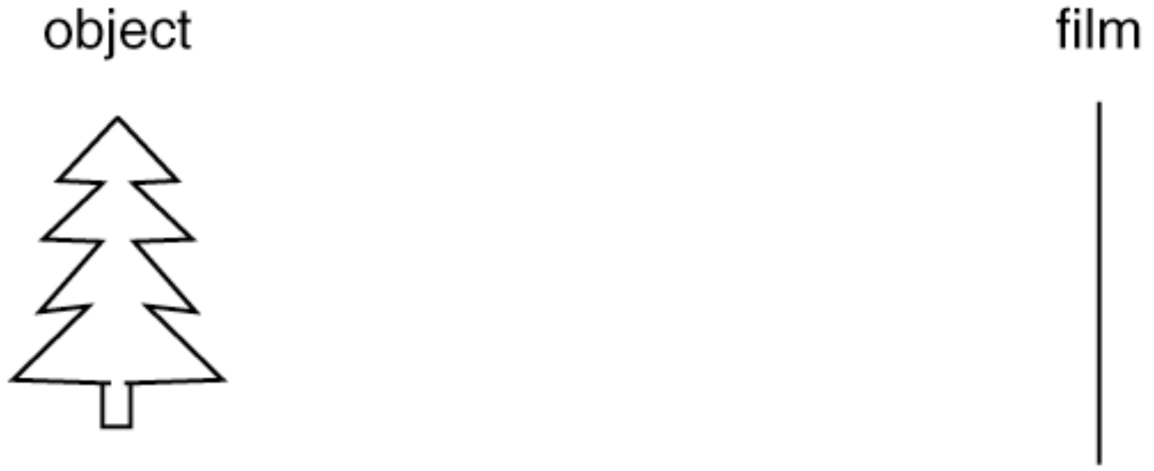
Image Formation

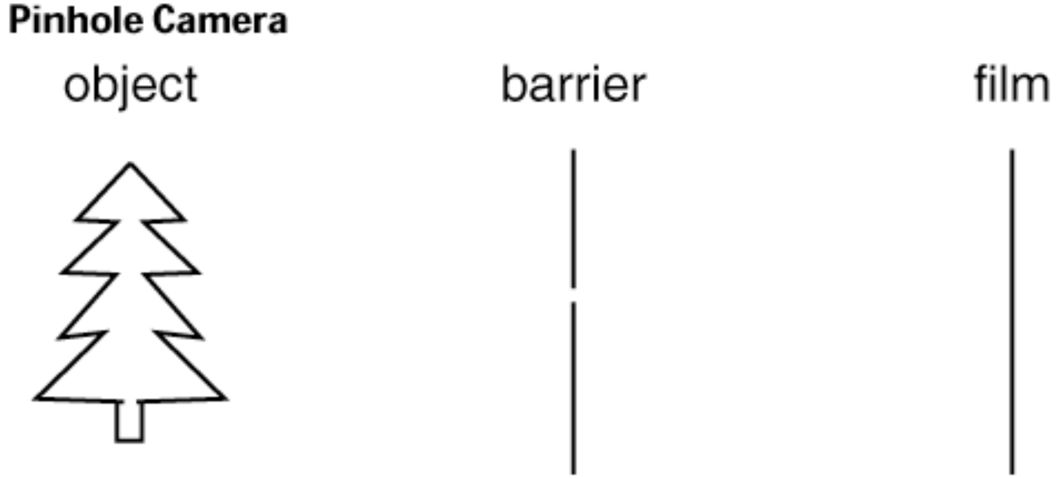
Forming an image

- First, we need some sort of sensor to receive and record light.
- Is this all we need?



- Do we get a useful image?

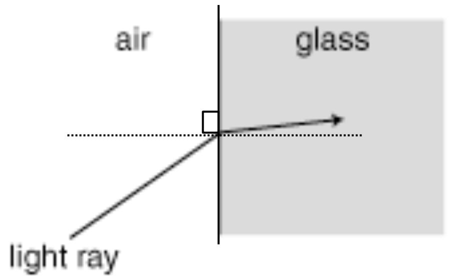
Restricting the Light

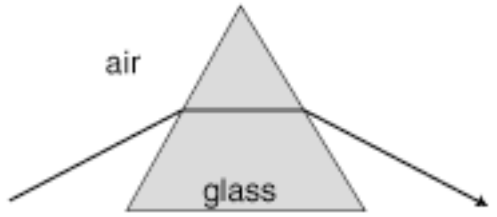


- Advantages:
  - easy to simulate
  - everything is in focus
- Disadvantages:
  - needs a bright scene (or long exposure)
  - everything is in focus

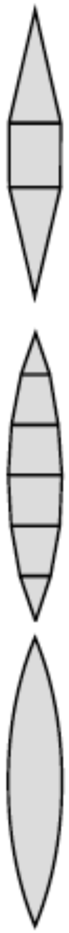
Collecting the light

- Instead of throwing away all but a single ray, let's collect a bunch of rays and concentrate them at a single point on the sensor.
- To do this, we need to be able to change the path of a light ray.
- Fortunately, we have **refraction**. Light passing from one medium into denser one will bend towards the **normal** of the interface.





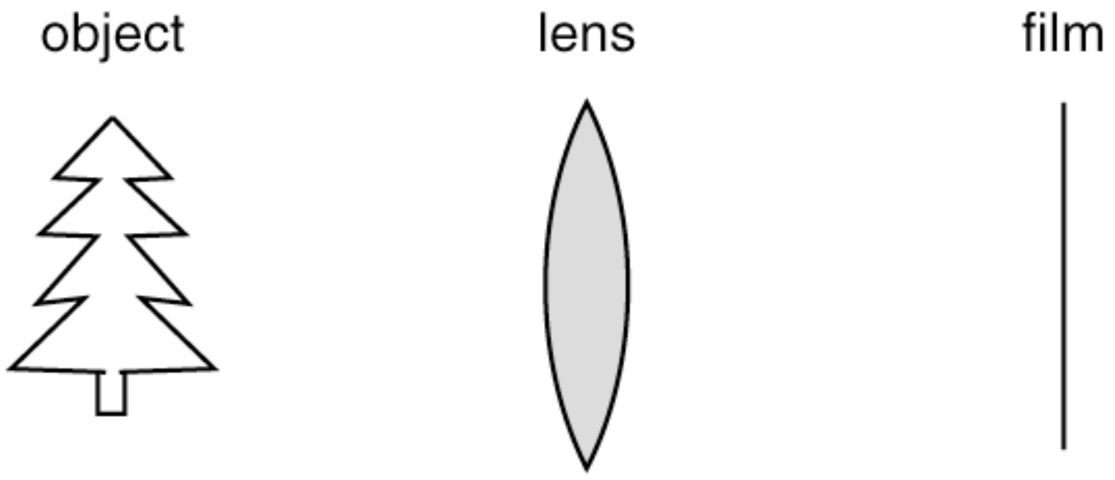
Stacking prisms



- We can use variously shaped prisms to take light rays of various angles and bend them to pass through a single point.
- As we use more and more prisms, the shape approaches a curve, and we get a **lens**.

Forming an image with a lens

- We can now replace the pinhole barrier with a lens, and we still get an image.



- Now there is a specific distance at which objects are “in focus”.
- By changing the shape of the lens, we change how it bends the light.

Optics

- **Focal point** - the point where parallel rays converge when passing through a lens.
- **Focal length** - the distance from the lens to the focal point.
- **Diopter** - the reciprocal of the focal length, measured in meters.

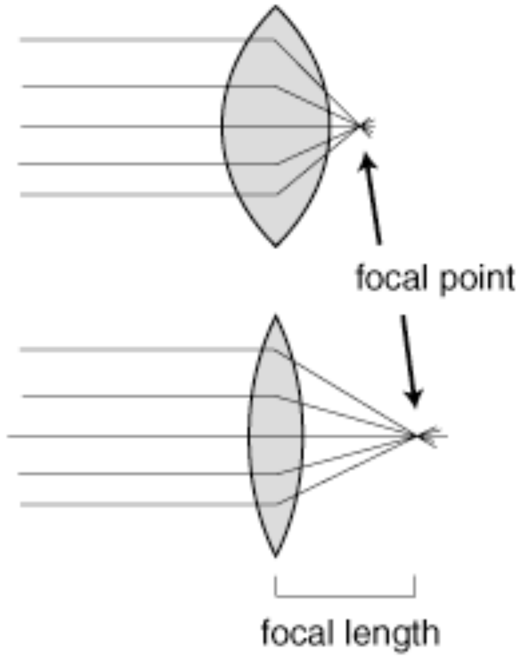


Image Processing

Reading

The following links are available on the course homepage

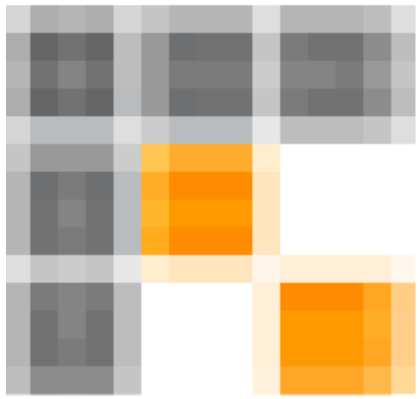
- http://www.dai.ed.ac.uk/HIPR2/noise.htm
- http://www.dai.ed.ac.uk/HIPR2/mean.htm
- http://www.cee.hw.ac.uk/hipr/html/median.html

Definitions

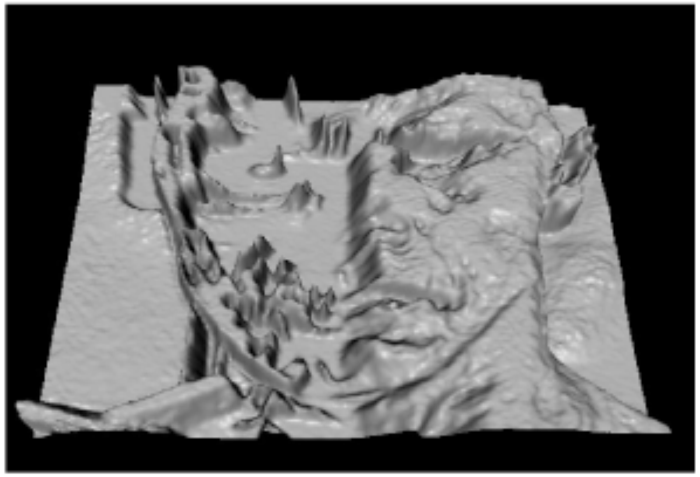
- Many graphics techniques operate only on images
- **Image processing**: operations that take images as input, produce images as output
- In its most general form, an **image** is a function f from R^2 to R
  - f (x, y) gives the intensity of a channel at position (x, y)
  - defined over a rectangle, with a finite range:
  - f : [a,b]x[c,d]   [0,1]
  - A color image is just three functions pasted together:
    - $\boldsymbol{f}$ (x, y) = (f_r(x, y), f_g(x, y), f_b(x, y))

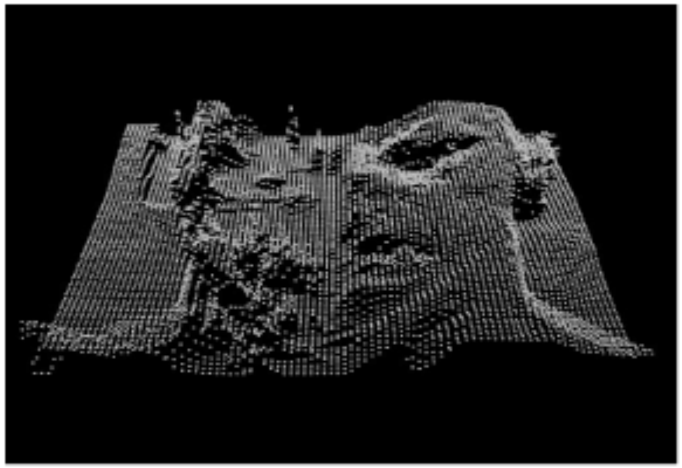
Images

- In computer graphics, we usually operate on **digital** (**discrete**) images
  - **Quantize** space into units (pixels)
  - Image is constant over each unit
  - A kind of step function
  - f : {0 … m -1}x{0 … n -1}   [0,1]
- An image processing operation typically defines an image f ′ in terms of an existing image f



Images as Functions





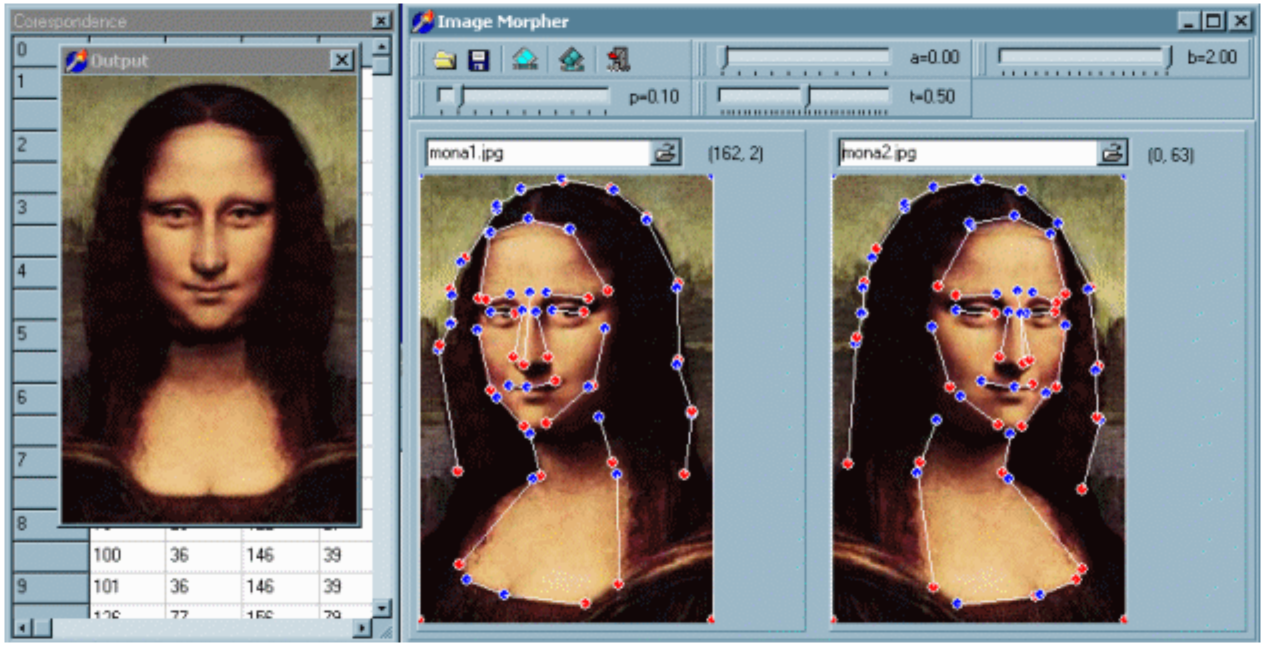
Pixel-to-pixel Operation

- The simplest operations are those that transform each pixel in isolation
- f ′ (x, y) = g (f (x,y))
- Example: threshold, RGB $\longrightarrow$ greyscale

$$\begin{array}{ccc} f\ (x,y) & & f`\ (x,y) \\ \begin{matrix} .4 & .3 & .2 & .1 \\ .3 & .8 & .9 & .2 \\ .2 & .5 & .7 & .3 \\ .1 & .0 & .2 & .4 \end{matrix} & \xrightarrow{\text{Threshold} >= 0.5} & \begin{matrix} .0 & .0 & .0 & .0 \\ .0 & .8 & .9 & .0 \\ .0 & .5 & .7 & .0 \\ .0 & .0 & .0 & .0 \end{matrix} \end{array}$$

Pixel Movement

- Some operations preserve intensities, but move around in the image
- $f'(x, y) = f\ (g(x,y), h(x,y))$
- Examples: many amusing warps of images



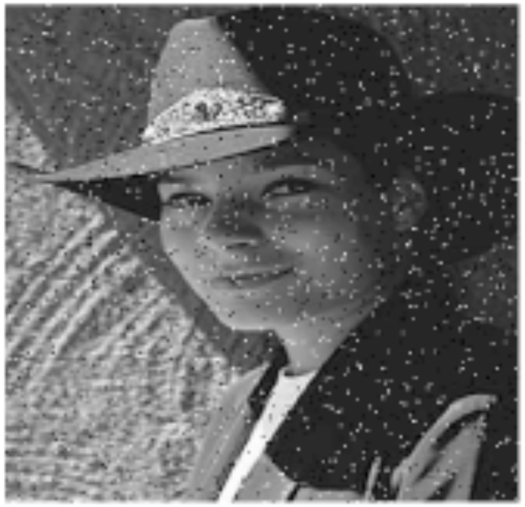
a=0.00 b=2.00
p=0.10 t=0.50
mona1.jpg (162, 2)
mona2.jpg (0, 63)

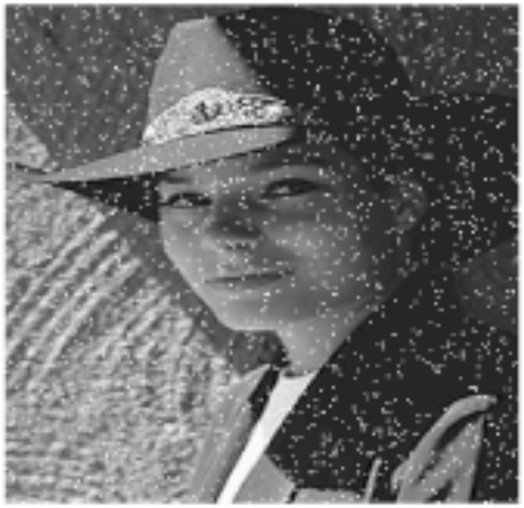
100	36	146	39
101	36	146	39

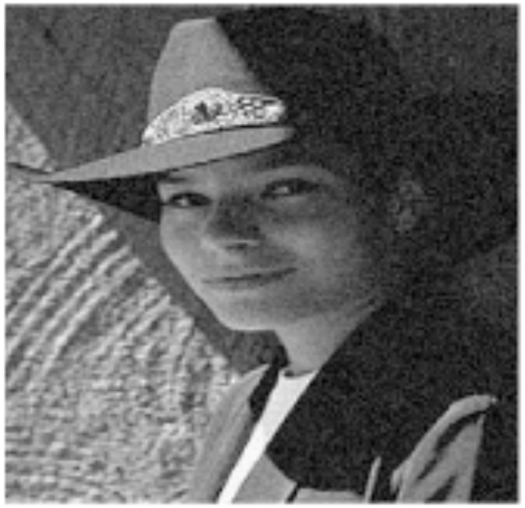
Noise

- Common types of noise

Original


Salt and pepper noise


Impulse noise


Gaussian noise

Noise

- Common types of noise:
  - **Salt and pepper noise**: random black and white pixels
  - **Impulse noise:** random white pixels
  - **Gaussian noise**: variations in intensity drawn from a Gaussian (normal) distribution

Noise Reduction

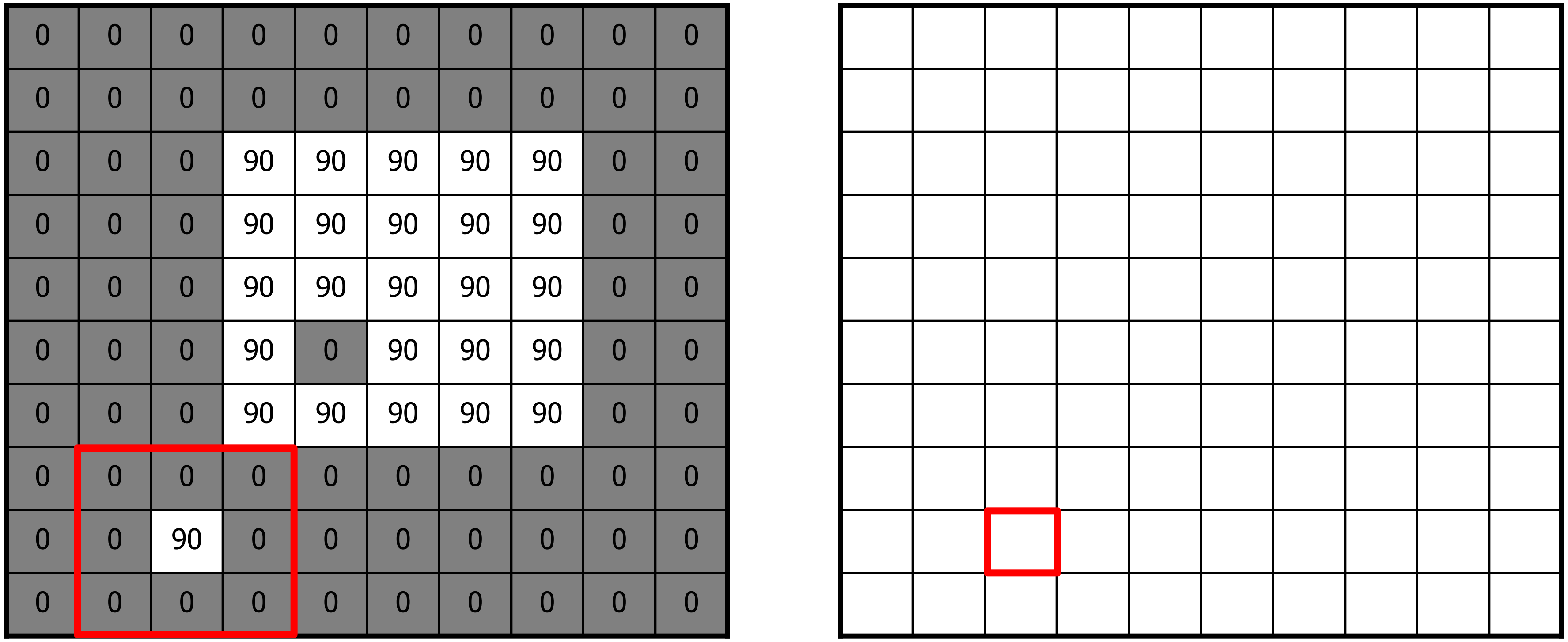
- Is there a way to “smooth” out the noise?

Reducing Noise

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

- Filtering
  - look at the neighborhood N around each pixel
  - replace each pixel with new value as a function of pixels in N
  - The behavio $g(i,j) = h(f,N)$ and f

Mean filtering



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x,y]$ $G[x,y]$

- Replace each pixel with an average of the pixels in the k×k box around it
  - 3×3 case: $G[x,y] = \frac{1}{9}\sum_{u=0}^{2}\sum_{v=0}^{2} F[x+u-1, y+v-1]$

Mean filtering

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

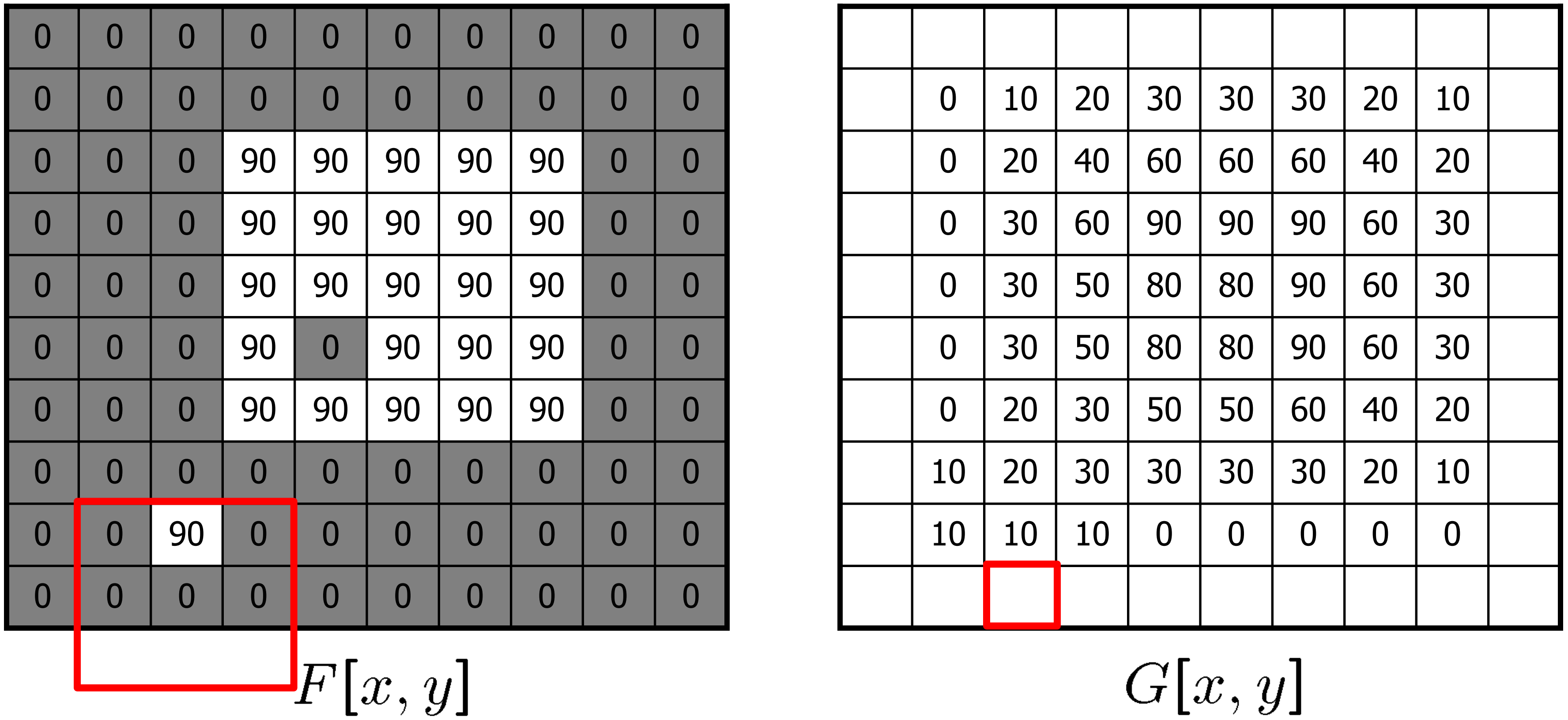
$F[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$G[x, y]$

- Replace each pixel with an average of the pixels in the k×k box around it
  - 3×3 case: $G[x, y] = \frac{1}{9} \sum_{u=0}^{2} \sum_{v=0}^{2} F[x + u - 1, y + v - 1]$

What about border pixels?



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

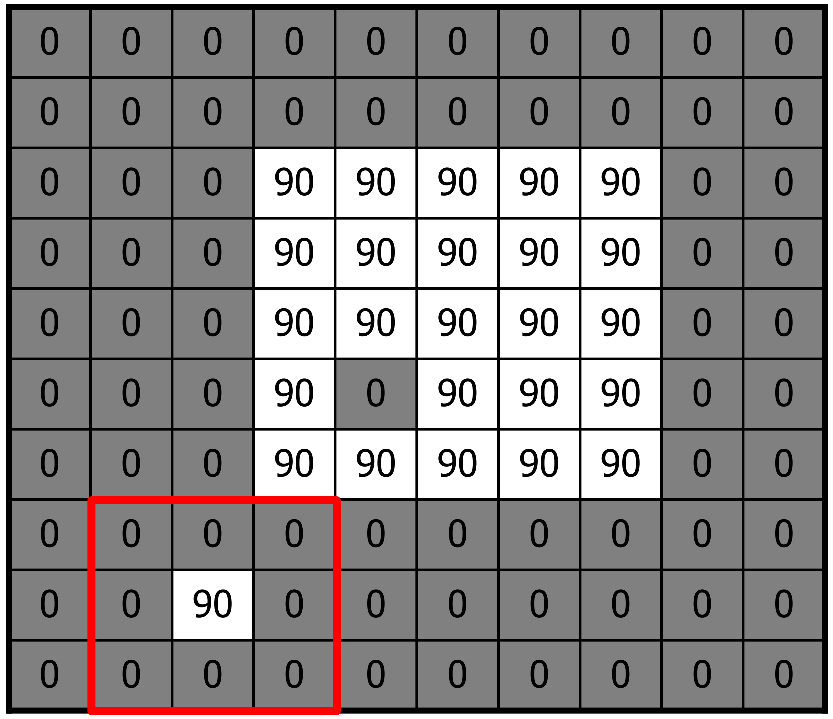
$G[x, y]$

- Some options
  - don't evaluate—image gets smaller each time a filter is applied
  - pad the image with more rows and columns on the top, bottom, left, and right
    - option 1: copy the border pixels: add [0 0 0] to F in above case
    - option 2: reflect the image about the border: add [0 90 0] to F in above case

Effect of filter size

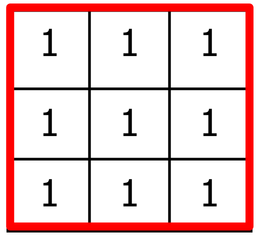
- What happens if we
- use a larger mean filter?
- 5×5? 7×7?

Weighted average filtering



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x,y]$



1	1	1
1	1	1
1	1	1

$H[u,v]$

- Replace each pixel with a weighted average of the pixels in the k×k box
  - 3×3 case: $G[x,y] = \sum_{u=0}^{2} \sum_{v=0}^{2} H[u,v] * F[x+u-1, y+v-1]$

Gaussian filter

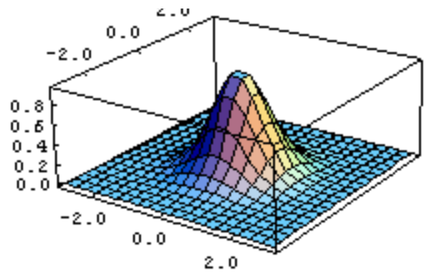
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$F[x,y]$$

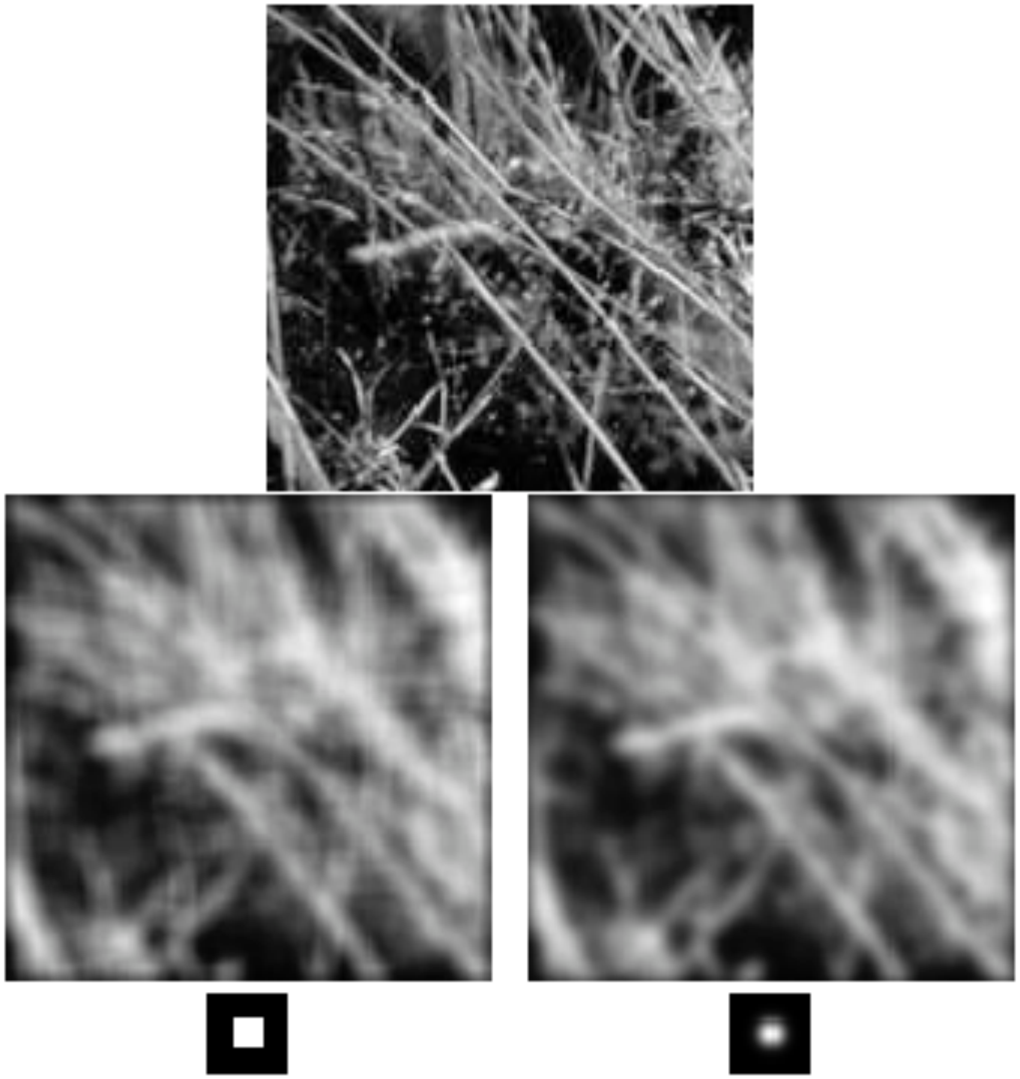
$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

$$H[u,v]$$

- This filter H is a good approximation to $h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$
- Properties of Gaussian
  - more weight to the center
  - good model of blurring in optical systems
  - σ corresponds to width of the Gaussian



Comparison of mean vs. gaussian filter



Convolution

- Convolution is a fancy way to combine two functions.
  - Think of *f* as an image and *g* as a "smear" operator
  - *g* determines a new intensity at each point (pixel) in terms of intensities at the neighborhood of that point (pixel)

$$\begin{aligned} h(x,y) &= f(x,y) * g(x,y) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x-x',y-y')dx'dy' \end{aligned}$$

Discrete Convolution

- For digital images, integration becomes summation. We can express convolution as a two-dimensional sum:

$$\begin{aligned} h[i,j] &= f[i,j] * g[i,j] \\ &= \sum_k \sum_l f[k,l]g[i-k,j-l] \end{aligned}$$

Convolution Representation

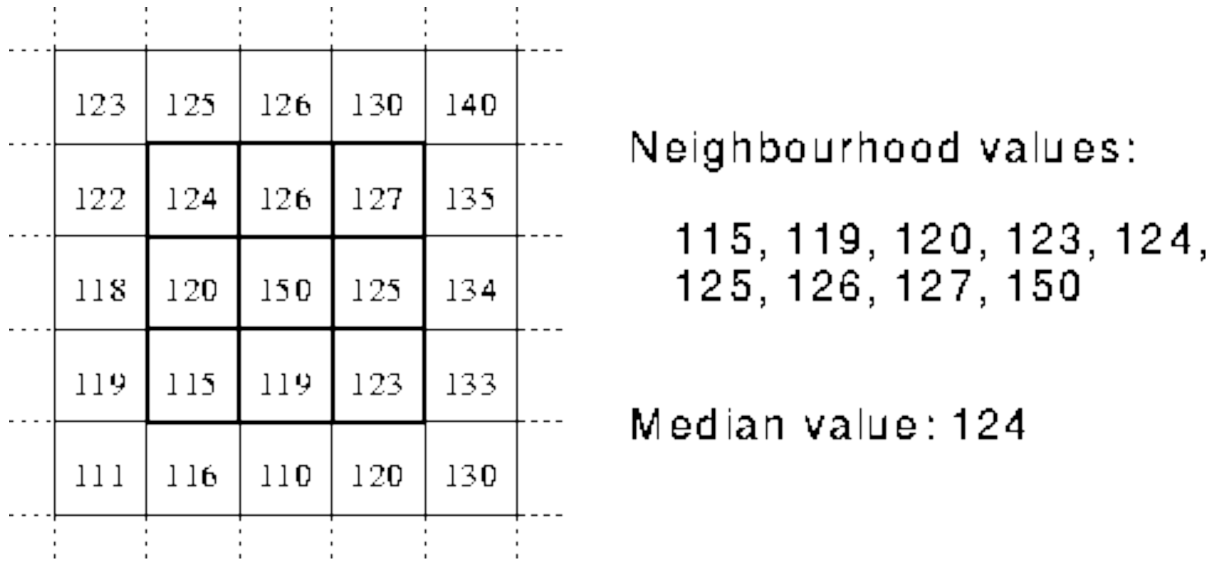
- Since *f* and *g* are defined over finite regions, we can write them out in two-dimensional arrays:

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

X .2	X 0	X .2
X 0	X .2	X 0
X .2	X 0	X .2

Median Filter

- A **Median Filter** operates over a kxk region by selecting the median intensity in the region.
  - What advantage does a median filter have over a mean filter? (answer available at “extra” - where you download the notes!)
  - Is a median filter a kind of convolution?



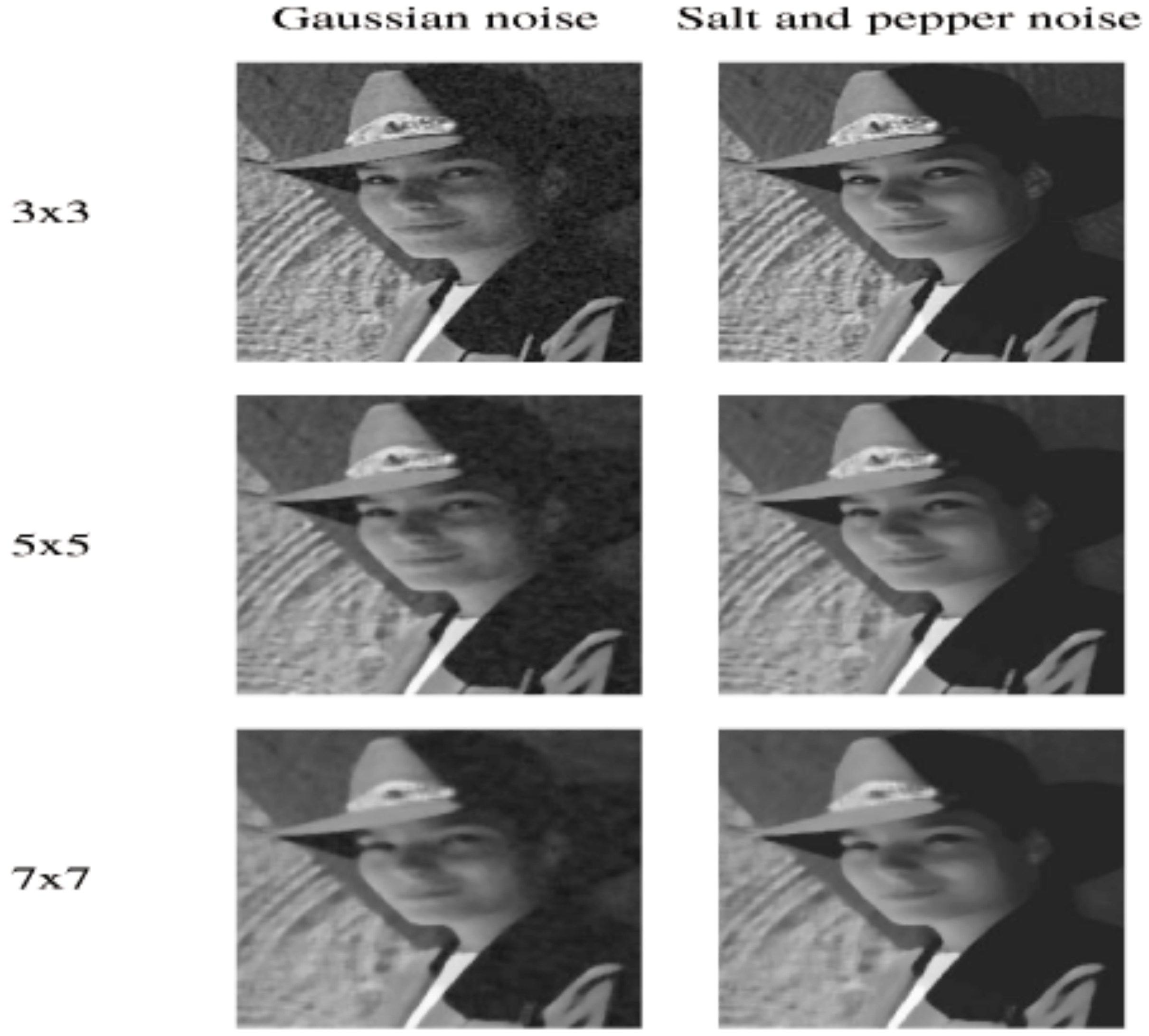
123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

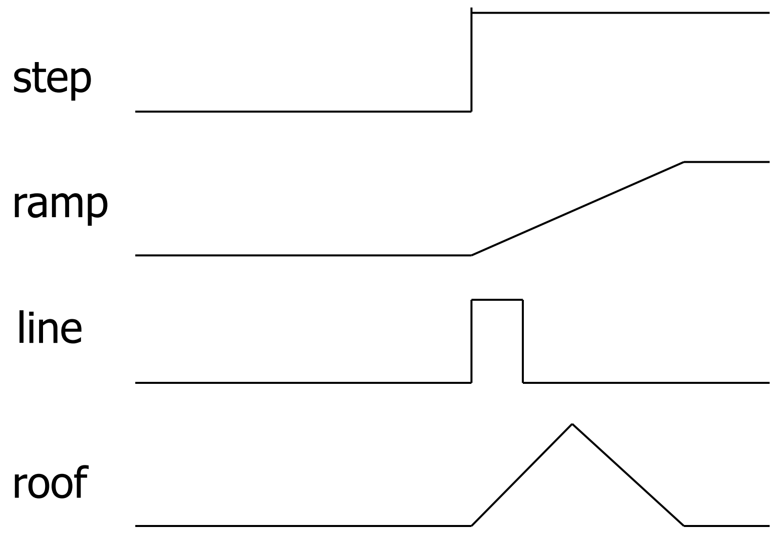
Median value: 124

Using Median Filters

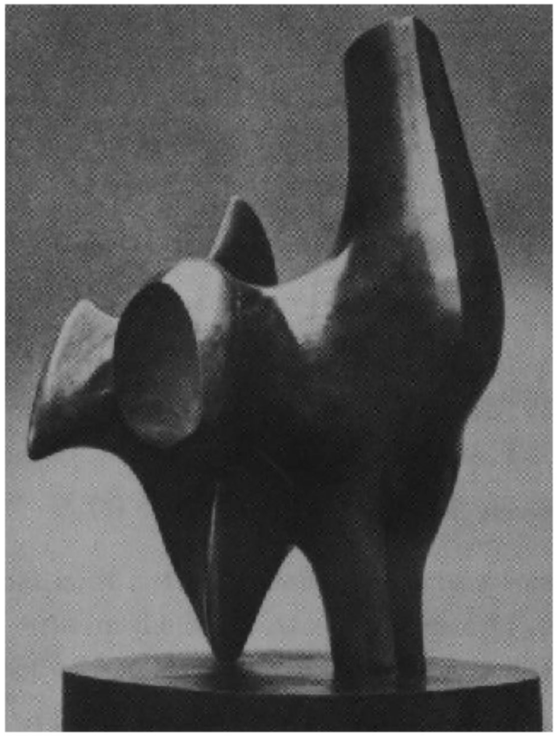


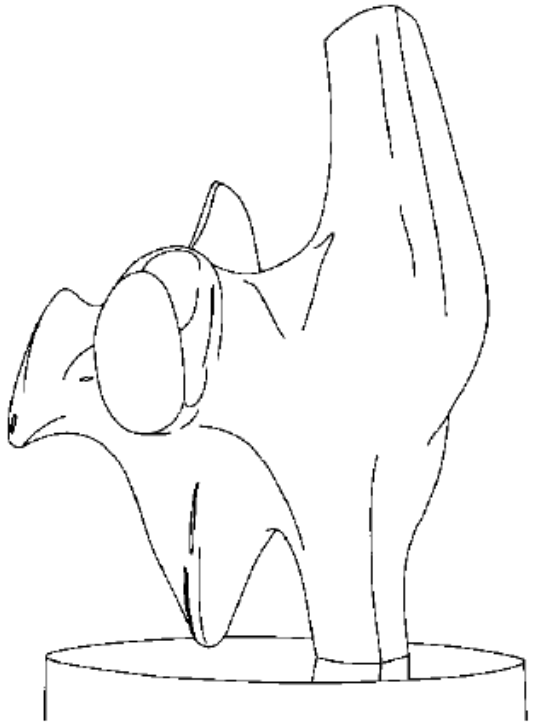
Edge Detection

- One of the most important uses of image processing is **edge detection**
  - Really easy for humans
  - Really difficult for computers
  - Fundamental in computer vision
  - Important in many graphics applications
- What defines an edge?



Edge detection





- How can you tell that a pixel is on an edge?

Edge Detection

- Edge detection algorithms typically proceed in four steps:
  - Filtering: cut down on noise
  - Enhancement: amplify the difference between edges and non-edges
  - Thresholding
  - Localization (optional): estimate geometry of edges beyond pixels

Gradient

- The **gradient** is the 2D equivalent of the derivative:

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

- Properties of the gradient
  - It's a vector
  - Points in the direction of maximum increase of *f*

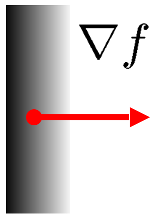
    (direction of the steepest descent)
  - Magnitude is rate of increase
- How can we approximate the gradient in a discrete image?

Image gradient

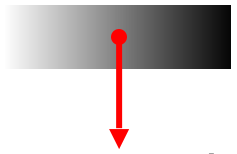
- The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

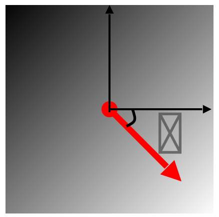
- The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Edge detection operator

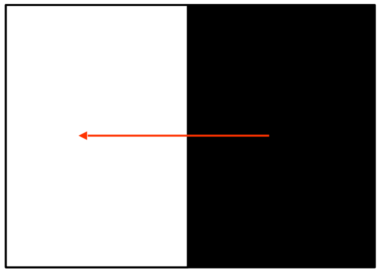
- A popular gradient magnitude computation is the **Sobel operator**:

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- We can then compute the magnitude of the vector $(G_x, G_y)^T$

Sobel Operator: Example



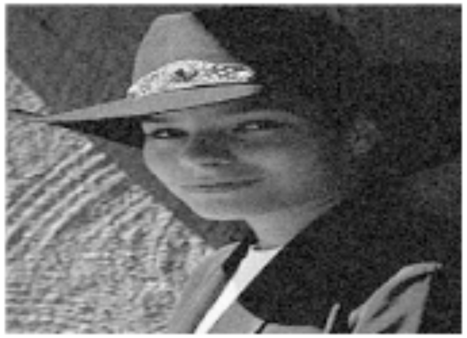
```
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
```

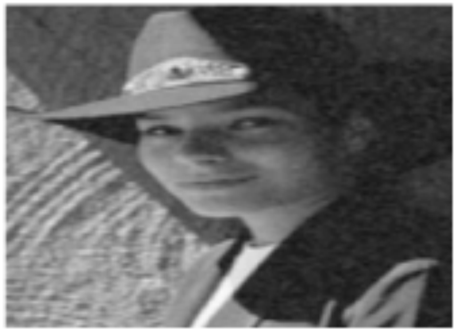
$$G_x = \begin{matrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{matrix} * \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} = -4$$

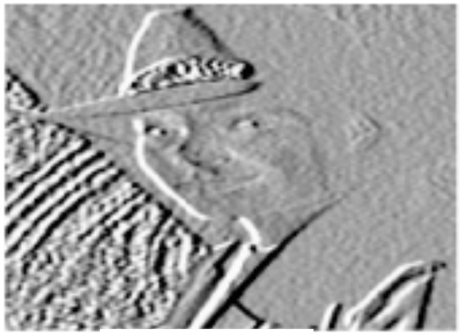
$$G_y = \begin{matrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{matrix} * \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} = 0$$

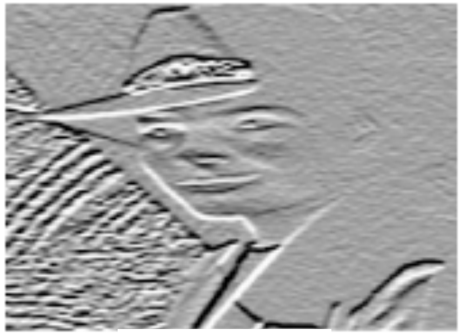
$$(G_x, G_y)^T = (-4, 0)^T$$

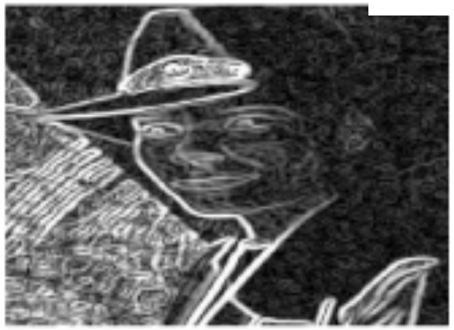
Using Sobel Operators

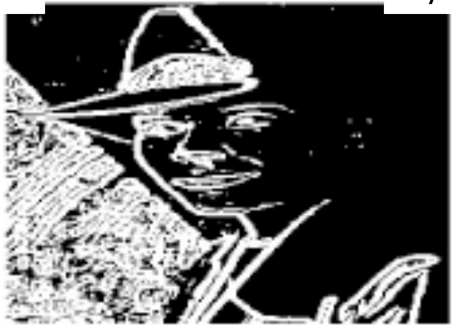

Original

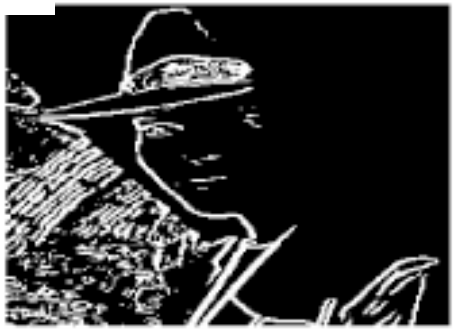

Smoothed


$G_x + 128$


$G_y + 128$

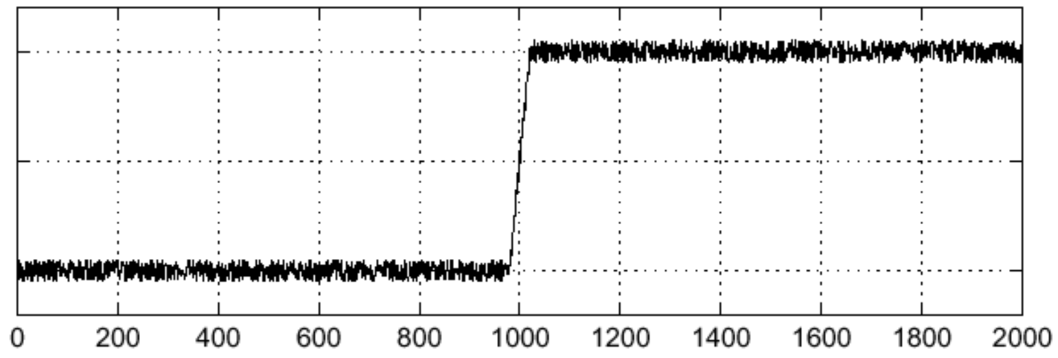

Magnitude

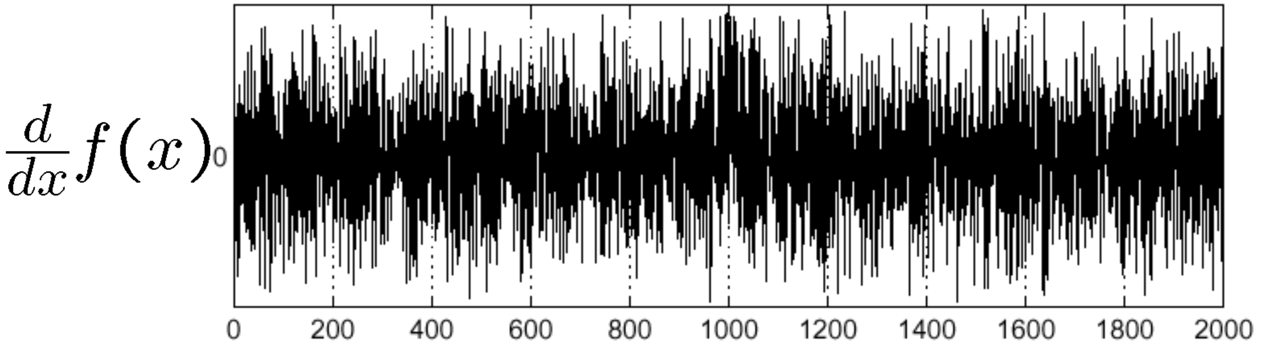

Threshold = 64


Threshold = 128

Effects of noise

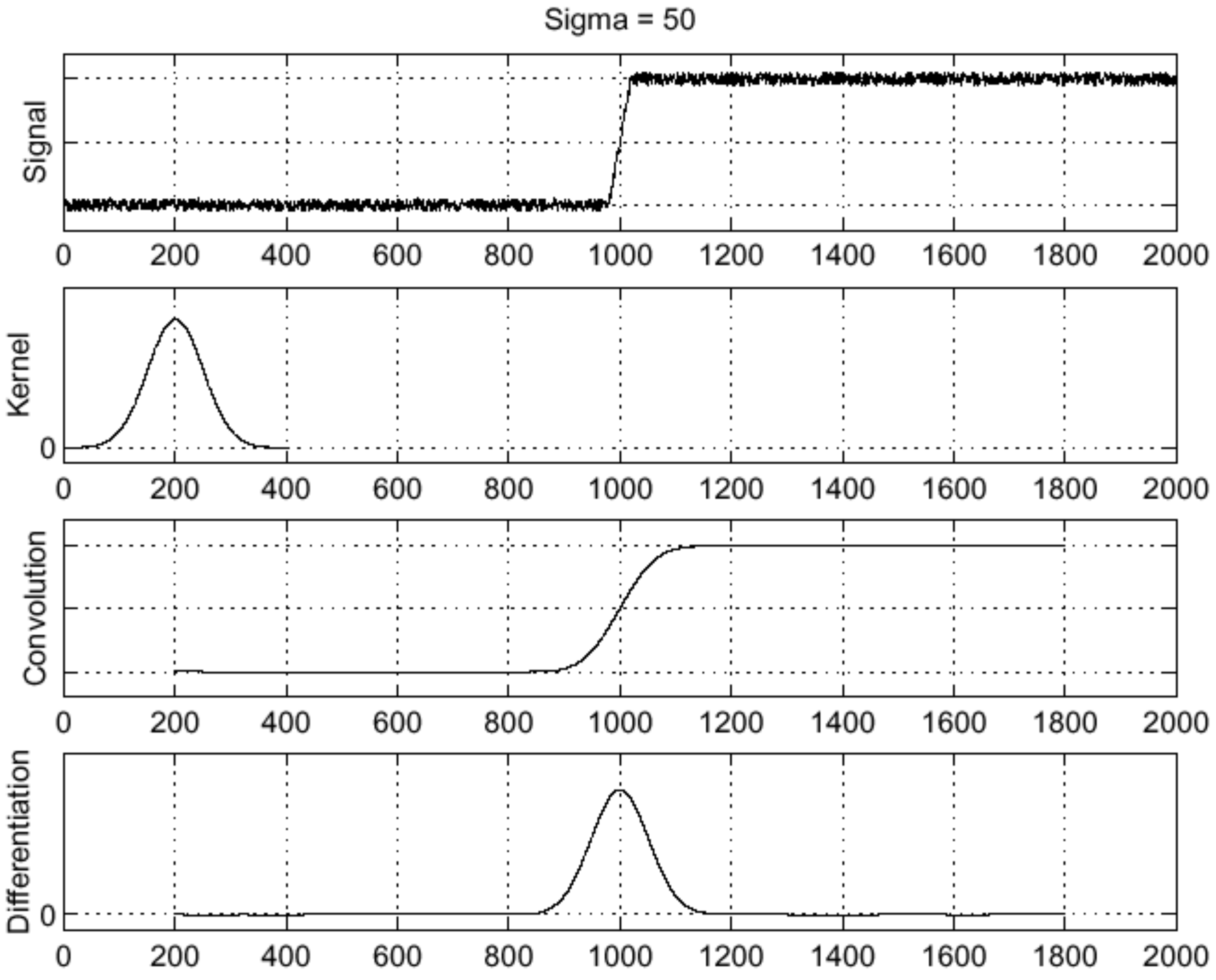
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

Solution: smooth first



$$\frac{\partial}{\partial x}(h \star f)$$

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Summary

What you should take from this lecture:

- Formal definitions of image and image processing
- Kinds of image processing: pixel-to-pixel, movement, convolution, others
- Types of noise and strategies for noise reduction
- Definition of convolution and how discrete convolution works
- The effects of mean, median and Gaussian filtering
- How edge detection is done
- Gradients and discrete approximations