Resampling

Image Scaling

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

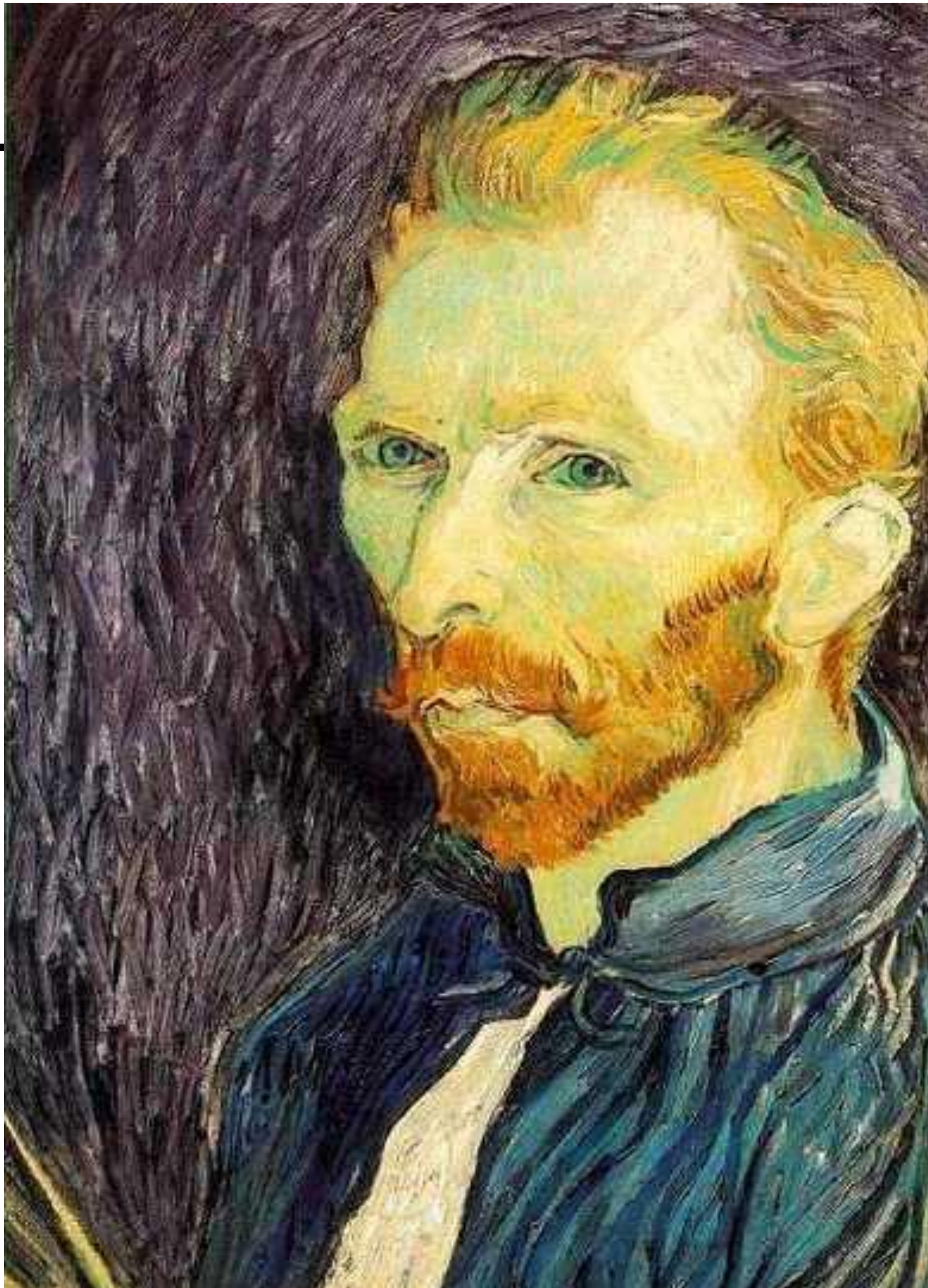
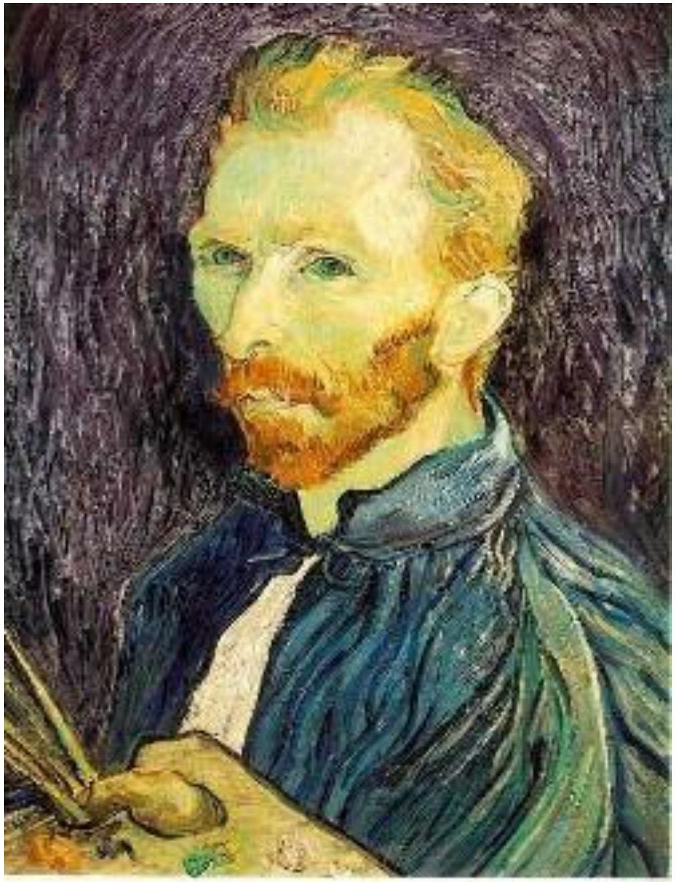


Image sub-sampling



1/4

1/8

Throw away every other row and column to create a 1/2 size image

Why does this look so crufty?

- Called **nearest-neighbor** sampling

Even worse for synthetic images

Sampling and the Nyquist rate

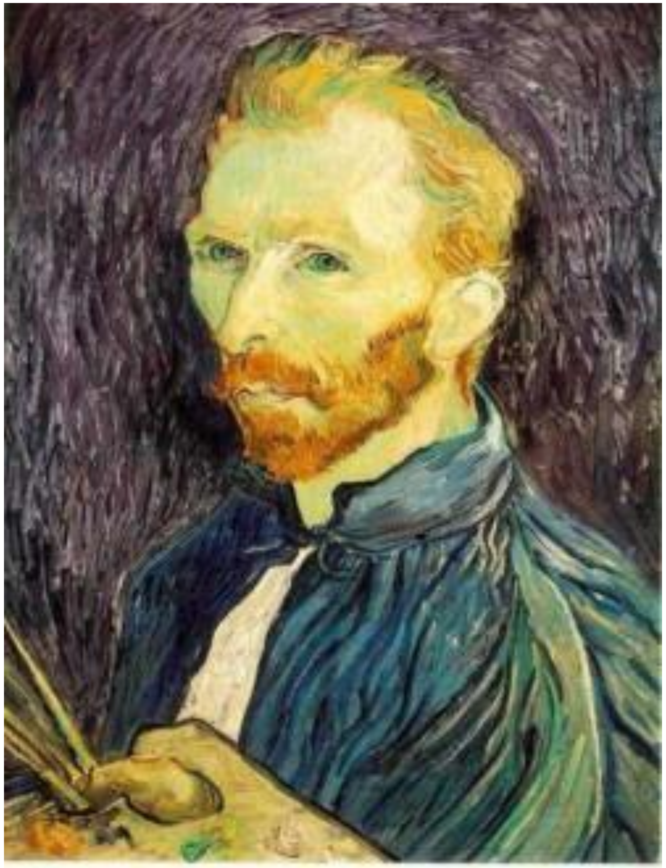
Aliasing can arise when you sample a continuous signal or image

- Demo applet http://www.cs.brown.edu/exploratories/freeSoftware/repository/edu/brown/cs/exploratories/applets/nyquist/nyquist_limit_java_plugin.html
- occurs when your sampling rate is not high enough to capture the amount of detail in your image
- 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
  - i.e., need more than two samples per period
- This minimum sampling rate is called the **Nyquist rate**

Subsampling with Gaussian pre-filtering



Gaussian 1/2

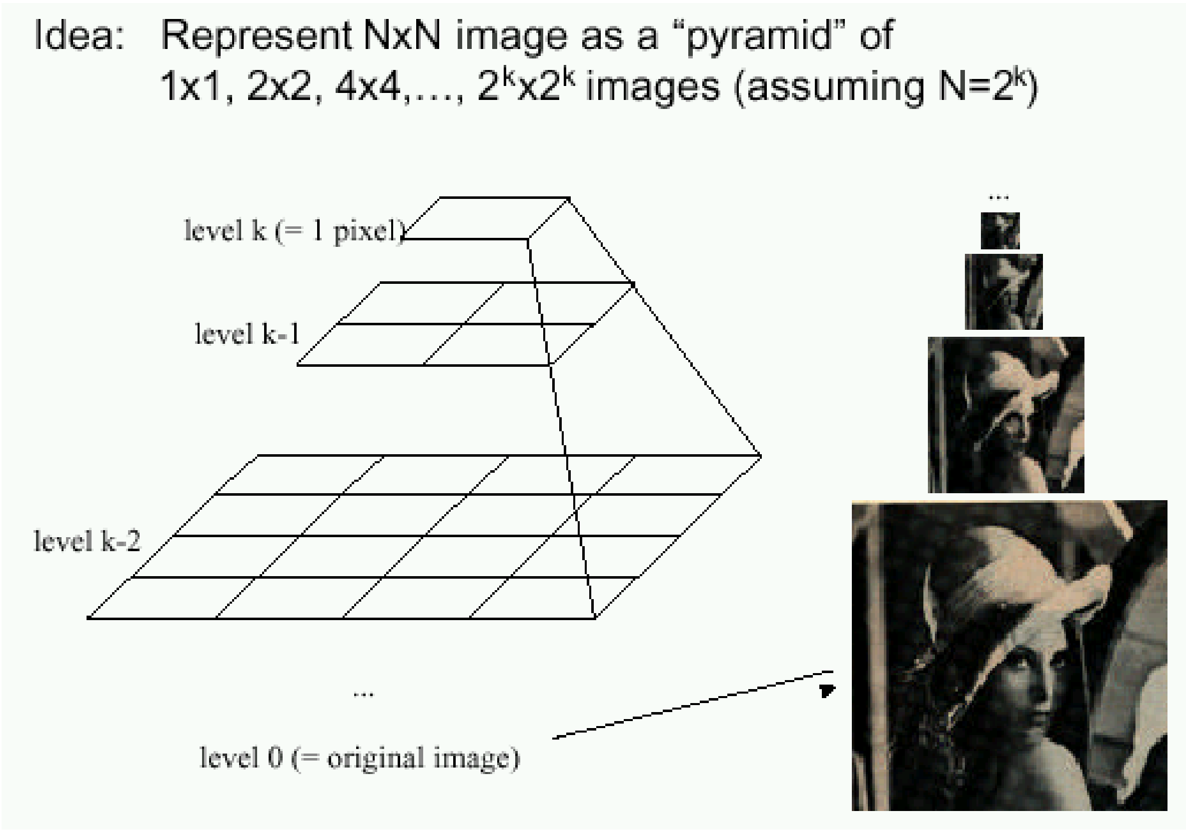
G 1/4

G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Some times we want many resolutions



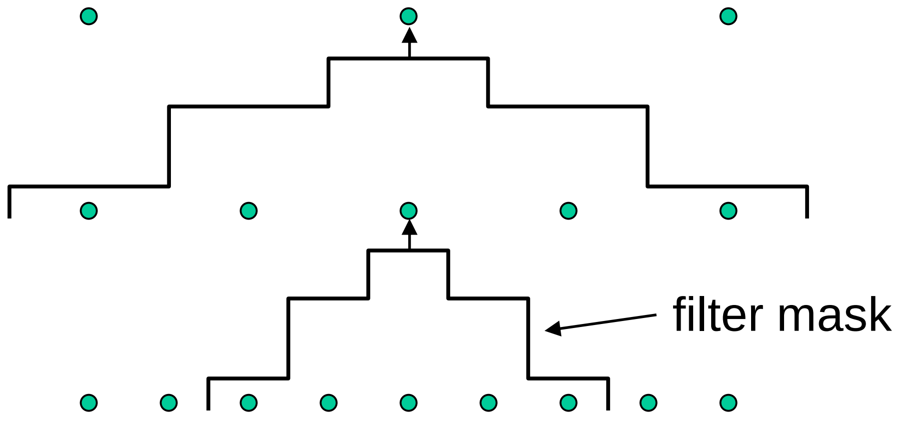
Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

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

Gaussian Pyramids have all sorts of applications in computer vision

- We'll talk about these later in the course

Gaussian pyramid construction



Repeat

- Filter
- Subsample

Until minimum resolution reached

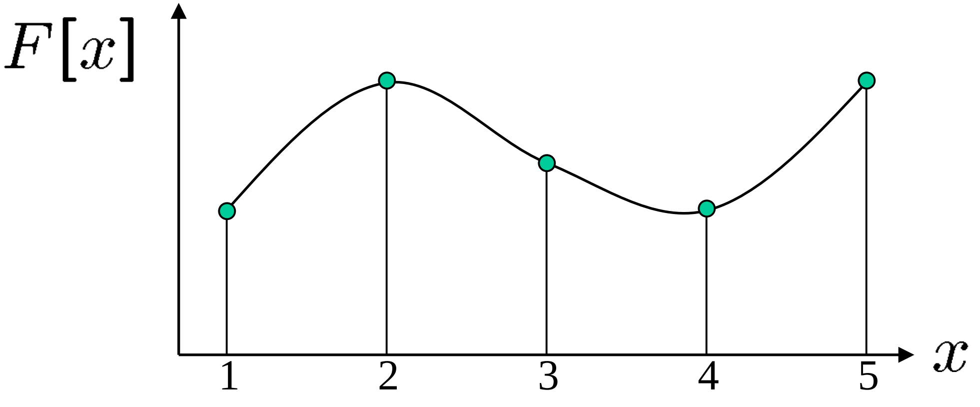
- can specify desired number of levels (e.g., 3-level pyramid)

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

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

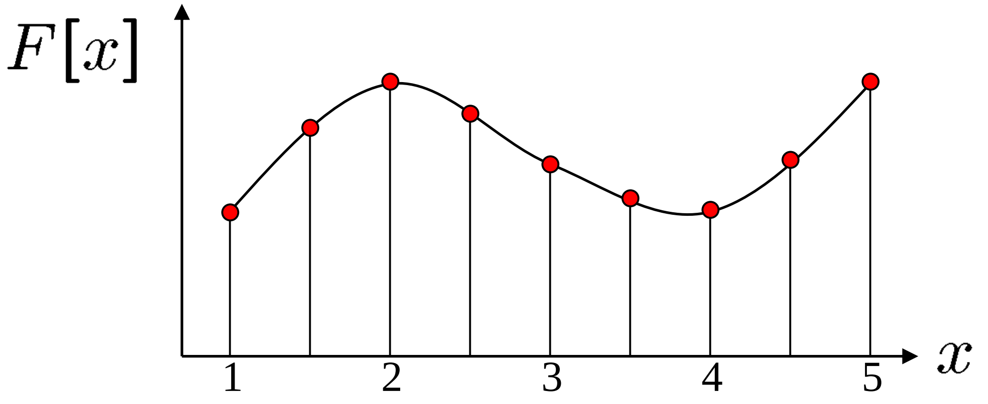
$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- 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 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

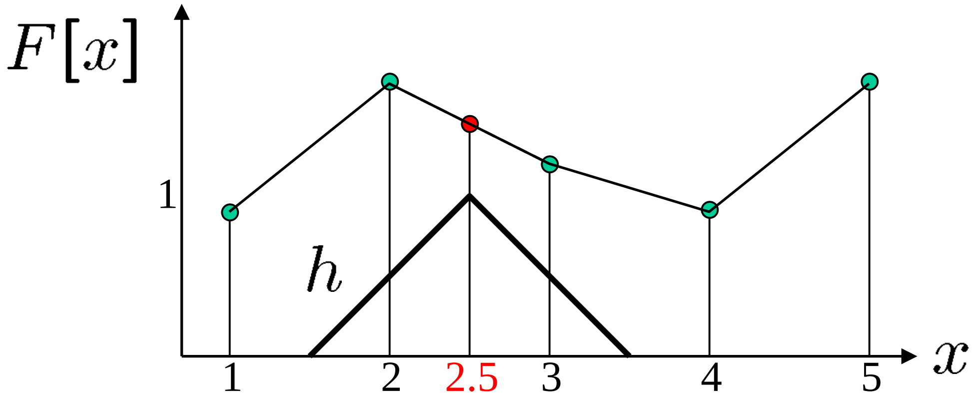
$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- 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 f

- Answer: guess an approximation $\tilde{f}$
- Can be done in a principled way: filtering



d = 1 in this example

Image reconstruction

- Convert F to a continuous function

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

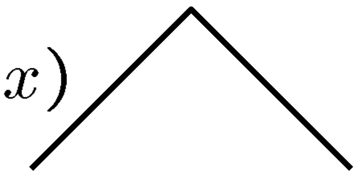
- Reconstruct by cross-correlation:

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

Resampling filters

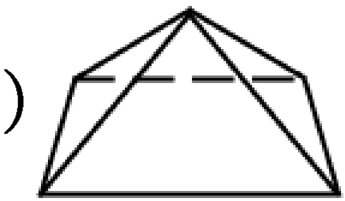
What does the 2D version of this hat function look like?

$h(x)$



performs
linear interpolation

$h(x, y)$

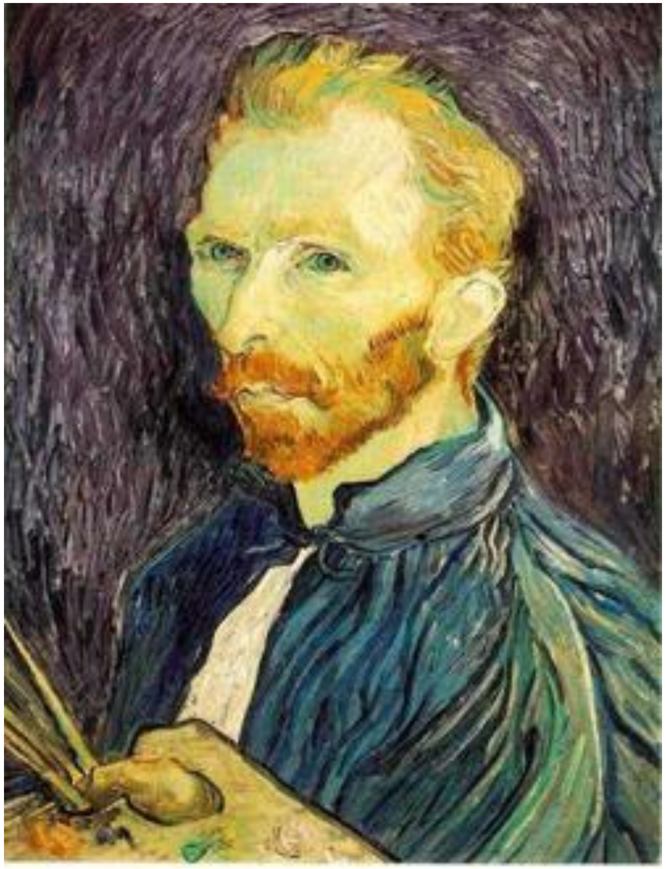


performs
bilinear interpolation

Better filters give better resampled images

- Bicubic is common choice
  - fit 3rd degree polynomial surface to pixels in neighborhood
  - can also be implemented by a convolution or cross-correlation

Subsampling with bilinear pre-filtering



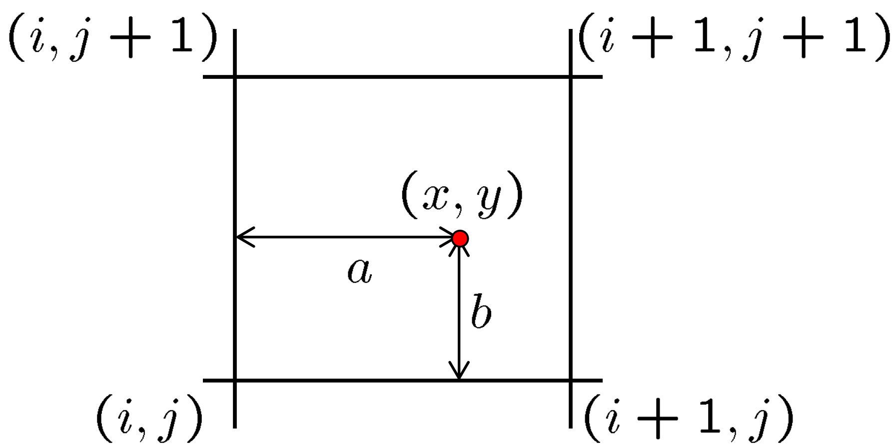
Bilinear 1/2

BL 1/4

BL 1/8

Bilinear interpolation

A common method for resampling images



$$
\begin{aligned}
F(x,y) = \; & (1-a)(1-b) \quad F(i,j) \\
& +a(1-b) \quad F(i+1,j) \\
& +ab \quad F(i+1,j+1) \\
& +(1-a)b \quad F(i,j+1)
\end{aligned}
$$