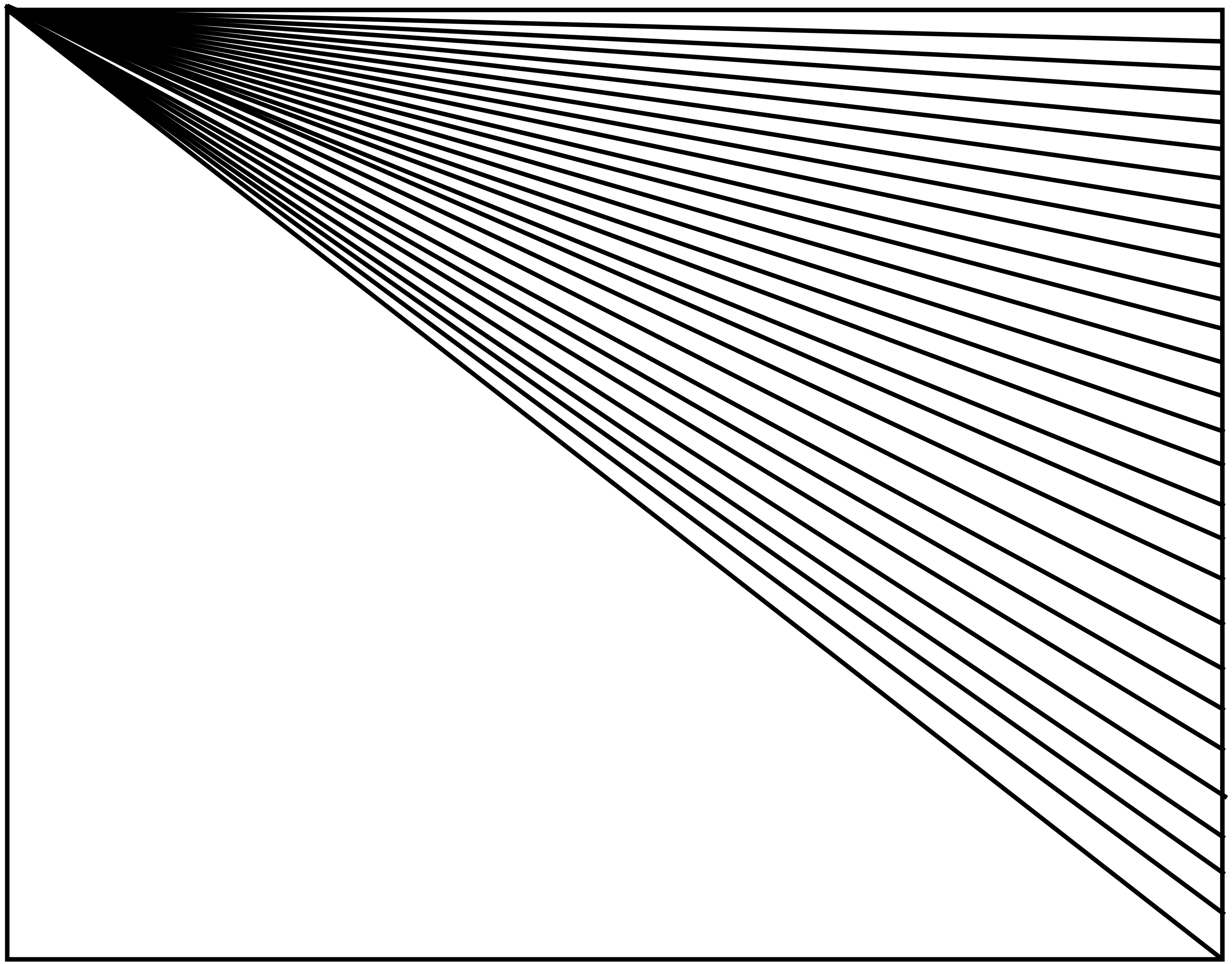
Resizing and resampling

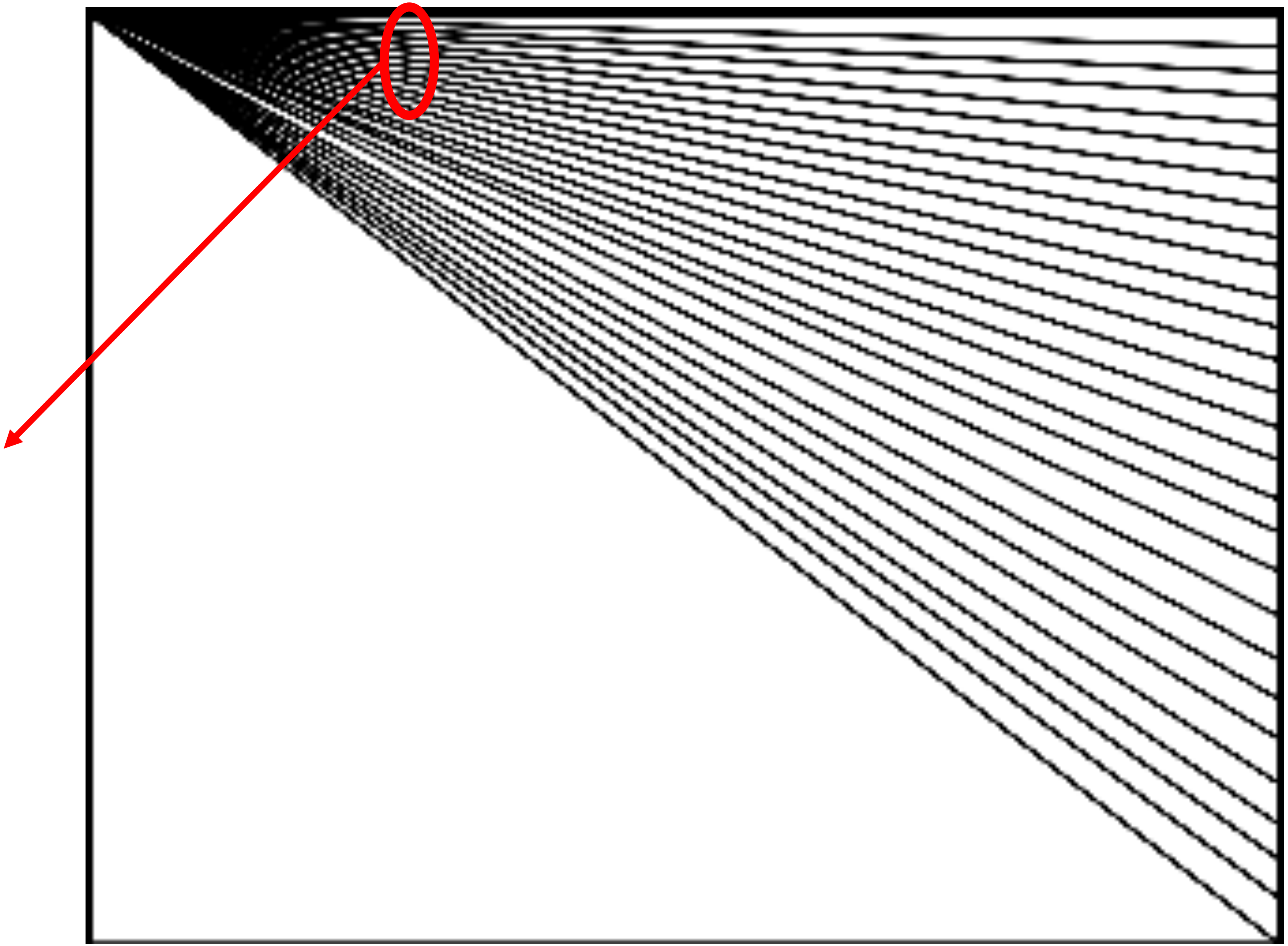
Aliasing

- Images are made up of high frequency and low frequency components
- High frequency components: pixel-to-pixel details
- Low frequency components: high-level structure
- What subsampling should do: remove pixel-to-pixel details, keep high-level structure
- What naïve subsampling does: converts pixel-to-pixel details to new coarse structures → problem

Aliasing

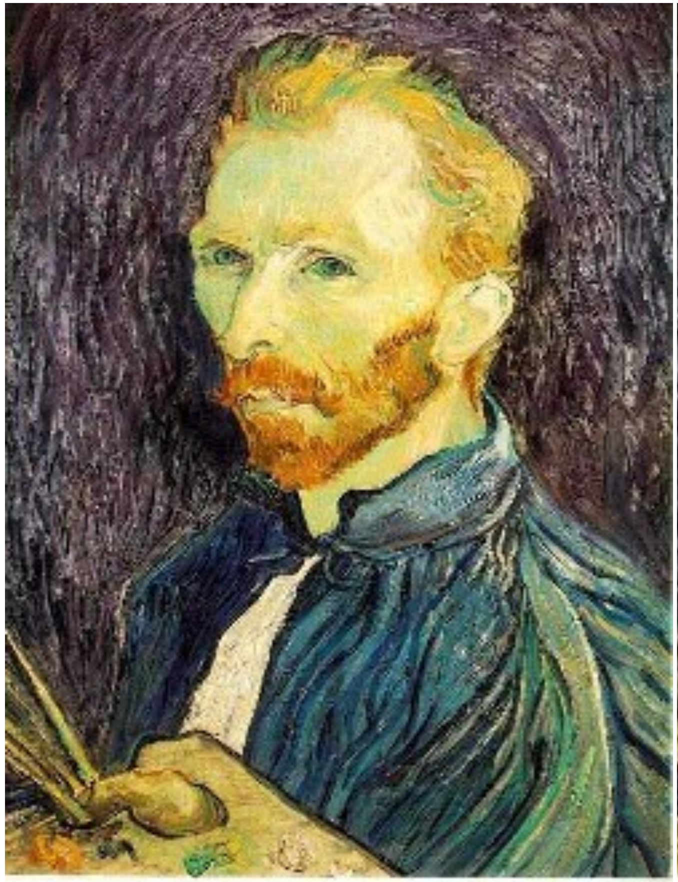


Aliasing



Aliasing artifacts

Image sub-sampling



1/2

1/4 (2x zoom)

1/16 (4x zoom)

Why does this look so crufty? Aliasing!

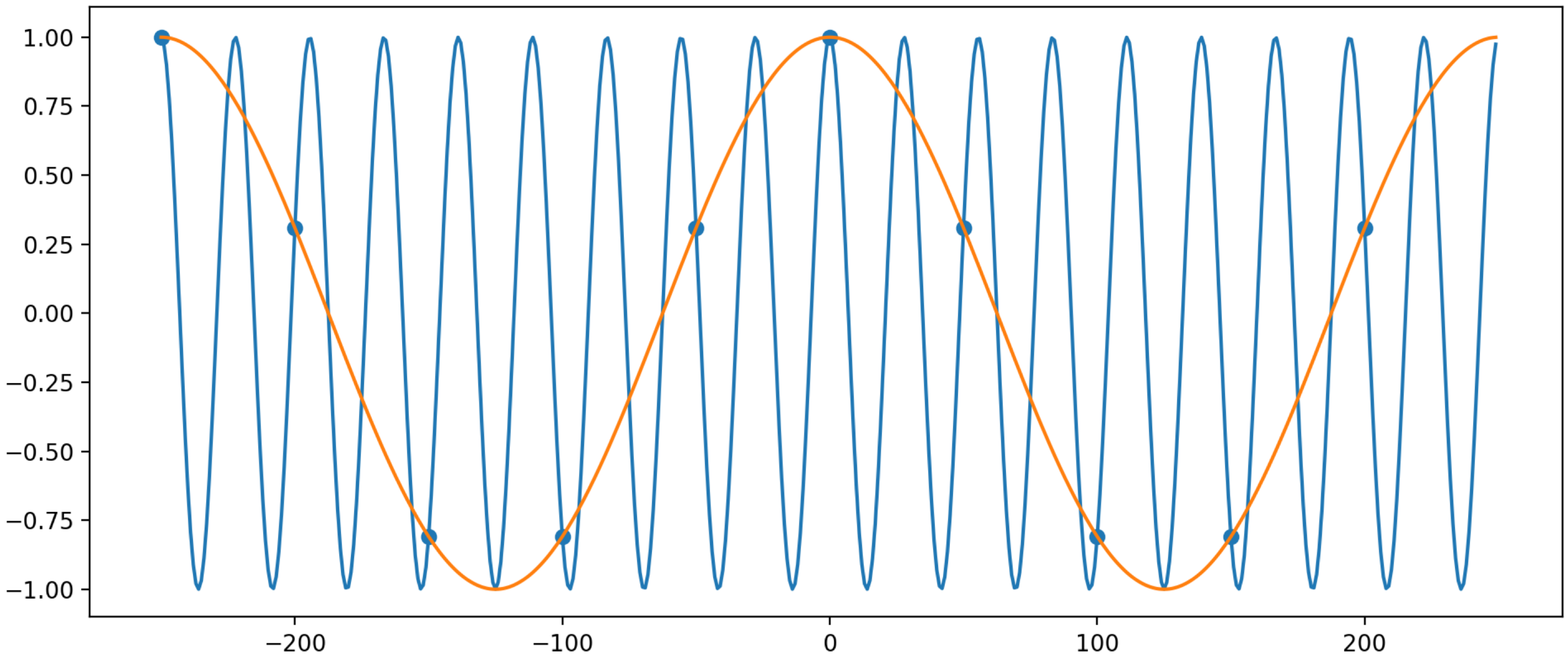
Source: S. Seitz

Why does aliasing happen?

- Consider sampling every P pixels
- $B_{\frac{N}{P}+k}(Pn) = e^{\frac{i2\pi\left(\frac{N}{P}+k\right)Pn}{N}} = e^{i2\pi n} e^{\frac{i2\pi kPn}{N}} = e^{\frac{i2\pi kPn}{N}} = B_k(Pn)$
- In fact $B_{\frac{mN}{P}+k}(Pn) = B_k(Pn)$
- The high frequency component $B_{\frac{N}{P}+k}$ gets aliased as the low frequency component B_k
- This is because $B_{\frac{N}{P}+k}$ computes an additional cycle between two samples and ends up in the same place as B_k

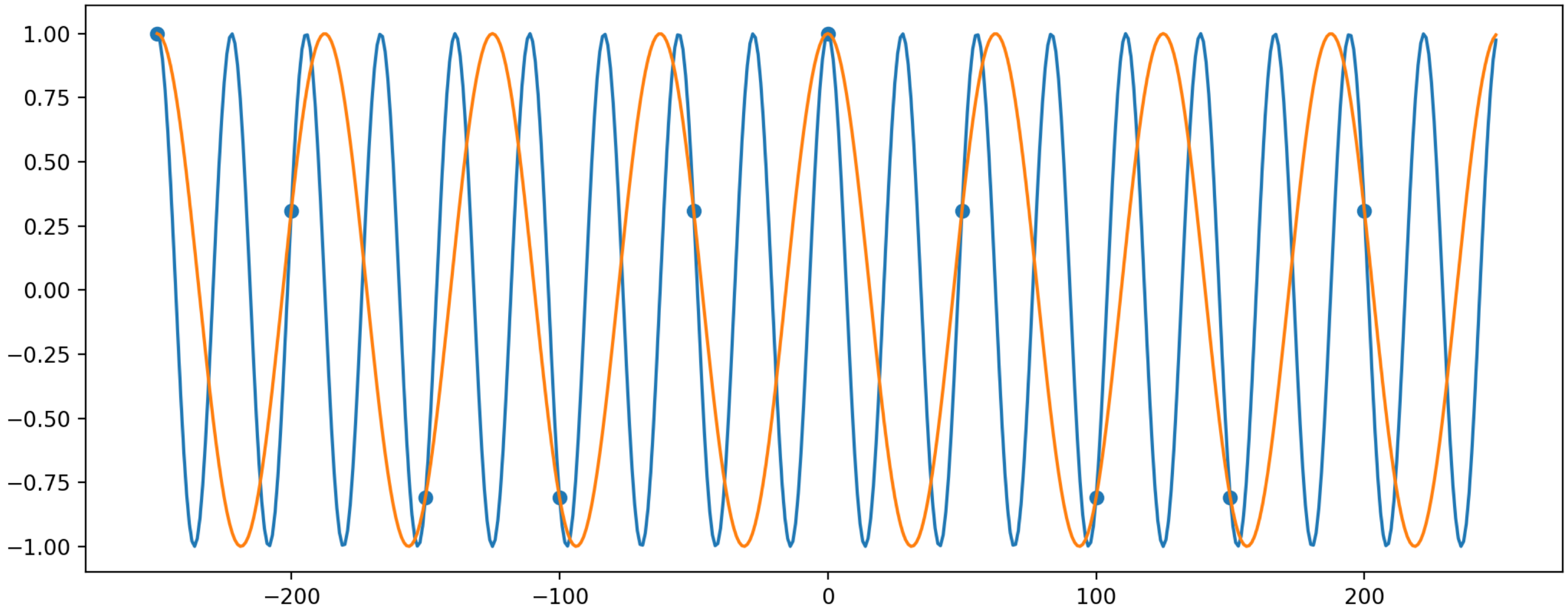
Why does aliasing happen?

Blue gets aliased as orange



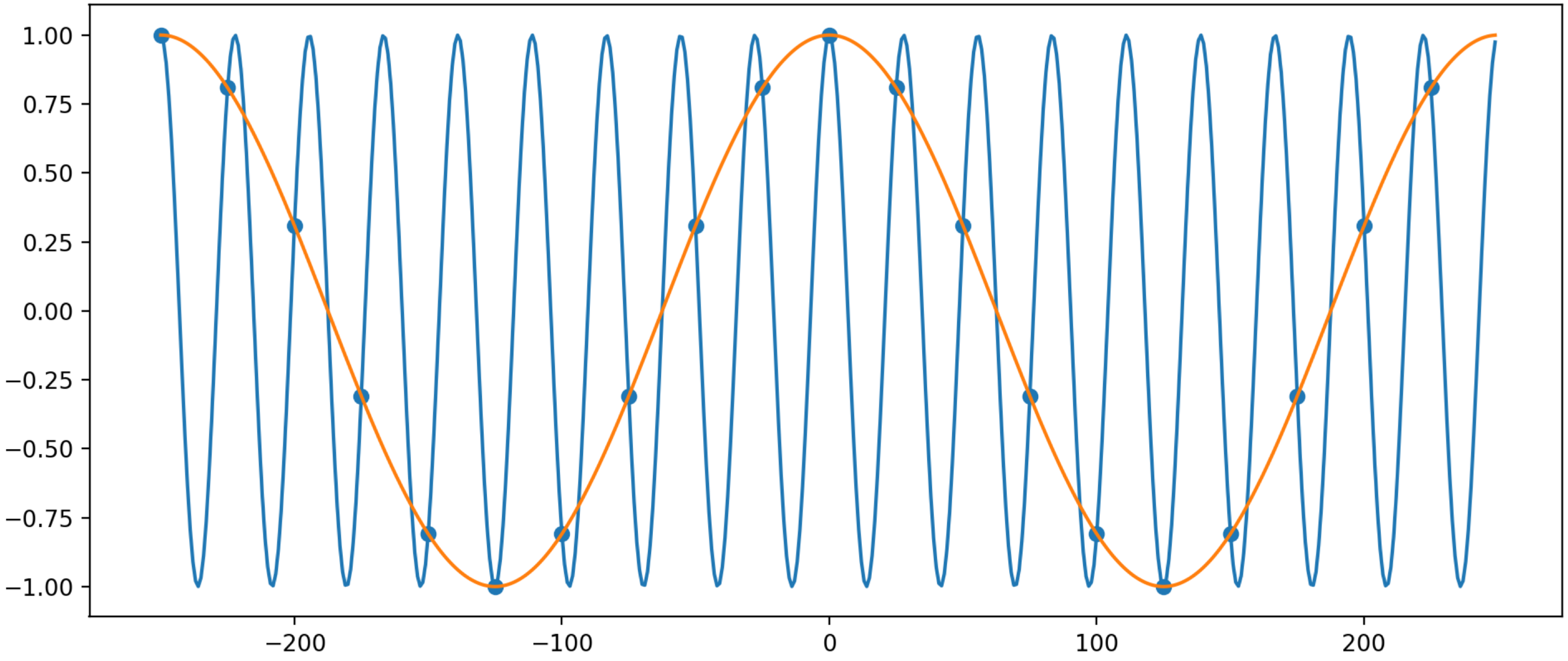
Why does aliasing happen?

Blue gets aliased as orange



Why does aliasing happen?

Blue gets aliased as orange

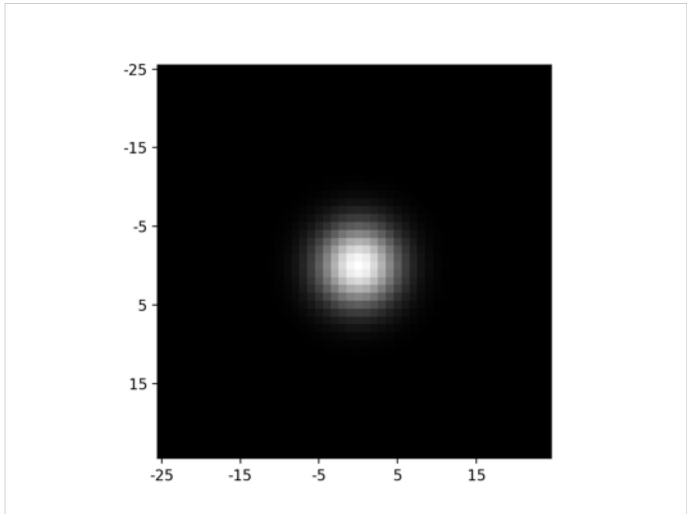


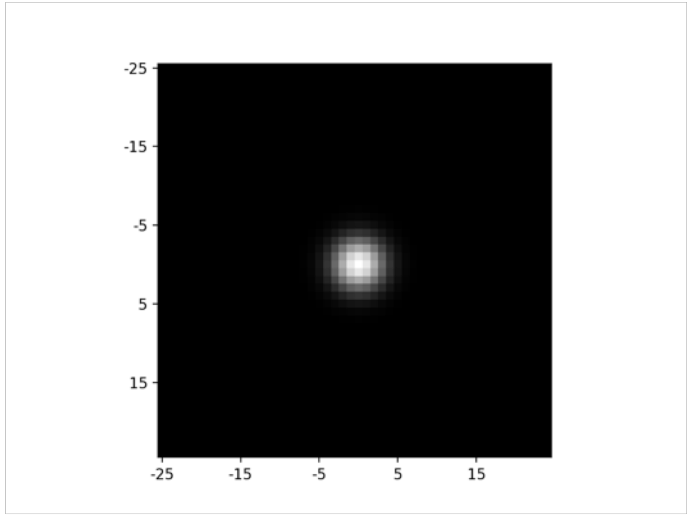
How to avoid aliasing

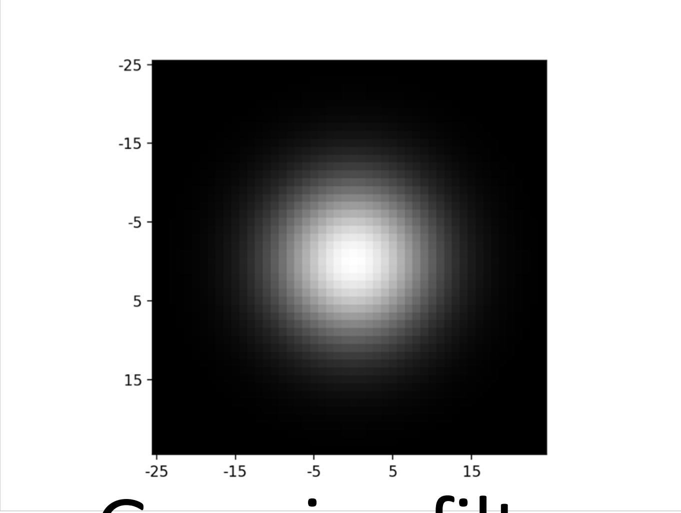
- To recover a sinusoid, need to sample at least twice per cycle
- For a general image, need to sample at least twice the rate of the highest frequency component
- **Nyquist sampling theorem:** $2\nu_{max} < \nu_{sample}$
- To subsample, *remove high frequency components*
- To remove high frequency components, *blur the image with a Gaussian*

Image

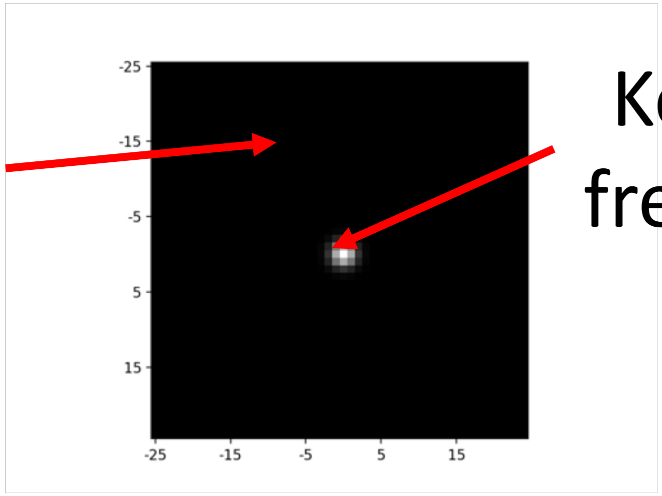
Fourier transform







Zeros out high frequencies



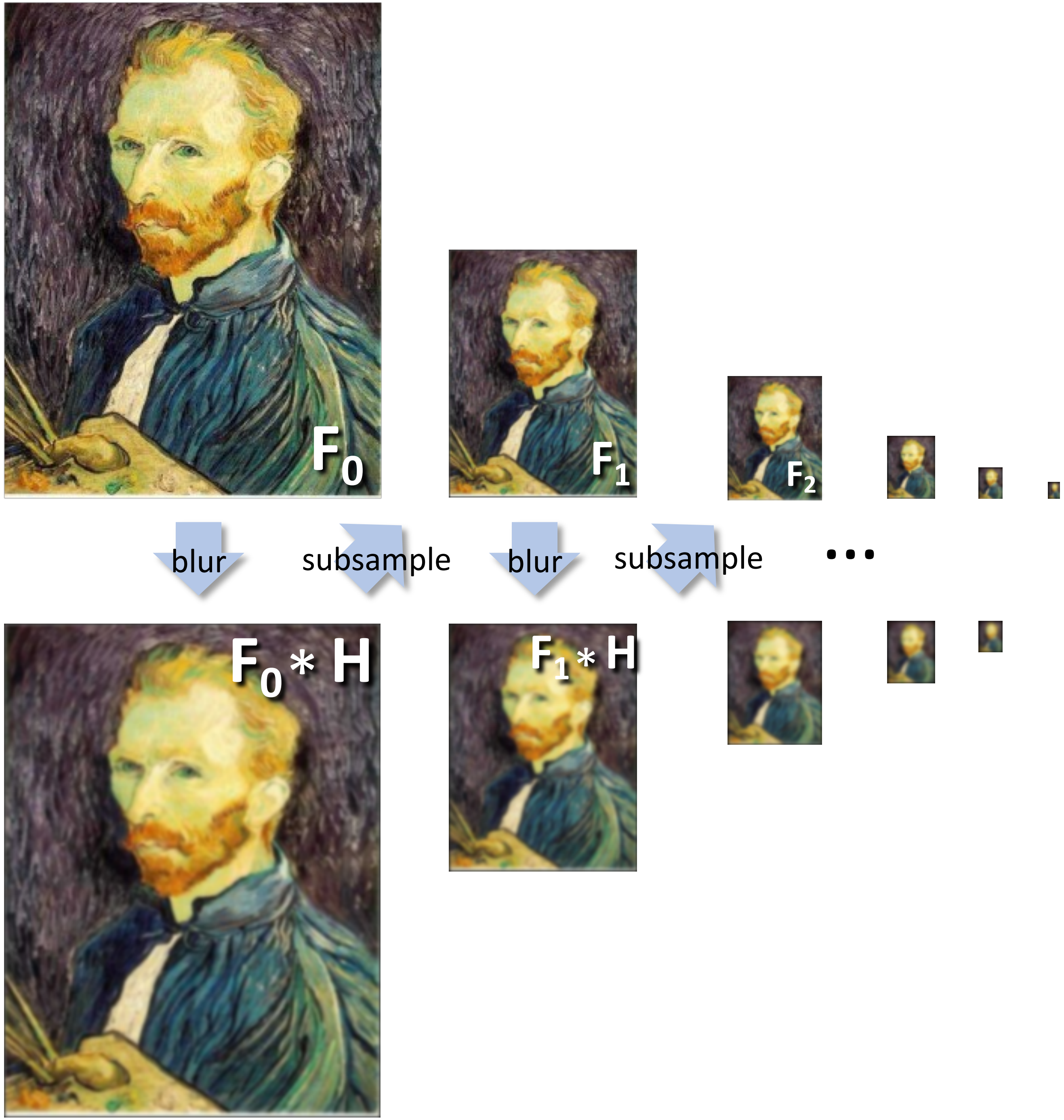
Keeps low frequencies

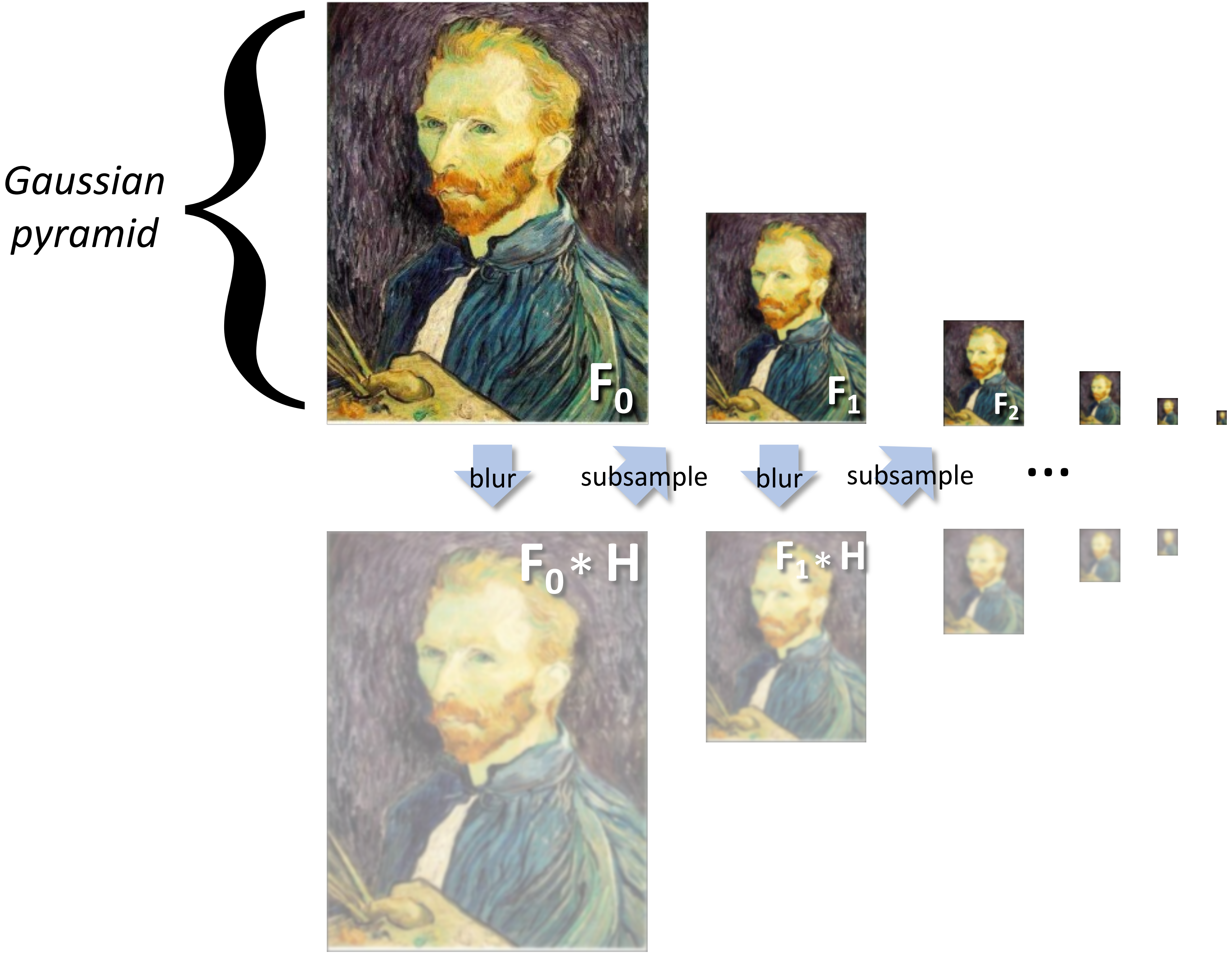
Gaussian filters

Fourier transform

Gaussian pre-filtering

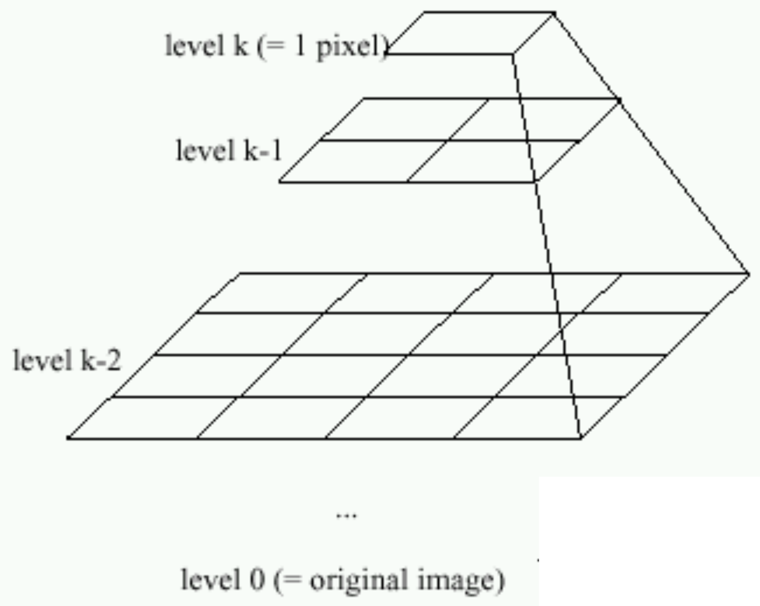
- Solution: filter the image, *then* subsample




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

Gaussian pyramids [Burt and Adelson, 1983]

Idea: Represent NxN image as a “pyramid” of 1×1, 2×2, 4×4,…, $2^k \times 2^k$ images (assuming $N=2^k$)

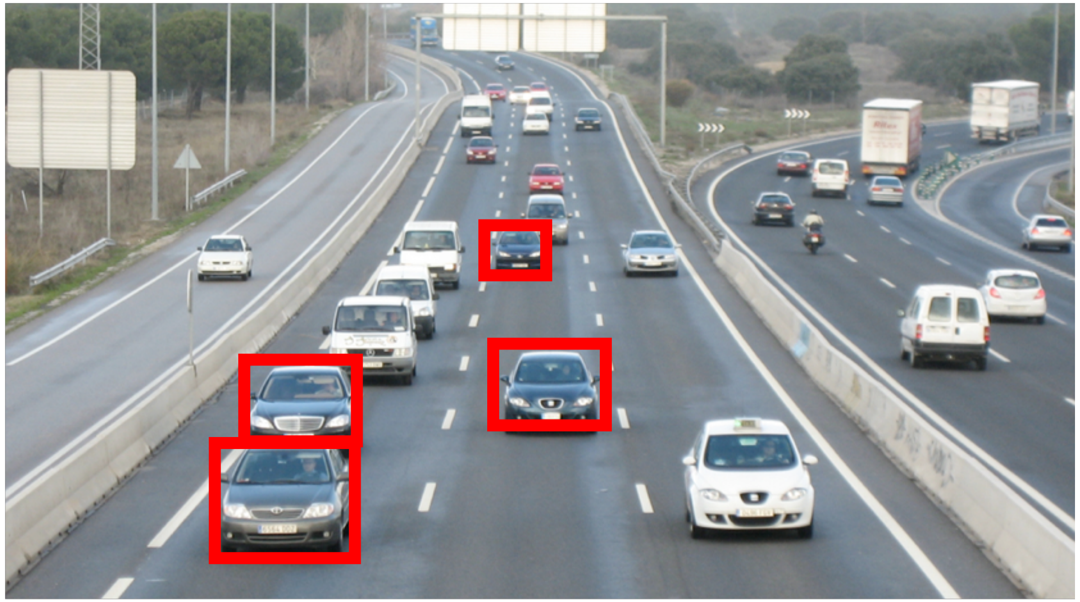


- In computer graphics, a *mip map* [Williams, 1983]

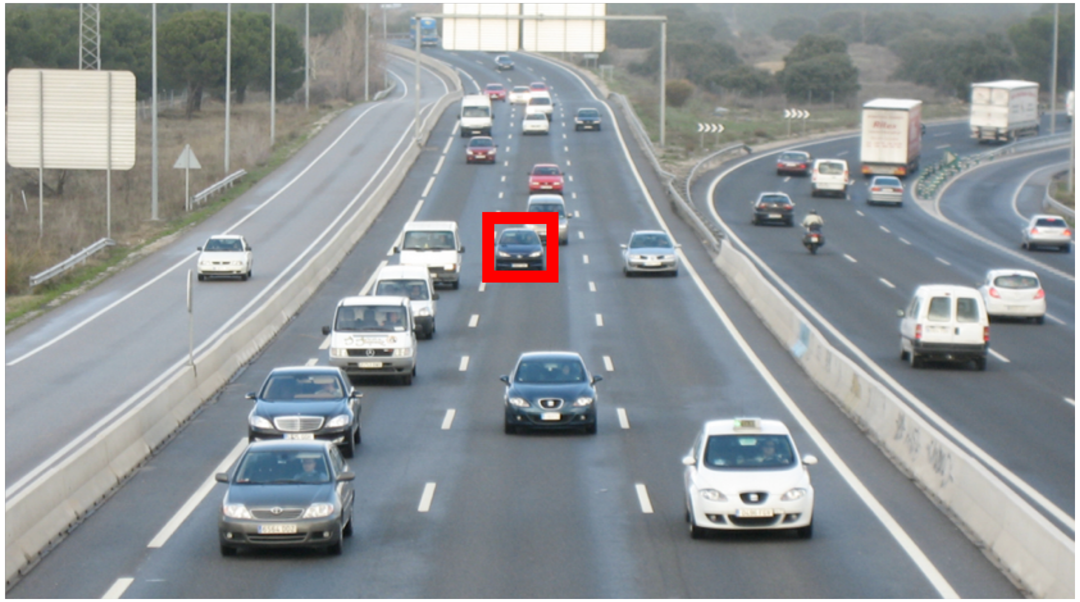
Gaussian Pyramids have all sorts of applications in computer vision

Source: S. Seitz

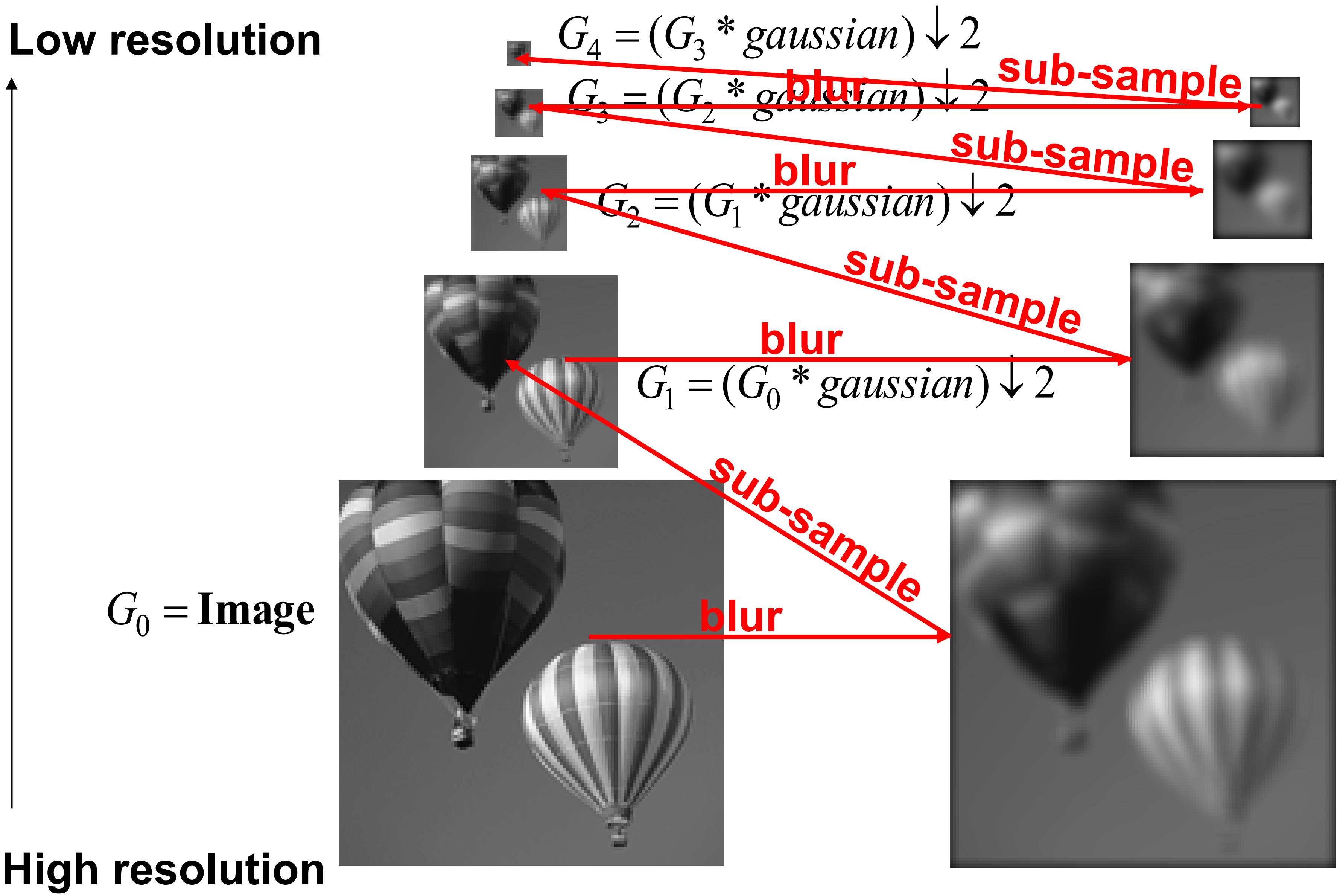
Gaussian pyramids - Searching over scales



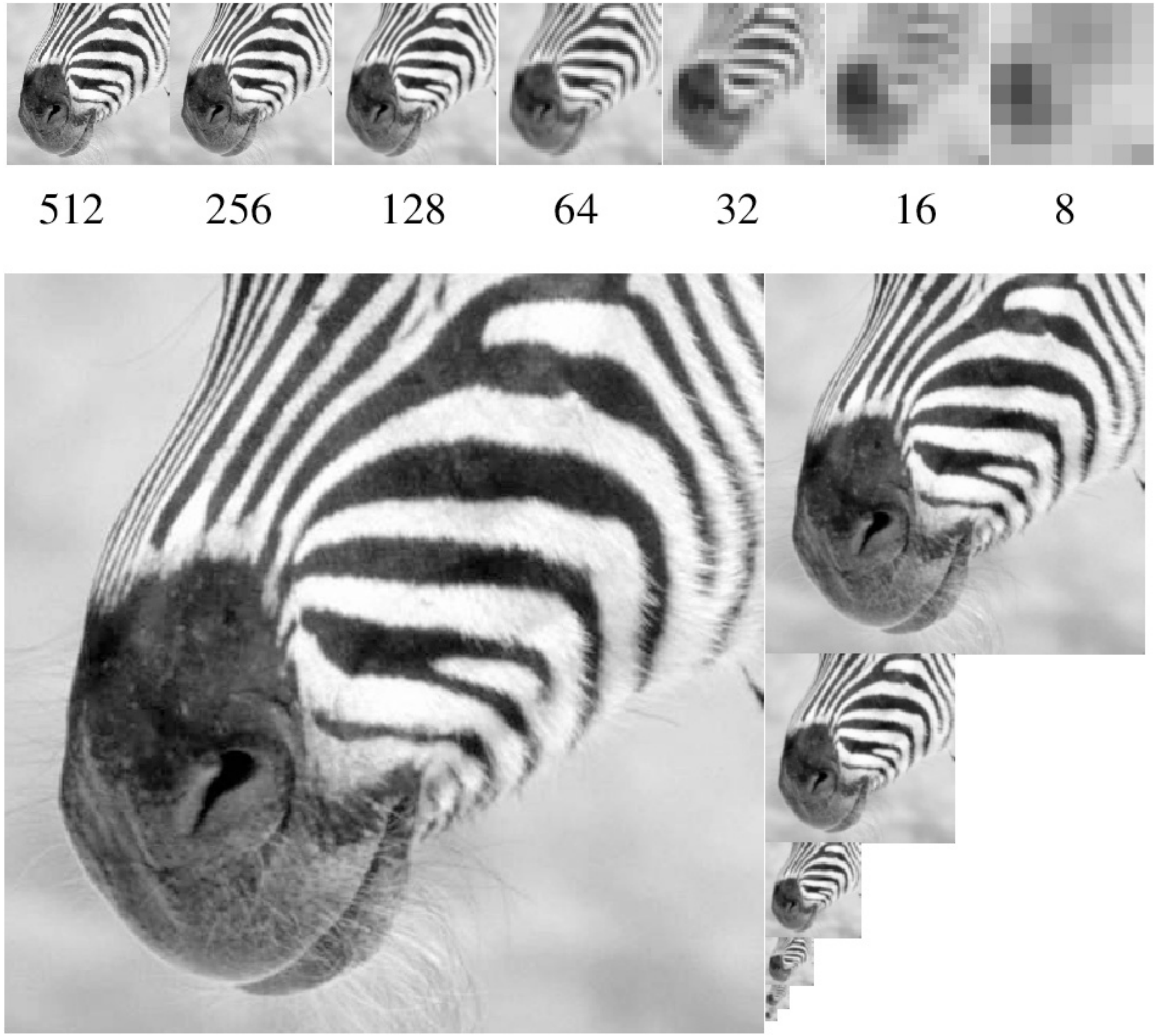
Gaussian pyramids - Searching over scales



The Gaussian Pyramid



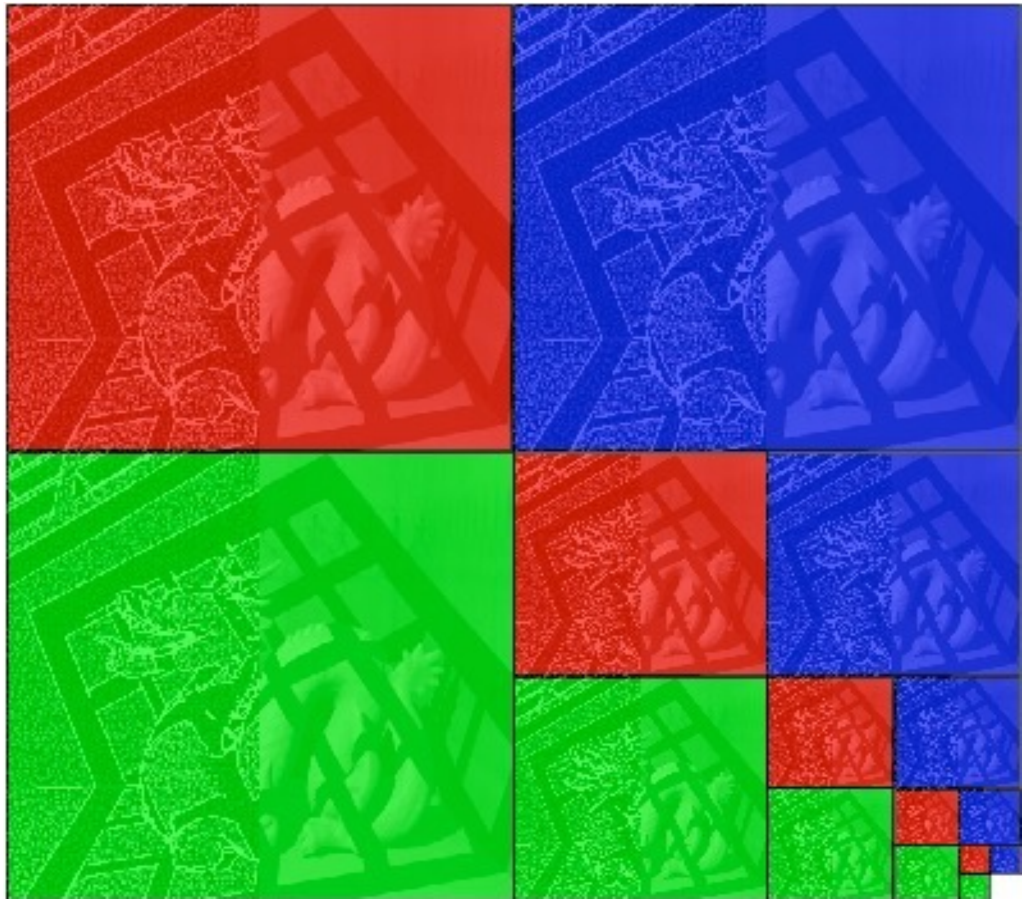
Gaussian pyramid and stack

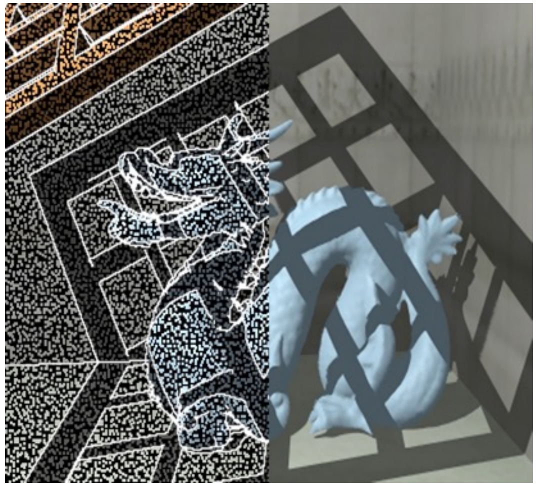


Source: Forsyth

Memory Usage

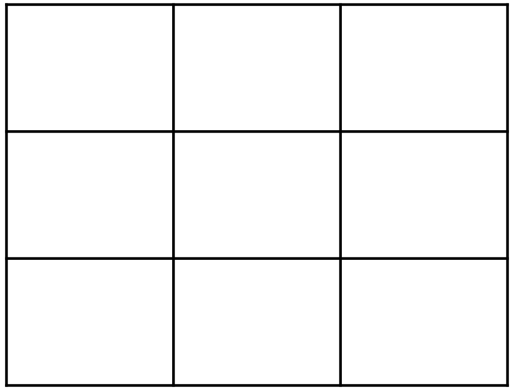
- Each color is a separate pyramid
- 3 pyramids fit into 2W x 2H image

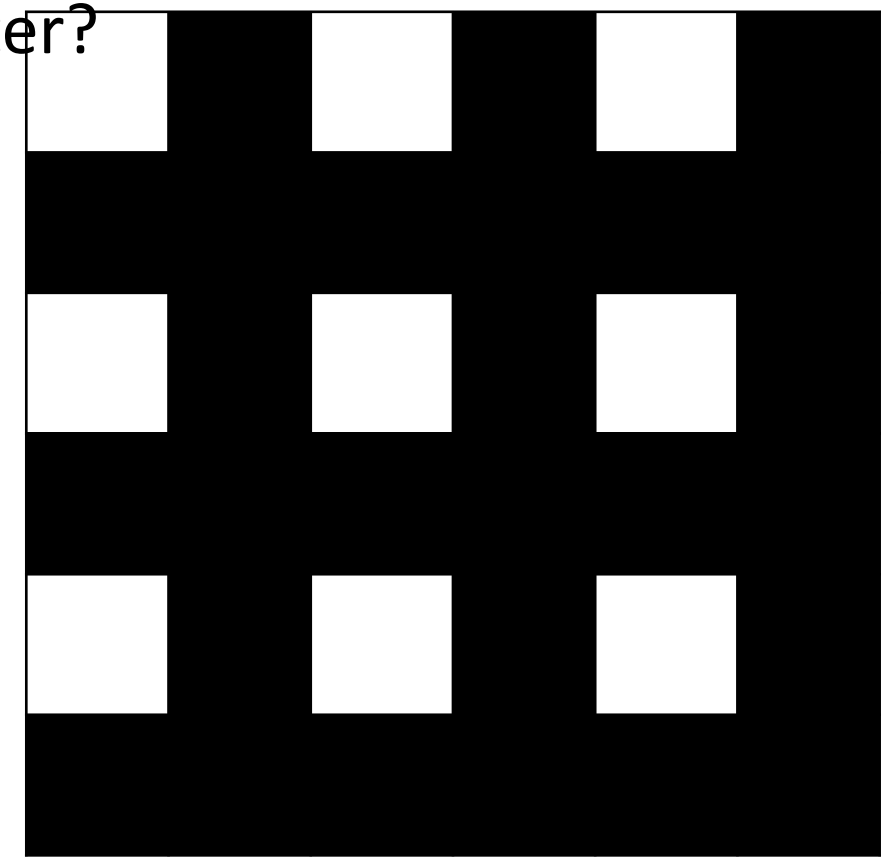




What about upsampling?

- Simple solution: Fill rest of the pixels with zeros
- Obviously wrong. How can we do better?





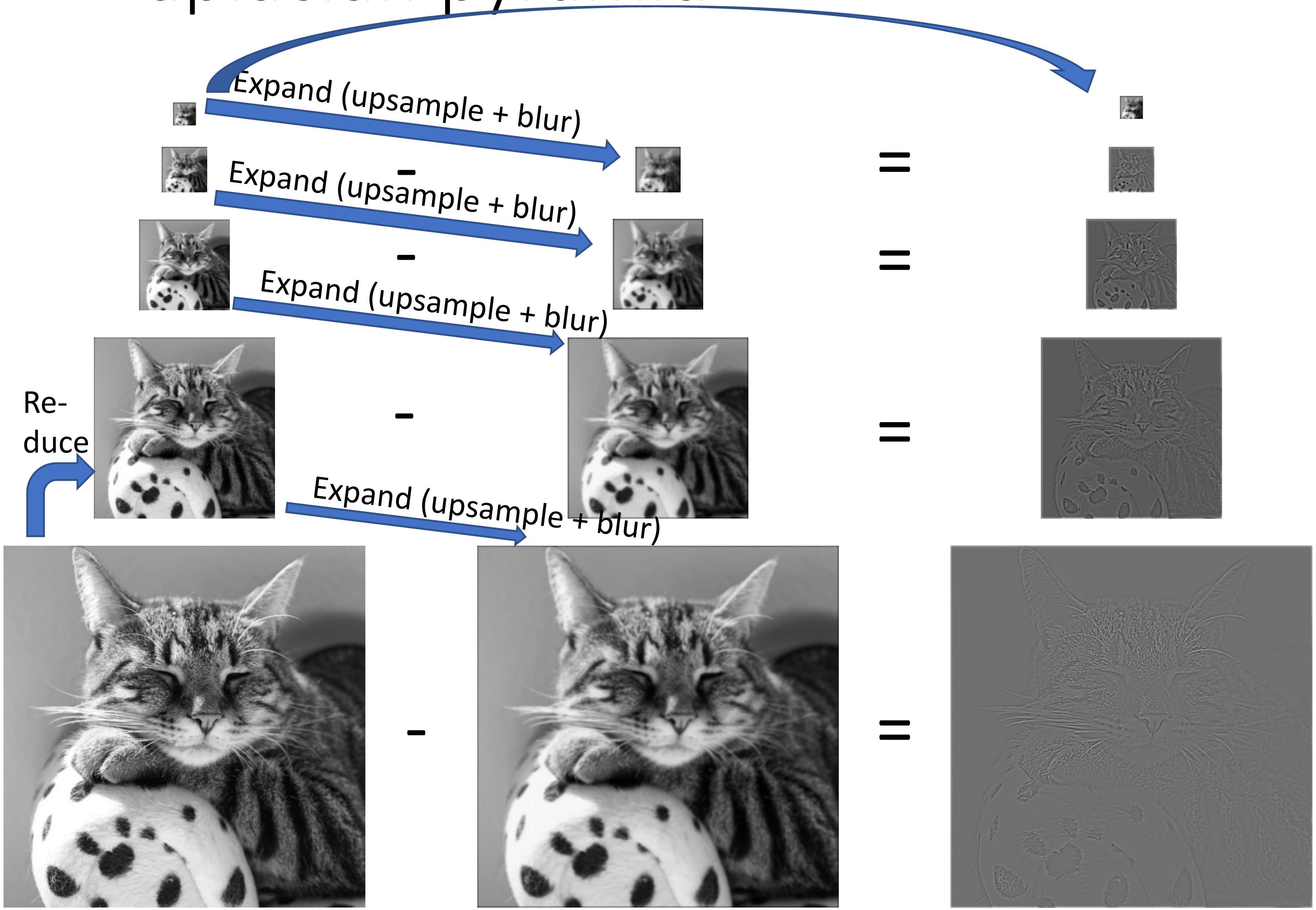
Upsampling

- Need to *interpolate* intermediate pixels. What is the best way to interpolate?
- Recall: before subsampling, we removed high frequencies
- Key idea: upsampled image should not have high frequencies either
- Gaussian blur again!

Upsampling

- Step 1: upsample and fill with 0s
- Step 2: Gaussian blur to interpolate
- Step 3: Scale correction
  - Gaussian blur is just weighted average
  - But we just introduced a bunch of zeros ==> need to scale up the resulting image

Laplacian pyramid



Laplacian pyramid

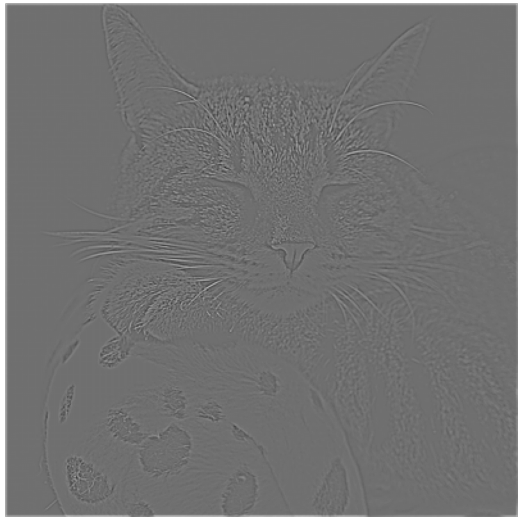
$L_4 = G_4 =$

$L_3 = G_3$ - expand$(G_4) =$

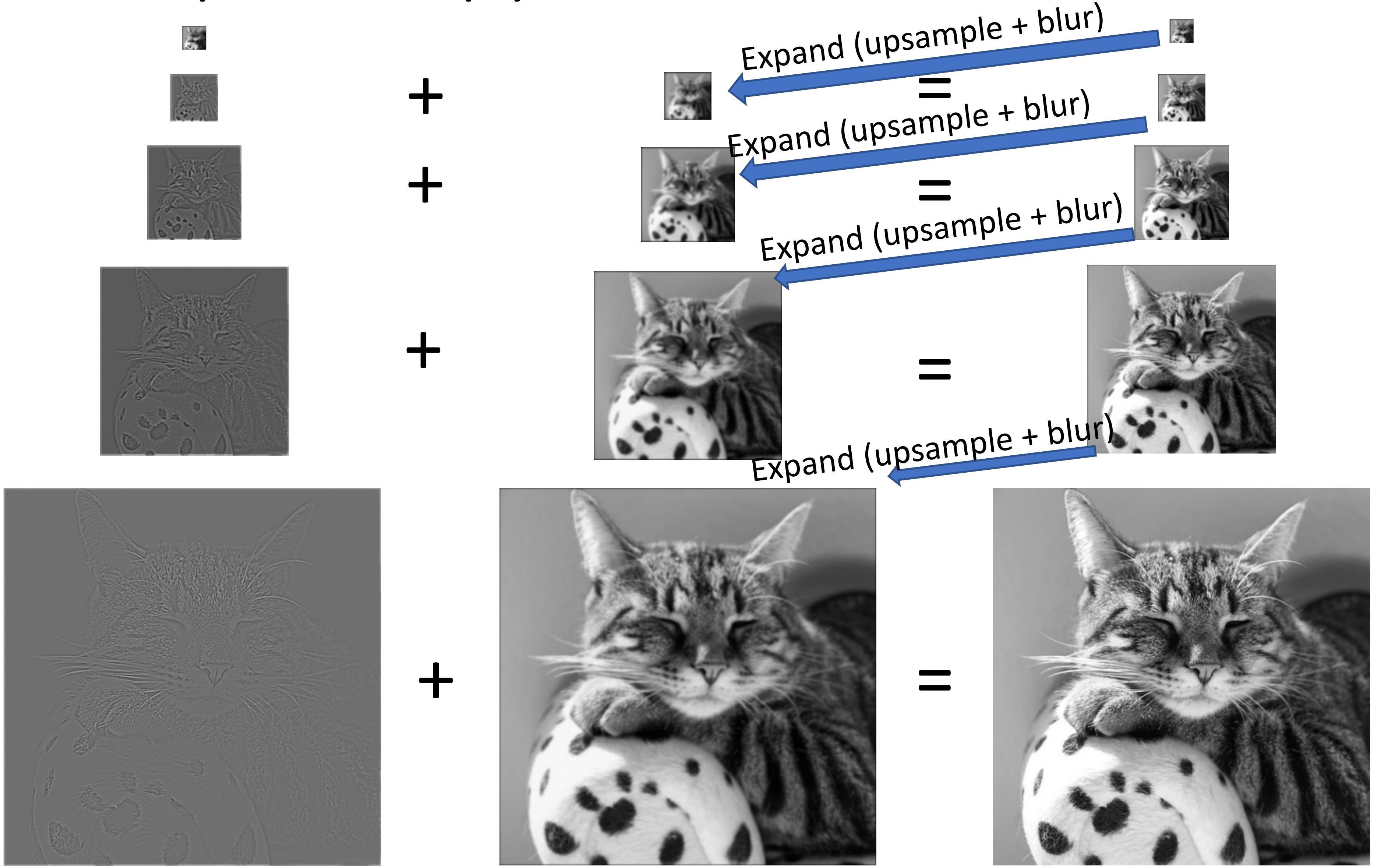
$L_2 = G_2$ - expand$(G_3) =$

$L_1 = G_1$ - expand$(G_2) =$

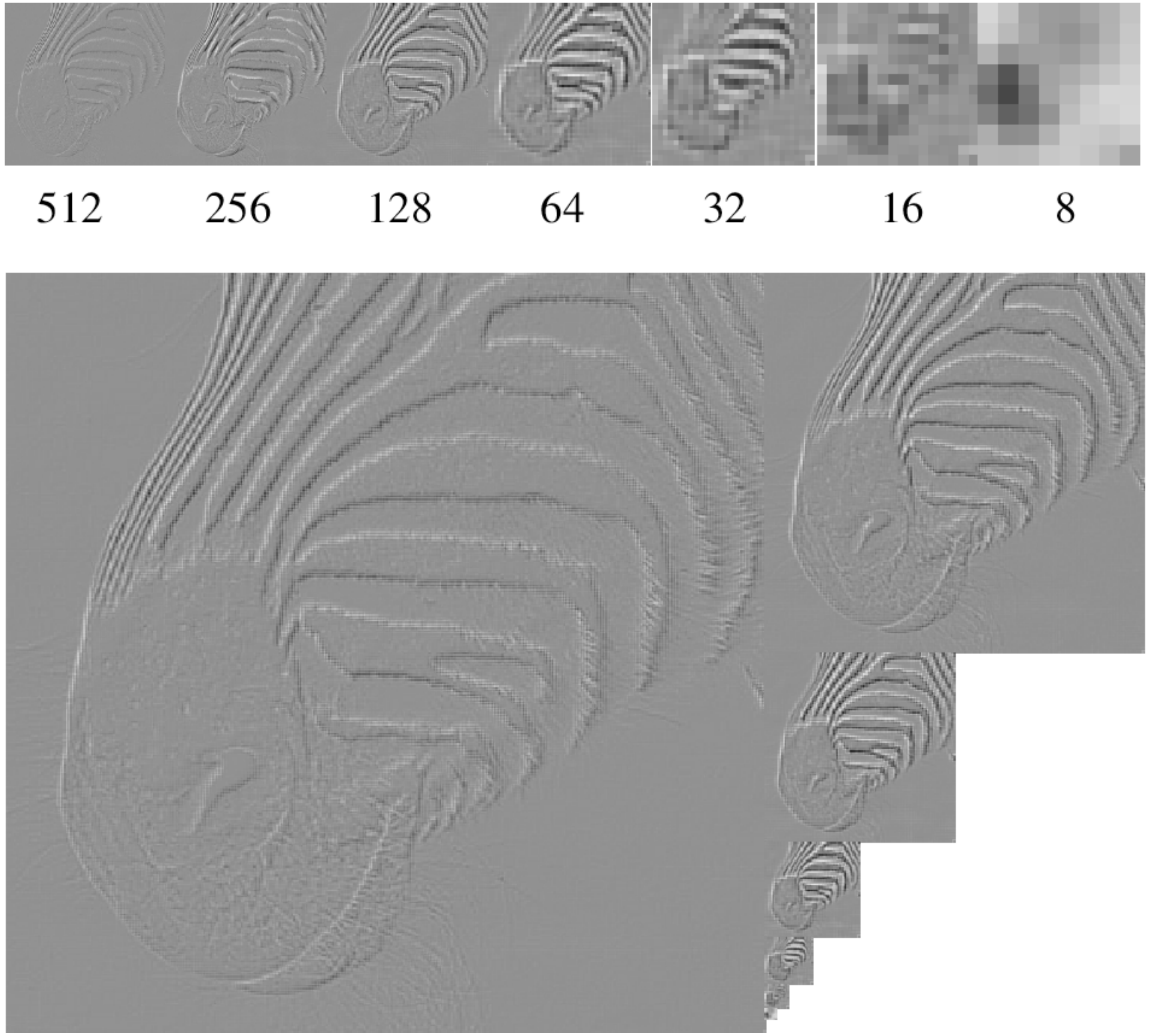
$L_0 = G_0$ - expand$(G_1) =$



Reconstructing the image from a Laplacian pyramid



Laplacian pyramid



Source: Forsyth

Low-pass and high-pass filtering

- Convolving with a Gaussian = remove high frequencies
- “Low-pass” filtering: low frequencies “pass” through filter, high frequencies don’t
- Identity – Low-pass filtered image = “High-pass filtering”

Hybrid images (PA1)

- From afar, images look tiny, we only see low frequencies
- Up close, we see high frequencies
- Low frequencies of one image + high frequencies of another