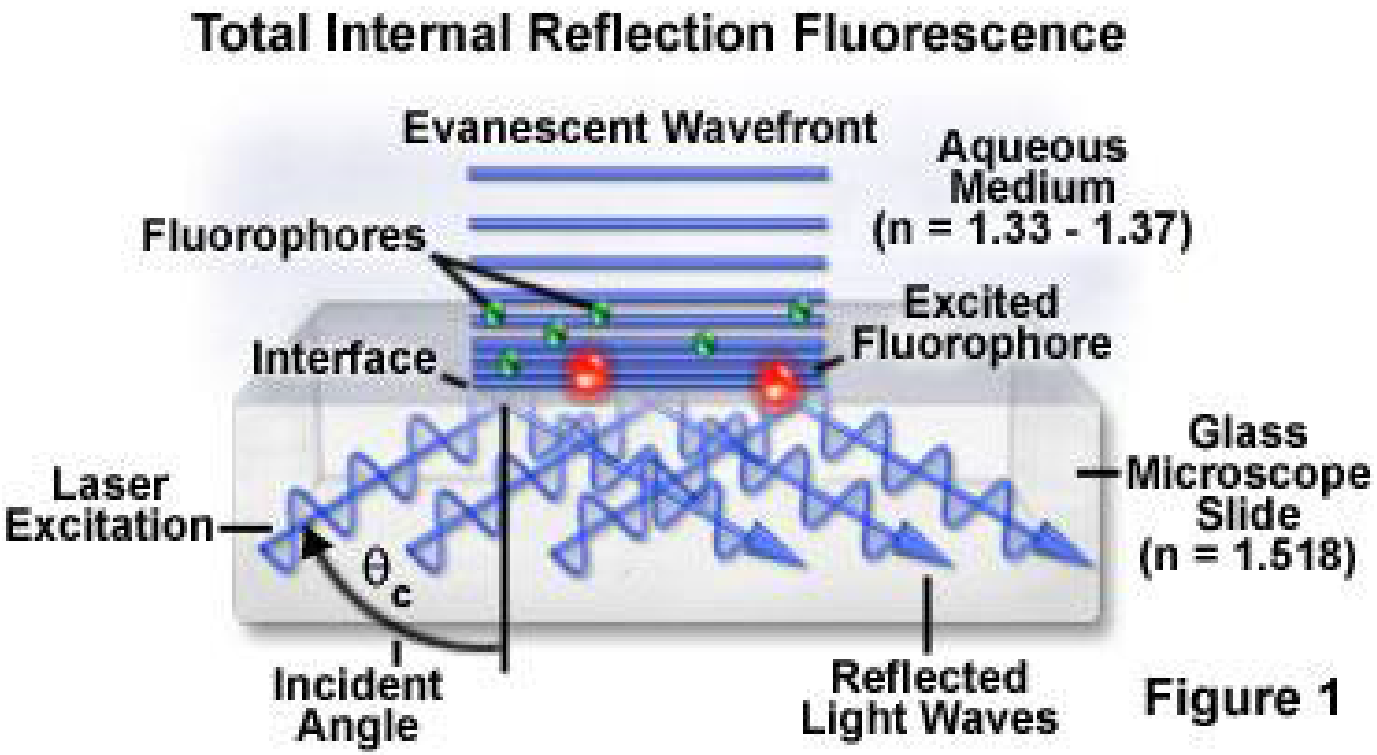
TIRF, geometric operators

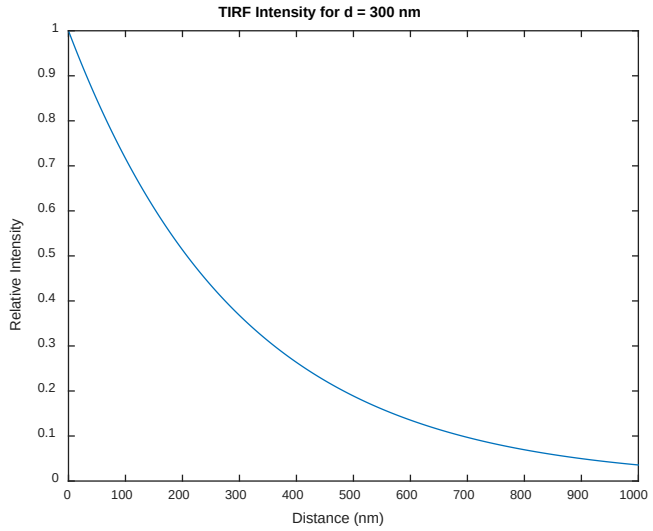
- Last class
  - FRET
  - TIRF
- This class
  - Finish up of TIRF
  - Geometric image processing

TIRF – light confinement



$$d_p = \lambda_0 \Big/ \left(2\pi \cdot n_1 \sqrt{\sin^2 \theta_1 - (n_2 / n_1)^2} \right)$$

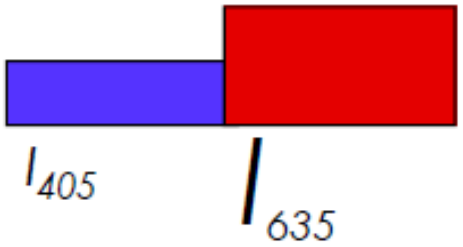
$$I(z) = I_0 e^{-z/d}$$

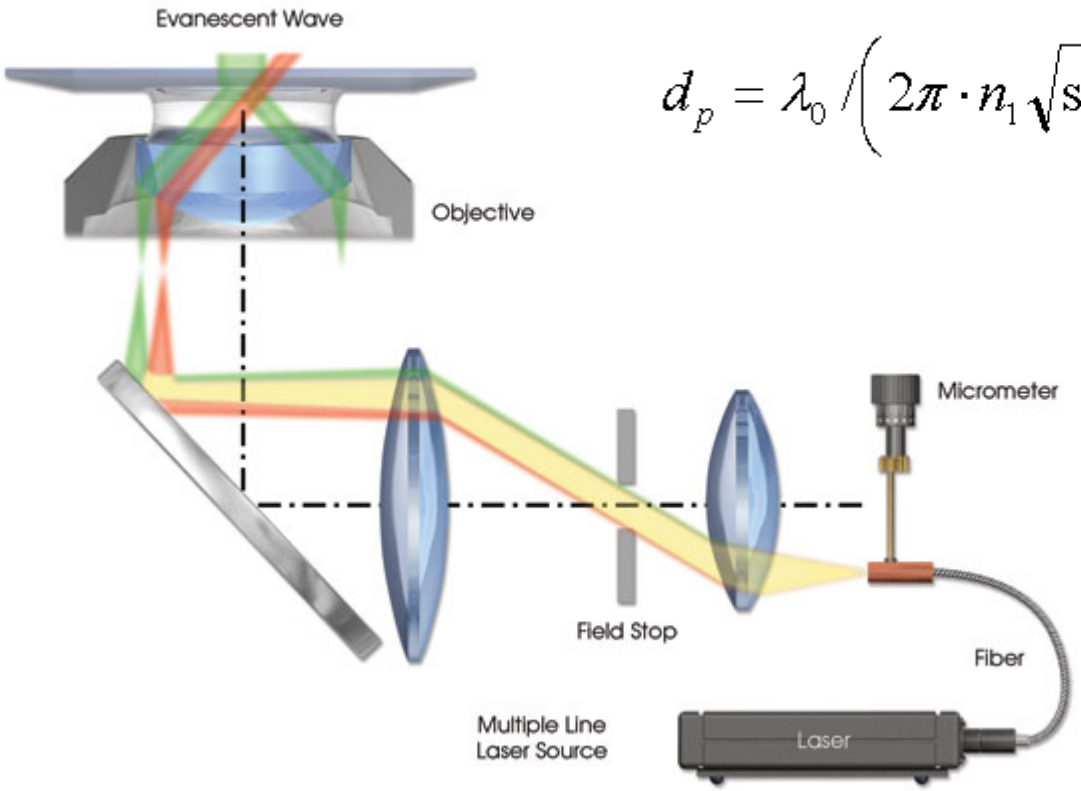


TIRF practicalities

- To change the output angle from the objective, we need to adjust the position at the focus point.

Different laser beam wavelenghts produce different penetration depths and different volume detected





$$d_p = \lambda_0 / \left(2\pi \cdot n_1 \sqrt{\sin^2 \theta_1 - (n_2 / n_1)^2} \right)$$

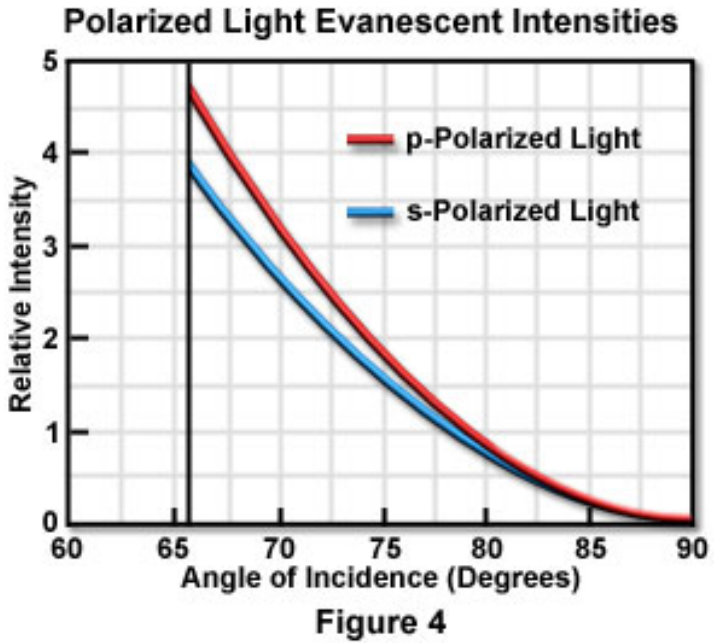
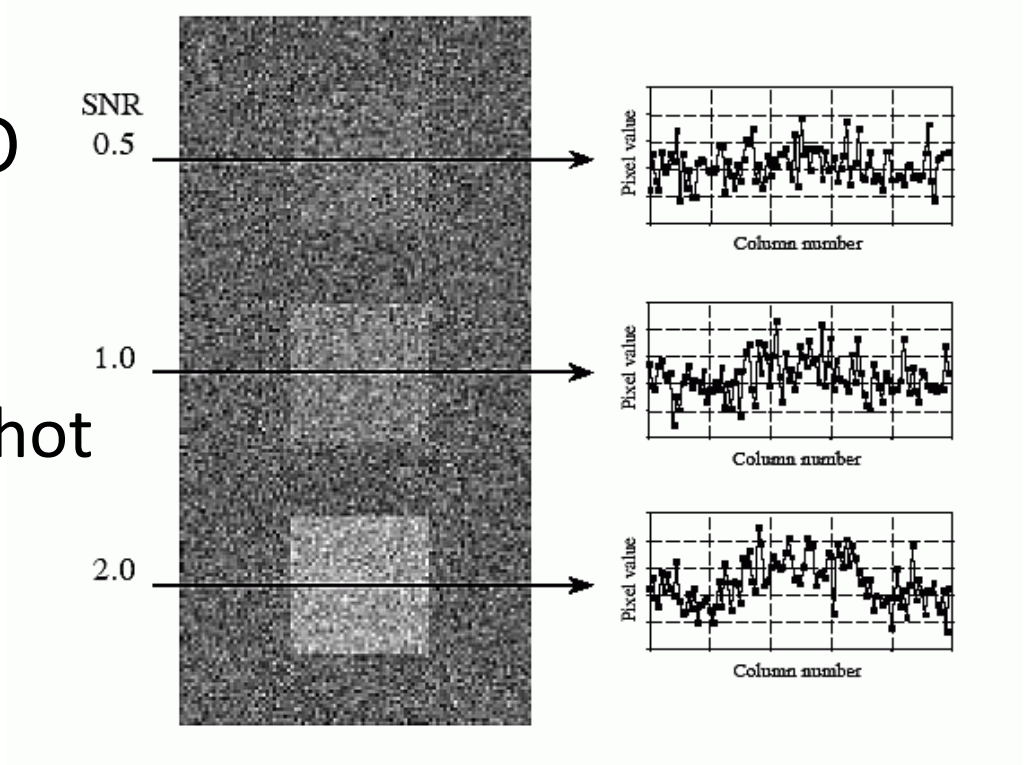


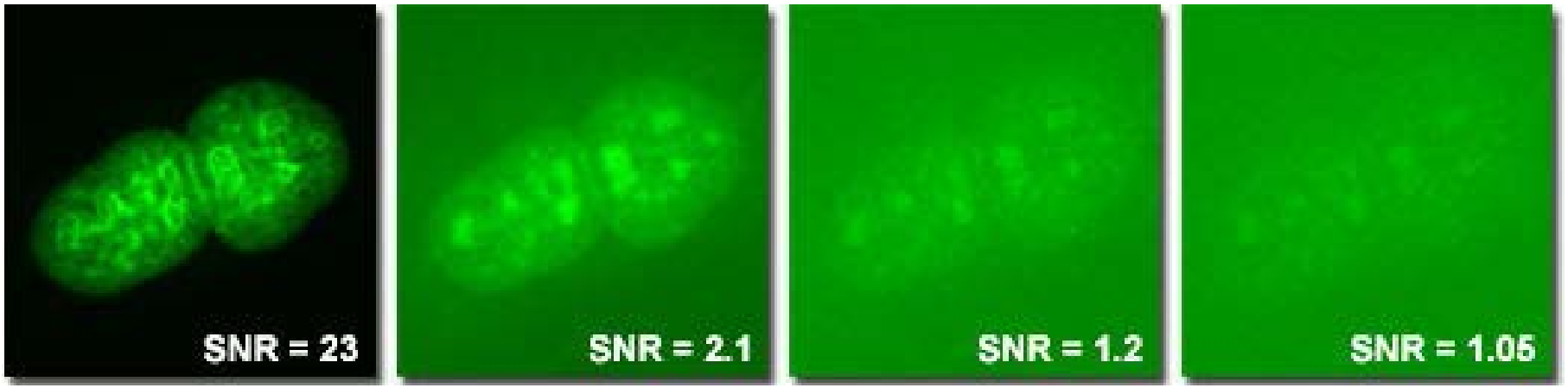
Figure 4

Effect of chromatic aberrations

Signal to noise ratio

- SNR = $\overline{\text{Signal}} - \overline{\text{background}} / \sigma_{bg}$
- Noise: background, dark counts, shot noise, scatter

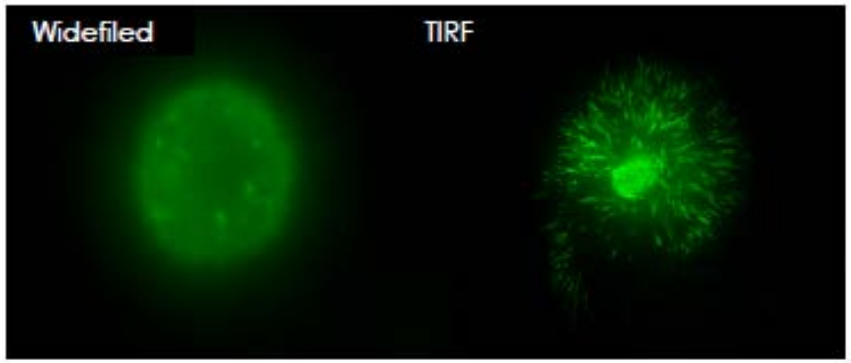




TIRF increases signal to noise by both increasing signal, and decreasing background

TIRF applications

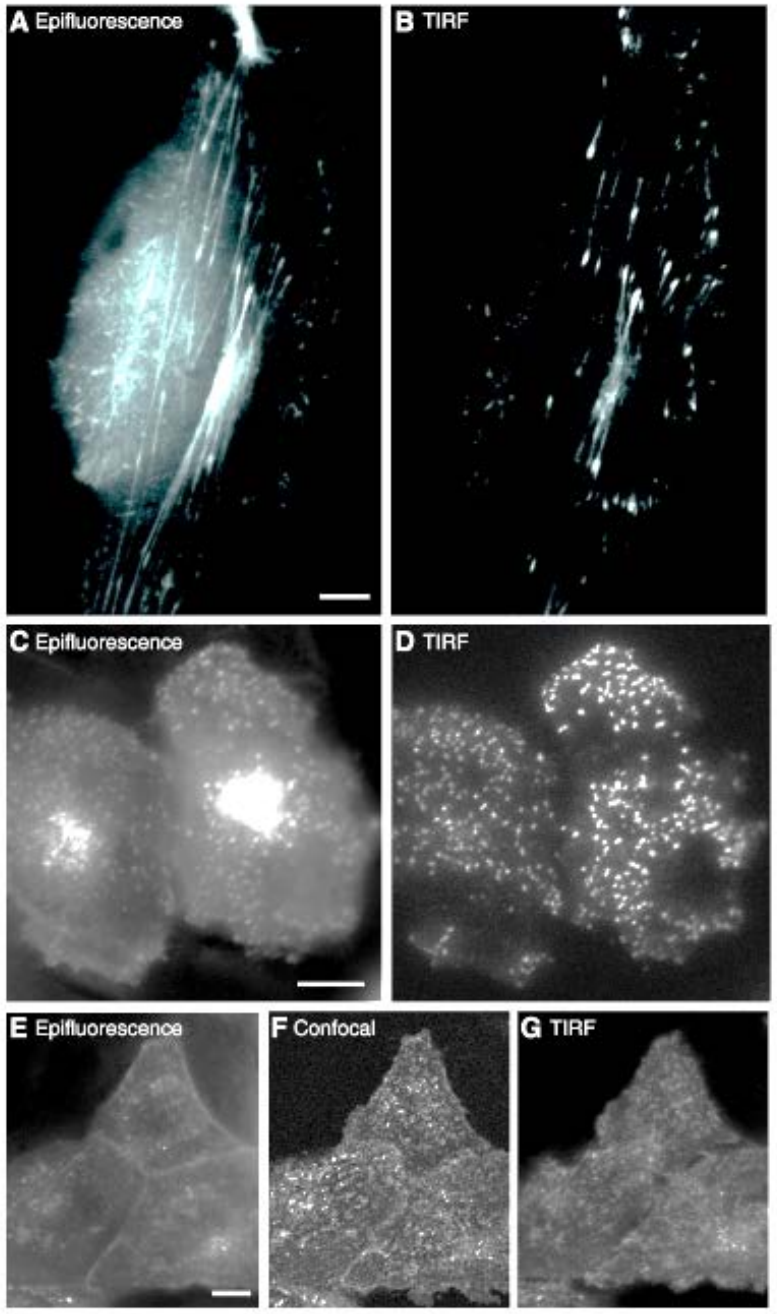
Higher Z resolution
Higher Signal to noise ratio
High contrast
Less bleaching/toxicity overall in the cell



Adhesion/cytoskeleton elements

Endocytosis

Diffusion



TIRF applications

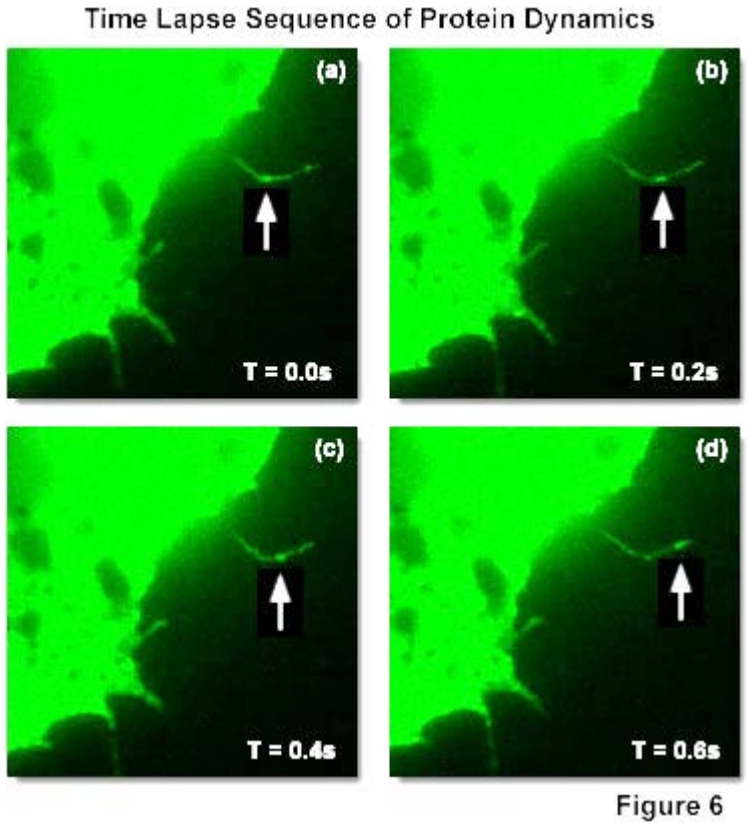


Figure 6

Protein movements in small cellular structures

Applications

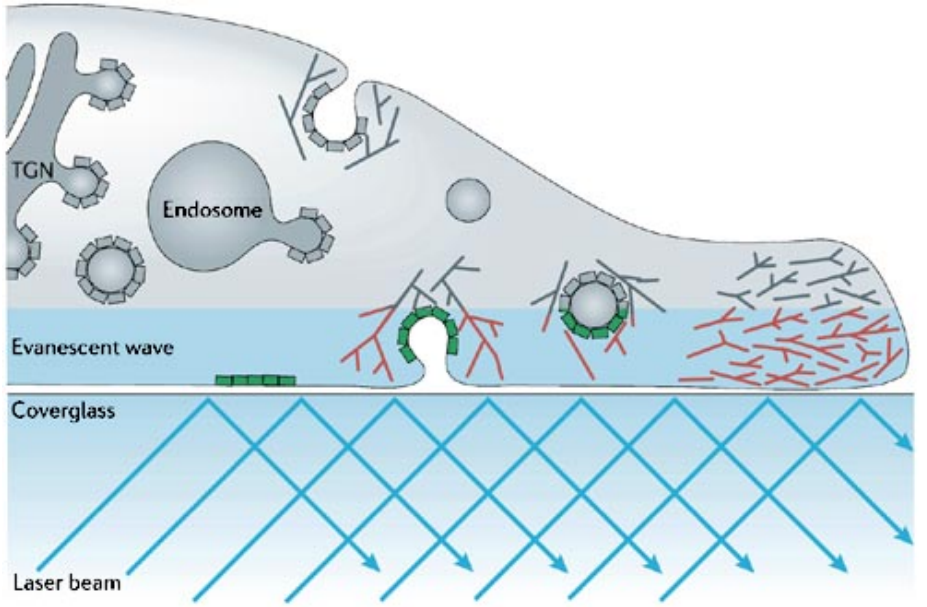
Membrane research

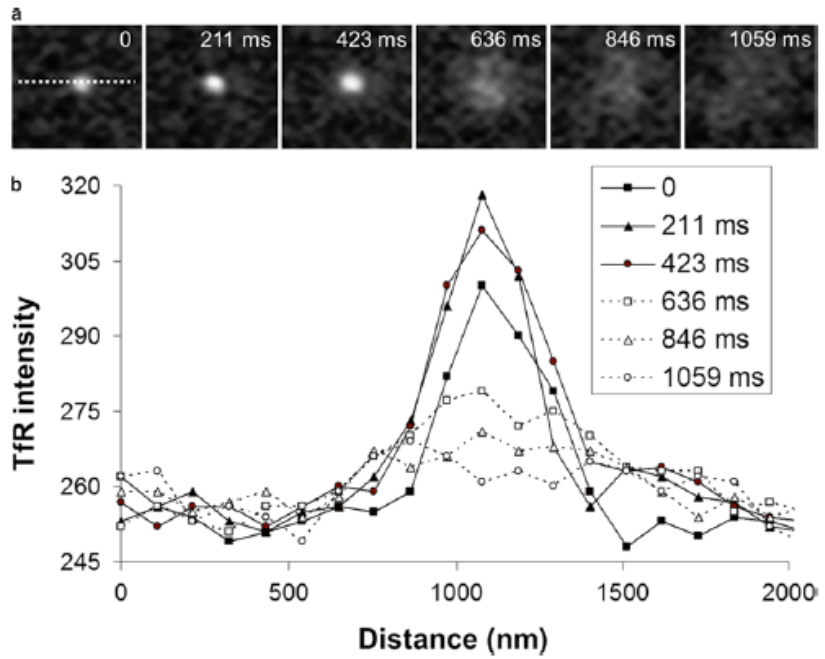
- Diffusion of molecules (e.g. Sytaxin)
- Kinetic of transporters
- Membrane fusion
- Cell/Cell interaction
- Cytoskeletaldynamics

Vesicle transport

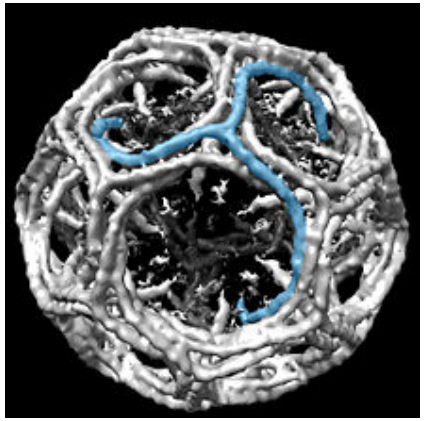
- Understanding of transport processes
- Localization of molecules
- Endocytosis and Exocytosis

Single molecule detection

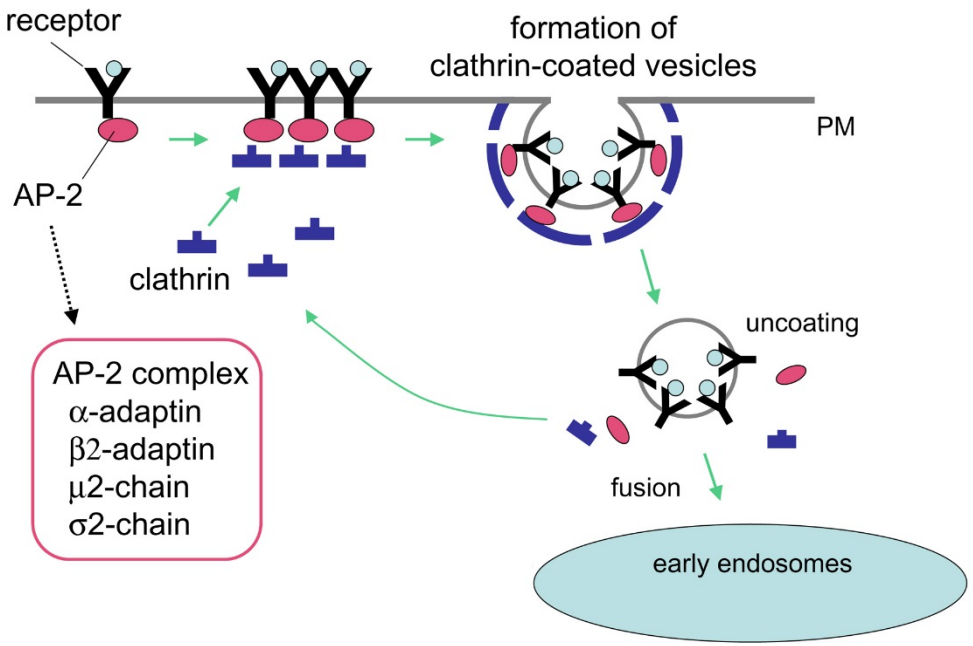


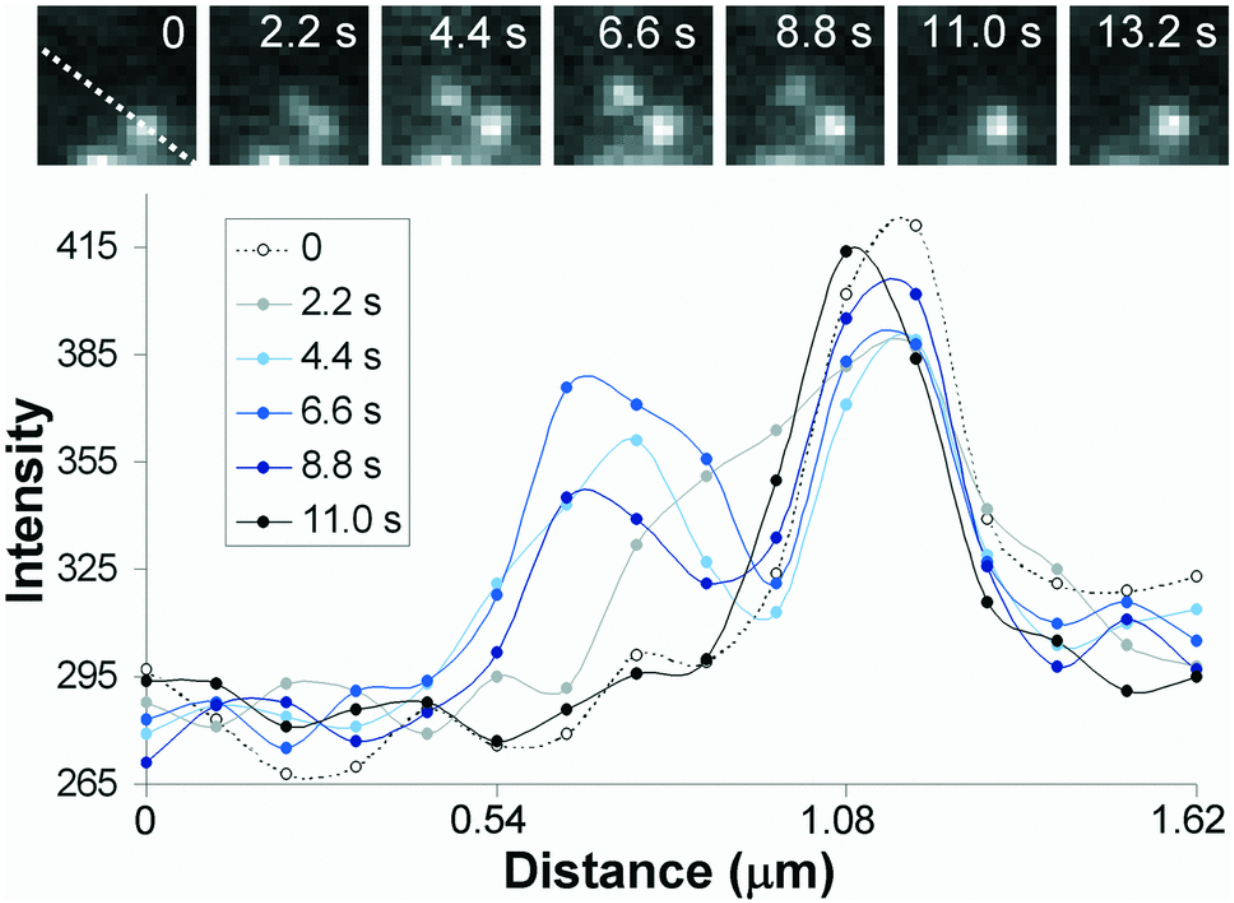


Clathrin mediated endocytosis



Clathrin-dependent endocytosis





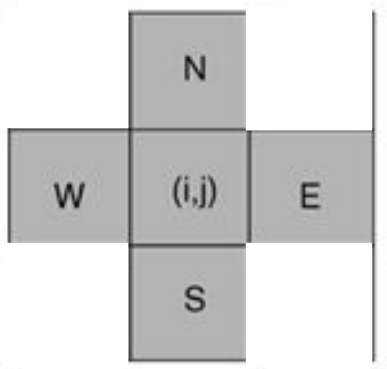
Geometric operators

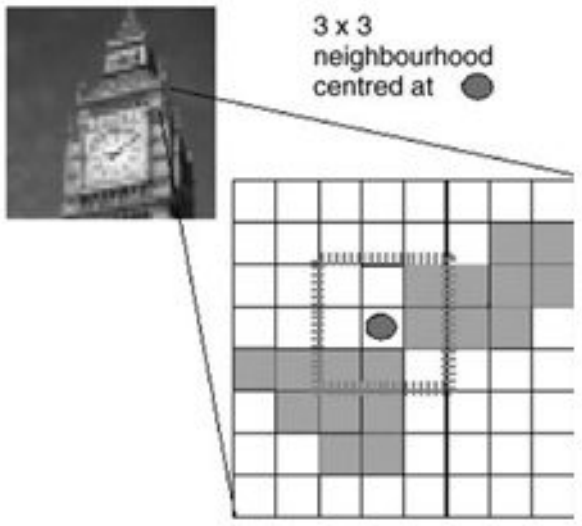
Similar to morphological operators

- Start with a filter mask (matrix)
- Apply filter to each pixel
- Reconstruct new image

Pixel neighborhood is area of filter

- Neighborhoods will are defined by surrounding pixels
- 4 Connectivity
- 8 Connectivity
- Neighborhoods are often 3 x 3, but can be arbitrary in size and shape





Each pixel in the neighborhood has a weight – defines type of operation

- 3x3 mean filter

$$F_{mean} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$= \begin{bmatrix}.11 & .11 & .11\\ .11 & .11 & .11\\ .11 & .11 & .11\end{bmatrix}$$

$$I_x = \sum_{i=1:9} I_i \ * \ F_{mean,i}$$

Original Image | 3x3 mean filter | 7x7 mean filter

As you might expect, it has the properties of smoothing out the image

Filters can be arbitrary

- If you want to maintain the same overall intensity, the total should sum to 1
- Also possible to have non-linear filters
- Can also have hard thresholds, perform some action if it is bright/dark enough

$$F_{mean} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$F_{arb} = \begin{bmatrix} 2 & 4 & -1 \\ 3 & 2 & 1 \\ 4 & 3 & -1 \end{bmatrix}$$

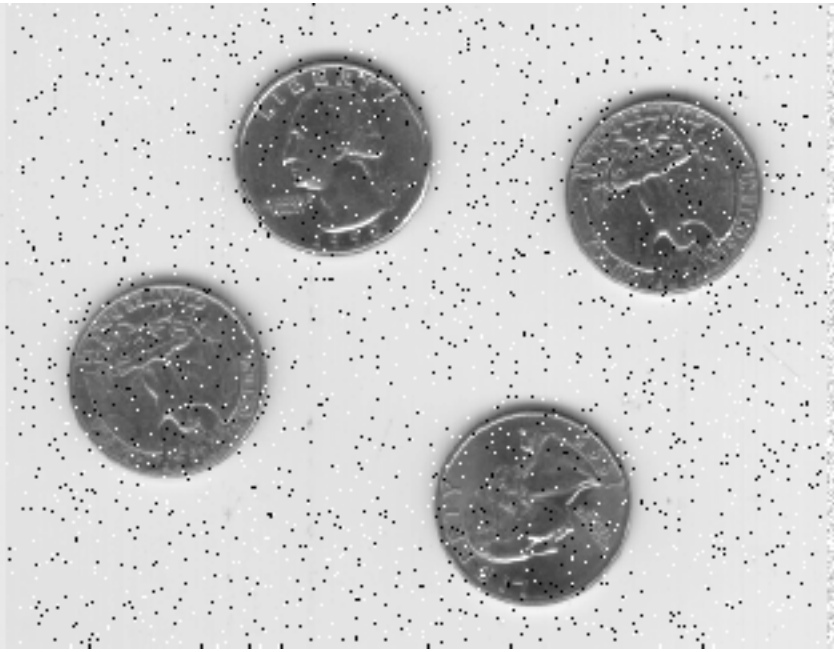
$$F_{arb} = \begin{bmatrix} I^2 & 4 & -1 \\ 3 & \sqrt{I} & 1 \\ 4 & 3 & -1 \end{bmatrix}$$

Filters are often used for removing noise

- Start with an image and introduce noise

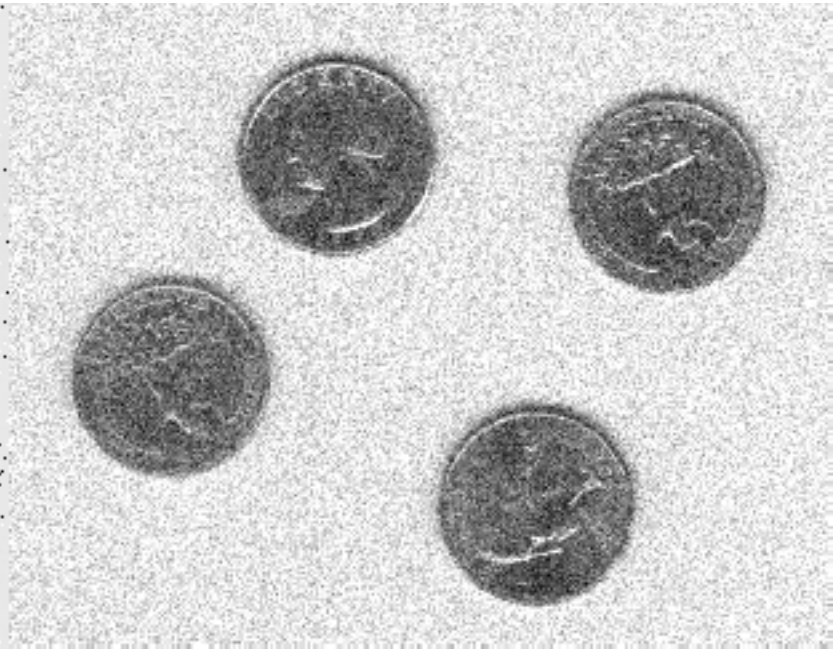
Original Image

Salt and Pepper noise



Salt and pepper noise adds highs and lows randomly in image

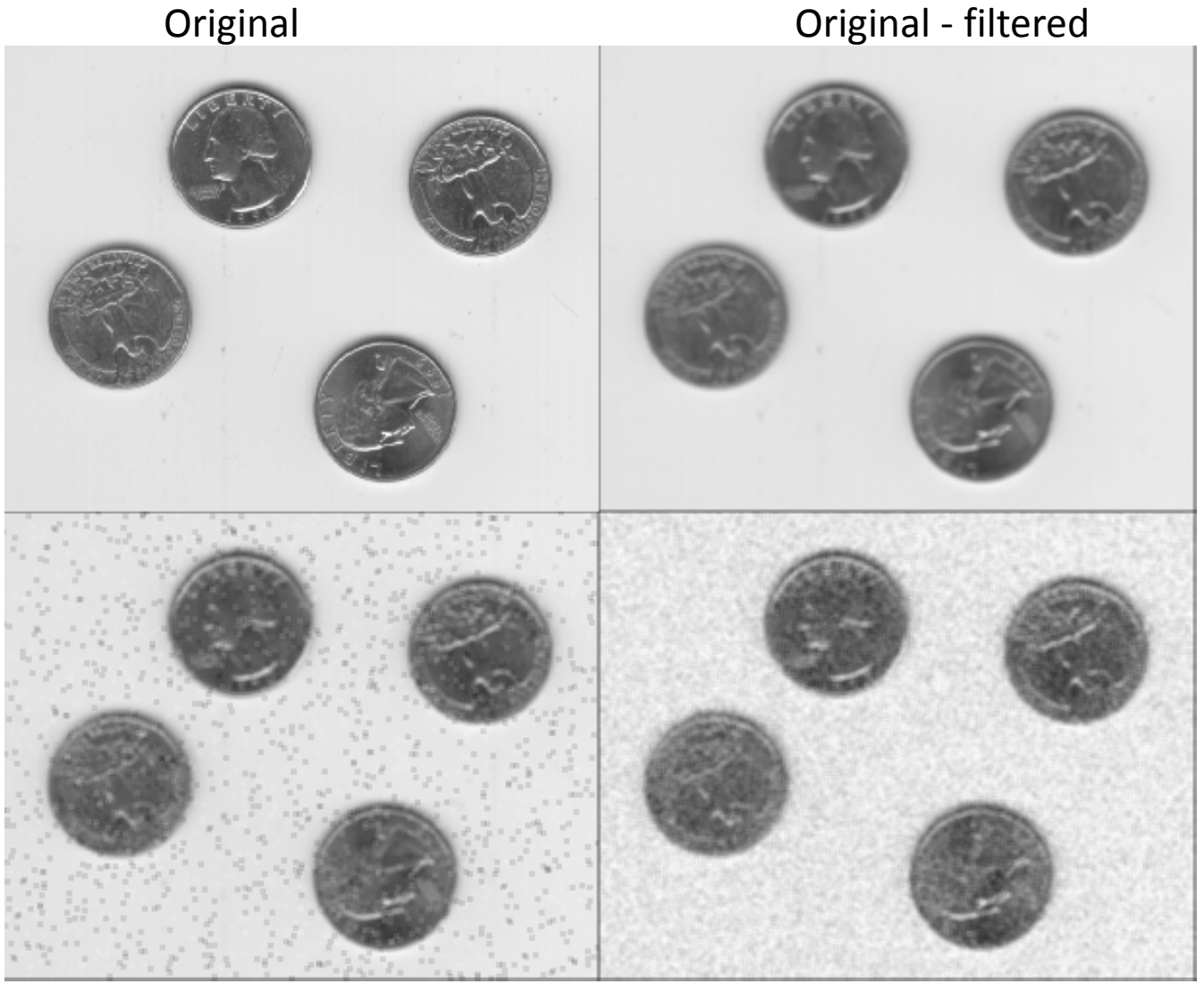
Gaussian noise



Gaussian noise is distributed on each pixel

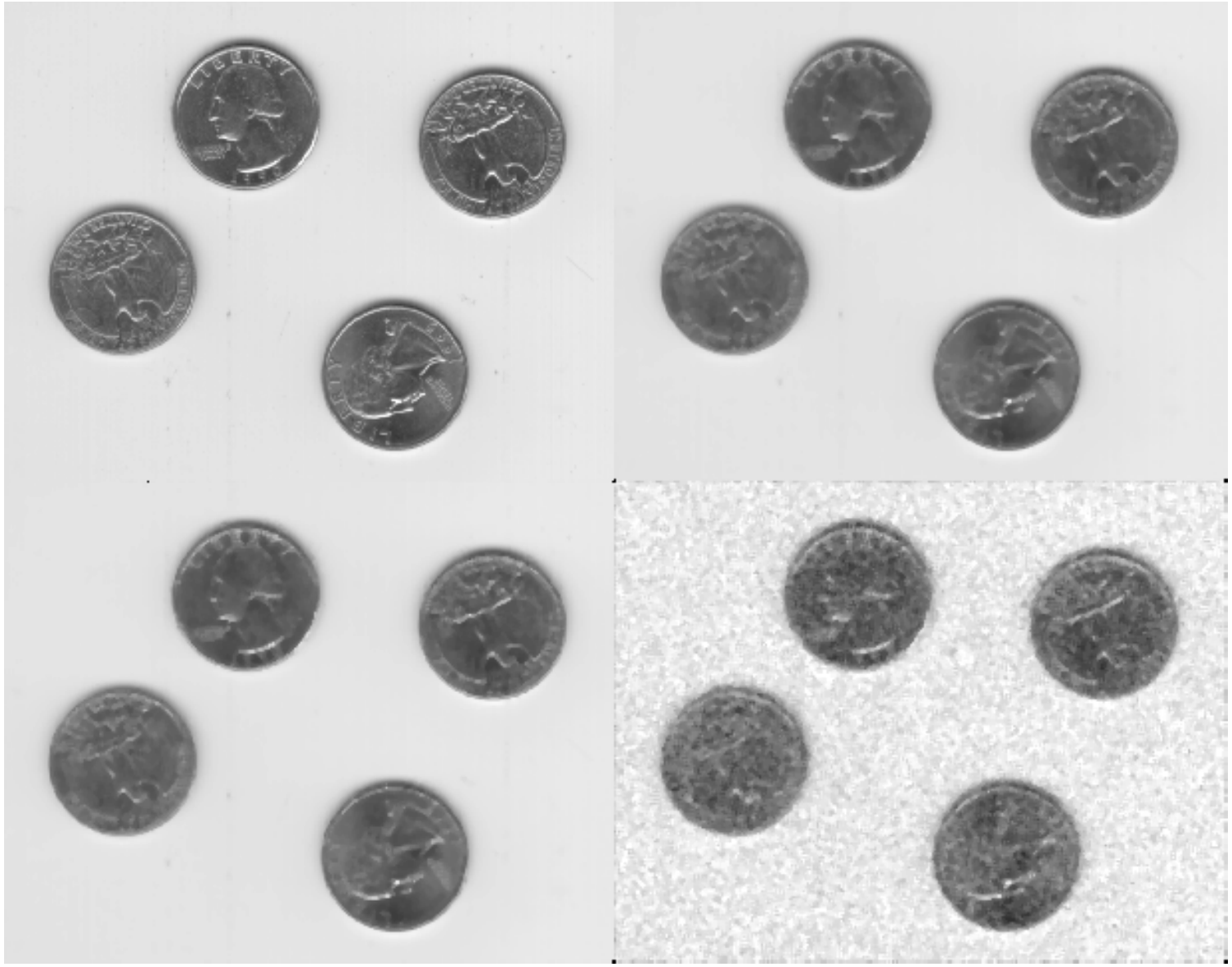
Mean filter

- Does reasonably well with Gaussian noise, but the extremes of the salt and pepper noise don’t work.
- The noise removal comes at the expense of the edge detail – high frequency portions of image
- Larger filters will further suppress noise, but at further expense of edge detail
- Often used with threshold – replace value IF change in value is below some set threshold…



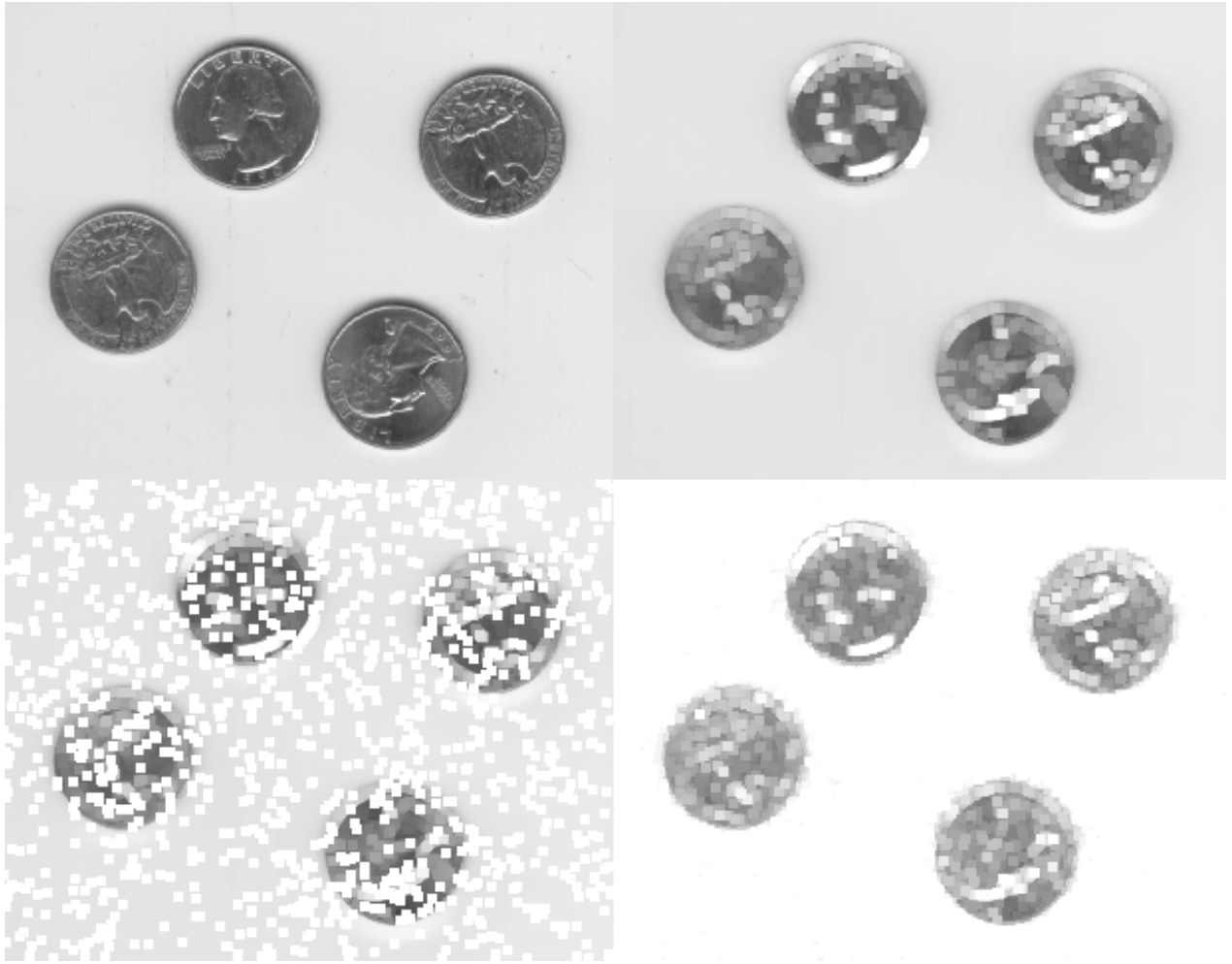
Median filtering

- Collect a neighborhood
- Rank values in order (from lowest to highest)
- Replace pixel with the median value of the neighborhood
- Higher computational cost than mean filter
- Better at preserving edges, high frequency detail
- Very good at removing salt and pepper noise



Median filtering is a subset of rank filters

- Collect a neighborhood
- Rank values in order (from lowest to highest)
- Do something with that value
  - Choose min or max
  - Threshold on values other than center pixel
- Similar to morphological operations we discussed earlier…



Gaussian 2D filters

- Kernel is formed from 2D Gaussian distribution

$$f(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- We have to determine the filter size, as well as the standard deviation (σ) of the Gaussian function
- Larger σ means that it will be blurred out more

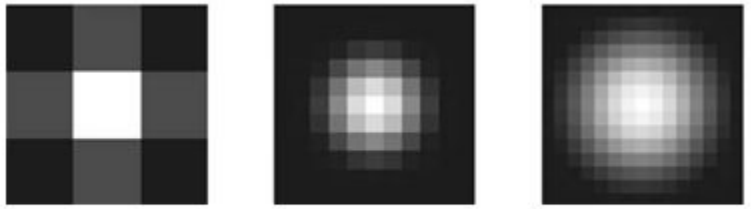


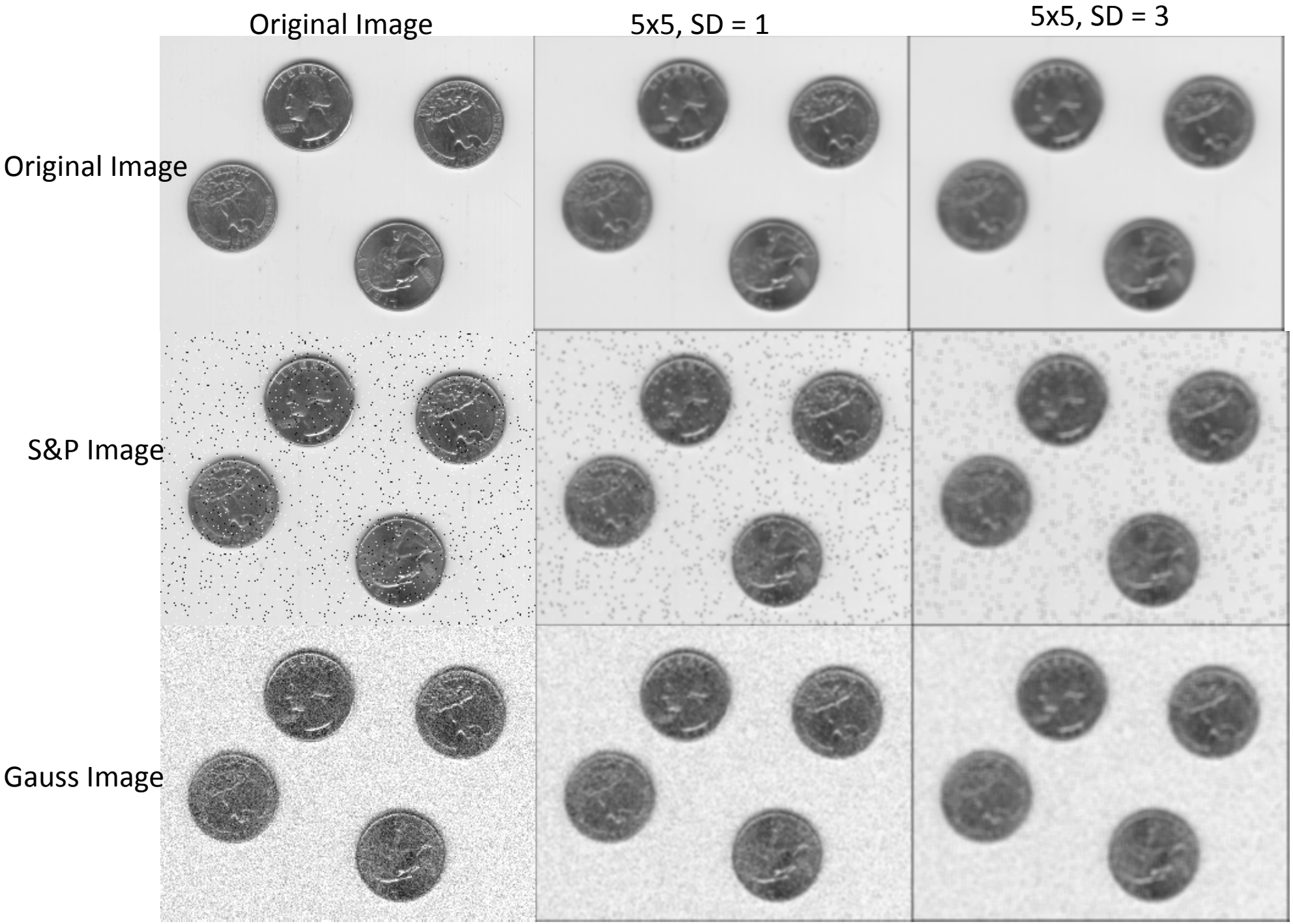
Figure 4.7 Gaussian filter kernels $3 \times 3\sigma = 1$, 11×11 $\sigma = 2$ and 21×21 $\sigma = 4$ (The numerical values shown are unnormalised)

5x5 Gaussian – SD = 1

0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

5x5 Gaussian – SD = 3

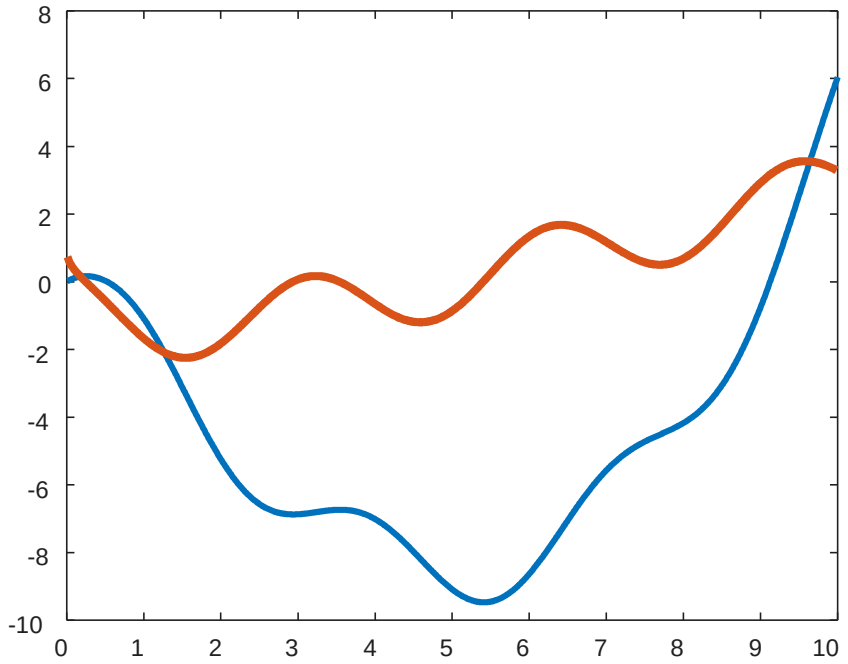
0.0318	0.0375	0.0397	0.0375	0.0318
0.0375	0.0443	0.0469	0.0443	0.0375
0.0397	0.0469	0.0495	0.0469	0.0397
0.0375	0.0443	0.0469	0.0443	0.0375
0.0318	0.0375	0.0397	0.0375	0.0318


Original Image
5x5, SD = 1
5x5, SD = 3
Original Image
S&P Image
Gauss Image

Filters for edge detection

- Derivatives will find changes of intensity in space
- If the intensity is flat, the derivative will be zero
- Derivative high if there is a large, sudden change in intensity
- Edges are defined by large, sudden changes in intensity

2-D derivative measure	Continuous case	Discrete case
$\frac{\partial f}{\partial x}$	$\lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$	$f(x+1,y) - f(x,y)$
$\frac{\partial f}{\partial y}$	$\lim_{\Delta y \to 0} \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$	$f(x,y+1) - f(x,y)$
$\nabla f(x,y)$	$\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$	$[f(x+1,y) - f(x,y), f(x,y+1) - f(x,y)]$



Prewitt and Sobel edges

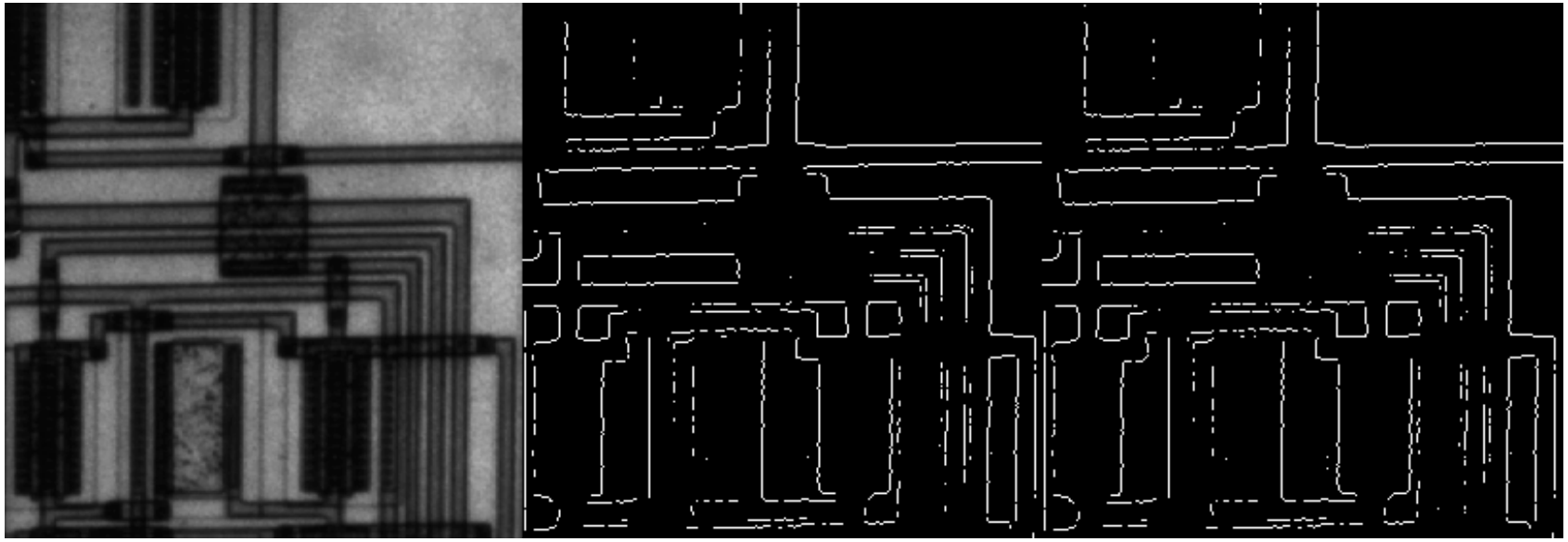
Derivative filters have a sum of 0, so that when the intensity is flat, they respond with a 0

The main difference between Prewitt and Sobel is that Sobel also does a pseudo-Gaussian smoothing in the opposite direction to the edge finding

	Prewitt	Sobel
X derivative	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
Y derivative	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Prewitt and Sobel edges

Sobel Prewitt

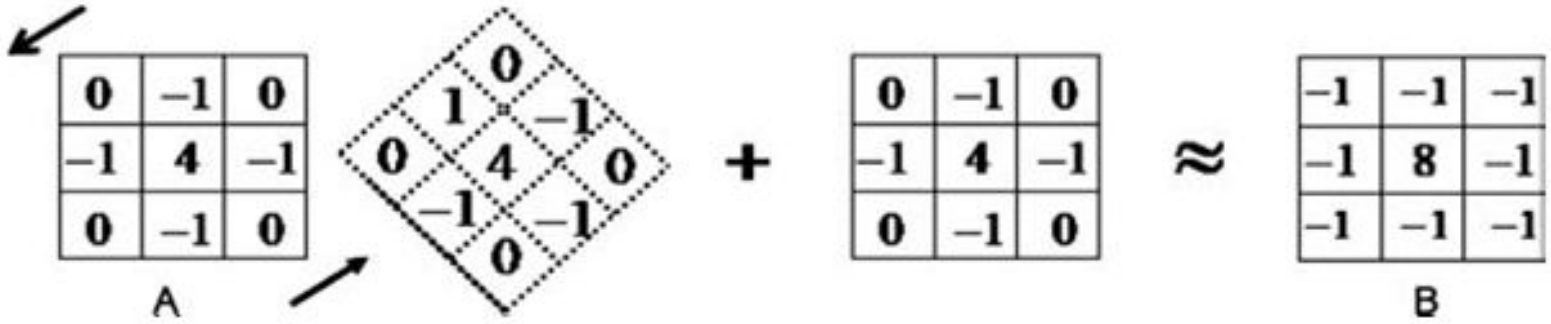


Laplacian is most sensitive edge finder

- Taking the derivative twice
- Involves derivatives in both x and y
- 3x3 filter is easy to construct, but add sensitivity to diagonal edges by summing a rotated filter

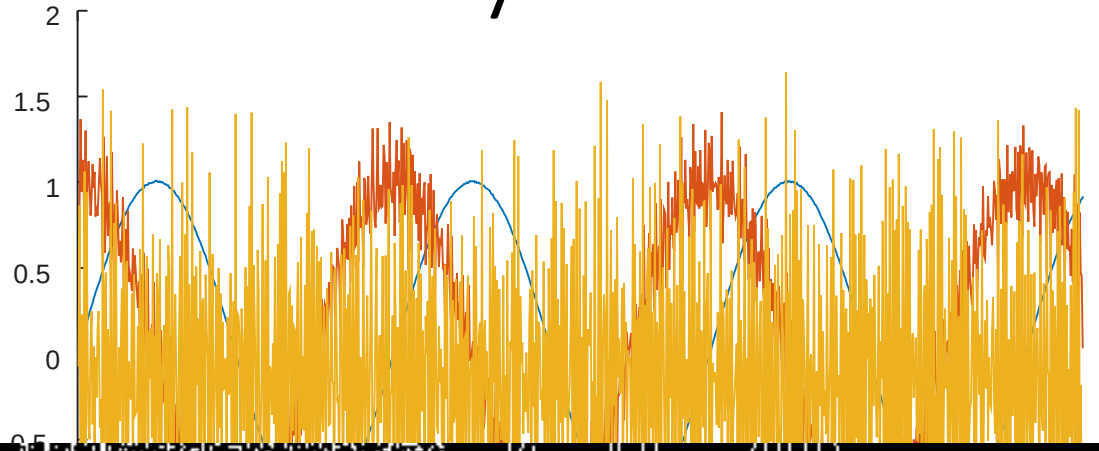
$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

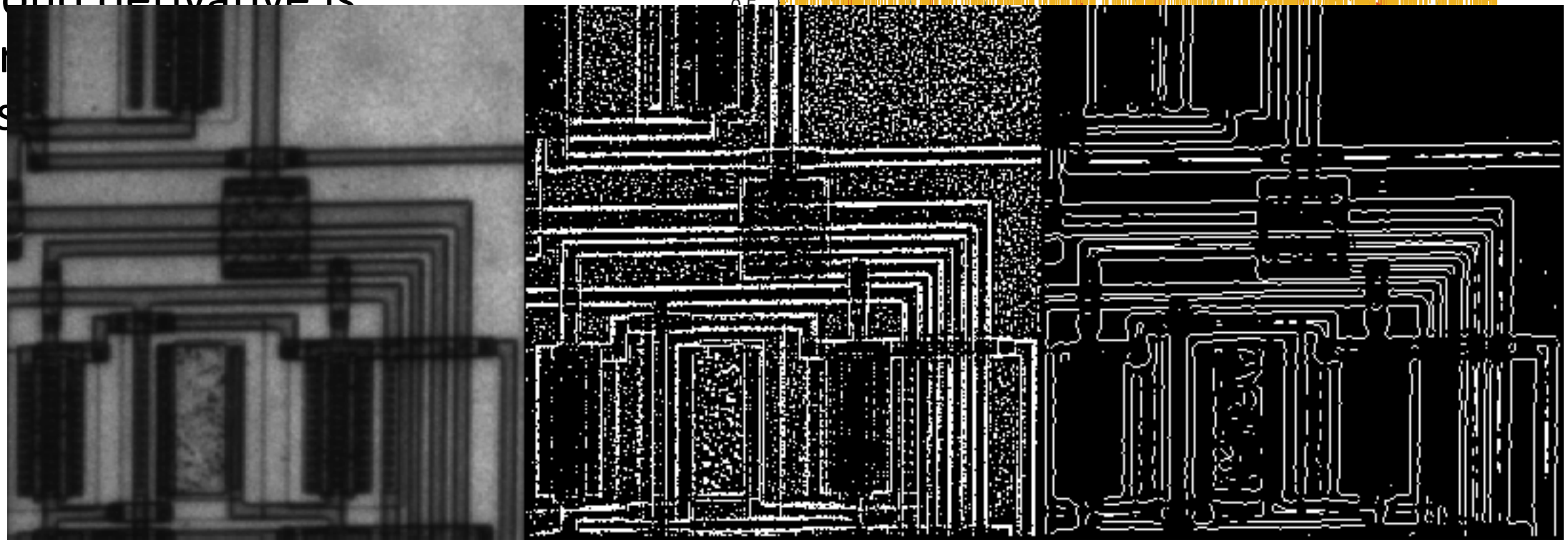
$$\nabla^2 f = f(x+1,y) + f(x-1,y) - 4f(x,y) + f(x,y+1) + f(x,y-1)$$



Laplacian is typically preceded by a smoothing (Gaussian)

- LoG filter
- Laplacian of Gaussian
- Second derivative is
  mor
  nois





Edges to sharpen images

- We can use LoG to find fine features in image
- We can then subtract original image from the edge image
- This will have the effect of increasing contrast at edges
- Sharpening

$$I_{sharp}(x,y) = I_0(x,y) - \nabla^2 I_0(x,y)$$

Unsharpen mask

$$I_{edge}(x,y) = I_{orig}(x,y) - I_{smooth}(x,y)$$

$$I_{unsharp}(x,y) = I_{orig}(x,y) + (k * I_{smooth}(x,y))$$

- Unsharpen is the process of blurring an image, and then subtracting the blurred image from the original

Original | Edge sharpened | Unsharpened

And on to Matlab…