Robert Collins
CSE486, Penn State

Lecture 3:
Linear Operators

Robert Collins
CSE486, Penn State

Administrivia

I have put some Matlab image tutorials on Angel. Please take a look if you are unfamiliar with Matlab or the image toolbox.

I have posted Homework 1 on Angel. It is due next Friday at beginning of class.

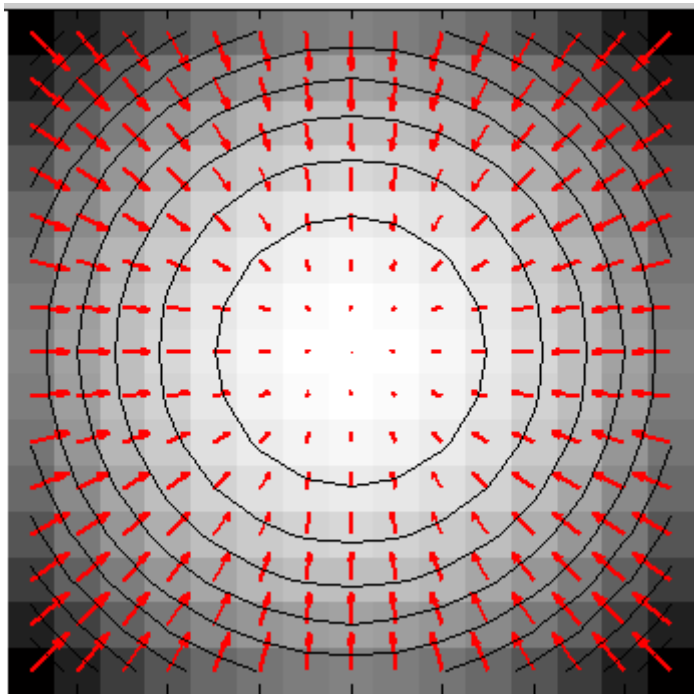
Robert Collins
CSE486, Penn State

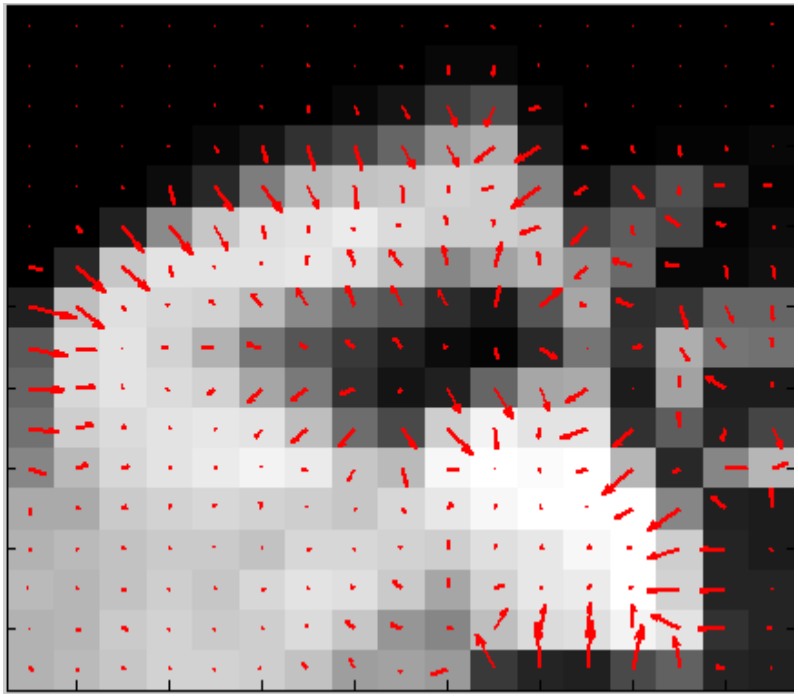
Recall: 2D Gradient

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).





Robert Collins
CSE486, Penn State

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

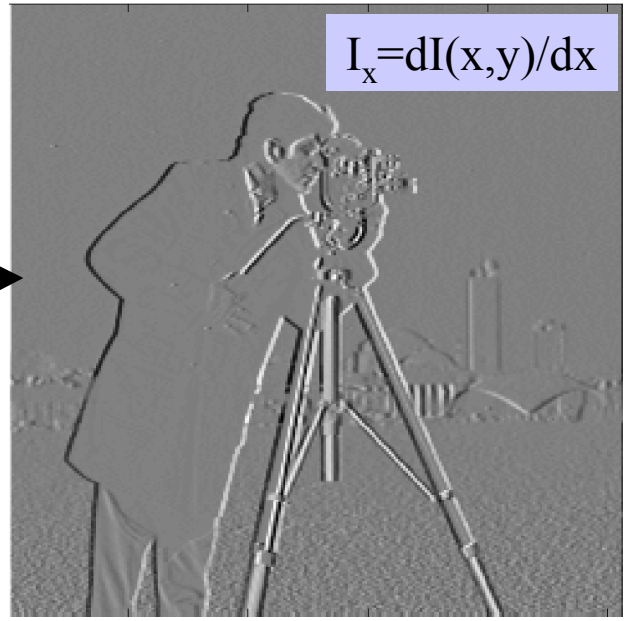
$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + O(h^2)$$

More accurate

Example: Spatial Image Gradients

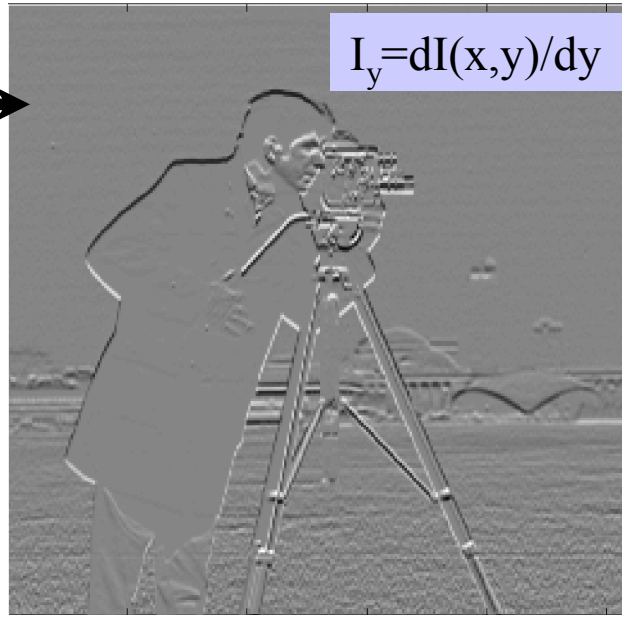
$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

Partial derivative wrt x



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

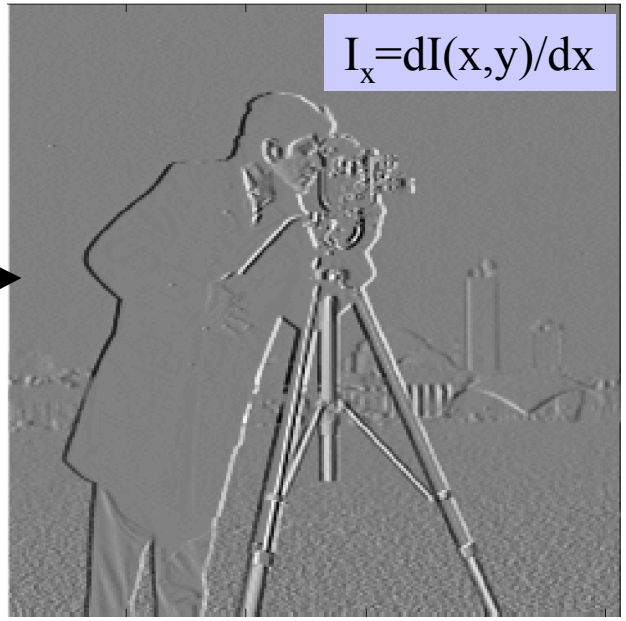
Partial derivative wrt y



Example: Spatial Image Gradients

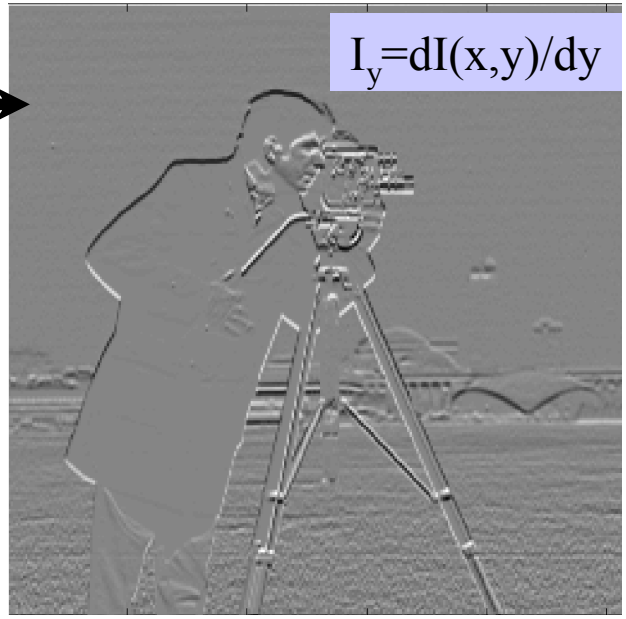
$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

Partial derivative wrt x



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

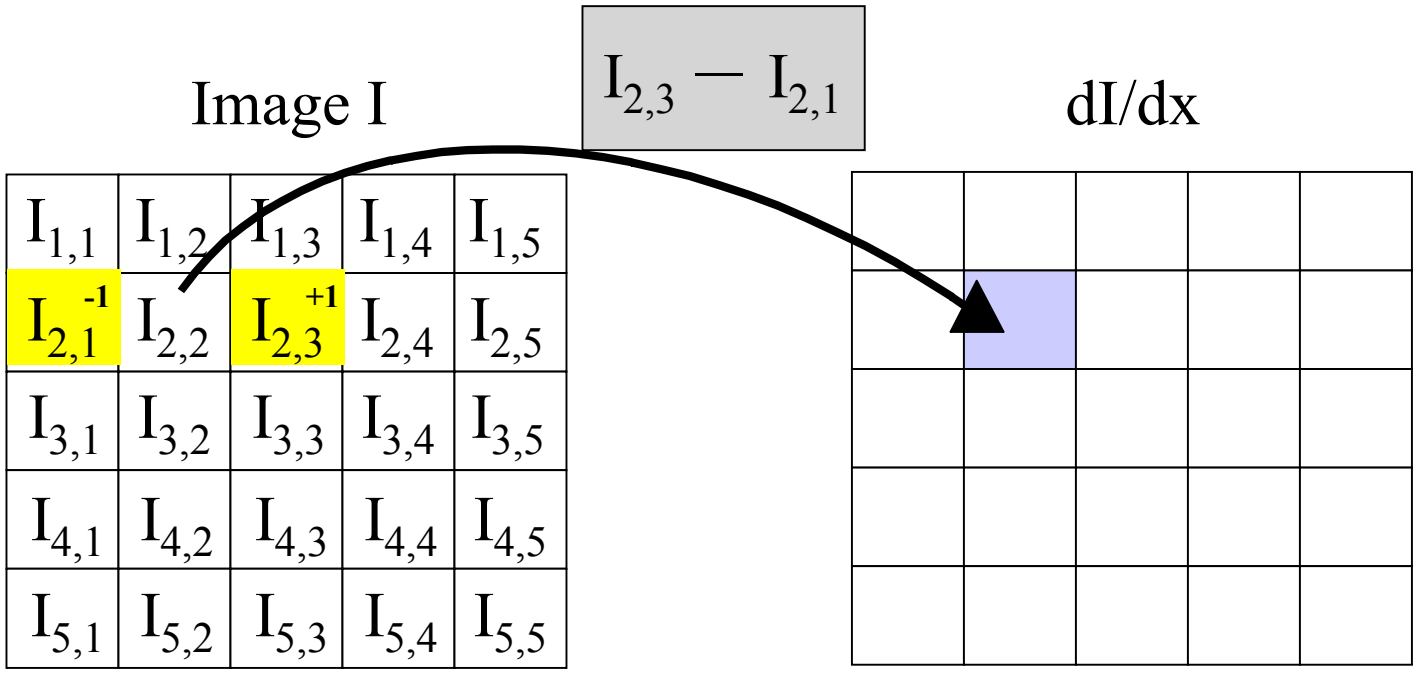
Partial derivative wrt y



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

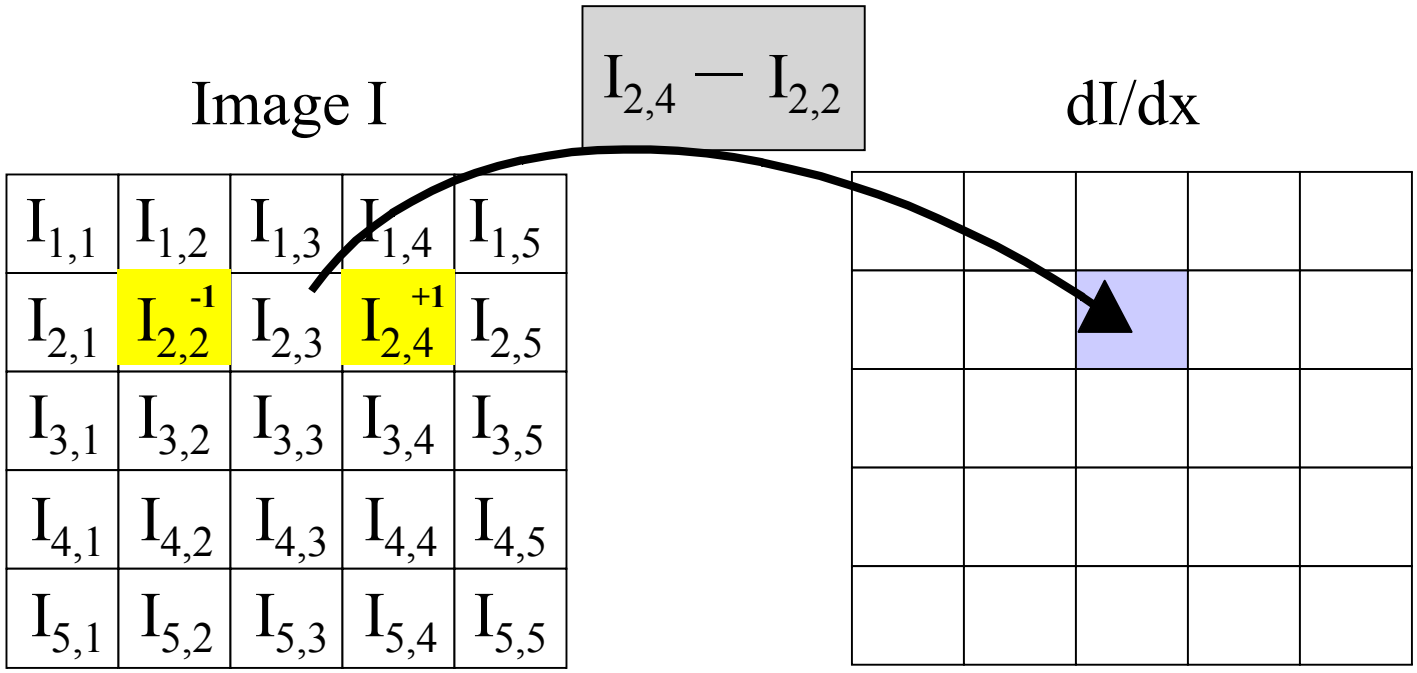
Robert Collins
CSE486, Penn State

More Specifically



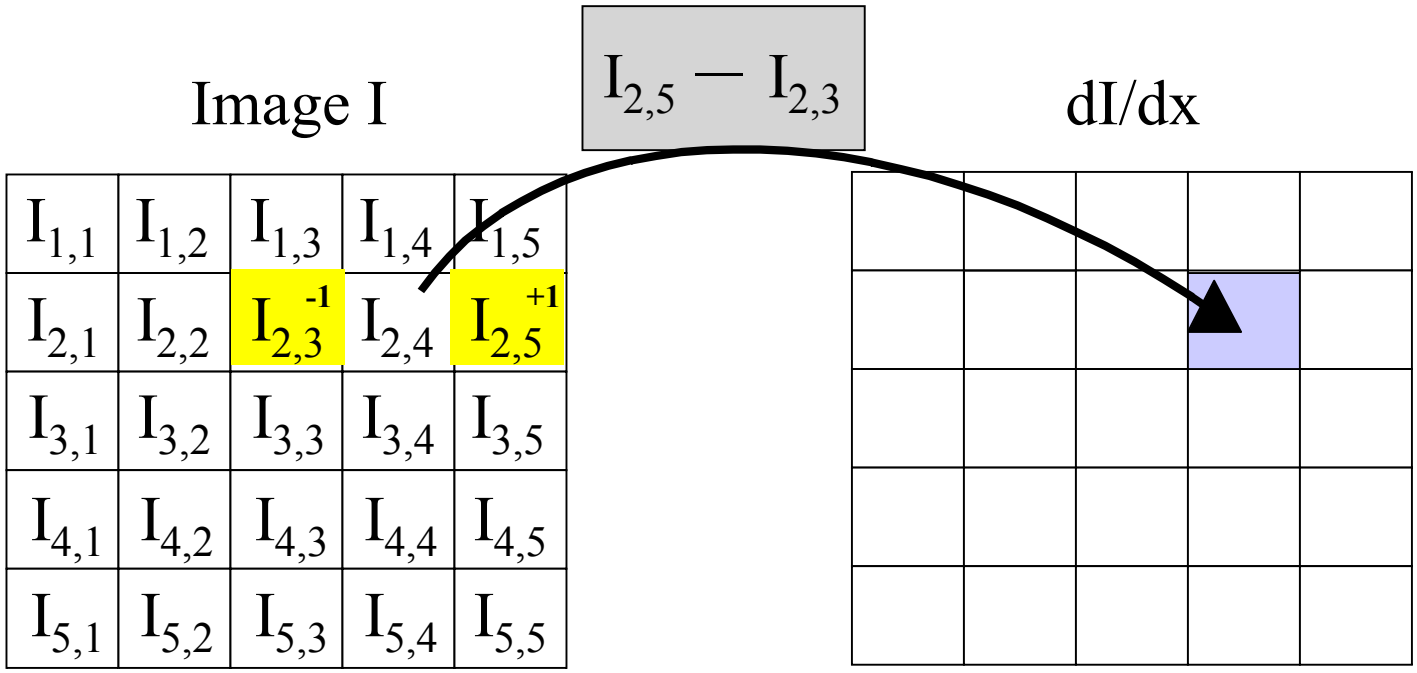
Robert Collins
CSE486, Penn State

More Specifically



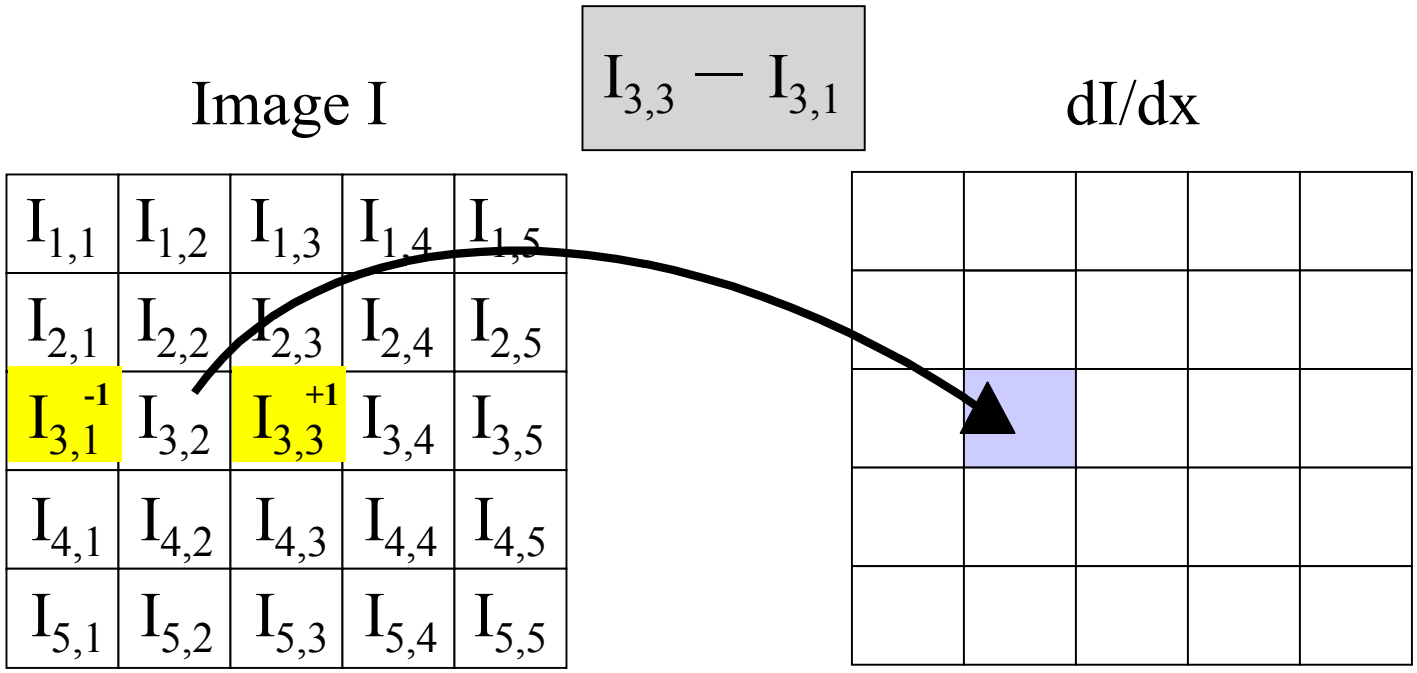
Robert Collins
CSE486, Penn State

More Specifically



Robert Collins
CSE486, Penn State

More Specifically



And so on …

Robert Collins
CSE486, Penn State

Linear Filters

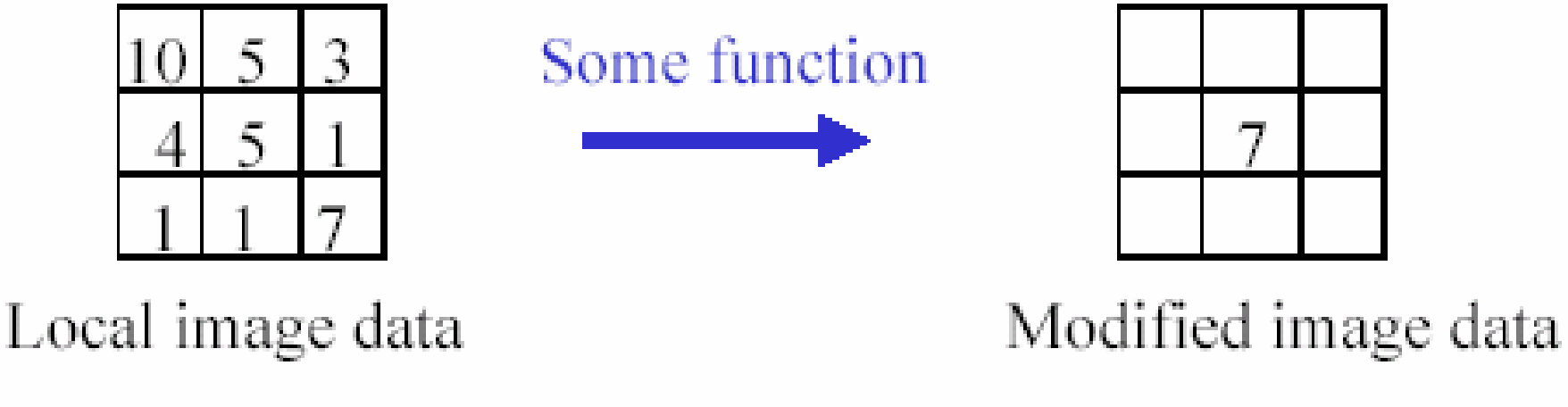
- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Example: smoothing by averaging
  - form the average of pixels in a neighbourhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood
- Example: finding a derivative
  - form a weighted average of pixels in a neighbourhood

Note: The “Linear” in “Linear Filters” means linear combination of neighboring pixel values.

Robert Collins
CSE486, Penn State

Image Filtering

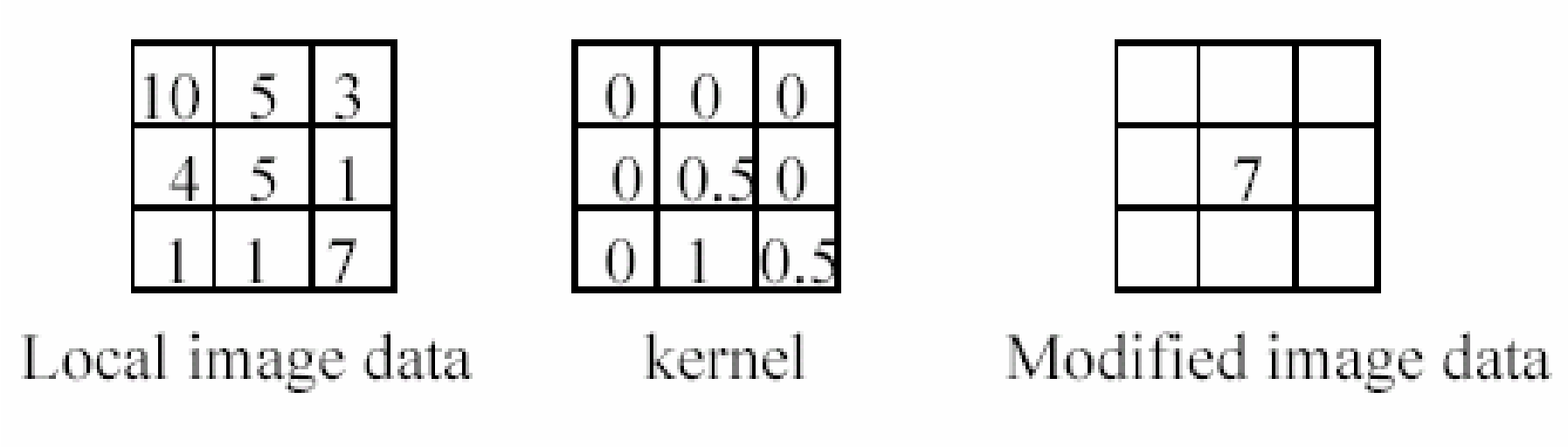
- Modify the pixels in an image based on some function of a local neighborhood of the pixels.



Robert Collins
CSE486, Penn State

Linear Filtering

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".



think of this as a weighted sum (kernel specifies the weights):

10*0+5*0+3*0+4*0+5*.5+1*0+1*0+1*1+7*.5 = 7

Robert Collins
CSE486, Penn State

Linear Filtering

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	7	

Modified image data

We don't want to only do this at a single pixel, of course, but want instead to "run the kernel over the whole image".

Robert Collins
CSE486, Penn State

Convolution (2D)

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

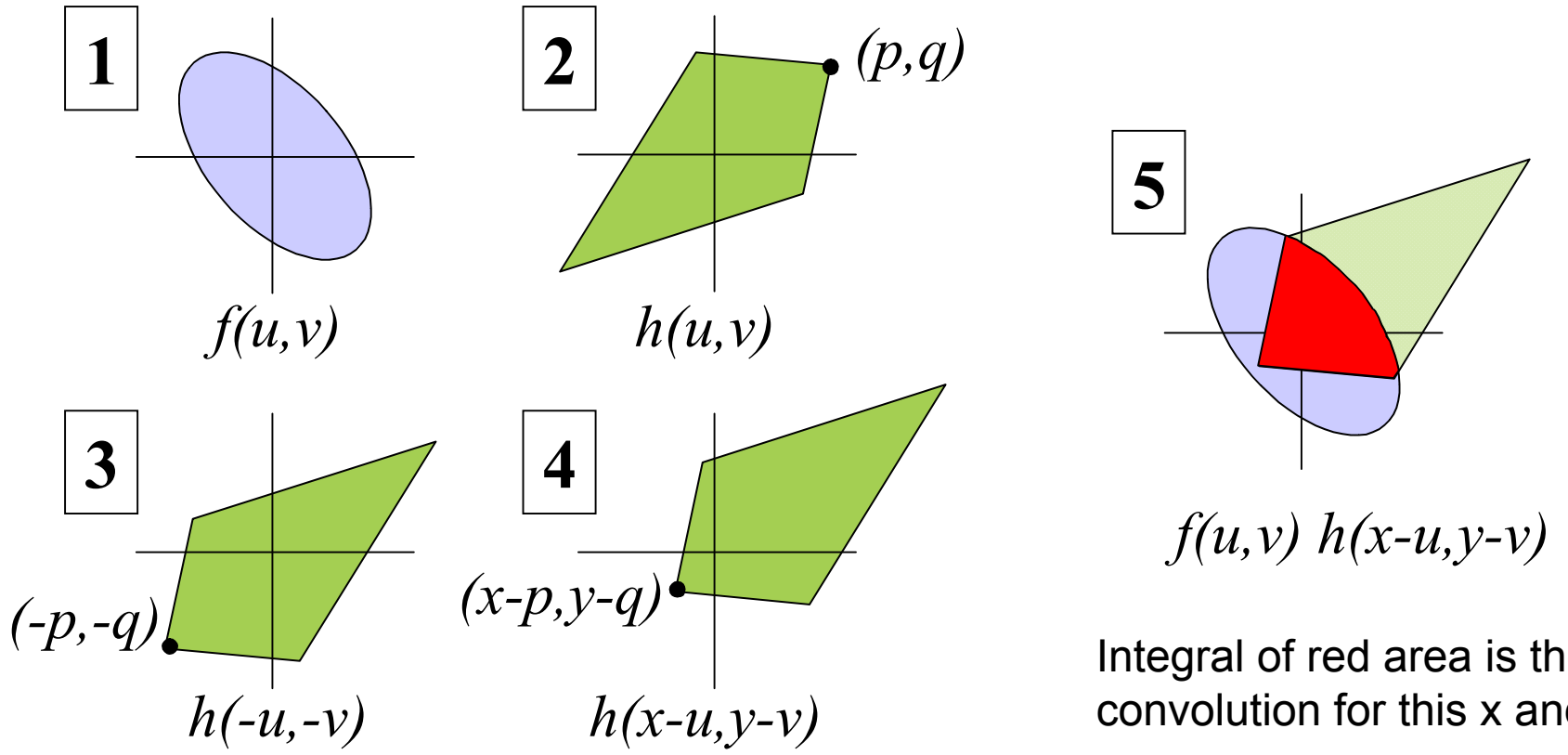
$$h(x,y) * f(x,y) \stackrel{C}{=} \int_v \int_u h(x-u, y-v) f(u,v) du dv$$
$$\stackrel{D}{=} \sum_i \sum_j h[x-i, y-j] f[i,j]$$

1) Note strange indexing into neighborhood of h(x,y). As a result, f behaves as if rotated by 180 degrees before combining with h.
2) That doesn't matter if f has 180 deg symmetry
3) If it *does* matter, use cross correlation instead.

Robert Collins
CSE486, Penn State

Convolution

$$h(x,y) * f(x,y) \stackrel{C}{=} \int_v \int_u h(x-u, y-v) f(u,v) du dv$$



Integral of red area is the convolution for this x and y

Adapted from Rosenfeld and Kak, Digital Picture Processing

Robert Collins
CSE486, Penn State

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

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

adapted from C. Rasmussen, U. of Delaware

Robert Collins
CSE486, Penn State

Convolution Example

f

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

Rotate ↓

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

Apply ↗

1	1	1		
-1	2 2	1 2	2	3
-1	-1 2	1 1	3	3
	2	2	1	2
	1	3	2	2

f * h

5	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

$2*2+1*2+(-1)*2+1*1 = 5$

Robert Collins
CSE486, Penn State

Convolution Example

f

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

Rotate ↓

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

Apply ↗

1	1	1	
-1 2	2 2	1 2	3
-1 2	-1 1	1 3	3
2	2	1	2
1	3	2	2

f * h

5	4	?	?
?	?	?	?
?	?	?	?
?	?	?	?

$2*2+1*2+(-1)*2+1*1 = 5$

$-1*2+2*2+1*2-1*2-1*1+1*3 = 4$

Robert Collins
CSE486, Penn State

Convolution Example

f

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

Rotate ↓

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

Apply ↗

	1	1	1
2	-1 2	2 2	1 3
2	-1 1	-1 3	1 3
2	2	1	2
1	3	2	2

f * h

5	4	4	?
?	?	?	?
?	?	?	?
?	?	?	?

$2*2+1*2+(-1)*2+1*1 = 5$

$-1*2+2*2+1*2-1*2-1*1+1*3 = 4$

$-1*2+2*2+1*3-1*1-1*3+1*3 = 4$

Robert Collins
CSE486, Penn State

Convolution Example

f

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

Rotate ↓

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

Apply ↗

		1	1	1
2	2	-1 2	2 3	1
2	1	-1 3	-1 3	1
2	2	1	2	
1	3	2	2	

f * h

5	4	4	-2
?	?	?	?
?	?	?	?
?	?	?	?

2*2+1*2+(-1)*2+1*1 = 5

-1*2+2*2+1*2-1*2-1*1+1*3= 4

-1*2+2*2+1*3-1*1-1*3+1*3= 4

-1*2+2*3-1*3-1*3 = -2

Robert Collins
CSE486, Penn State

Convolution Example

f

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

Rotate ↓

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

Apply ↗

1	1 2	1 2	2	3
-1	2 2	1 1	3	3
-1	-1 2	1 2	1	2
	1	3	2	2

f * h

5	4	4	-2
9	?	?	?
?	?	?	?
?	?	?	?

2*2+1*2+(-1)*2+1*1 = 5
-1*2+2*2+1*2-1*2-1*1+1*3= 4
-1*2+2*2+1*3-1*1-1*3+1*3= 4
-1*2+2*3-1*3-1*3 = -2
1*2+1*2+2*2+1*1-1*2+1*2= 9

Robert Collins
CSE486, Penn State

Convolution Example

f

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

Rotate ↓

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

Apply ↗

1 2	1 2	1 2	3
-1 2	2 1	1 3	3
-1 2	-1 2	1 1	2
1	3	2	2

f * h

5	4	4	-2
9	6	?	?
?	?	?	?
?	?	?	?

2*2+1*2+(-1)*2+1*1 = 5

-1*2+2*2+1*2-1*2-1*1+1*3= 4

-1*2+2*2+1*3-1*1-1*3+1*3= 4

-1*2+2*3-1*3-1*3 = -2

1*2+1*2+2*2+1*1-1*2+1*2= 9

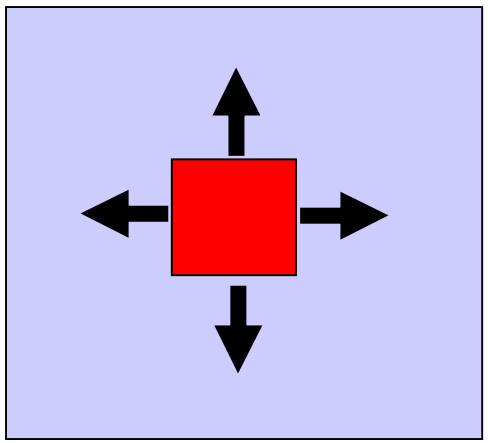
1*2+1*2+1*2-1*2+2*1+1*3-1*2-1*2+1*1= 6

Convolution Example

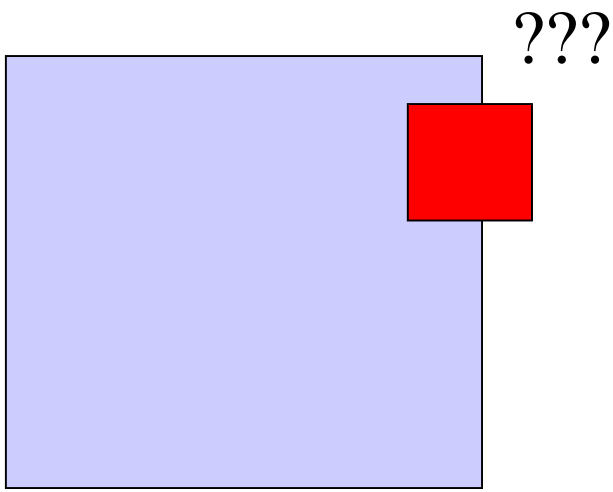
and so on…

Practical Issue: Border Handling

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



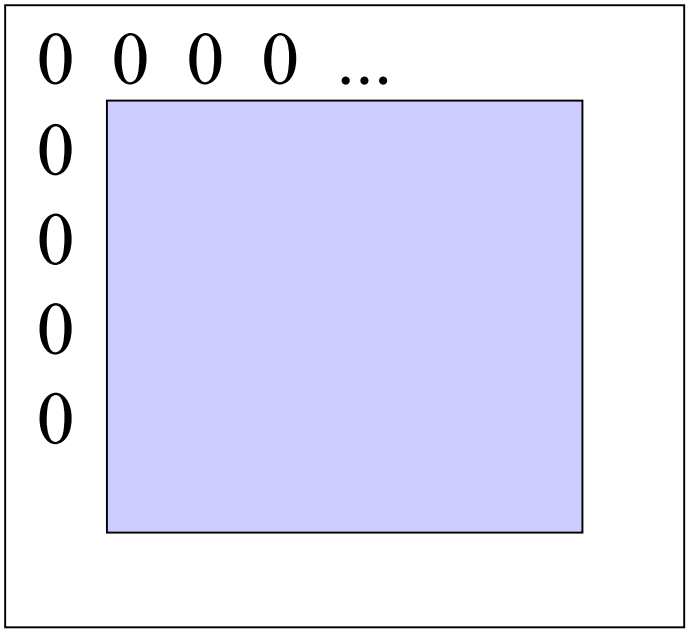
for interior pixels where there is full overlap, we know what to do.



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

Practical Issue: Border Handling

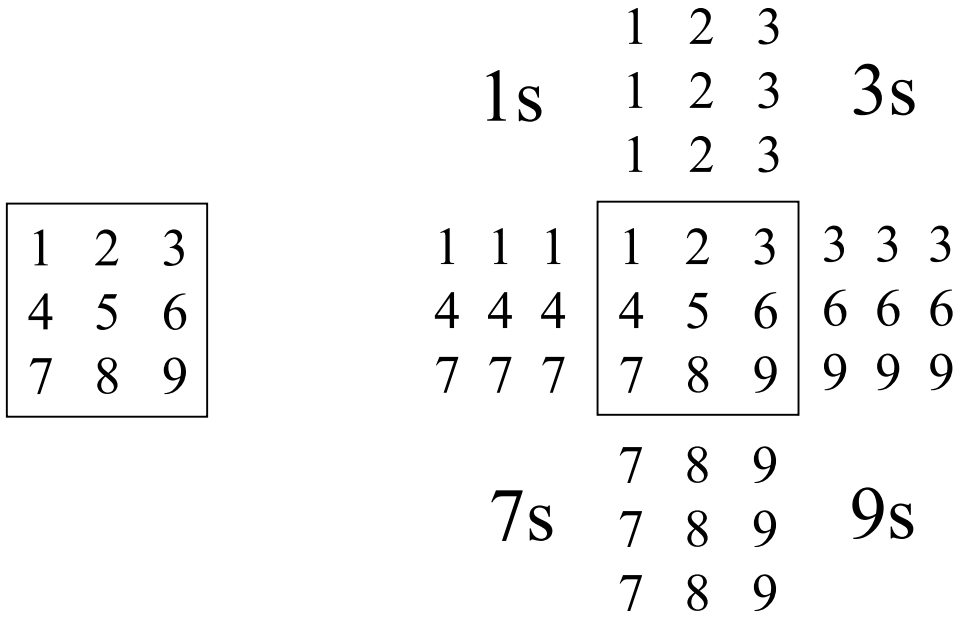
- Different border handling methods specify different ways of defining values for pixels that are off the image.
- One of the simplest methods is **zero-padding**, which we used by default in the earlier example.



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:



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

Robert Collins
CSE486, Penn State

Convolution in Matlab

Could use conv and conv2, but newer versions use:

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

Robert Collins
CSE486, Penn State

Properties of Convolution

Commutative: f * g = g * f

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

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)

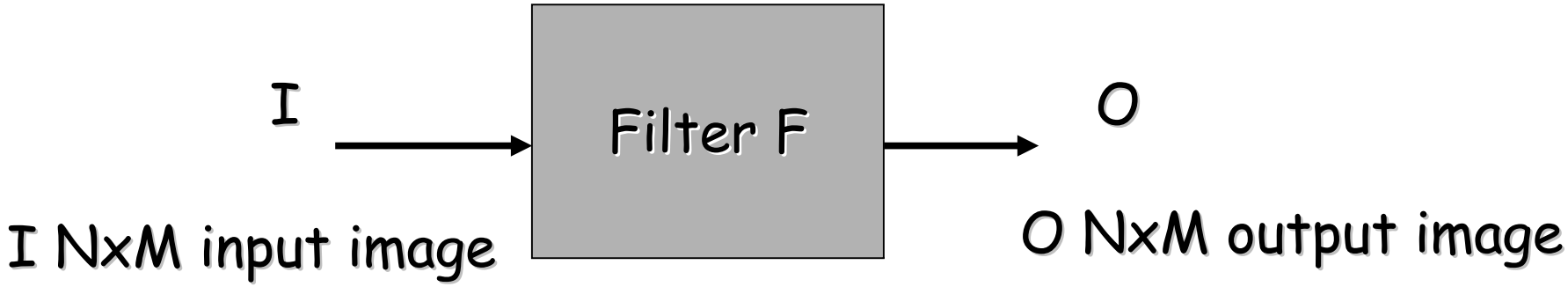
Differentiation rule:

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

You will see some of these again!

Robert Collins
CSE486, Penn State

Linear Filtering



F (2m+1)x(2m+1) mask

e.g. F= $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ for m=1

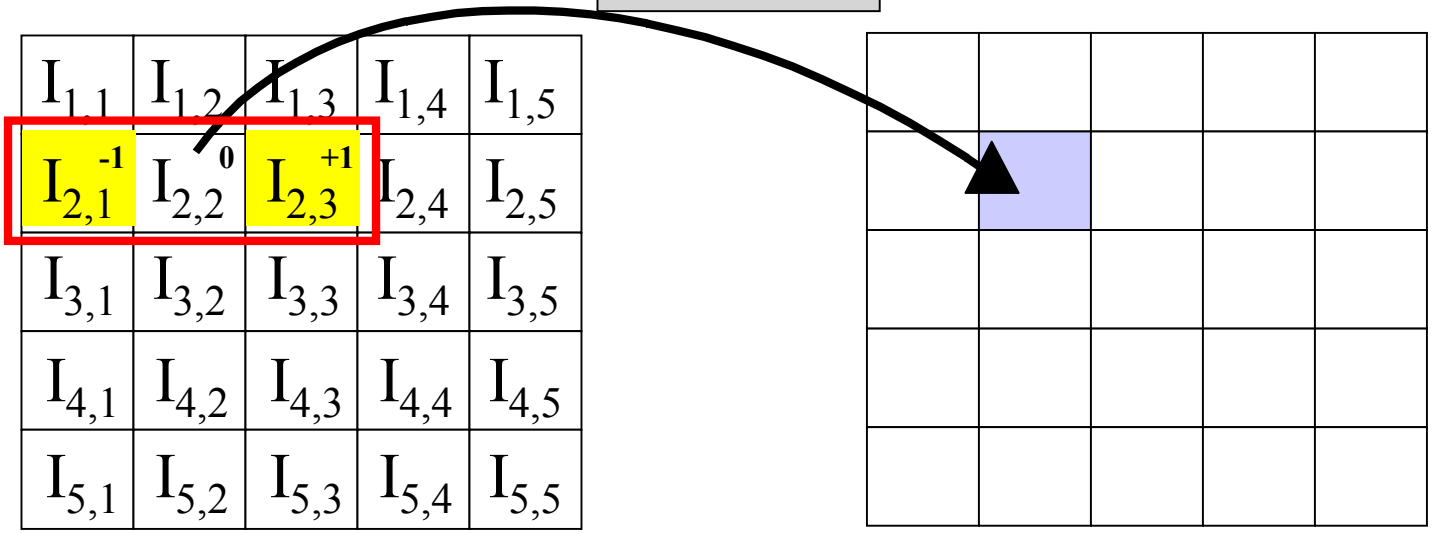
$$O(i,j) = \sum_{h=-m}^{h=m} \sum_{k=-m}^{k=m} I(i-h, j-k) \cdot F(h,k)$$

Robert Collins
CSE486, Penn State

Back to the Gradient…

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

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



Robert Collins
CSE486, Penn State

Finite Difference Filters

Finite Differences computed using convolution kernels

Vertical Edges:

Convolve with:

-1	0	1

Horizontal Edges:

Convolve with:

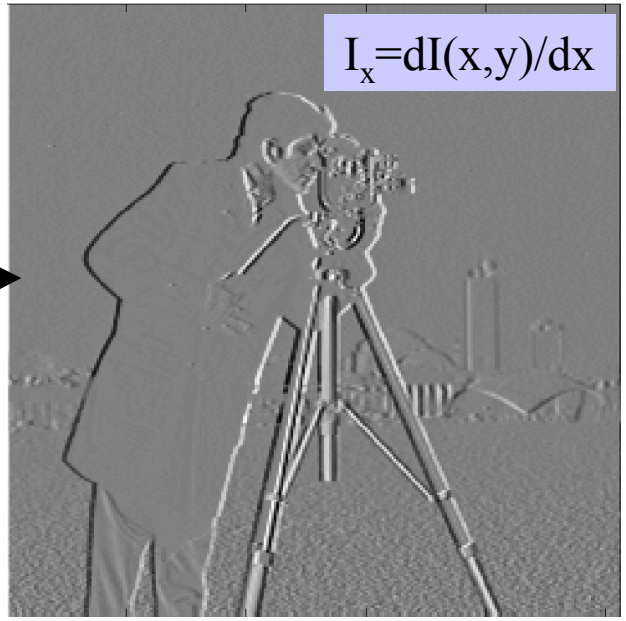
-1
0
1

Question for class:
Is this correct?

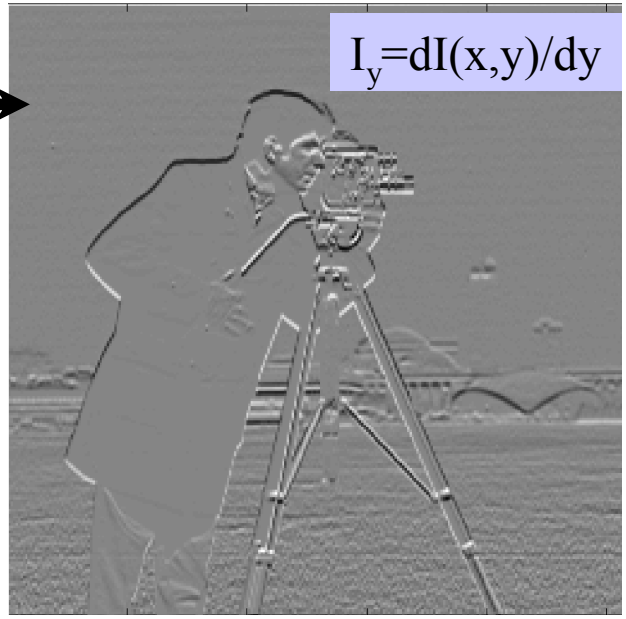
Example: Spatial Image Gradients

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

Partial derivative wrt x



Partial derivative wrt y



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

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