methods...
- **Replication** – replace each off-image pixel with the value from the nearest pixel that IS in the image.

Example:

1	2	3
4	5	6
7	8	9

1s			1	2	3	3s		
			1	2	3			
			1	2	3			
1	1	1	1	2	3	3	3	3
4	4	4	4	5	6	6	6	6
7	7	7	7	8	9	9	9	9
			7	8	9			
7s			7	8	9	9s		
			7	8	9			

Practical Issue: Border Handling

- Other methods...
- **Reflection** – reflect pixel values at the border (as if there was a little mirror there)

Example:

1	2	3
4	5	6
7	8	9

9	8	7	7	8	9	9	8	7
6	5	4	4	5	6	6	5	4
3	2	1	1	2	3	3	2	1
3	2	1	1	2	3	3	2	1
6	5	4	4	5	6	6	5	4
9	8	7	7	8	9	9	8	7
9	8	7	7	8	9	9	8	7
6	5	4	4	5	6	6	5	4
3	2	1	1	2	3	3	2	1

Practical Issue: Border Handling

- Other methods...
- **Wraparound** – when going off the right border of the image, you wrap around to the left border. Similarly, when leaving the bottom of the image you reenter at the top. Basically, the image is a big donut (or torus).

Example:

1	2	3
4	5	6
7	8	9

1	2	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9
1	2	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9
1	2	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9

Convolution in Matlab

Imfilter(image,template{,option1,option2,…})

Boundary options: constant, symmetric, replicate, circular
Output size options: same as image, or full size (includes partial values computed when mask is off the image).
Corr or conv option: convolution rotates the template (as we have discussed). Correlation does not.

Type "help imfilter" on command line for more details

Correlation vs Convolution

Convolution

$$F * I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x-i, y-j)$$

The difference is this indexing

Correlation, also called cross-correlation

$$F \circ I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x+i, y+j)$$

Why not just use correlation?

- Partly historical – signal processing
- Partly mathematical – see next slide

Properties of Convolution

Commutative: f * g = g * f

Associative: (f * g) * h = f * (g * h)

Not true (in general) for correlation

Distributive: (f + g) * h = f * h + g * h

Linear: (a f + b g) * h = a f * h + b g * h

Shift Invariant: f(x+t) * h = (f * h)(x+t)

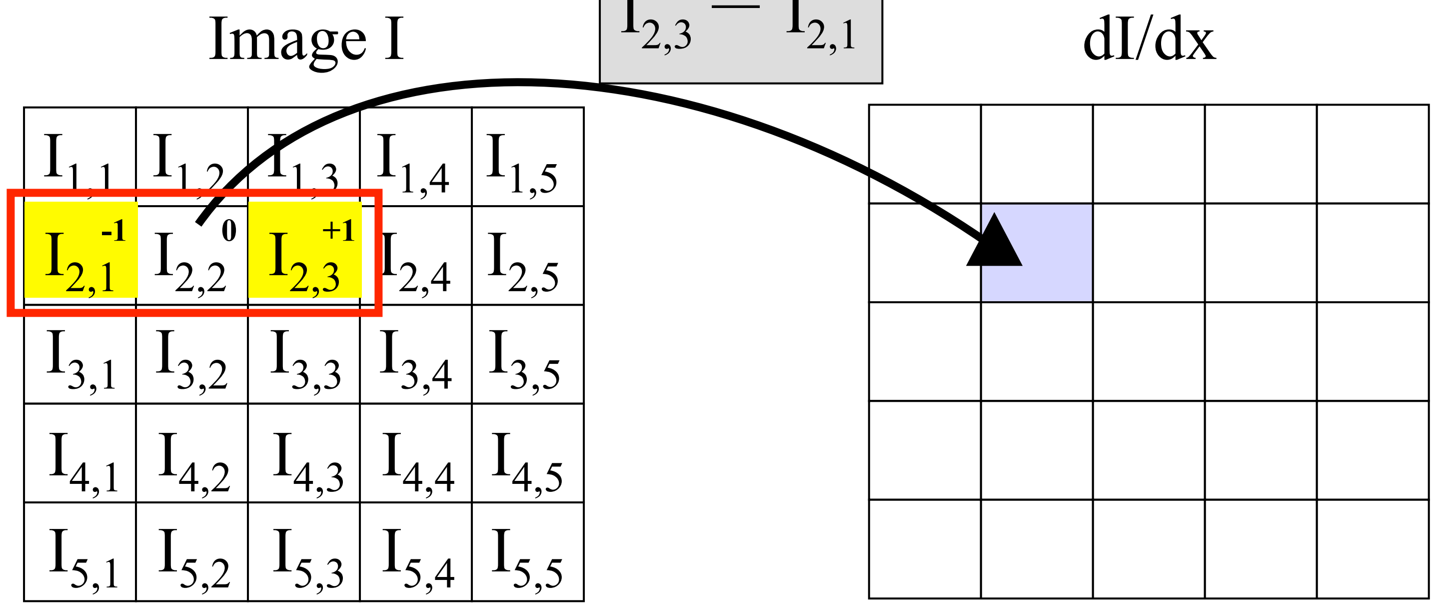
Differentiation rule:

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

Back to the Gradient…

$$(-1) * I_{2,1} + (0) * I_{2,2} + (+1) * I_{2,3}$$

$$I_{2,3} - I_{2,1}$$



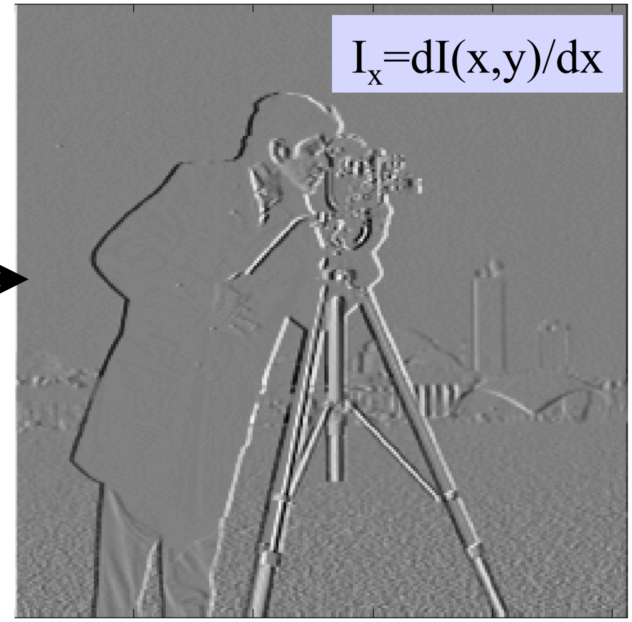
So, do I want to convolve image I with filter [-1 0 1]?

No! Since convolution flips the filter, this filter does not compute what we want. In fact, it computes the negative of the what we want. To compute precisely what we specified, we should use [1 0 -1] as the convolution filter, or else use correlation rather than convolution to do the computation.

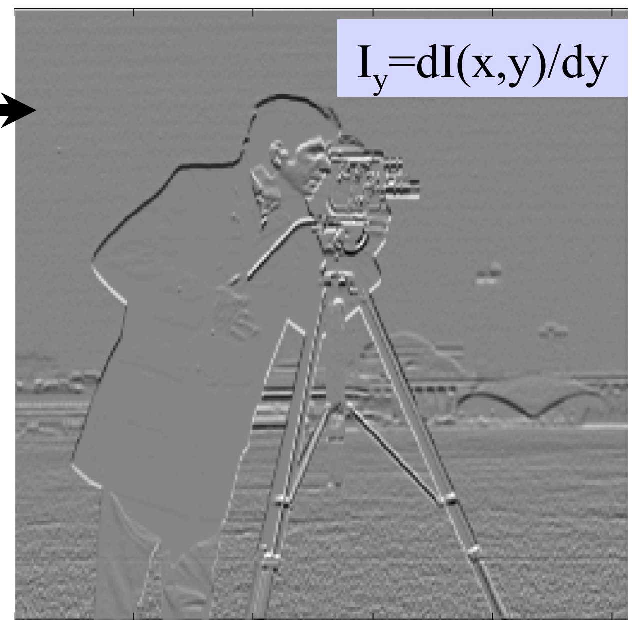
Example: Spatial Image Gradients

$$I_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * I$$

Partial derivative wrt x



Partial derivative wrt y

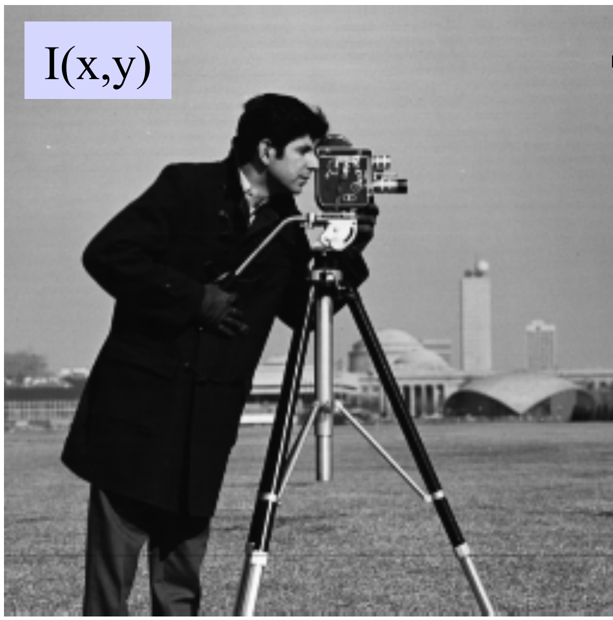


$$I_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * I$$

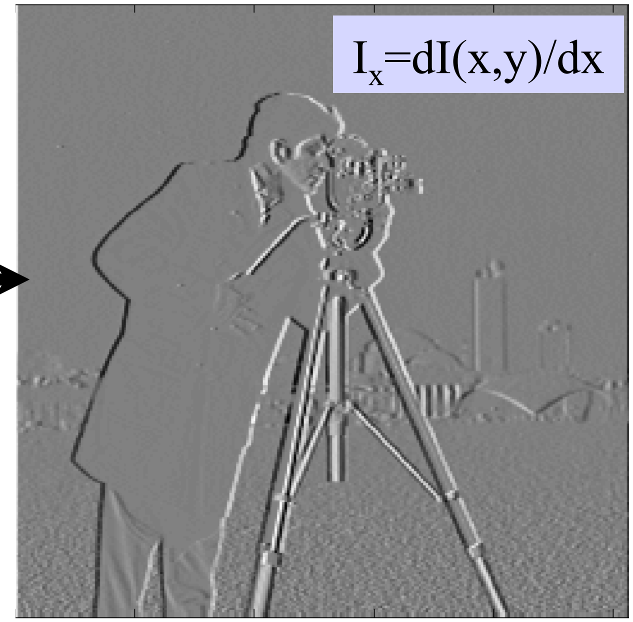
Note the kernel coefficients are rotated to counteract the implicit rotating that convolution will be doing to them.

Example: Spatial Image Gradients

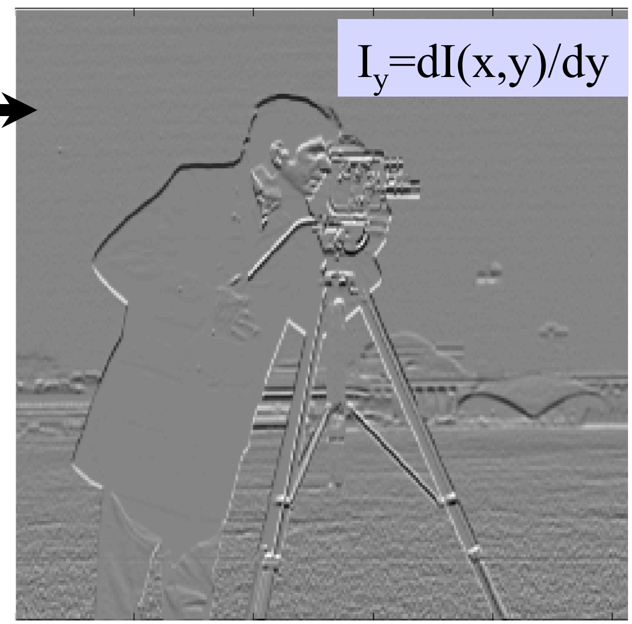
$$I_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * I$$



Partial derivative wrt x



Partial derivative wrt y



$$I_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * I$$

Also note that there is a difference between convolving with a 1xn row filter and an nx1 col filter.

Finite Difference Filters

Finite Differences computed using convolution filters

To compute dI/dx :

Convolve with:

Central difference

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + O(h^2)$$

1/2	0	-1/2

Forward difference

$$\frac{f(x+h)-f(x)}{h} = f'(x) + O(h)$$

1	-1	0

Backward difference

$$\frac{f(x)-f(x-h)}{h} = f'(x) + O(h)$$

0	1	-1

To compute dI/dy convolve with the transpose of these.

Be Aware!

Many authors / presenters (including me sometimes) are sloppy about the 180 degree rotation.

Finite Differences computed using convolution kernels

To compute dI/dx :

	Convolve with:		
Central difference $\frac{f(x+h)-f(x-h)}{2h} = f'(x) + O(h^2)$	1/2	0	-1/2
Forward difference $\frac{f(x+h)-f(x)}{h} = f'(x) + O(h)$	1	-1	0
Backward difference $\frac{f(x)-f(x-h)}{h} = f'(x) + O(h)$	0	1	-1

The justification for this is that when they say "convolution" (which rotates the filter), what they really have in mind is cross correlation (which does not rotate the filter). It's an easy mistake to make, as otherwise the operations are identical.

However, as a student, you don't have the luxury of making that mistake unpenalized.