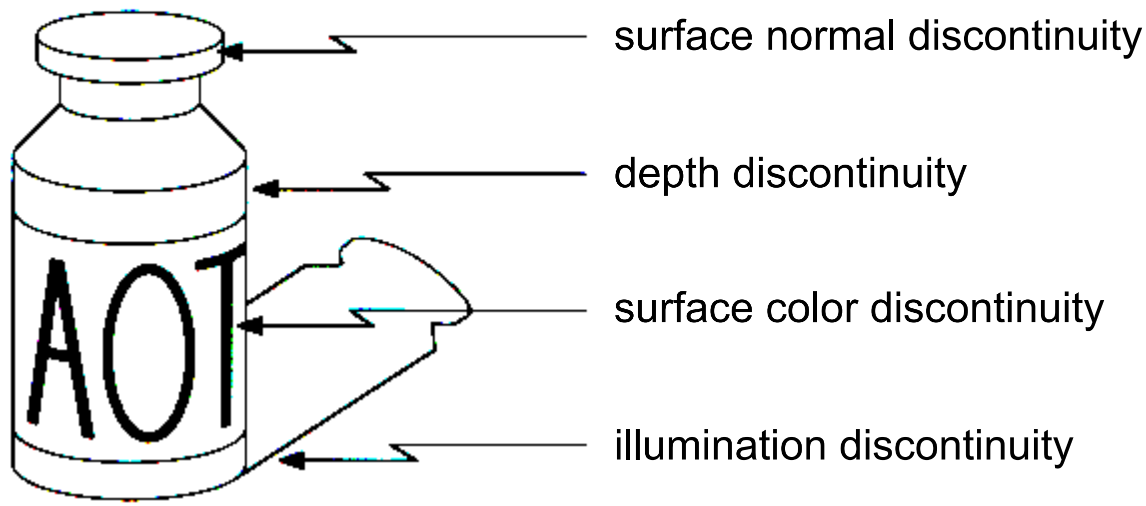
Edges and Scale

From Sandlot Science

Today's reading

- Cipolla & Gee on edge detection (available online)
- Szeliski 3.4.1 – 3.4.2

Origin of Edges



Edges are caused by a variety of factors

Detecting edges

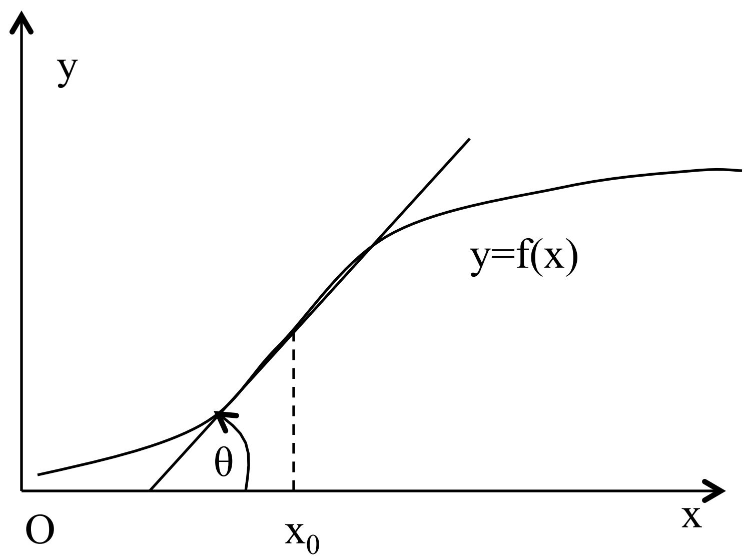
What's an edge?

- intensity discontinuity (= rapid change)

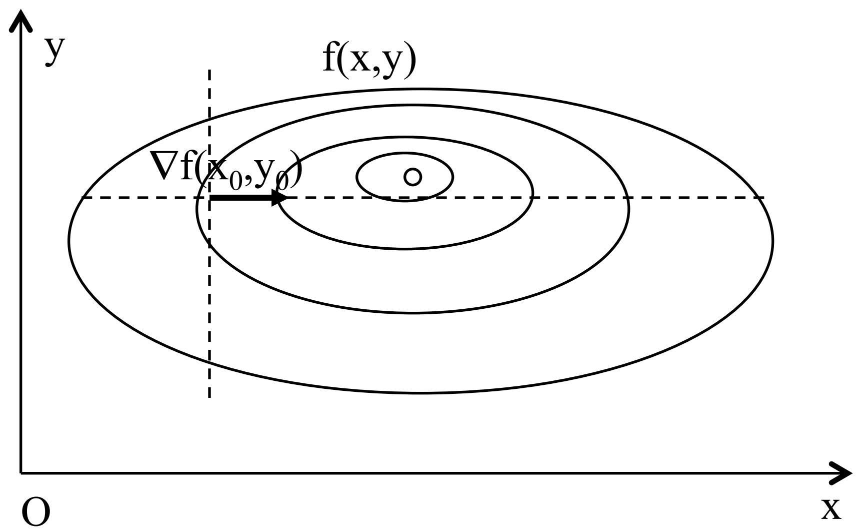
How can we find large changes in intensity?

- gradient operator seems like the right solution

Math Refresher: Vectors and Derivatives



$f'(x)=df/dx=\tan\theta$



Partial derivatives: $\partial f/\partial x$, $\partial f/\partial y$

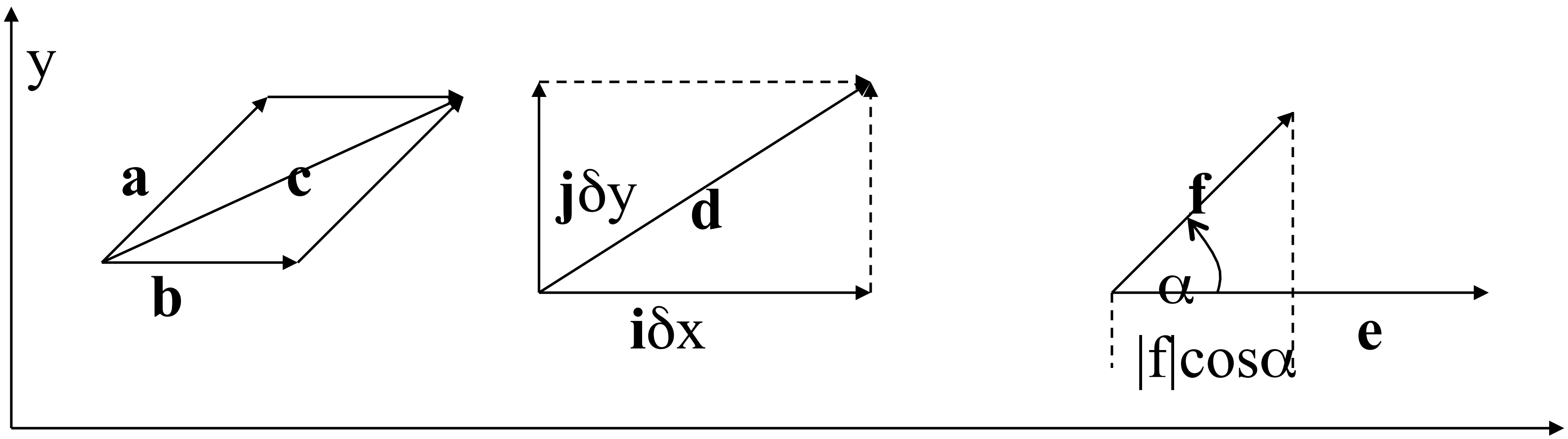
$[\partial f(x,y)/\partial x|_{y=y_0}=f'(x,y_0)]$

Gradient:

$\nabla f(x,y)= \mathbf{i}\partial f/\partial x + \mathbf{j}\partial f/\partial y$

[**i**,**j** – unit vectors in x,y directions]

Math Refresher: Vectors and Derivatives (cont.)



$\mathbf{c}=\mathbf{a}+\mathbf{b}$

$\mathbf{a}=\mathbf{c}-\mathbf{b}$

$\mathbf{d}=\mathbf{i}\delta x+\mathbf{j}\delta y$

$\mathbf{f}=\mathbf{i}x_1+\mathbf{j}y_1$, $\mathbf{e}=\mathbf{i}x_2+\mathbf{j}y_2$

Inner product:

$\mathbf{f}\cdot\mathbf{e} = x_1x_2+y_1y_2 = |\mathbf{f}|\cdot|\mathbf{e}|\cos\alpha$

Directional derivative: $\partial f/\partial n = \nabla f\cdot n$

$\mathbf{a}=\mathbf{i}x_1+\mathbf{j}y_1$, $\mathbf{b}=\mathbf{i}x_2+\mathbf{j}y_2$

$\mathbf{c}=\mathbf{a}+\mathbf{b}=\mathbf{i}(x_1+x_2)+\mathbf{j}(y_1+y_2)$

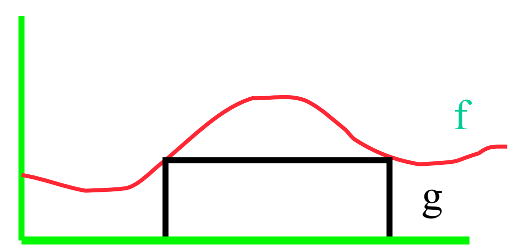
Smoothing and convolution

The convolution of two functions, f(x) and g(x) is defined as

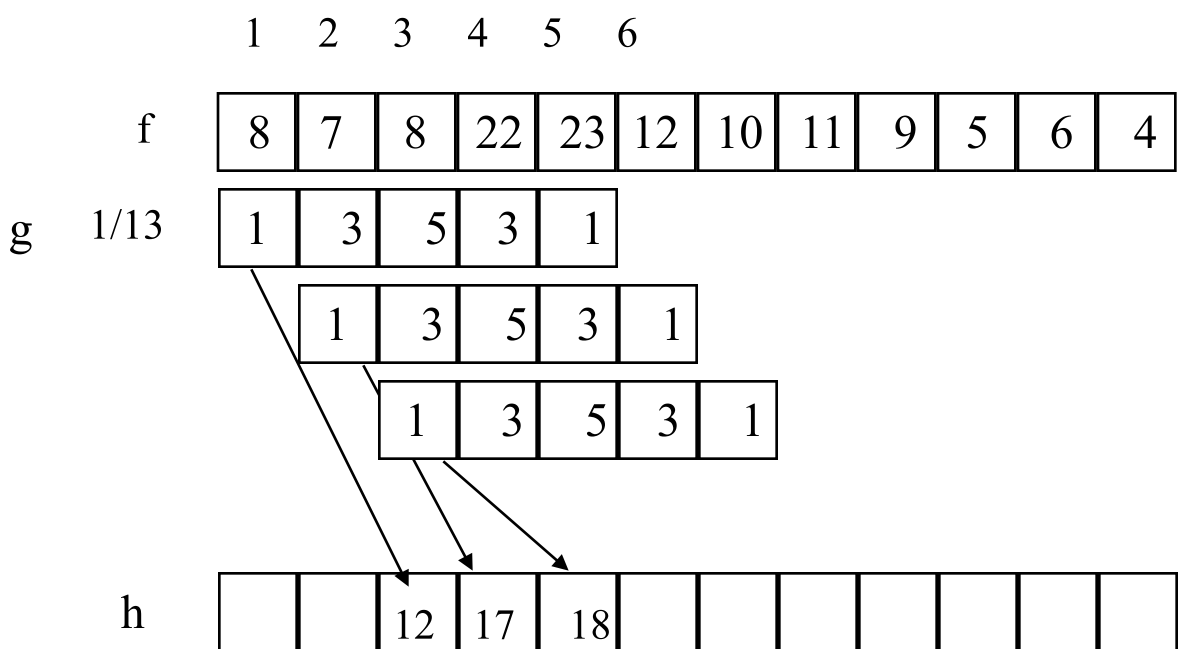
$$h(x) = \int_{-\infty}^{\infty} g(x')f(x - x')dx' = g(x) * f(x)$$

When the functions f and g are discrete and when g is nonzero only over a finite range [-n,n] then this integral is replaced by the following summation:

$$h(i) = \sum_{j=-n}^{n} g(j)f(i + j)$$



Example of 1-d convolution



$$h(4) = \sum_{j=-2}^{2} g(j) f(4 + j)$$

$$= g(-2) f(2) + g(-1) f(3) + g(0) f(4) + g(1) f(5) + g(2) f(6)$$

Smoothing and convolution

These integrals and summations extend simply to functions of two variables:

$$h(i,j) = f(i,j) * g = \sum_{k=-n}^{n} \sum_{l=-n}^{n} g(k,l) f(i+k, j+l)$$

Convolution computes the weighted sum of the gray levels in each *nxn* neighborhood of the image, *f*, using the matrix of weights *g*.

Convolution is a so-called linear operator because

- $g*(af_1 + bf_2) = a(g*f_1) + b(g*f_2)$

2-D convolution

$$h(5,5) = \sum_{k=-1}^{1}\sum_{l=-1}^{1} g(k,l) f(5+k,5+l)$$

$$= g(-1,-1)f(4,4) + g(-1,0)f(4,5) + g(-1,1)f(4,4)$$

$$+ g(0,-1)f(5,4) + g(0,0)f(5,5) + g(0,1)f(5,6)$$

$$+ g(1,-1)f(6,4) + g(1,0)f(6,5) + g(1,1)f(6,6)$$

Separability

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} \qquad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} \qquad \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

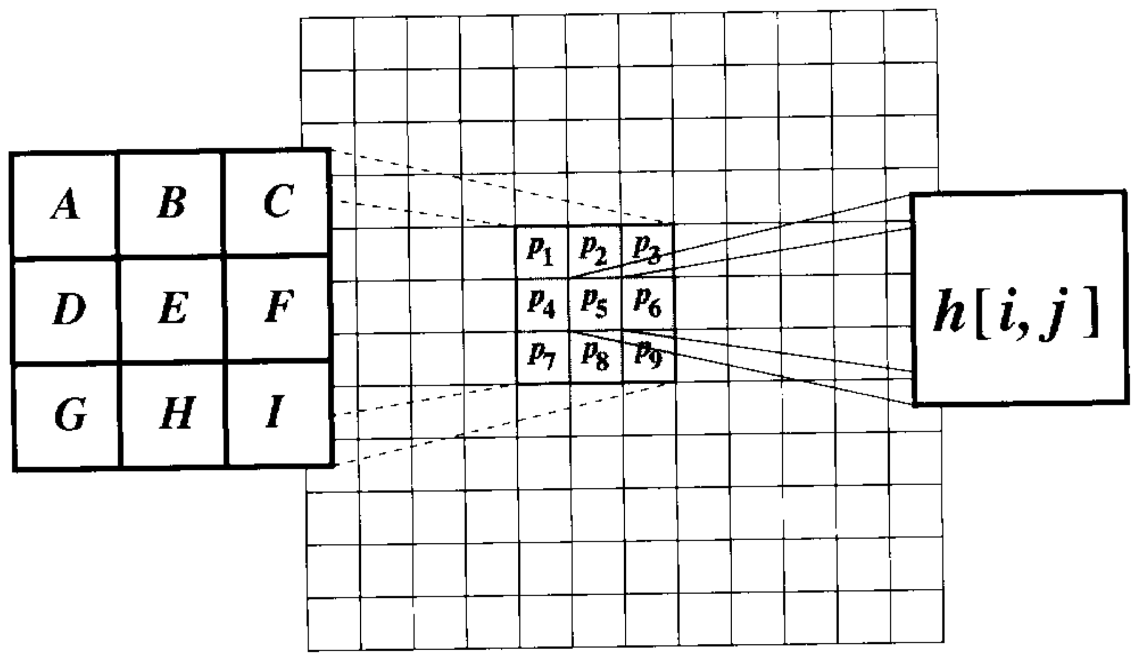
$= 2 + 6 + 3 = 11$

$= 6 + 20 + 10 = 36$

$= 4 + 8 + 6 = 18$

65

Smoothing and convolution

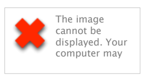


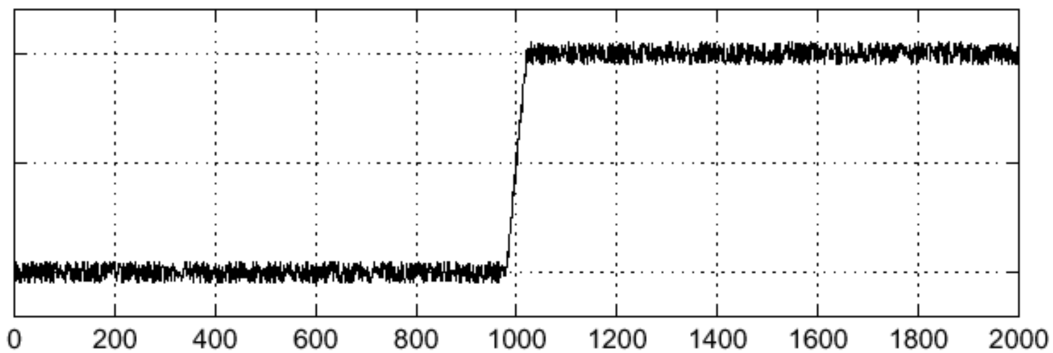
$$h[i,j] = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

Effects of noise

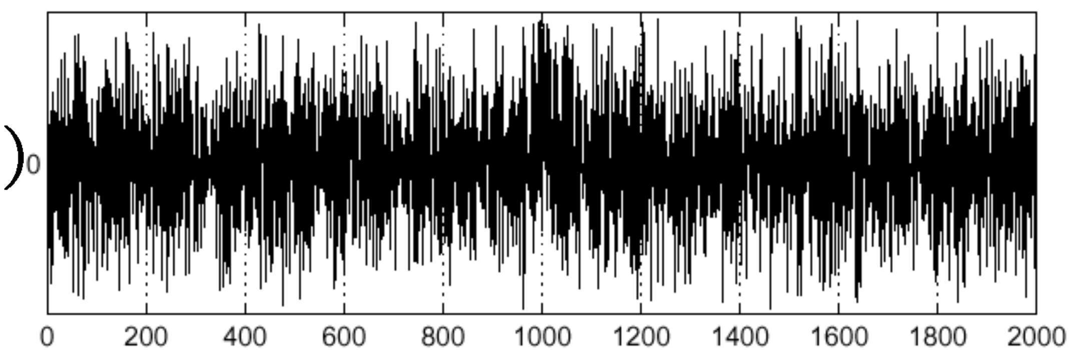
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



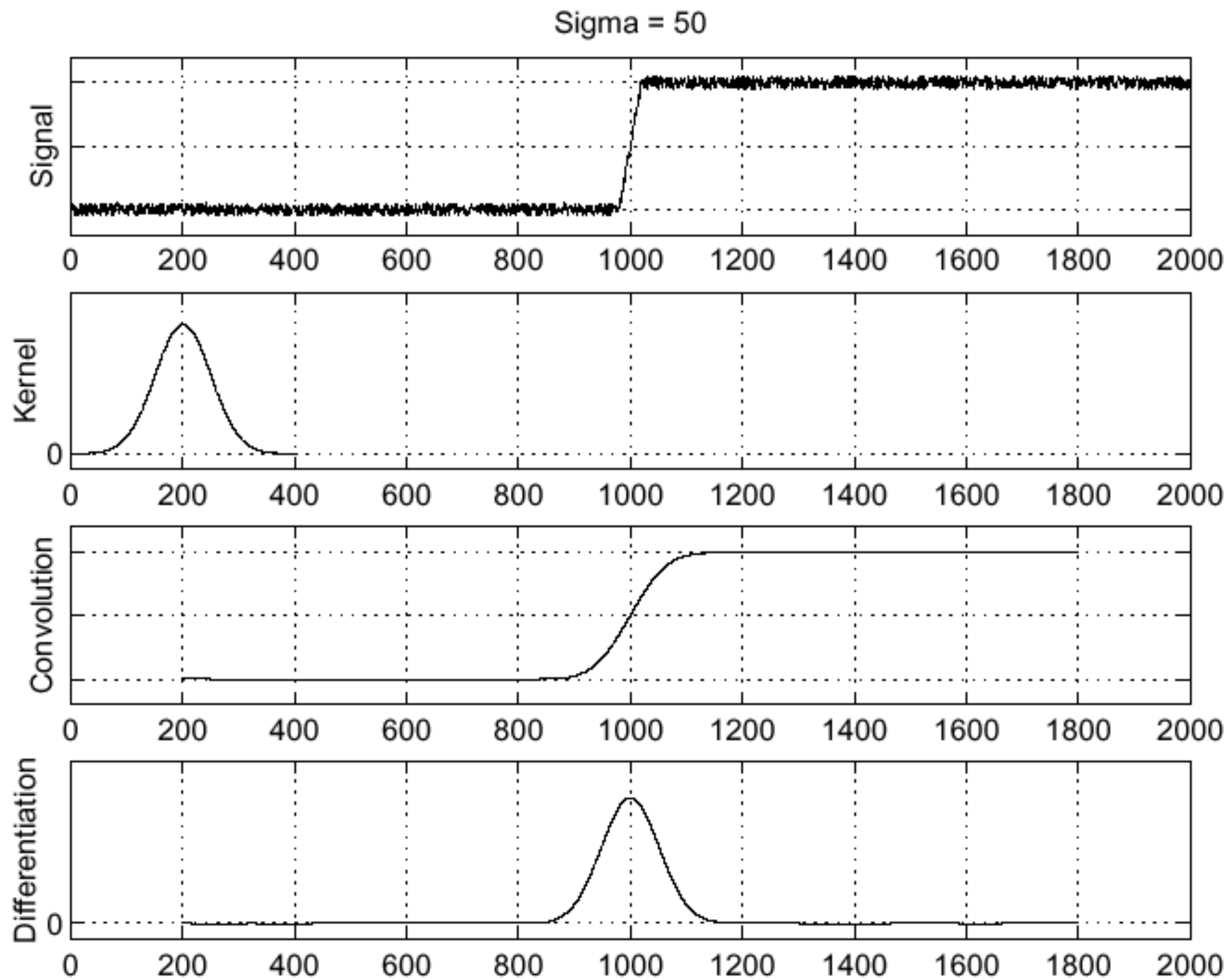


$$\frac{d}{dx}f(x)$$



Where is the edge?

Solution: smooth first



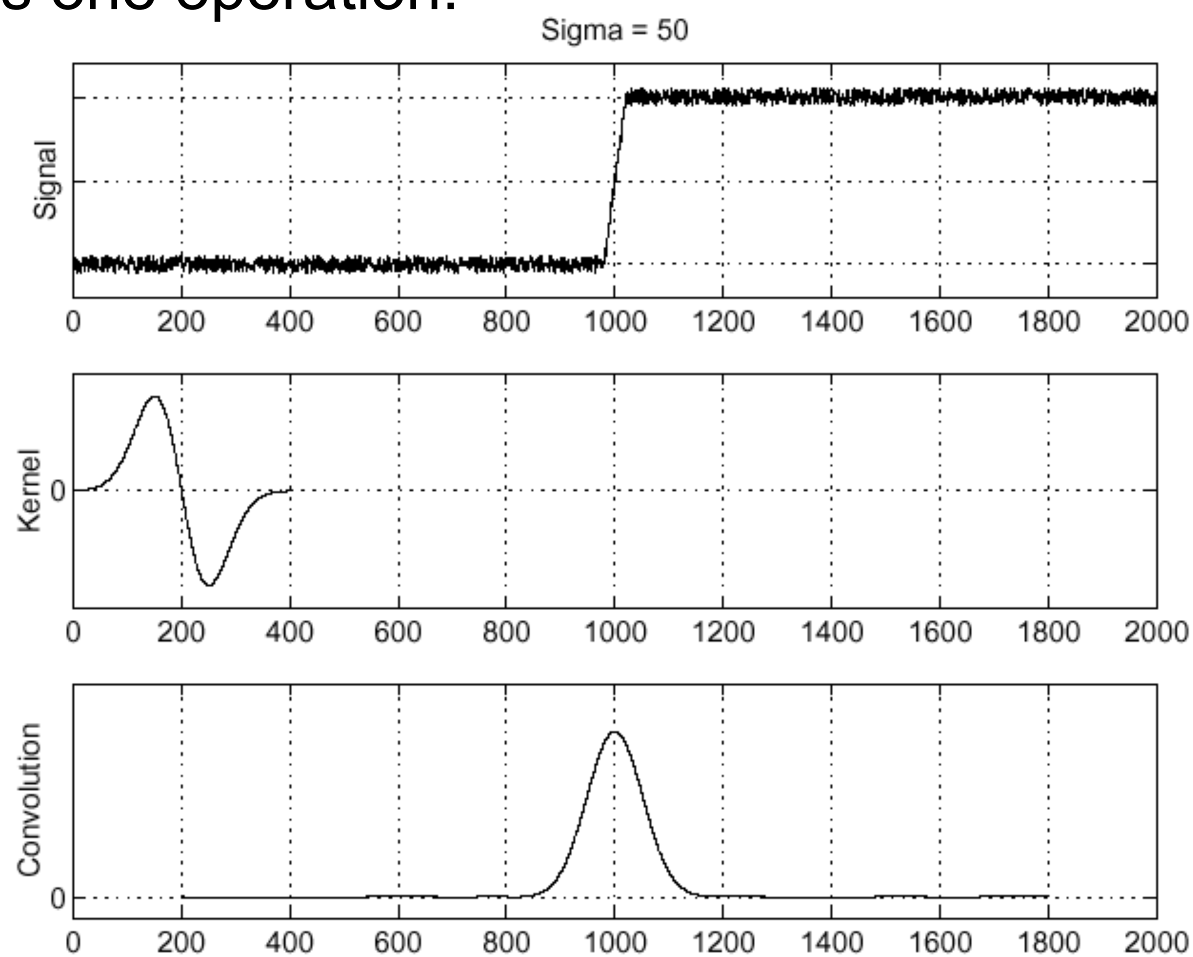
$$\frac{\partial}{\partial x}(h \star f)$$

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

Associative property of convolution

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

This saves us one operation:

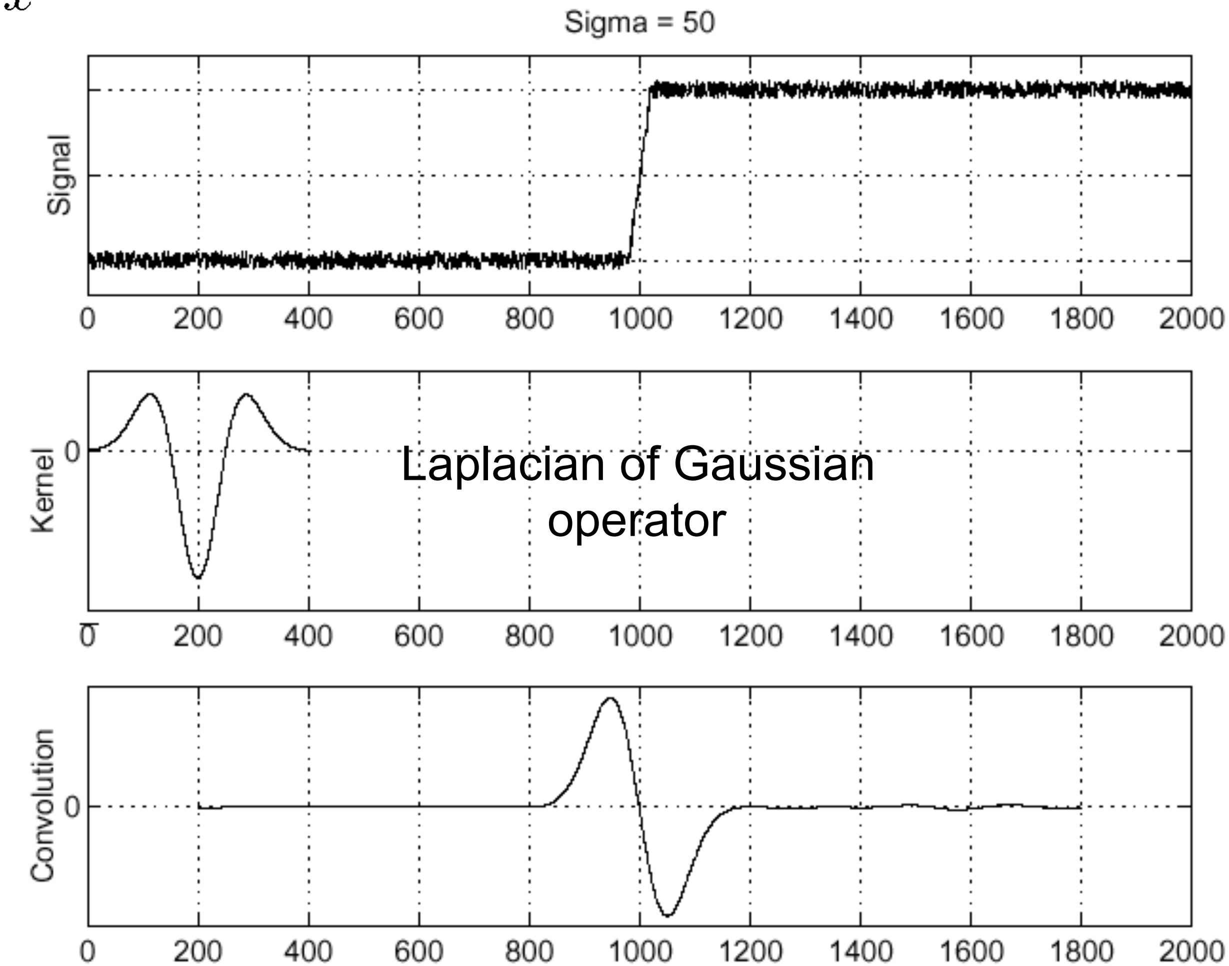


$$\frac{\partial}{\partial x}h$$

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

Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

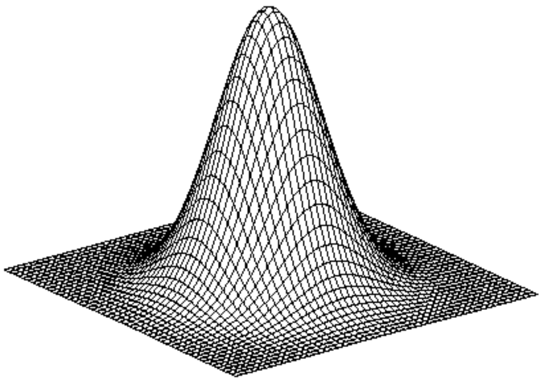


$$\frac{\partial^2}{\partial x^2}h$$

$$\left(\frac{\partial^2}{\partial x^2}h\right) \star f$$

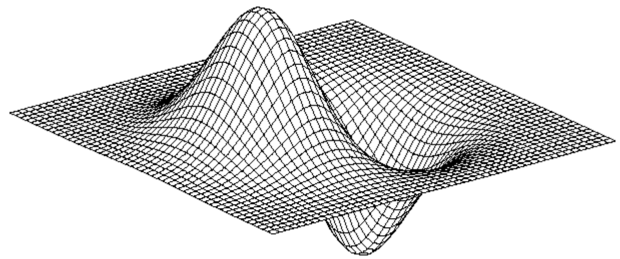
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

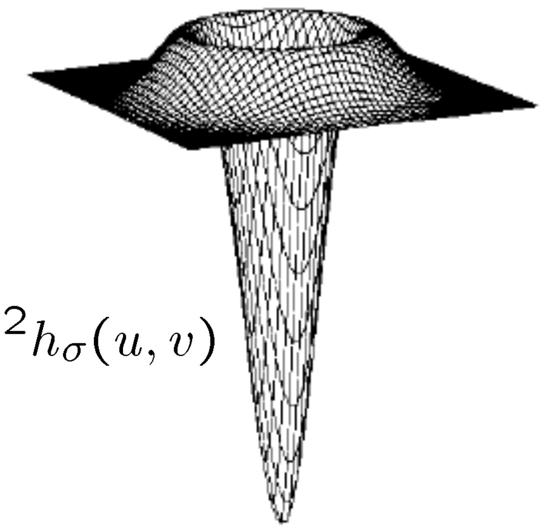
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



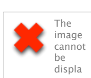
derivative of Gaussian

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

Laplacian of Gaussian



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



is the **Laplacian** operator:

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

The Sobel operator

Common approximation of derivative of Gaussian

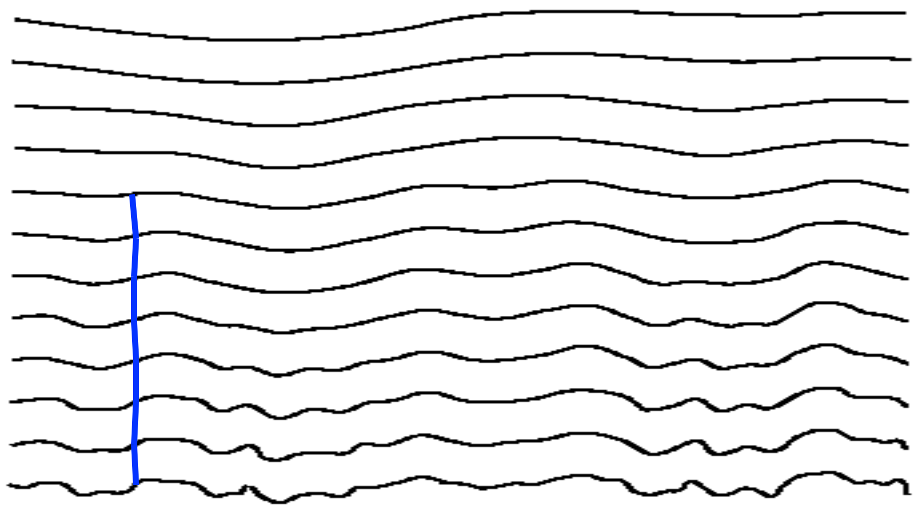
-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

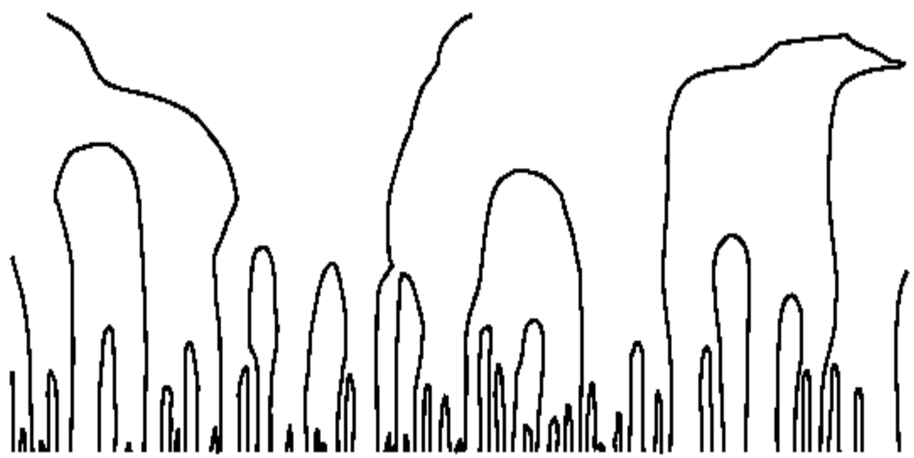
- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

The effect of scale on edge detection

larger

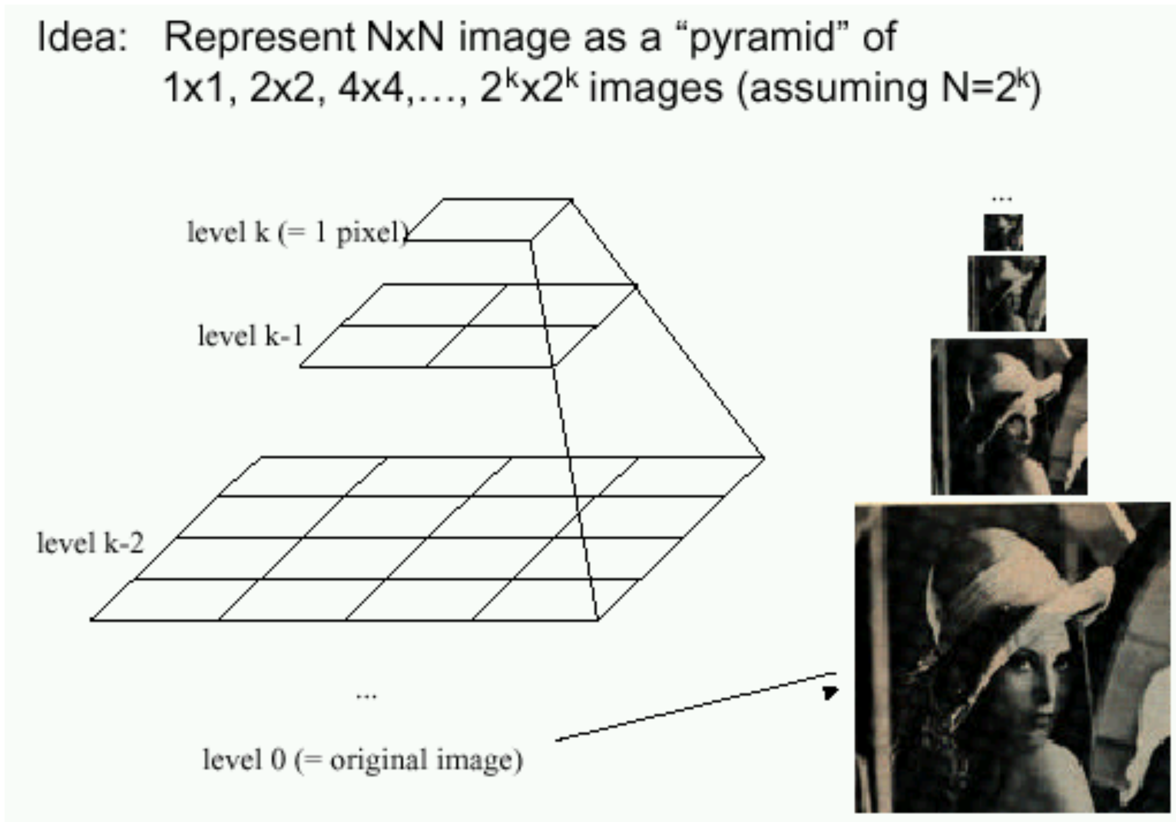


larger



Scale space (Witkin 83)

Some times we want many resolutions

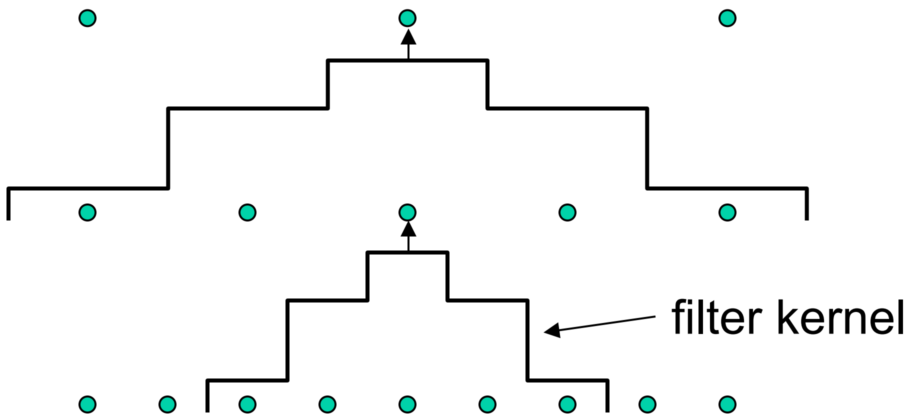


Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision

Gaussian pyramid construction



Repeat

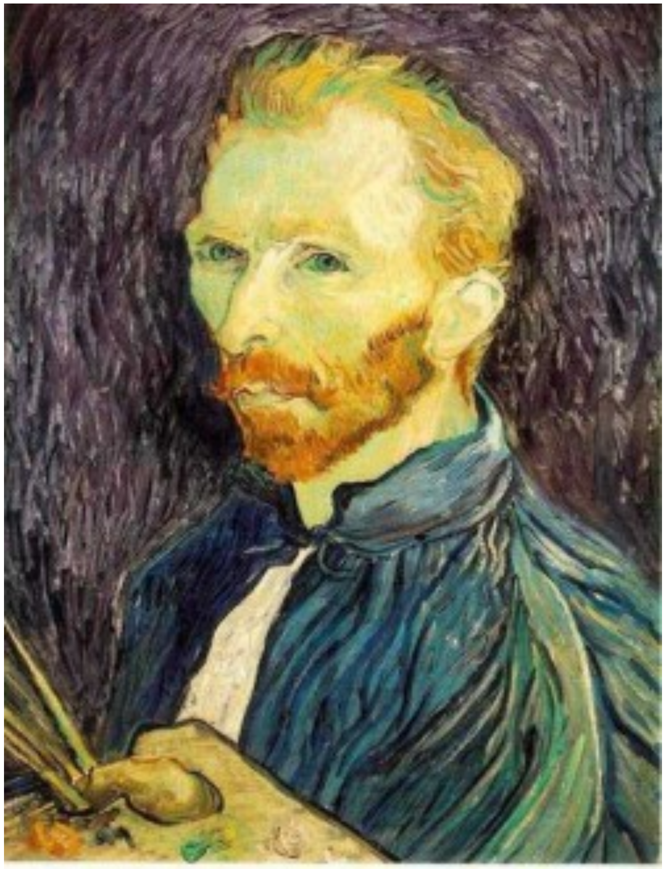
- Filter
- Subsample

Until minimum resolution reached

- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

Subsampling with Gaussian pre-filtering



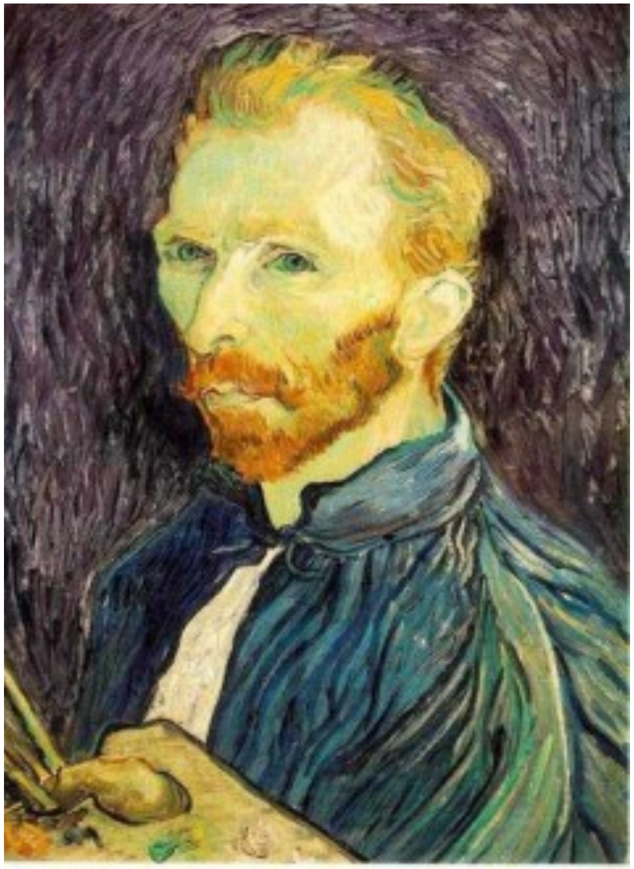
Gaussian 1/2

G 1/4

G 1/8

Filter the image, *then* subsample

Subsampling with Gaussian pre-filtering



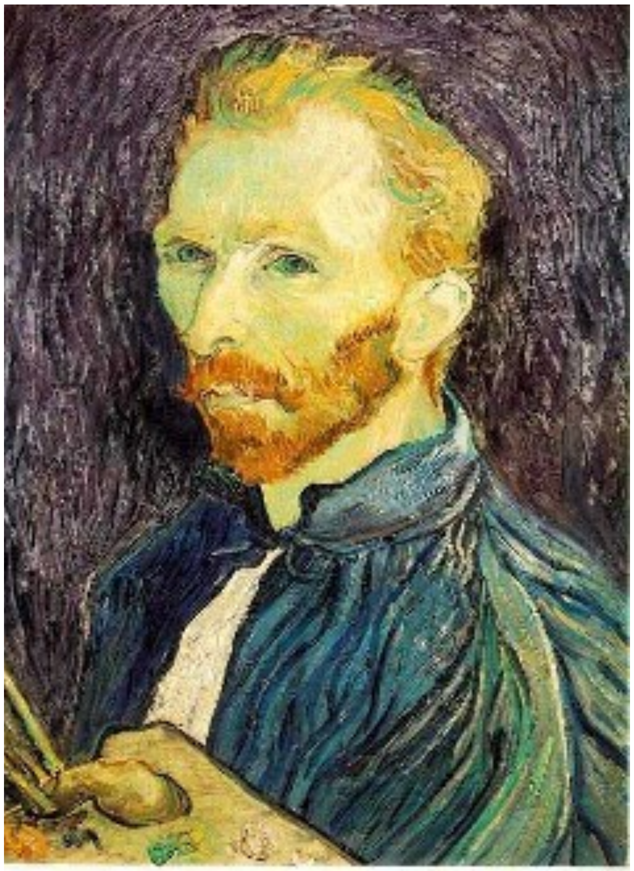
Gaussian 1/2

G 1/4

G 1/8

Filter the image, *then* subsample

Subsampling *without* pre-filtering

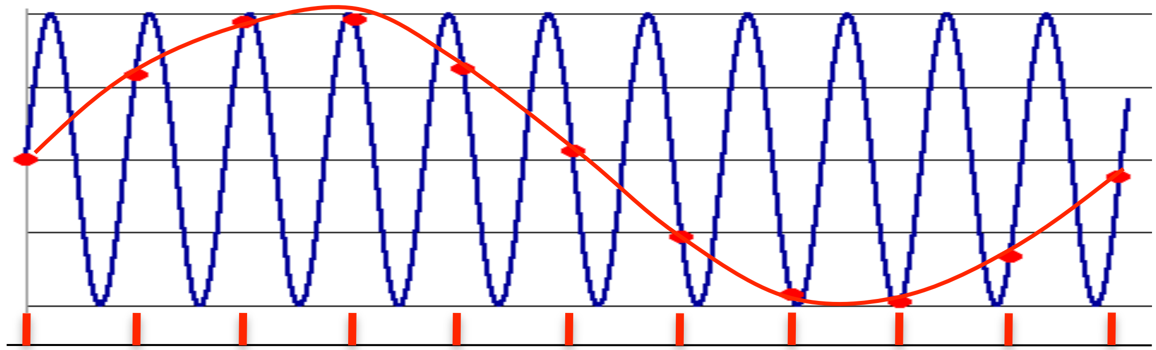


1/2

1/4 (2x zoom)

1/8 (4x zoom)

Sampling and the Nyquist rate



Aliasing can arise when you sample a continuous signal or image

- occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an *alias*
- formally, the image contains structure at different scales
  - called "frequencies" in the Fourier domain
- the sampling rate must be high enough to capture the highest frequency in the image

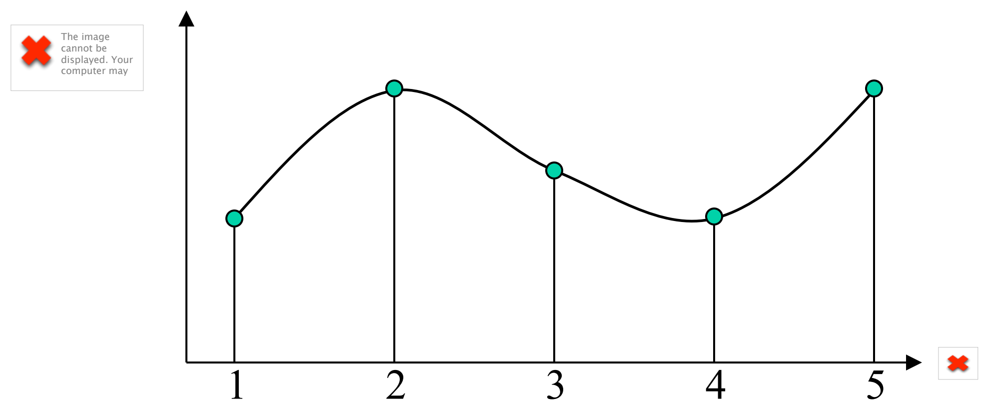
To avoid aliasing:

- sampling rate ≥ 2 * max frequency in the image
  - said another way: ≥ two samples per cycle
- This minimum sampling rate is called the **Nyquist rate**

Image resampling

So far, we considered only power-of-two subsampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?



d = 1 in this example

Recall how a digital image is formed

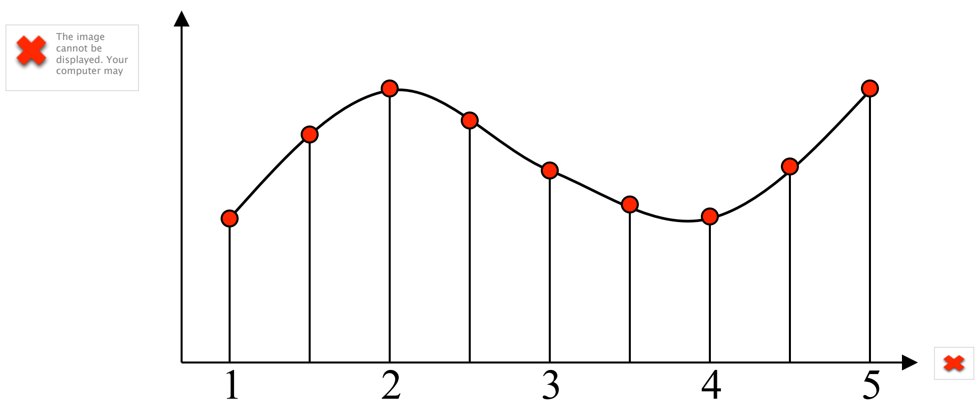
$$F[x, y] = \mathrm{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Image resampling

So far, we considered only power-of-two subsampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?



d = 1 in this example

Recall how a digital image is formed

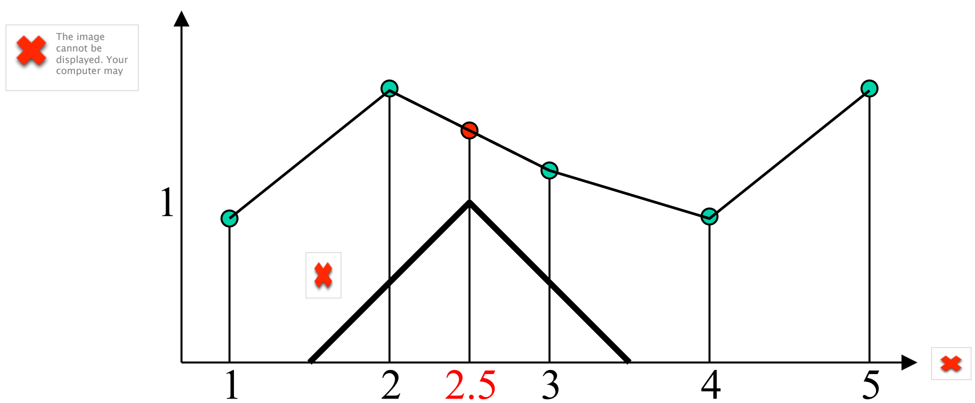
$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Image resampling

So what to do if we don't know

- Answer: guess an approximation
- Can be done in a principled way: filtering



d = 1 in this example

Image reconstruction

- Convert to a continuous function

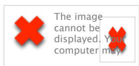
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

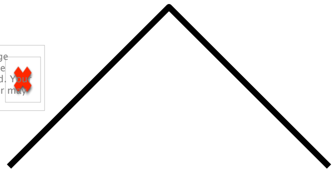
- Reconstruct by cross-correlation:

$$\tilde{f} = h \otimes f_F$$

Resampling filters

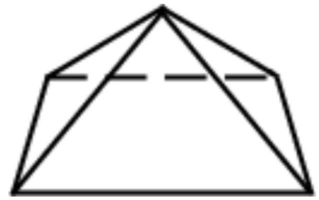
What does the 2D version of this hat function look like?



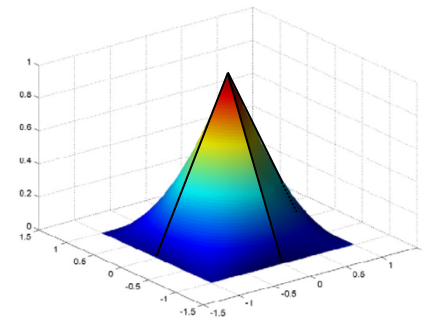


performs
linear interpolation

$h(x, y)$



(tent function) performs
bilinear interpolation



Often implemented without cross-correlation

- E.g., http://en.wikipedia.org/wiki/Bilinear_interpolation

Better filters give better resampled images

- Bicubic is common choice
  - fit 3rd degree polynomial surface to pixels in neighborhood

Example: Subsample at $\sqrt{2}$

Subsample at $\sqrt{2}$ rather than 2

$N, N/\sqrt{2}, N/2, N/2\sqrt{2}, N/4, \ldots$

$1024, 724, 512, 362, 256, \ldots$

Q: How do you determine values for lower resolution images?
A: Use bilinear interpolation

Given

$f(x,y), f(x+1,y), f(x,y+1), f(x+1,y+1), 0<a<1, 0<b<1$

Estimate $f(x+a,y+b)$ as

$$f_1 = (1-a)f(x,y) + af(x+1,y)$$
$$f_2 = (1-a)f(x,y+1)+af(x+1,y+1)$$
$$f(x+a,y+b) = (1-b)f_1 + bf_2$$