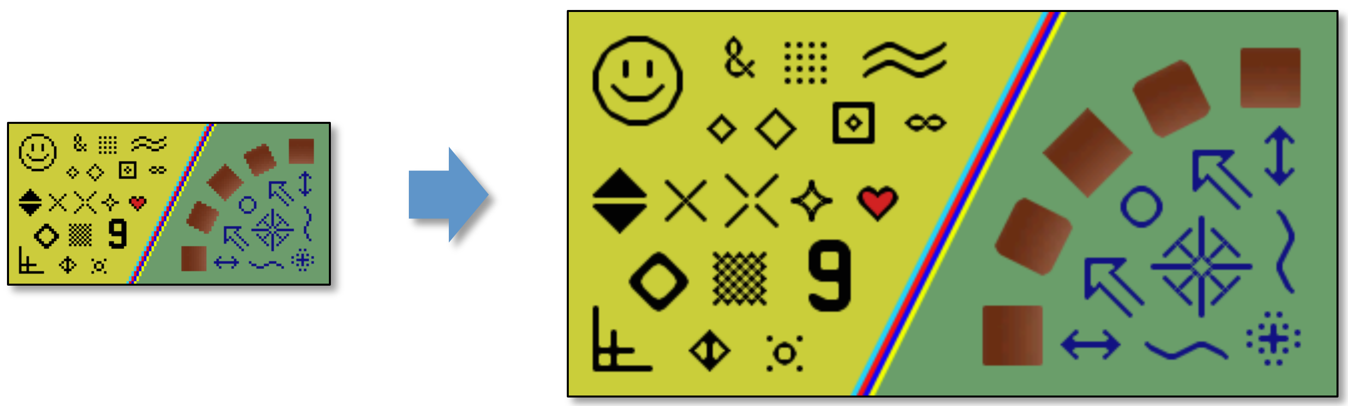
CS4670: Computer Vision

Kavita Bala

Lecture 4: Image Resampling and Reconstruction



Announcements

- PA 1 is out
  - **Due next Thursday**
- Demo info etc. online
- Prelim March 5th (Thu)

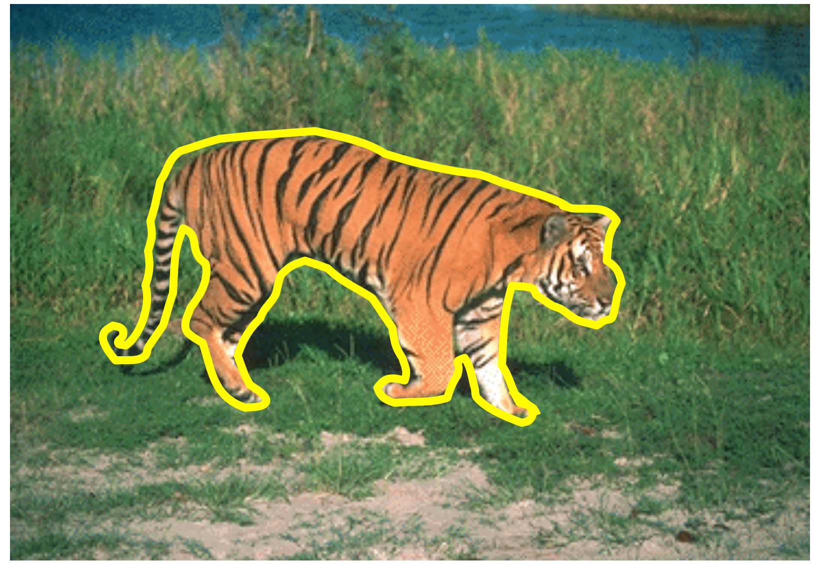
PA 1

Image Scissors

Aging Helen Mirren

- Today's Readings
  - Intelligent Scissors, Mortensen et. al, SIGGRAPH 1995

Extracting objects



- How can this be done?
  - hard to do manually
    - By selecting each pixel on the boundary
  - hard to do automatically (“image segmentation”)
  - pretty easy to do semi-automatically

Image Scissors (with demo!)

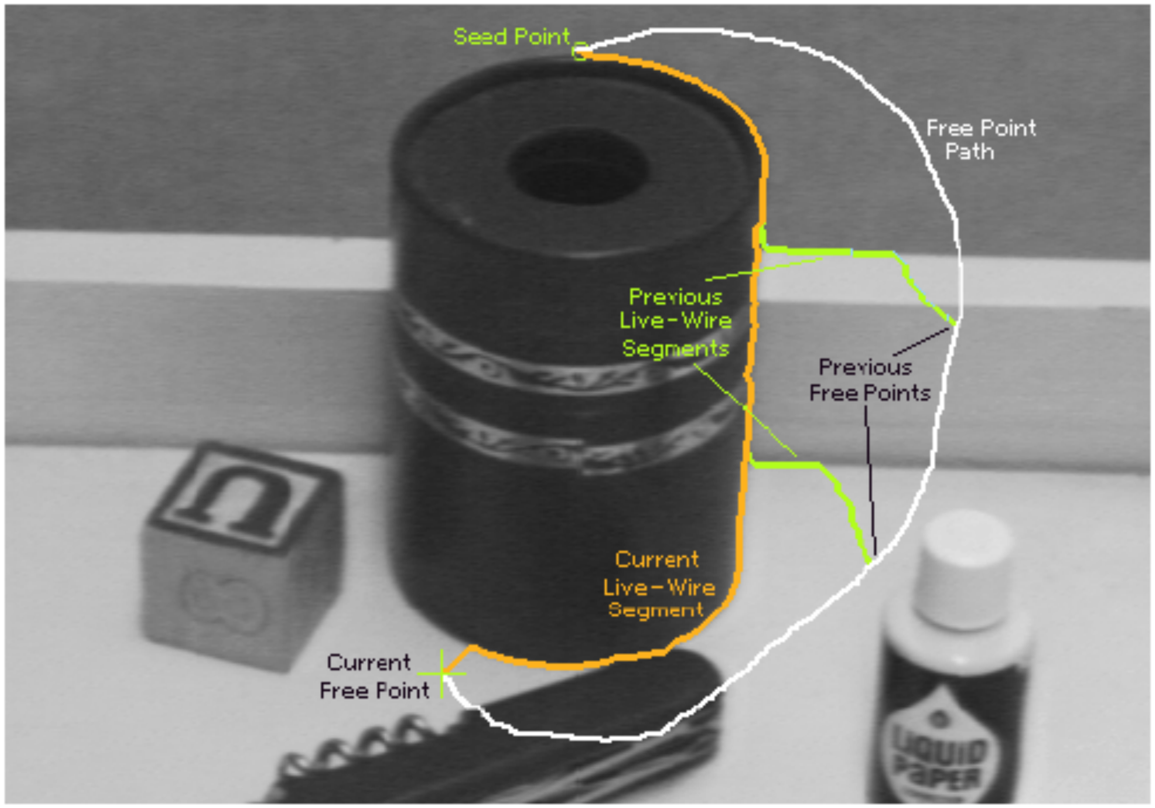
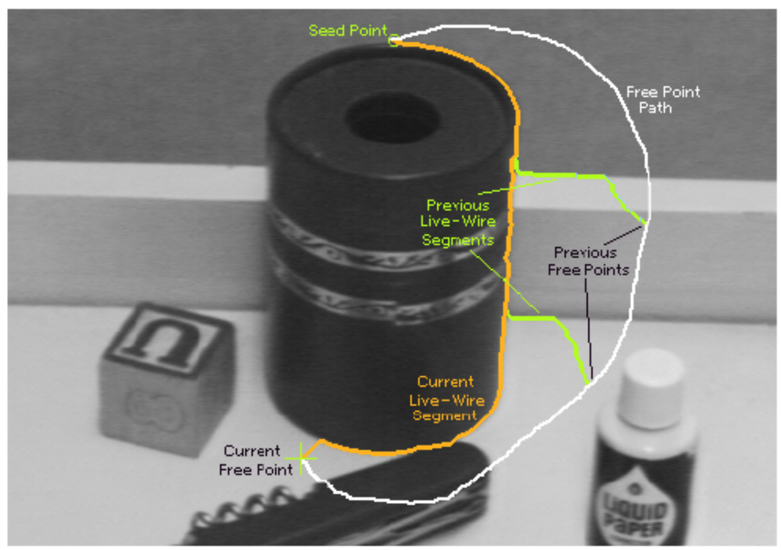


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

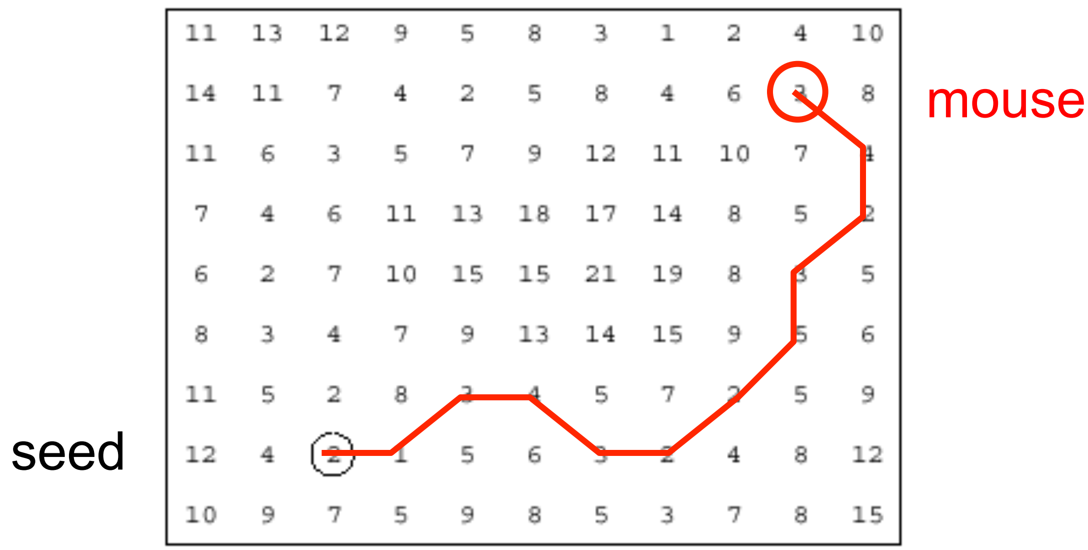
Intelligent Scissors

- Approach answers basic question
  - Q: how to find a path from seed to mouse that follows an object boundary as closely as possible?
  - A: define a path that stays as close as possible to *edges*



Intelligent Scissors

- Basic Idea
  – Define edge score for each pixel
    - edge pixels have low cost
  – Find lowest cost path from seed to mouse

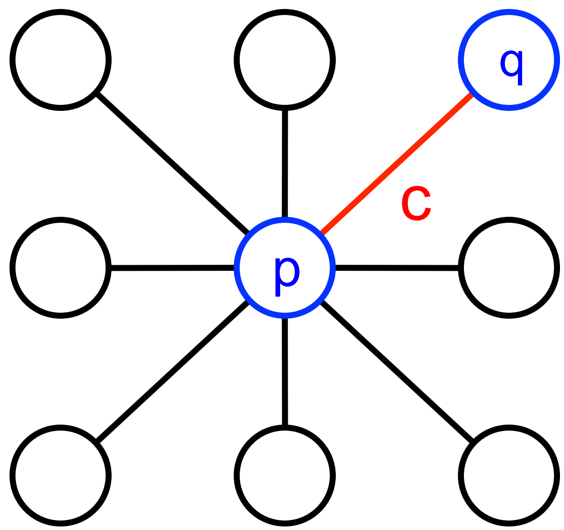


Questions

- How to define costs?
- How to find the path?

Let’s look at this more closely

- Treat the image as a graph



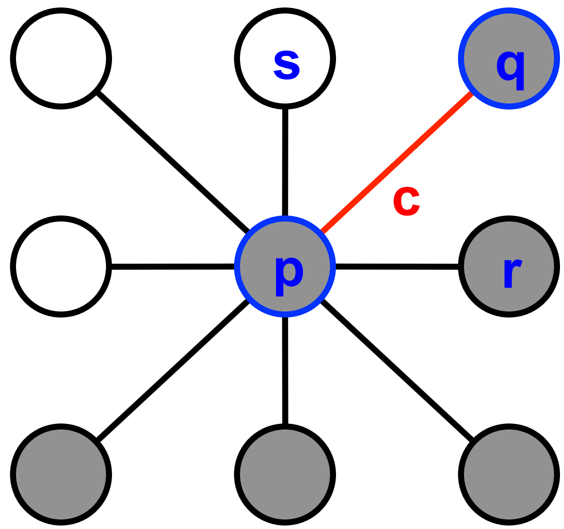
Graph

- node for every pixel p
- link between every adjacent pair of pixels, p,q
- cost c for each link

Note: each link has a cost

- this is a little different than the figure before where each pixel had a cost

Defining the costs

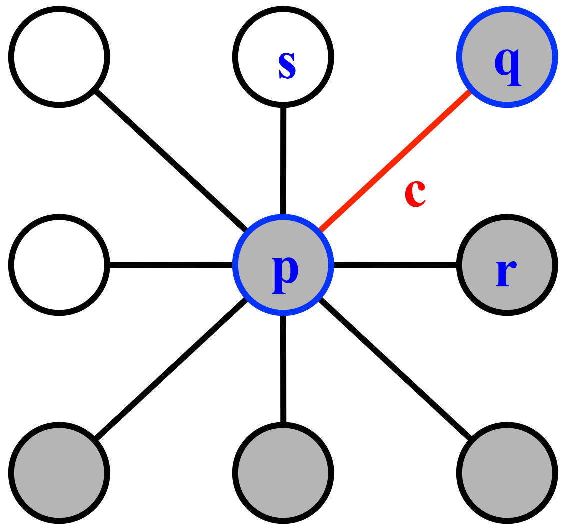


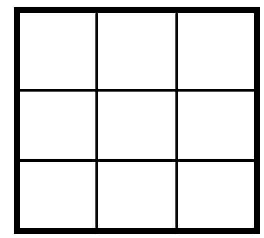
Want to hug image edges: how to define cost of a link?

- good (low-cost) links follow intensity edges
– want intensity to change rapidly ⊥ to the link

- $\mathbf{c} \propto -\frac{1}{\sqrt{2}}$ |intensity of **r** – intensity of **s**|

Defining the costs





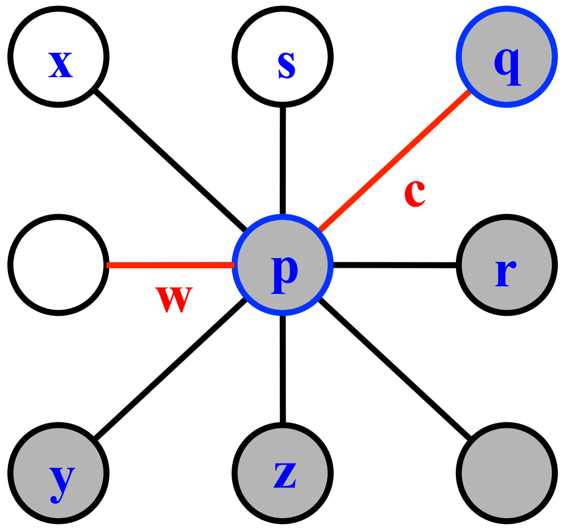
H_c

c can (almost) be computed using a cross-correlation filter

- assume it is centered at p

Defining the costs

$$H_w = \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 1 & \\ \hline & & \\ \hline -1 & -1 & \\ \hline \end{array}$$



$$H_c = \frac{1}{\sqrt{2}}\begin{array}{|c|c|c|}\hline & 1 & \\ \hline & & -1 \\ \hline & & \\ \hline \end{array}$$

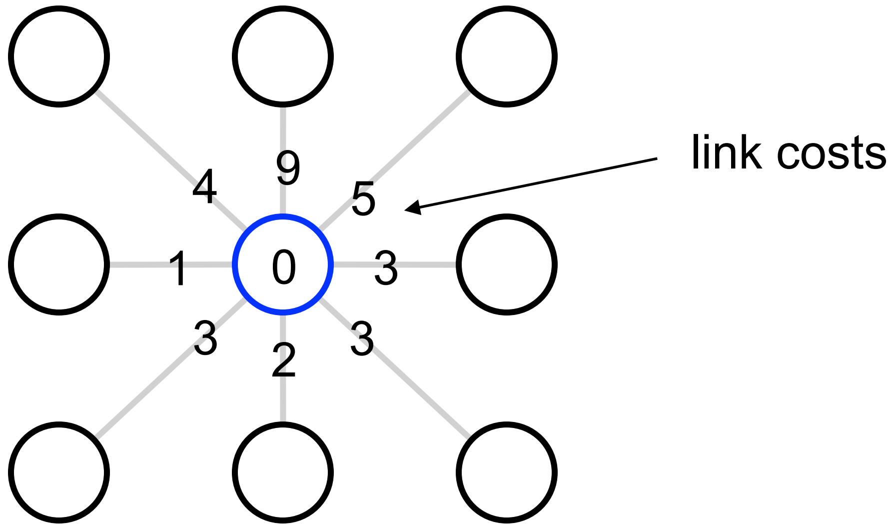
c can (almost) be computed using a cross-correlation filter

- assume it is centered at p

A couple more modifications

- Scale the filter response by length of link c. Why?
- Make c positive
  - Set c = (max-|filter response|)*length
  - where max = maximum |filter response| over all pixels in the image

Dijkstra's shortest path algorithm



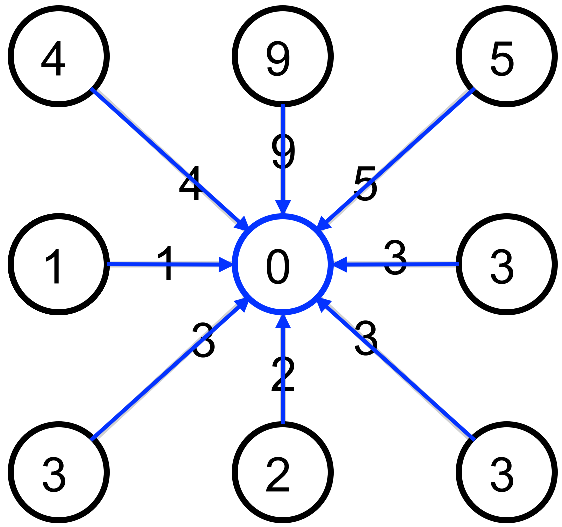
Algorithm

1. init node costs to ∞, set p = seed point, cost(p) = 0
2. expand p as follows:

   for each of p's neighbors q that are not expanded

   set cost(q) = min(cost(p) + c_{pq}, cost(q))

Dijkstra's shortest path algorithm



Algorithm

1. init node costs to ∞, set p = seed point, cost(p) = 0
2. expand p as follows:

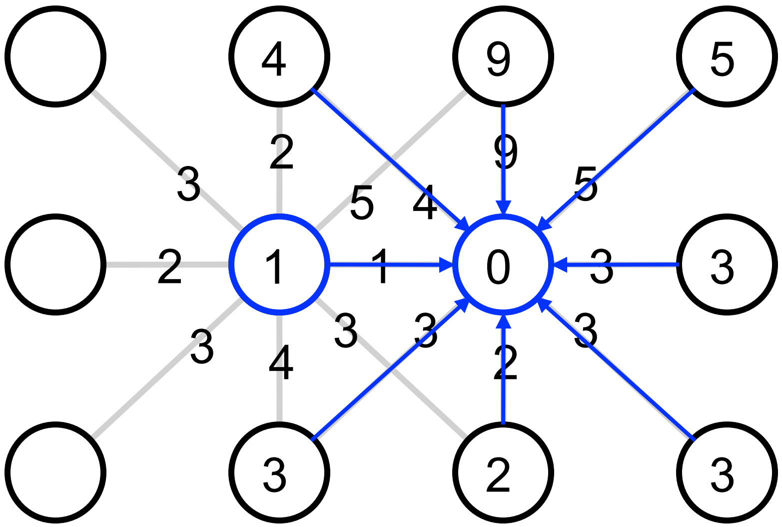
   for each of p's neighbors q that are not expanded

      set cost(q) = min(cost(p) + c_{pq}, cost(q))

      if q's cost changed, make q point back to p

      put q on the ACTIVE list   (if not already there)

Dijkstra's shortest path algorithm



Algorithm

1. init node costs to ∞, set p = seed point, cost(p) = 0
2. expand p as follows:

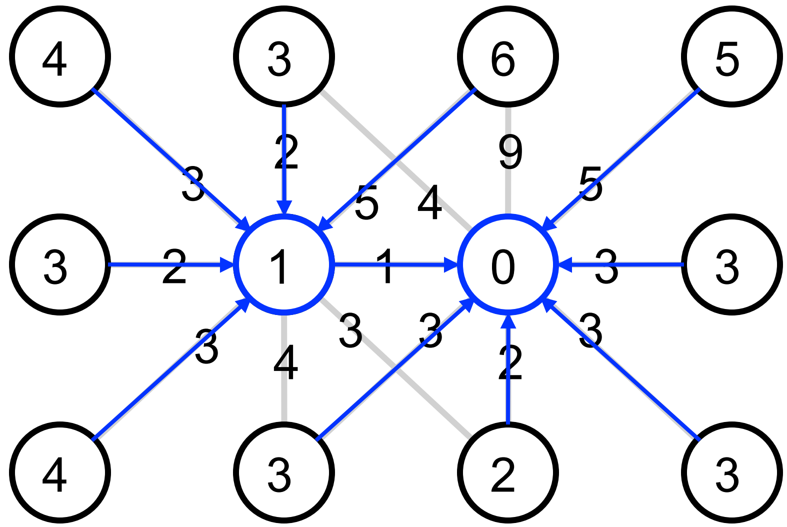
   for each of p's neighbors q that are not expanded

   set cost(q) = min(cost(p) + c_{pq}, cost(q))

   if q's cost changed, make q point back to p

   put q on the ACTIVE list (if not already there)
3. set r = node with minimum cost on the ACTIVE list
4. repeat Step 2 for p = r

Dijkstra’s shortest path algorithm



Algorithm

1. init node costs to ∞, set p = seed point, cost(p) = 0
2. expand p as follows:

   for each of p’s neighbors q that are not expanded

   set cost(q) = min(cost(p) + c_{pq}, cost(q))

   if q’s cost changed, make q point back to p

   put q on the ACTIVE list (if not already there)

3. set r = node with minimum cost on the ACTIVE list
4. repeat Step 2 for p = r

Dijkstra's shortest path algorithm

- Properties
  - It computes the minimum cost path from the seed to every node in the graph. This set of minimum paths is represented as a tree
  - Running time, with N pixels:
    - $O(N^2)$ time if you use an active list
    - O(N log N) if you use an active priority queue (heap)
    - takes fraction of a second for a typical (640x480) image
  - Once this tree is computed once, we can extract the optimal path from any point to the seed in O(N) time.
    - it runs in real time as the mouse moves
  - What happens when the user specifies a new seed?

Example Results

Kuan-chuan Peng

Le Zhang

Questions?

Image

This image is too big to fit on the screen. How can we generate a half-sized version?

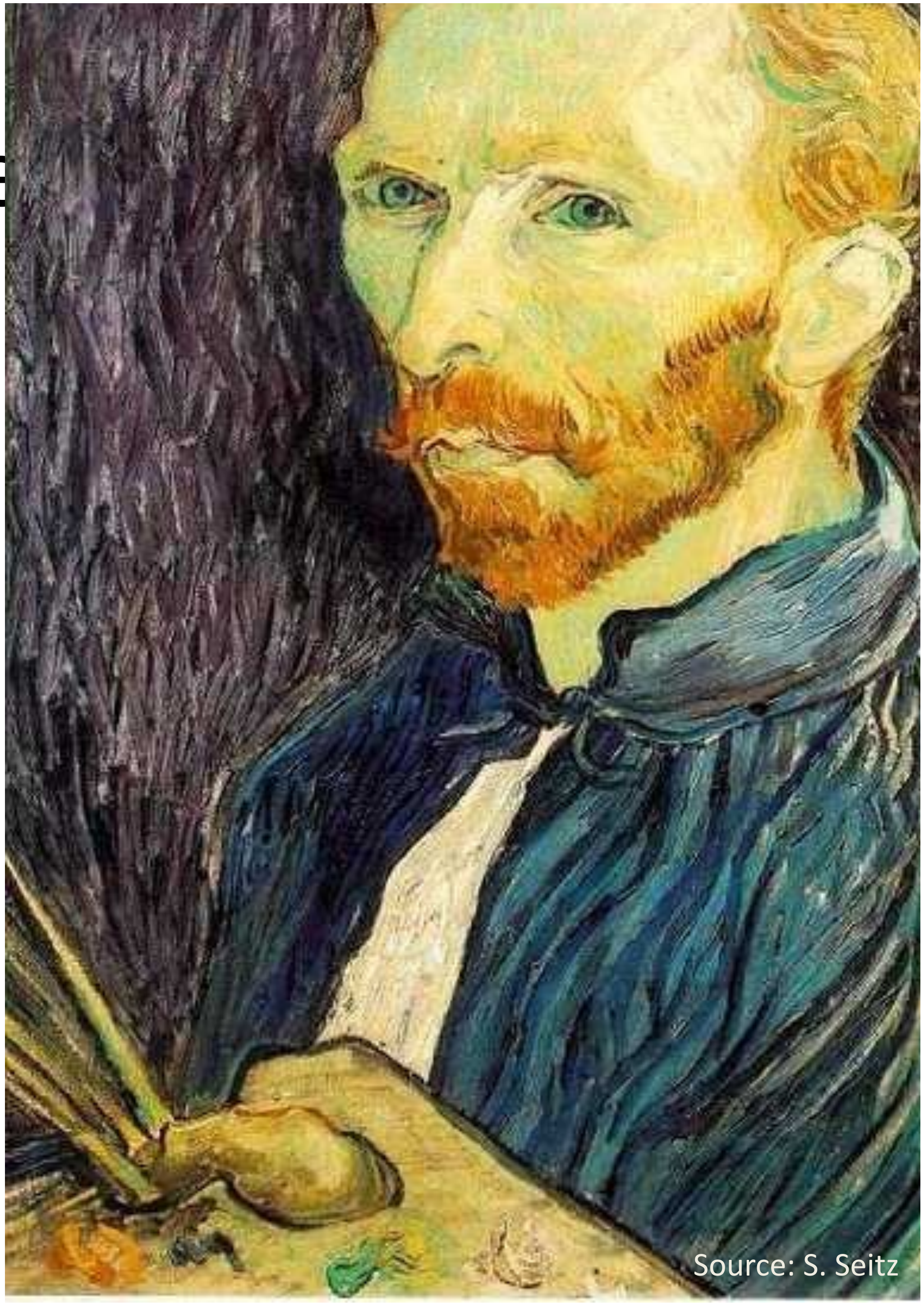
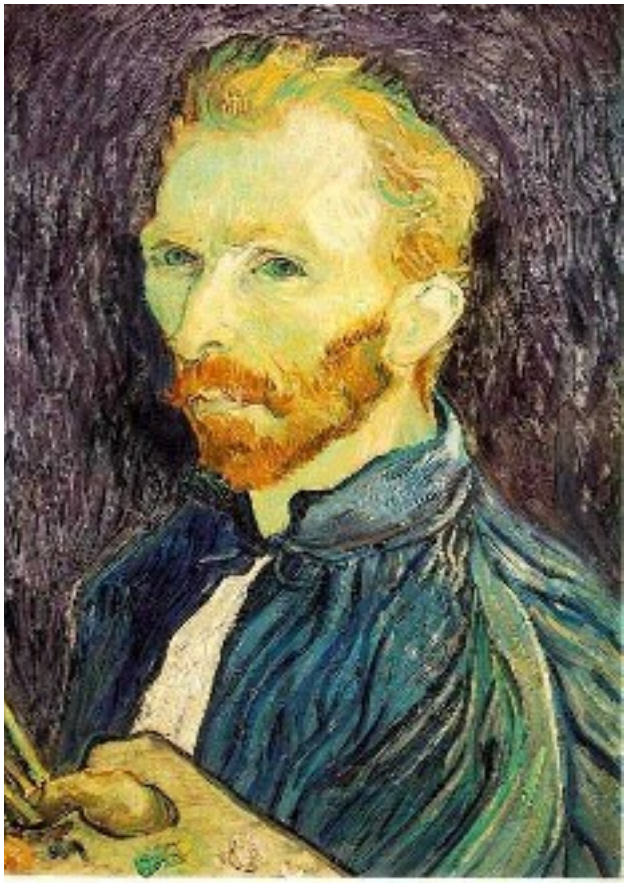


Image sub-sampling



1/2

1/4 (2x zoom)

1/16 (4x zoom)

Why does this look so crufty?

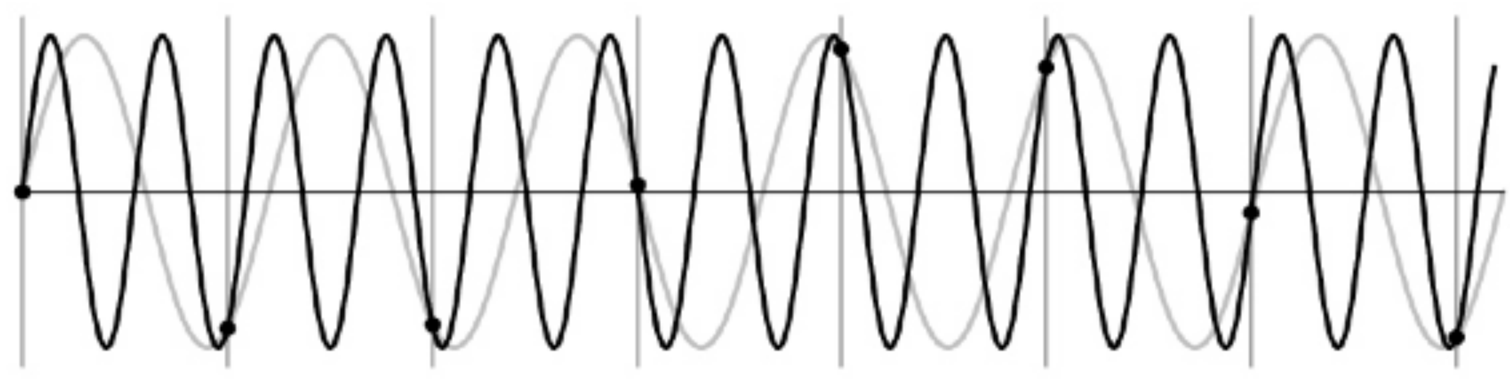
Source: S. Seitz

Image sub-sampling

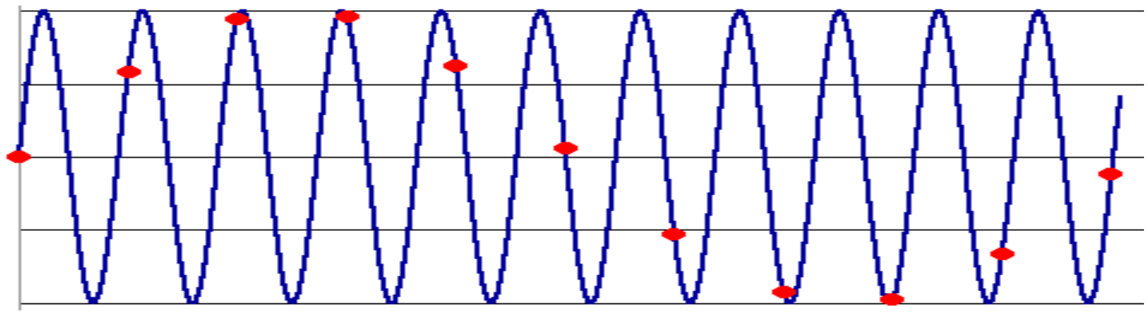
Source: F. Durand

What is aliasing?

- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals "traveling in disguise" as other frequencies



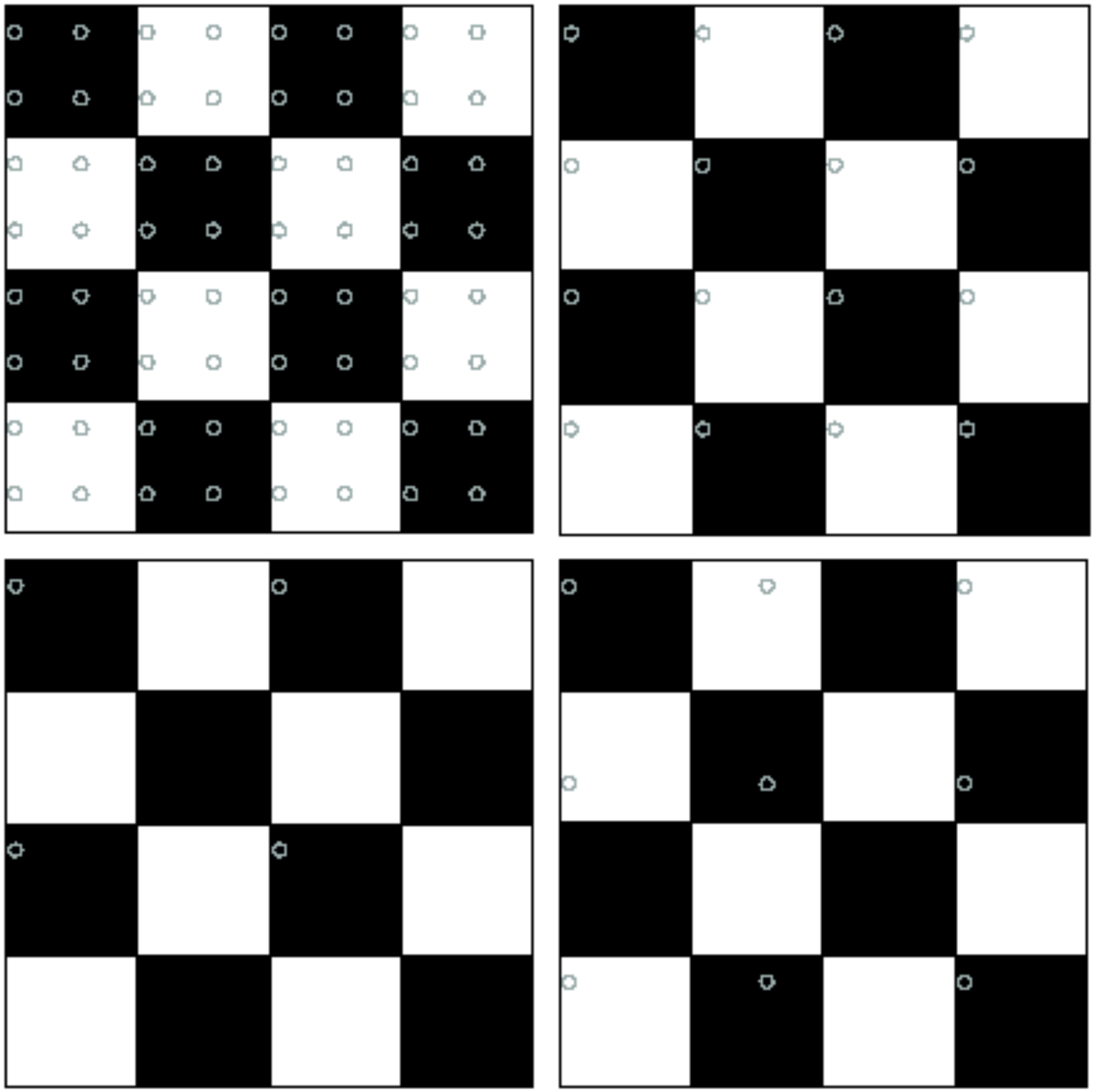
Aliasing



- 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**

Source: L. Zhang

Nyquist limit – 2D example



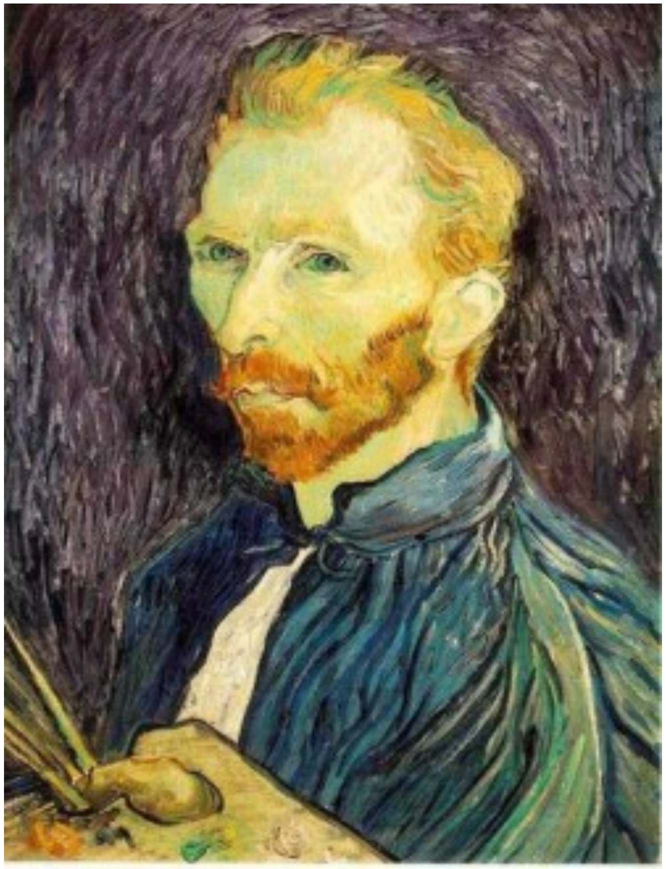
Good sampling

Bad sampling

Aliasing

- When downsampling by a factor of two
  - Original image has frequencies that are too high
- How can we fix this?

Gaussian pre-filtering



Gaussian 1/2

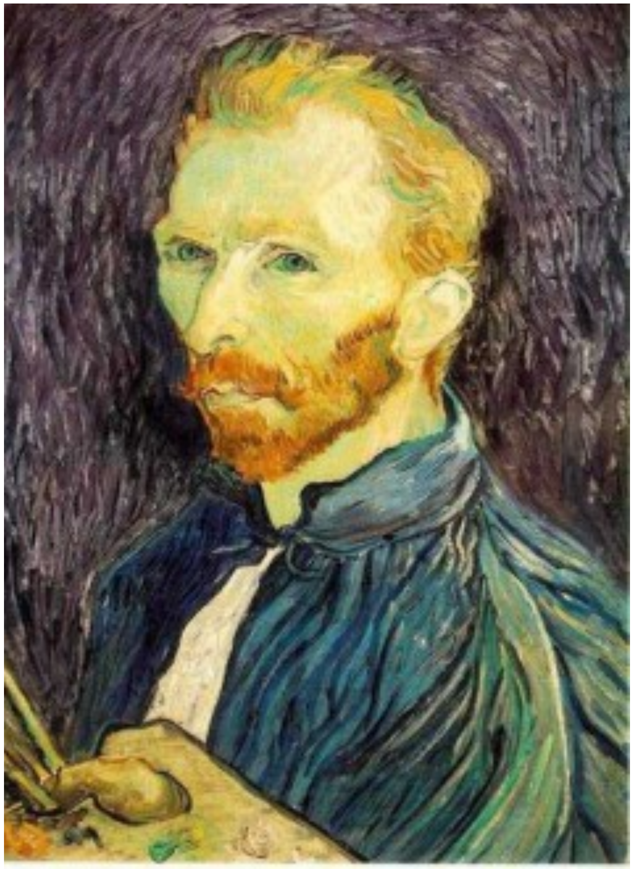
G 1/4

G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

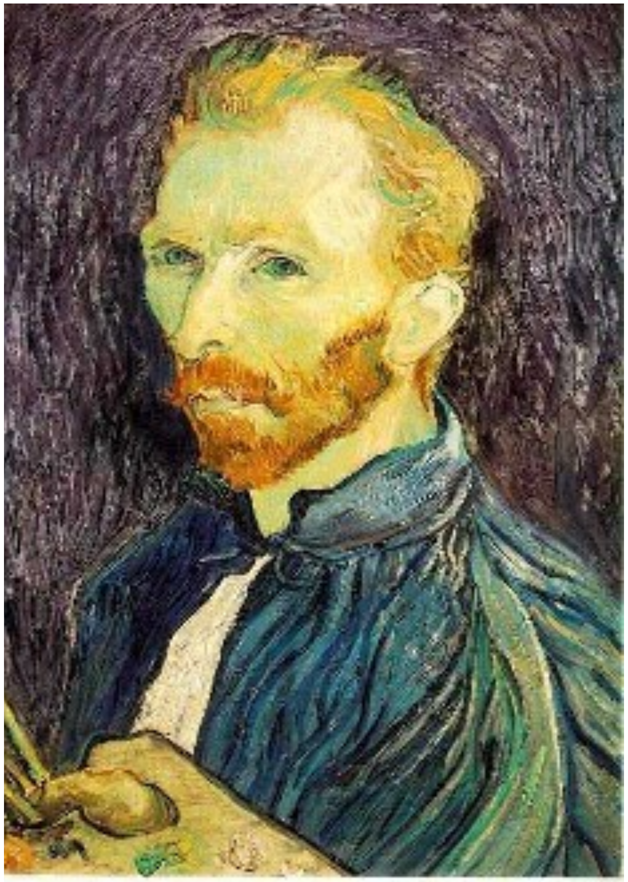
G 1/4

G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Compare with...



1/2

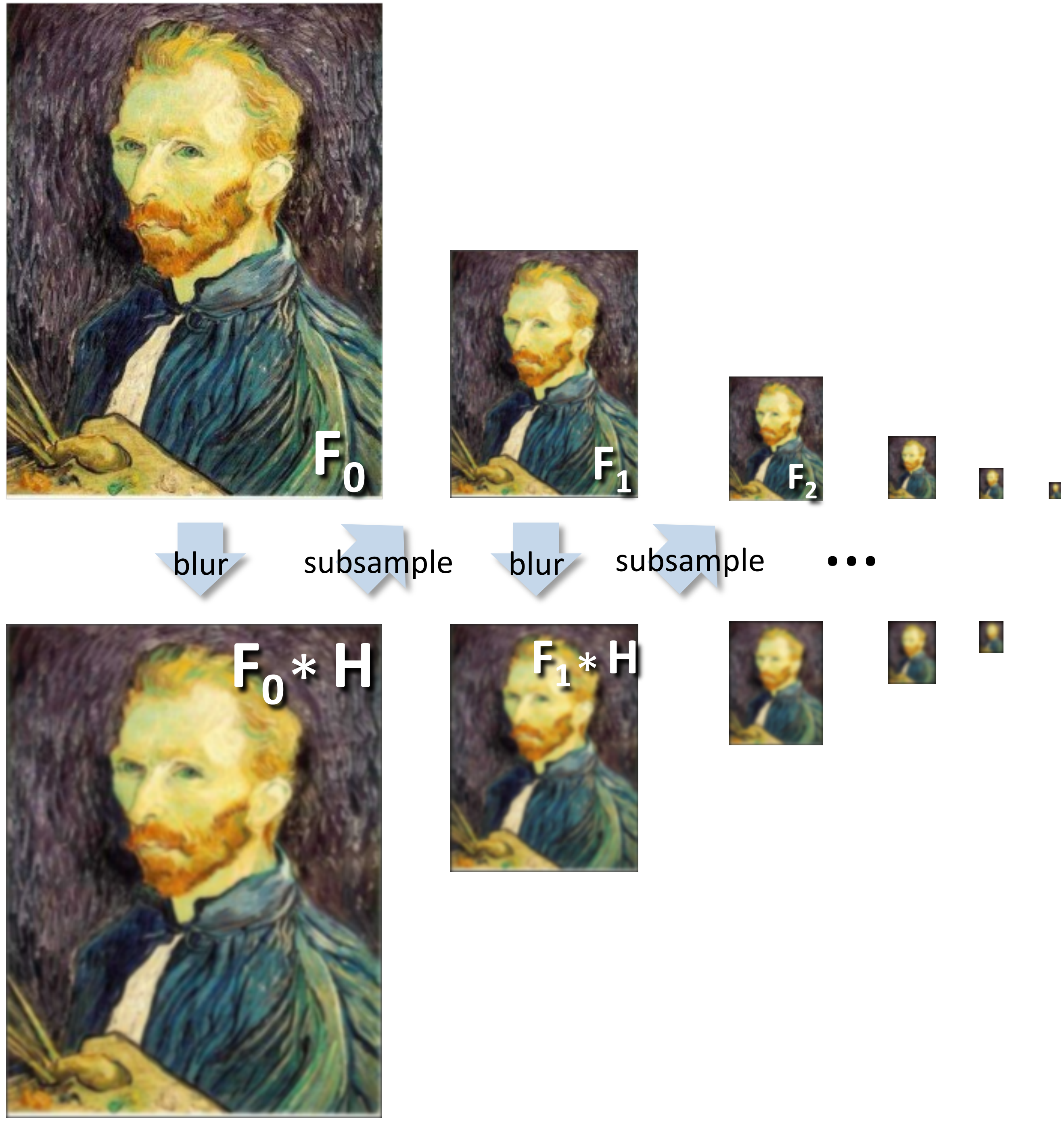
1/4 (2x zoom)

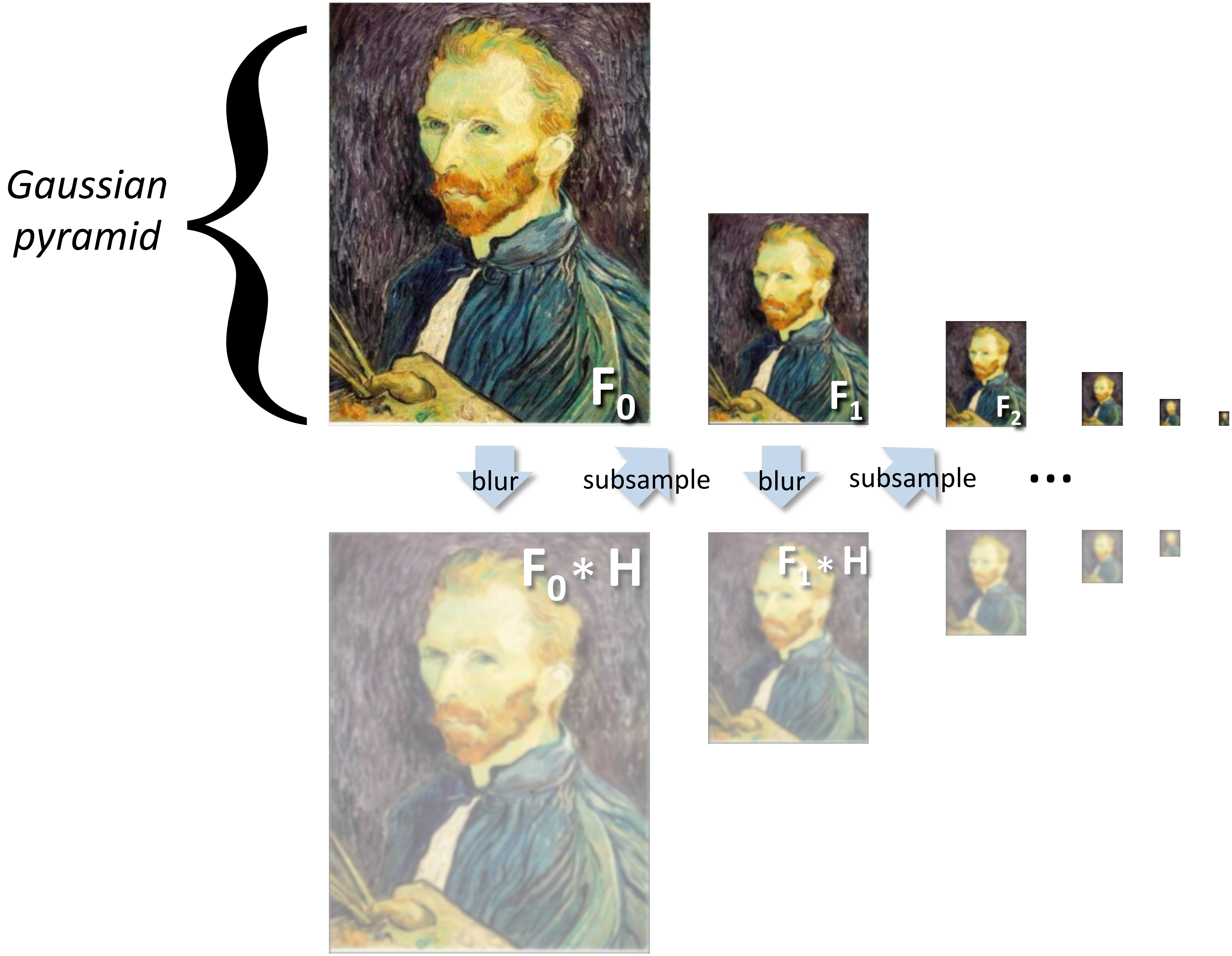
1/16 (4x zoom)

Source: S. Seitz

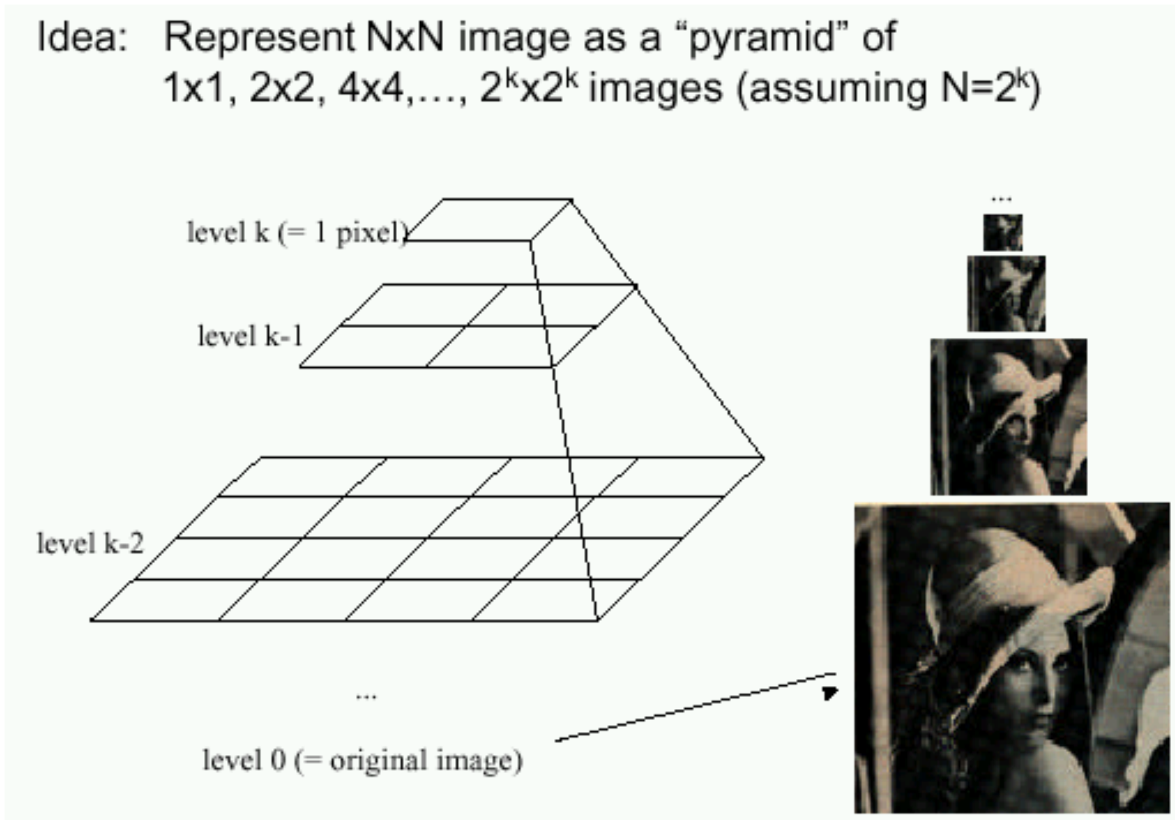
Gaussian pre-filtering

- Solution: filter the image, *then* subsample




Gaussian pyramid
F0
F1
F2
blur
subsample
blur
subsample
...
F0 * H
F1 * H

Gaussian pyramids [Burt and Adelson, 1983]

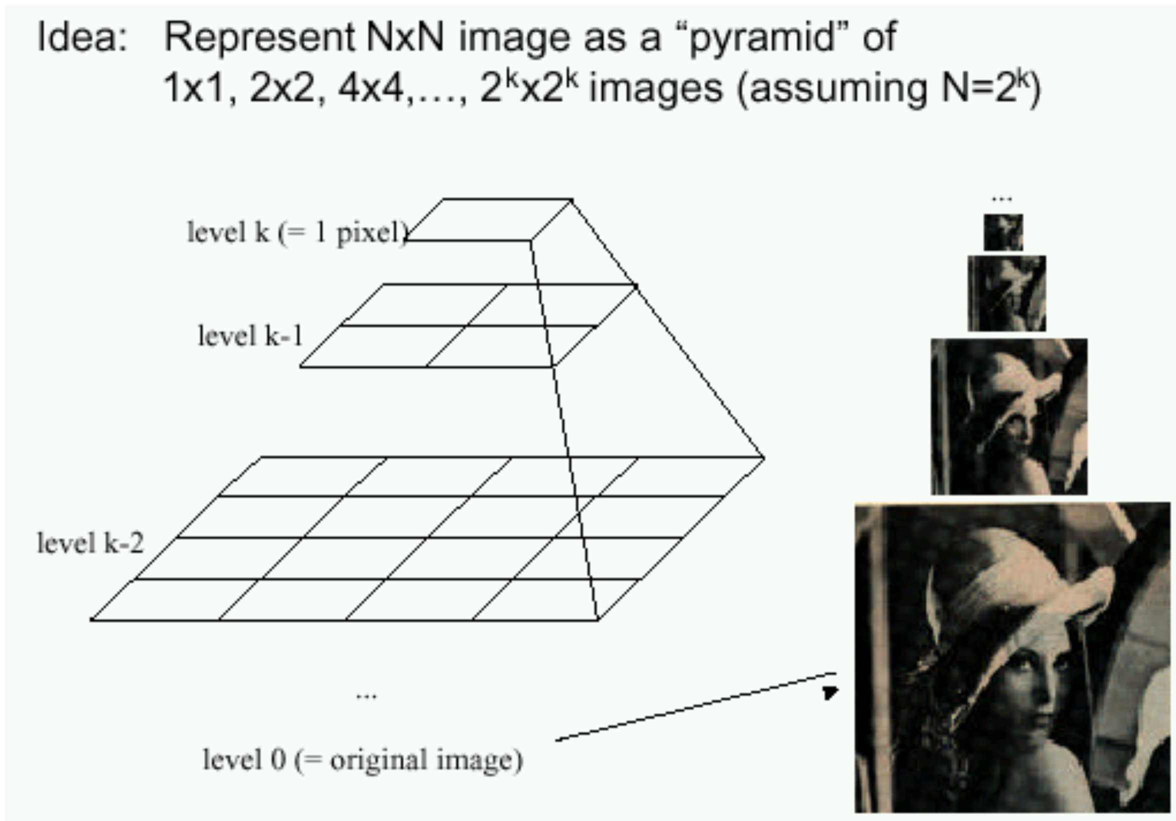


- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision

Source: S. Seitz

Gaussian pyramids [Burt and Adelson, 1983]

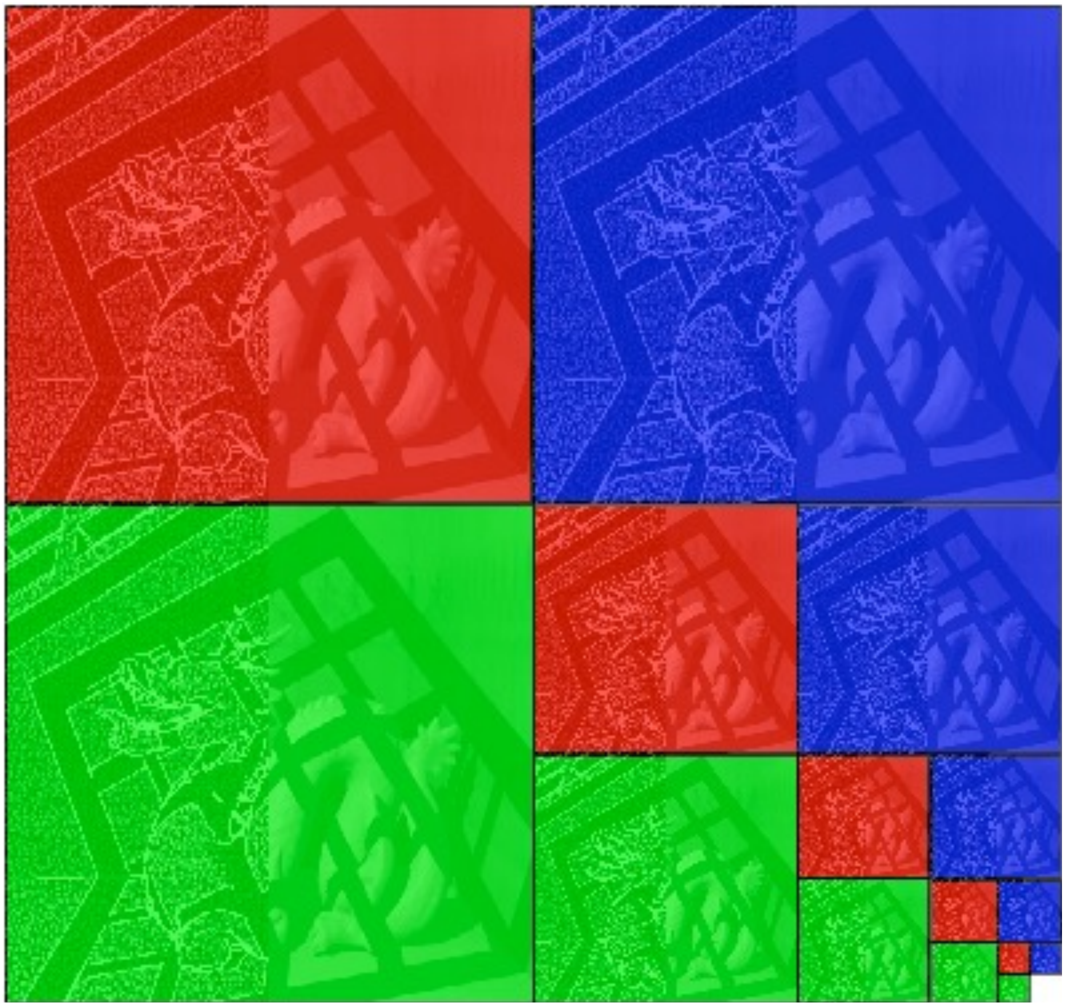


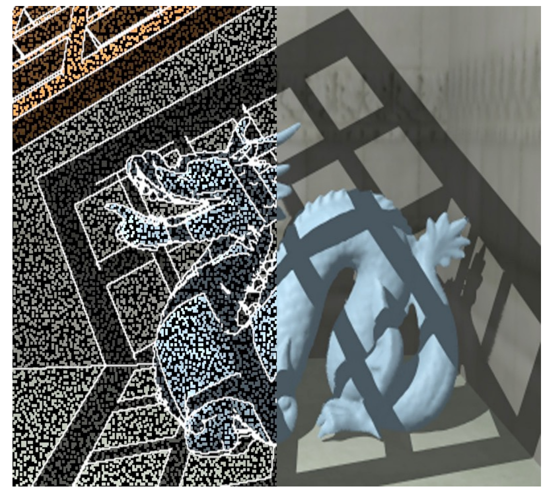
- How much space does a Gaussian pyramid take compared to the original image?

Source: S. Seitz

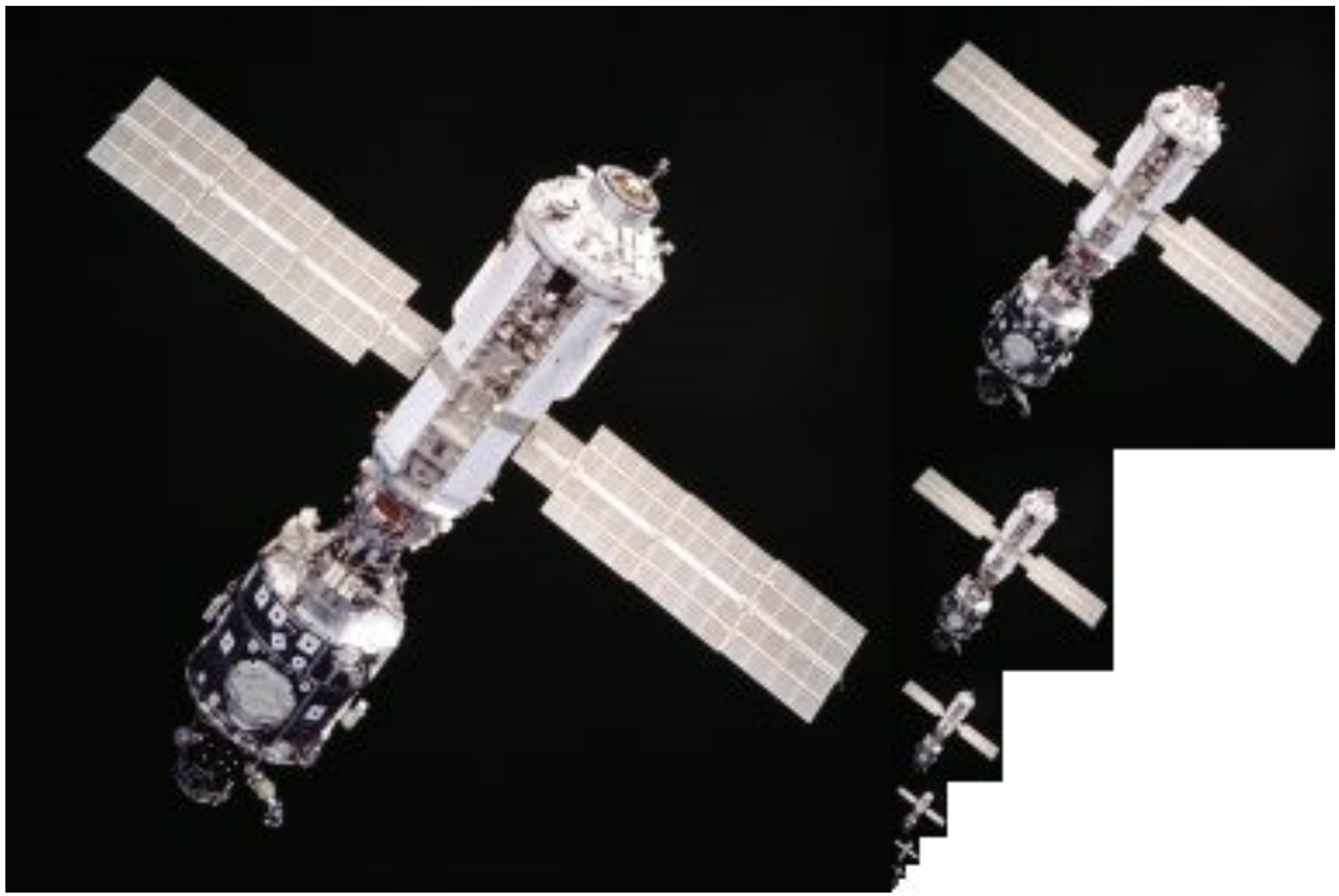
Memory Usage

- What is the size of the pyramid?





Gaussian Pyramid

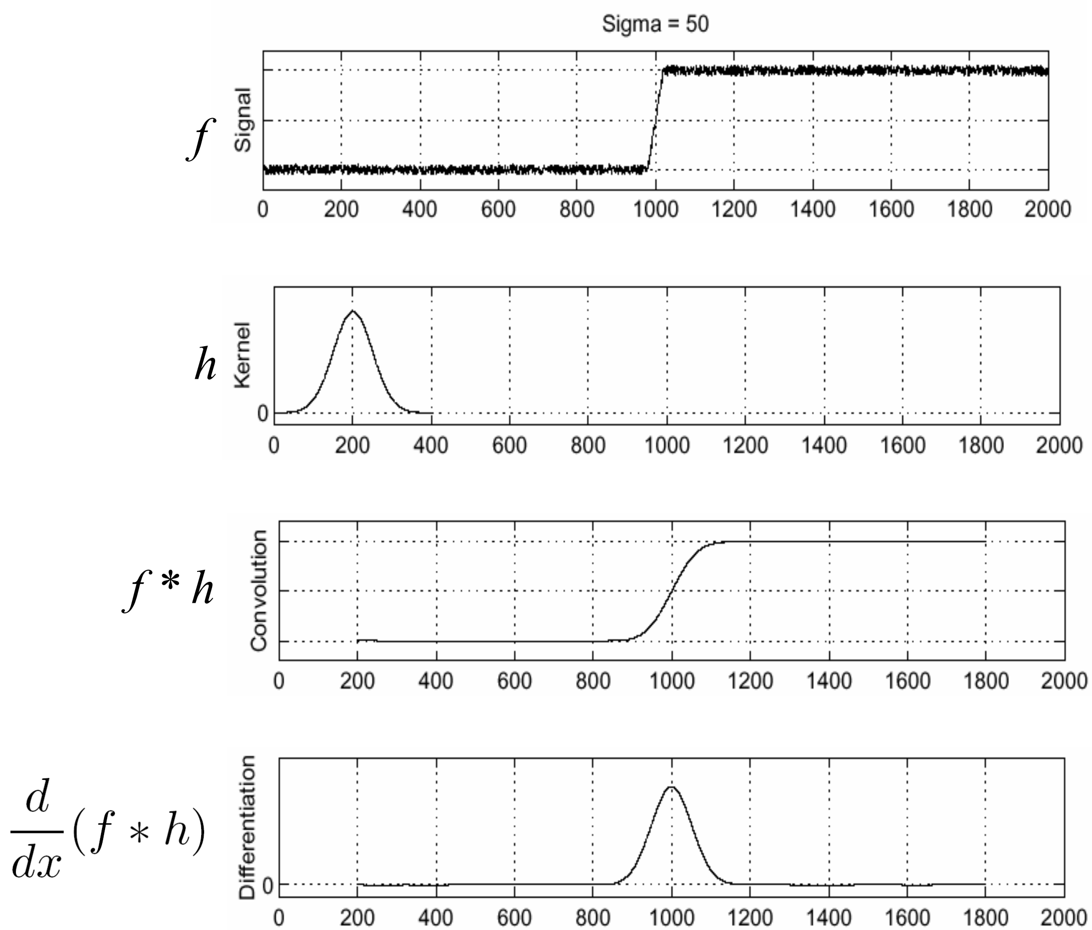


Aside: Laplacian

- Laplacian: divergence of gradient

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

- 0 at edge



The Laplacian Pyramid

$$L_i = G_i - \text{expand}(G_{i+1})$$

$$G_i = L_i + \text{expand}(G_{i+1})$$