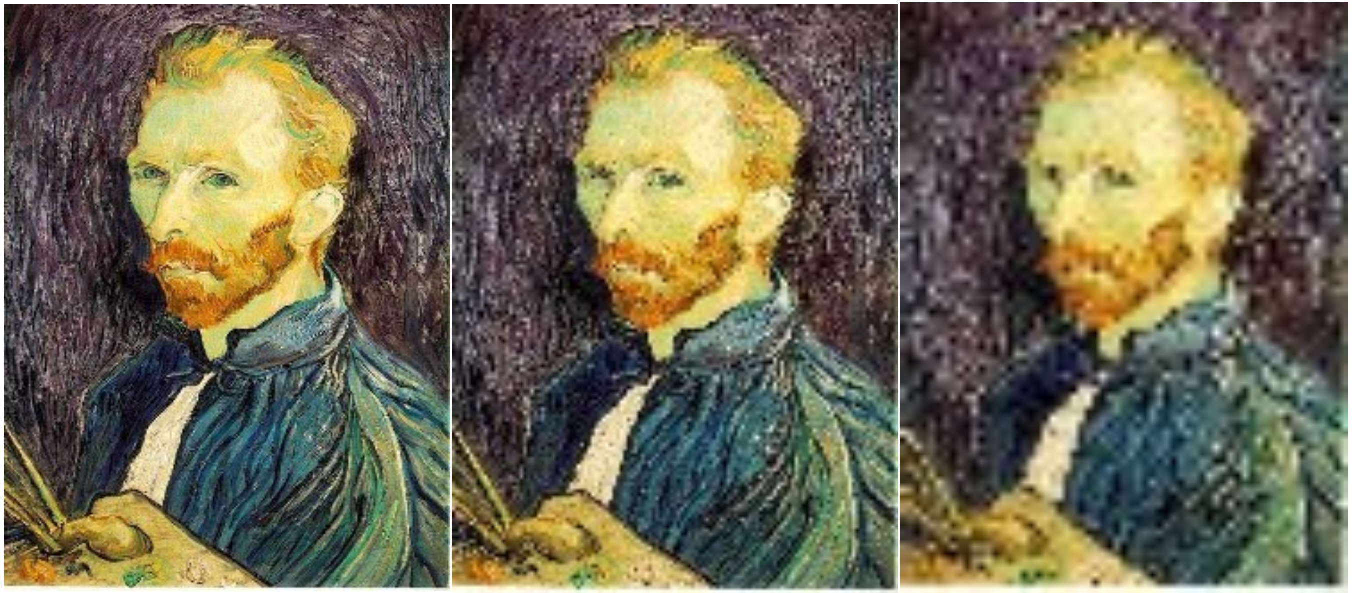
CSCI 497P/597P: Computer Vision



Lecture 6:

~~Sobel Filter~~

Image Frequency Content

~~Downsampling and~~ Gaussian Pyramids

Laplacian Pyramids

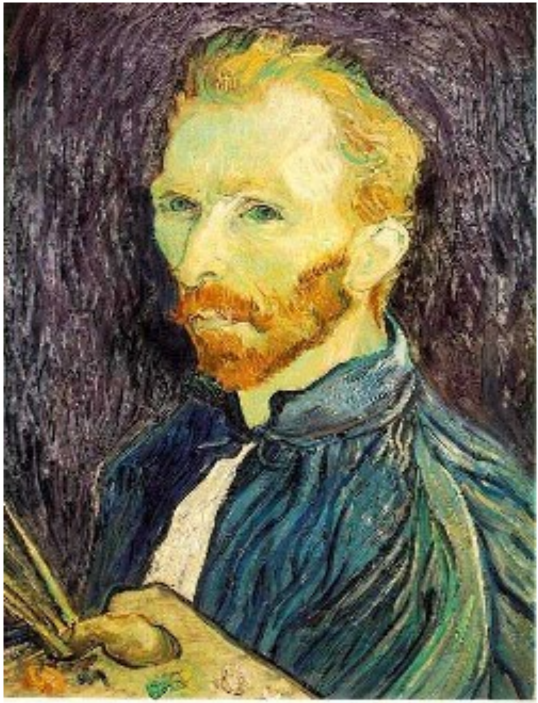
Upsampling

Announcements

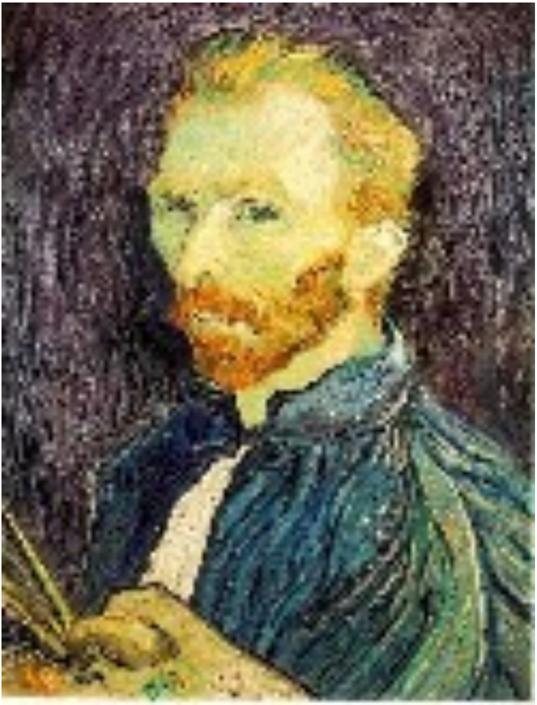
Goals

- Know how and why to construct a Gaussian Pyramid
- Know how and why to construct a Laplacian Pyramid
- Know how to upsample images naively
- Know how to upsample images using reconstruction filters.
    - Understand how to upsample using bilinear interpolation and how it relates to reconstruction filters.

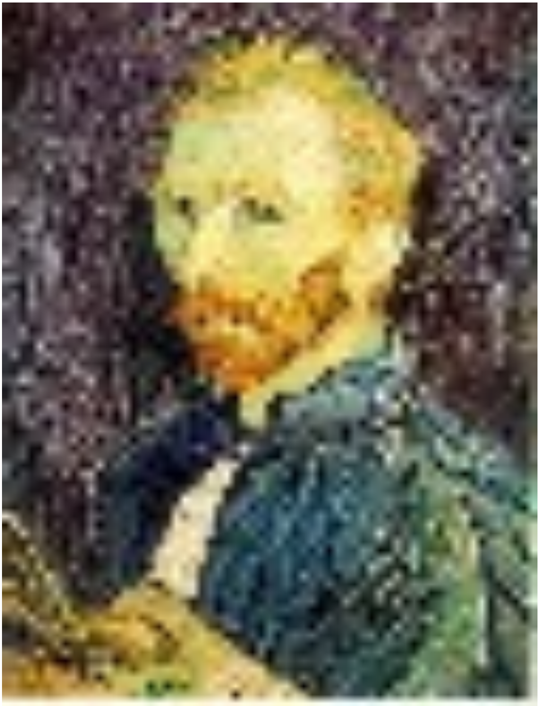
Image Subsampling



1/2



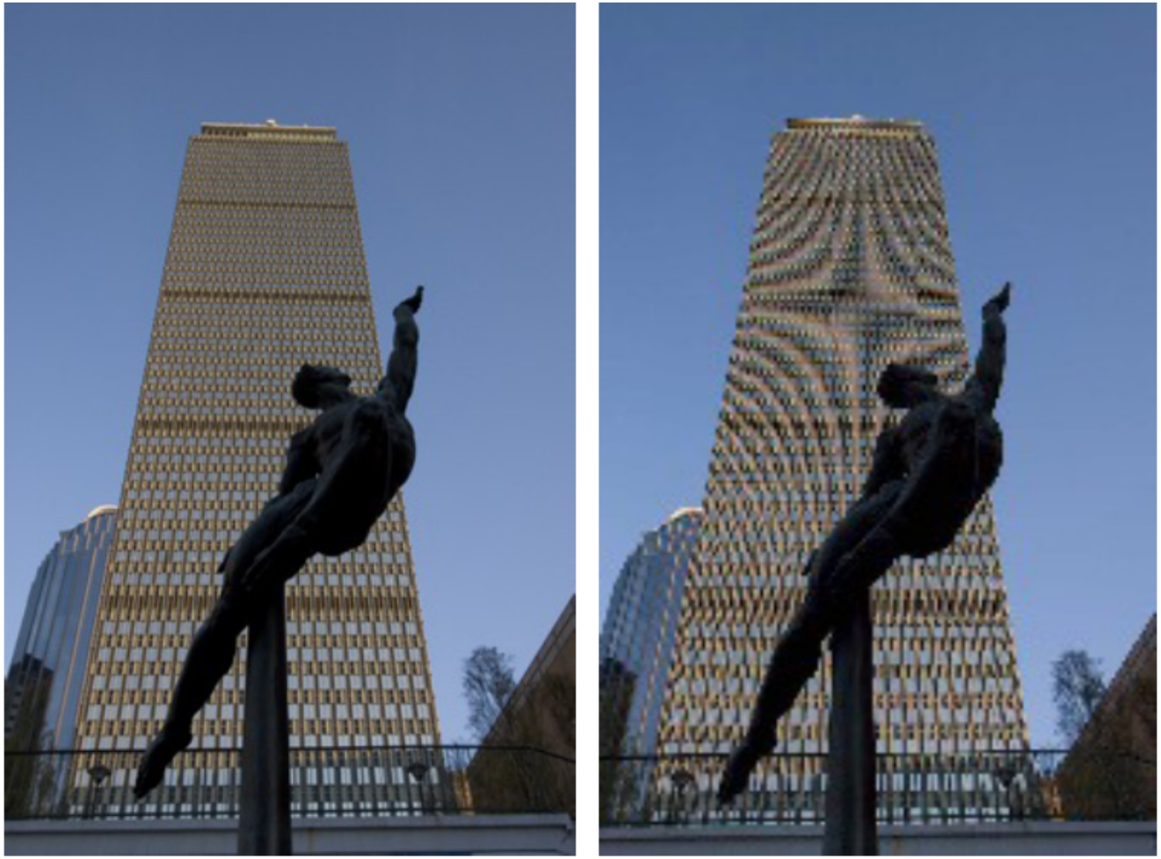
1/4



1/8

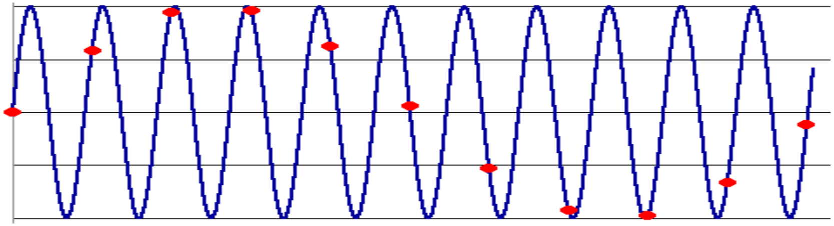
Why does this look so crufty?

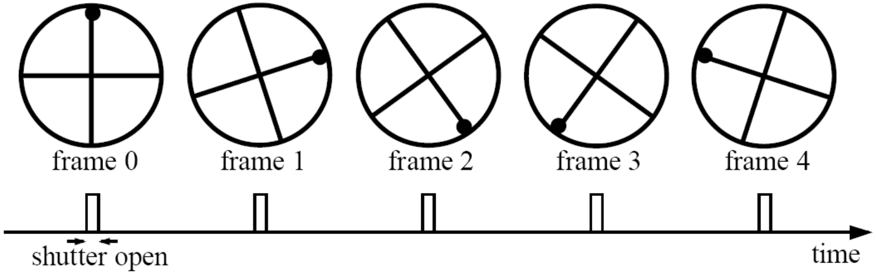
Subsampling: Another example



Source: Fredo Durand

Aliasing

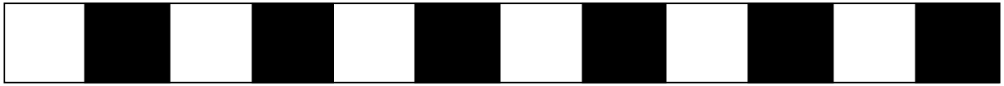




https://en.wikipedia.org/wiki/Wagon-wheel_effect

Aliasing

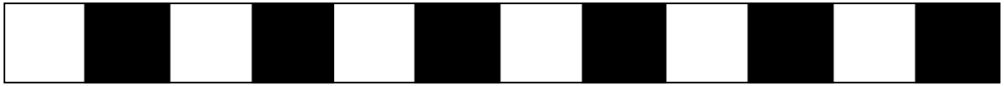
- Let's look back at our highest-frequency scanline:



- If we need to represent this image with only 6 pixels (half size), What's the "right" (i.e., best we can do) answer?

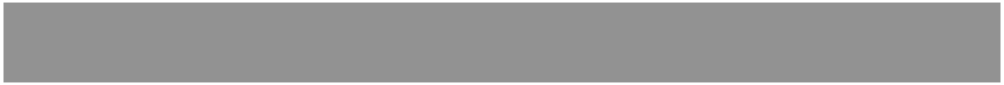
Aliasing

- Let's look back at our highest-frequency scanline:



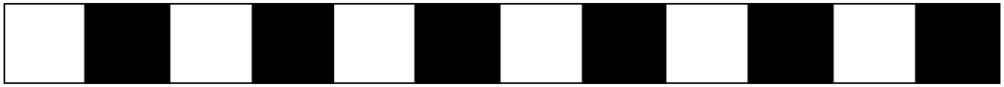
- If we need to represent this image with only 6 pixels (half size), What's the "right" (i.e., best we can do) answer?

- If we walked far away, what we'd see is:

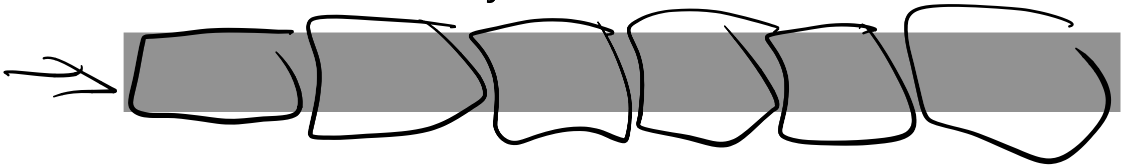


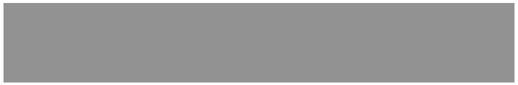
Aliasing

- Let's look back at our highest-frequency scanline:



- If we need to represent this image with only 6 pixels (half size), What's the "right" (i.e., best we can do) answer?
- If we walked far away, what we'd see is:



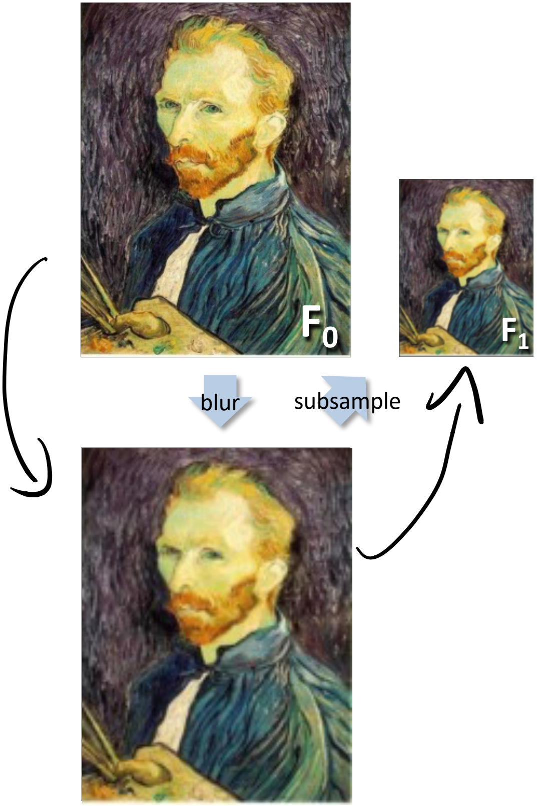
- Subsample that, and we get: 

Downsampling

Blurring *removes* high frequencies.

So, to make an image smaller:

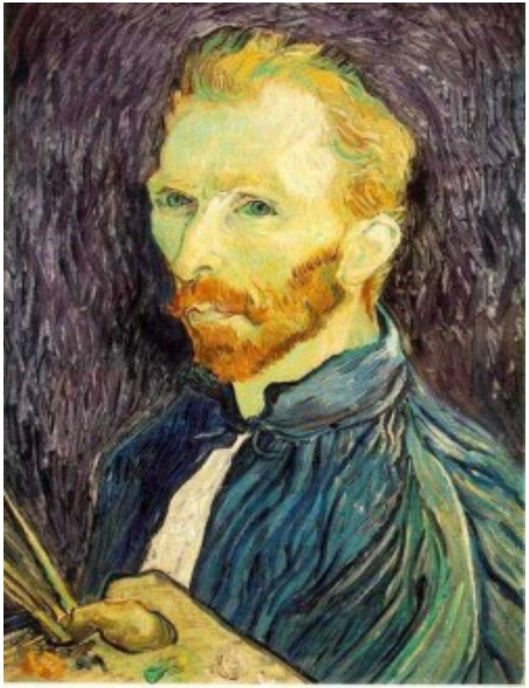
1. blur (pre-filter) the image
2. *then* subsample it.



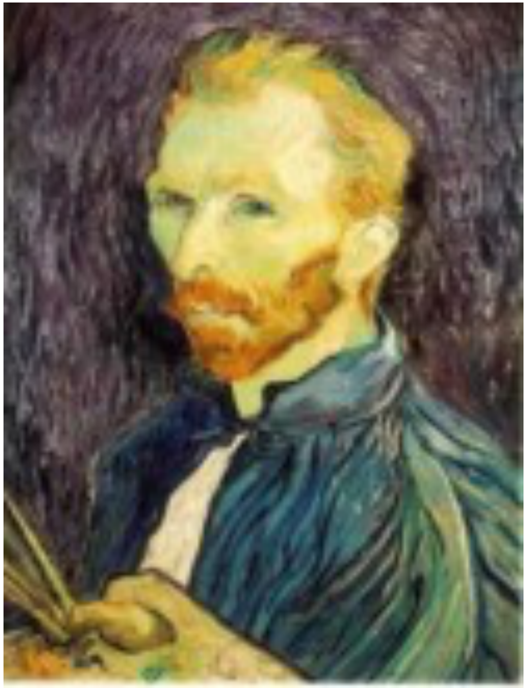
F_0: Original image

F_1: Downsampled image

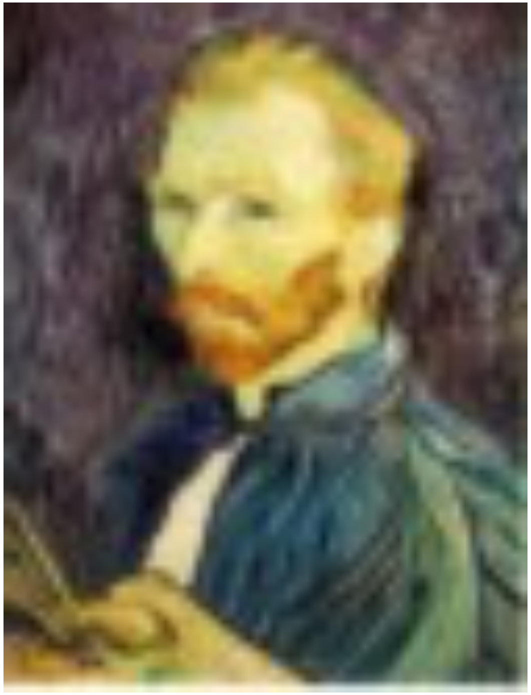
Downsampling with Gaussian Pre-filtering



1/2

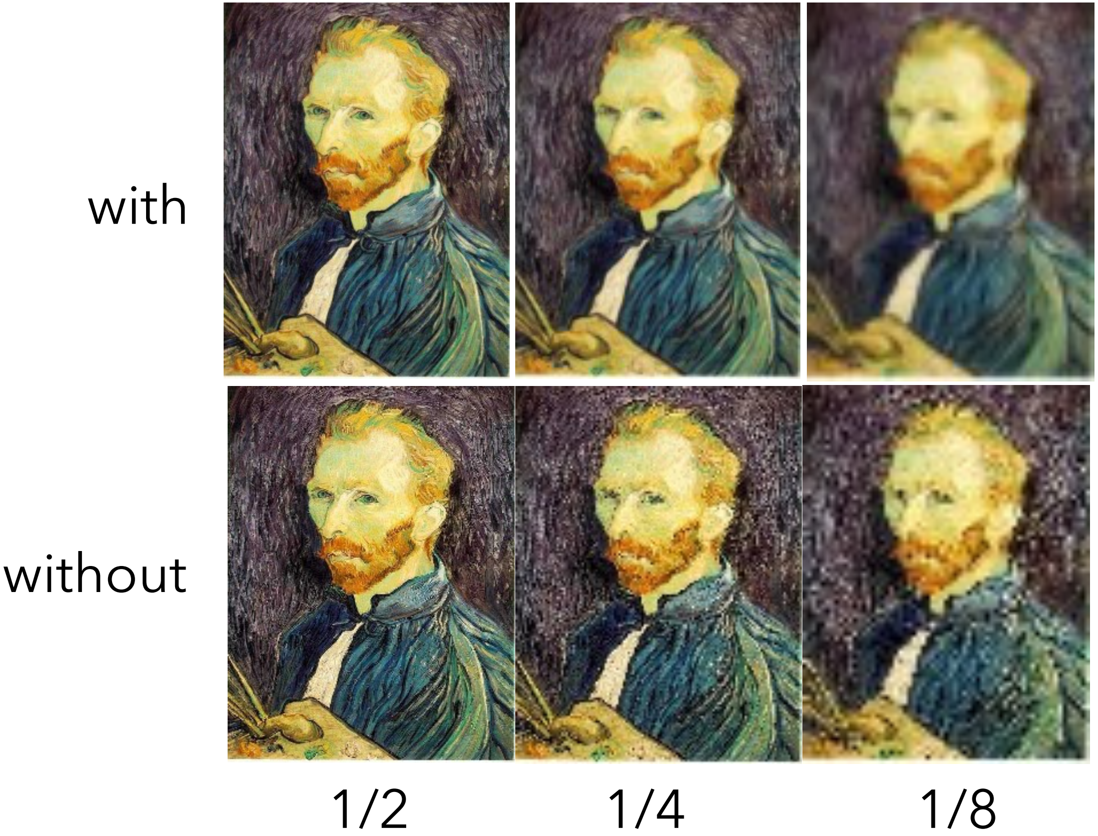


1/4



1/8

Downsampling with Gaussian Pre-filtering

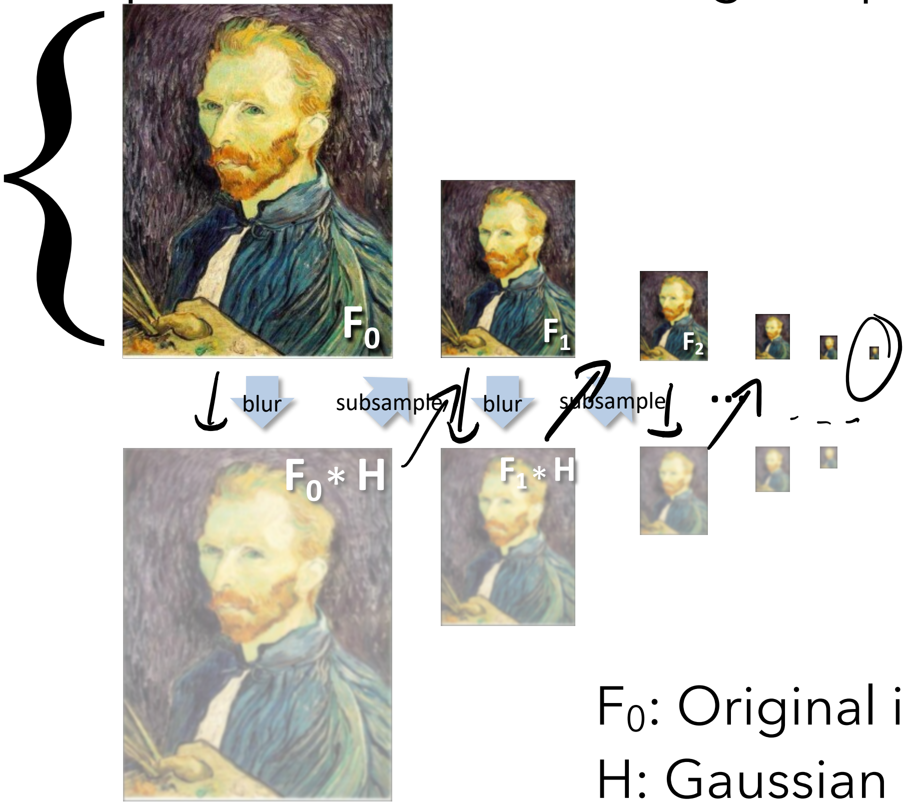


Gaussian Pyramid

To create a multi-scale representation of an image, repeat:

1. Store
2. Blur
3. Subsample

Gaussian pyramid

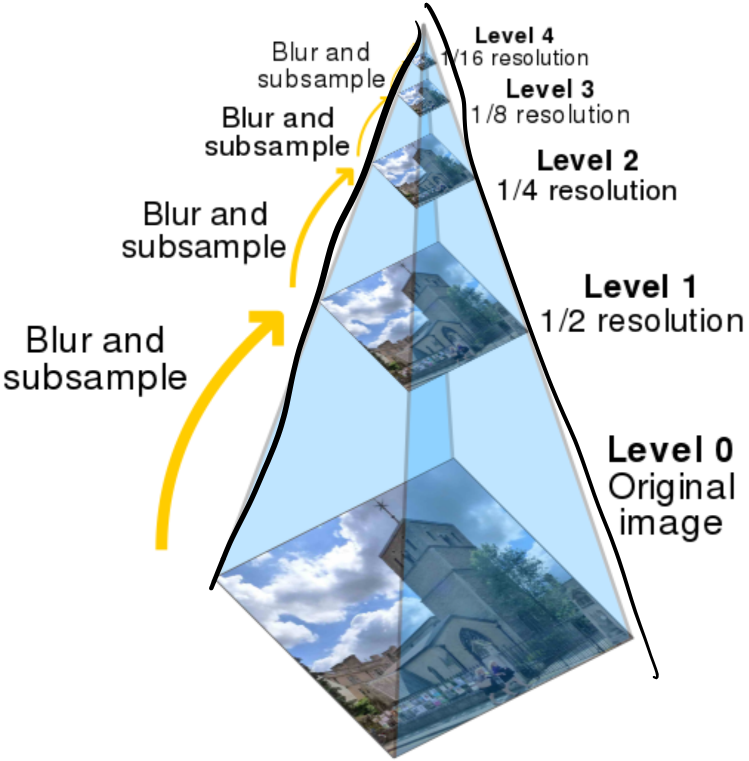


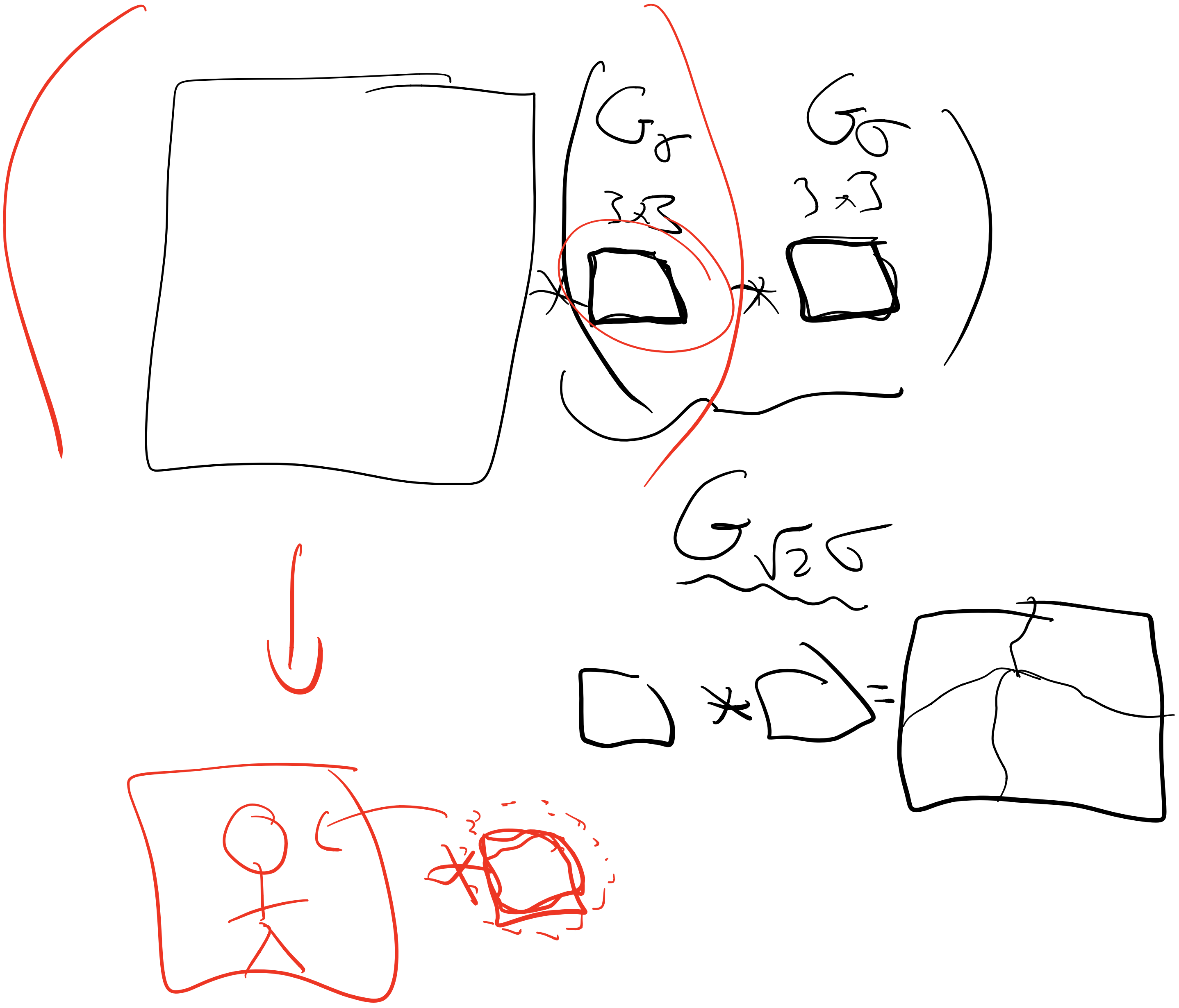
F_0: Original image

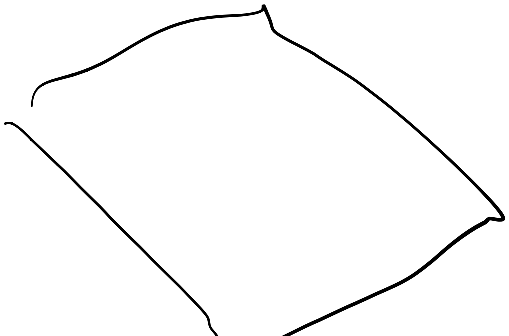
H: Gaussian blur

Gaussian Pyramid

A more pyramid-shaped view:

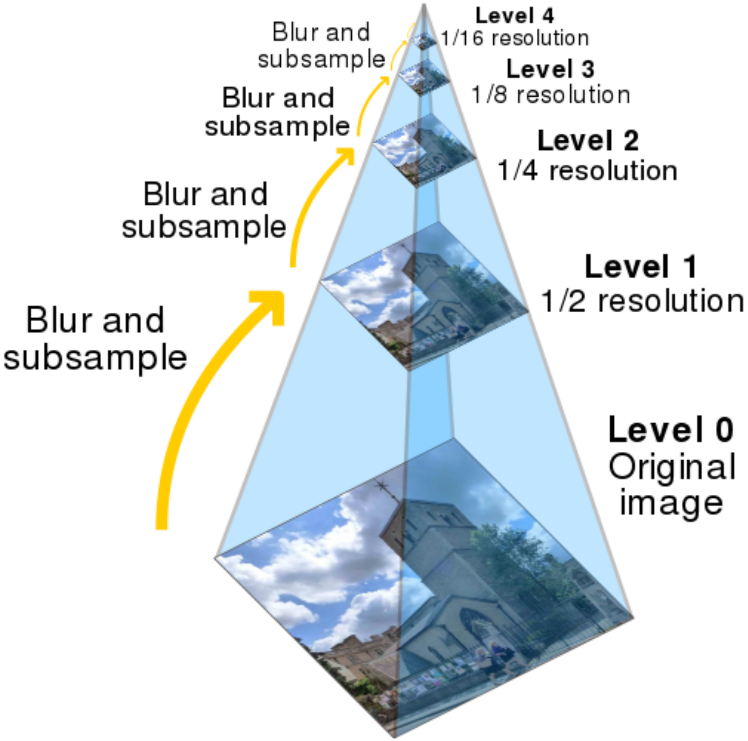



G_γ
G_σ
3×3
3×3
*
*
$G_{\sqrt{2}\sigma}$
*
=
*

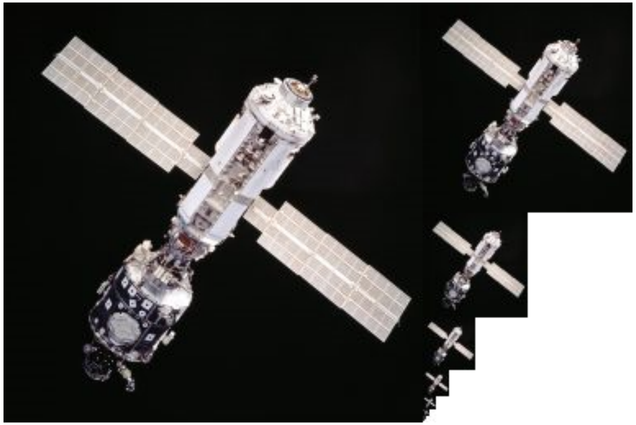


Gaussian Pyramid

A more pyramid-shaped view:



A storage-oriented view:



Gaussian Pyramid: But why?

- You have a (edge, object, whatever) detector. You want to run it at multiple scales:

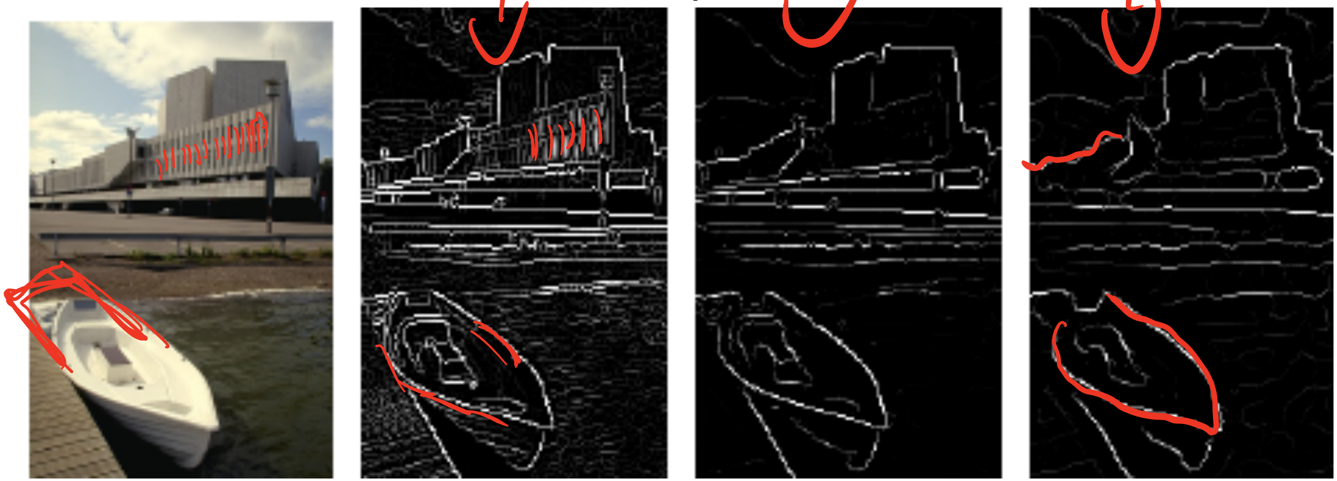
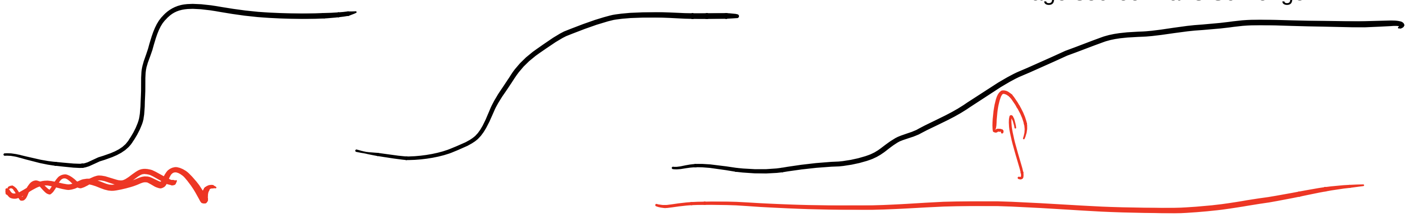
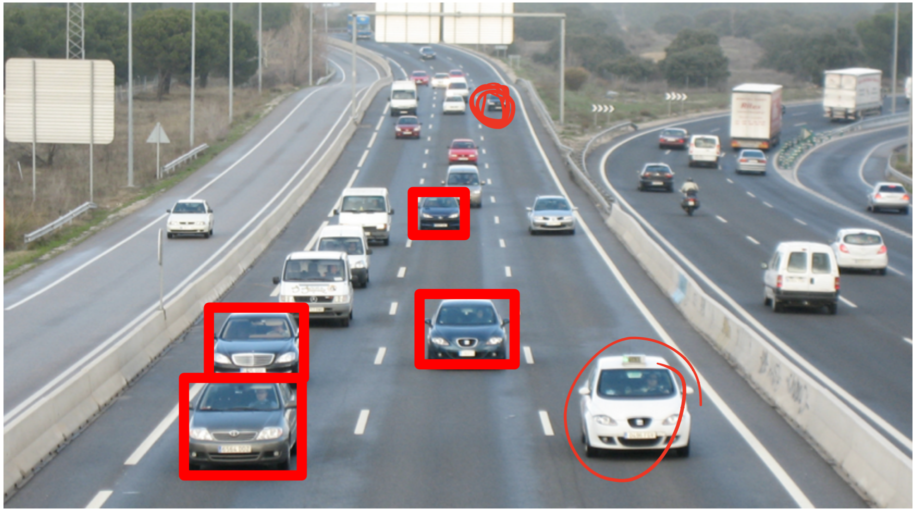


Image source: Baris Sumengen



Gaussian Pyramid: But why?

- You have a (edge, object, whatever) detector.
  You want to run it at multiple scales:



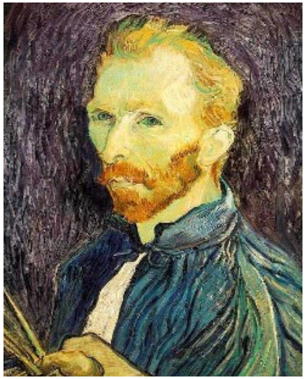
Some terminology

From the signal processing / electrical engineering field:

- A low-pass filter preserves low frequencies and eliminates (or attenuates) high frequencies.

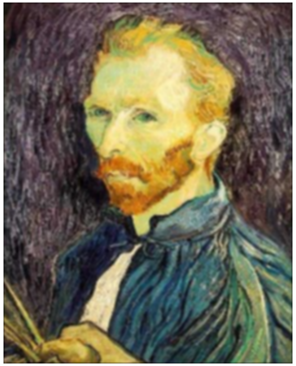
- A high-pass filter preserves high frequencies and eliminates (or attenuates) low frequencies.

$f =$



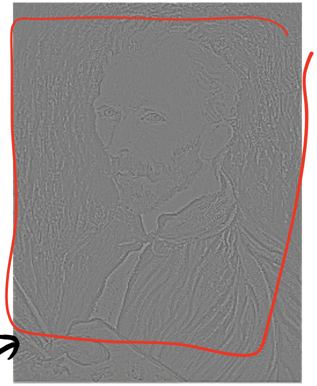
$f * w =$

(Gaussian Blur, e.g.)



$f * (---)$

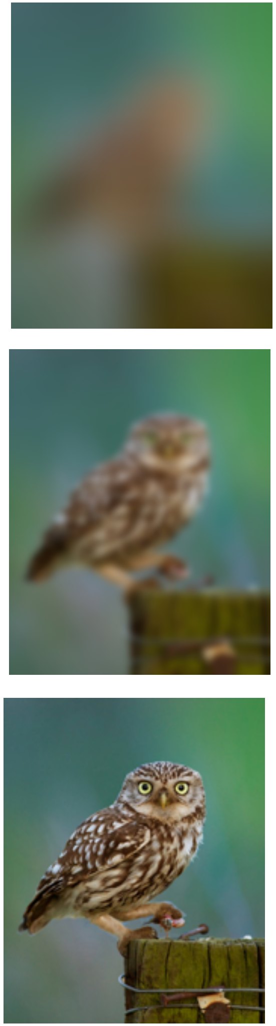
$f - (f * w) =$



(We could also derive a single filter to accomplish this)

Frequency content in a Gaussian Pyramid

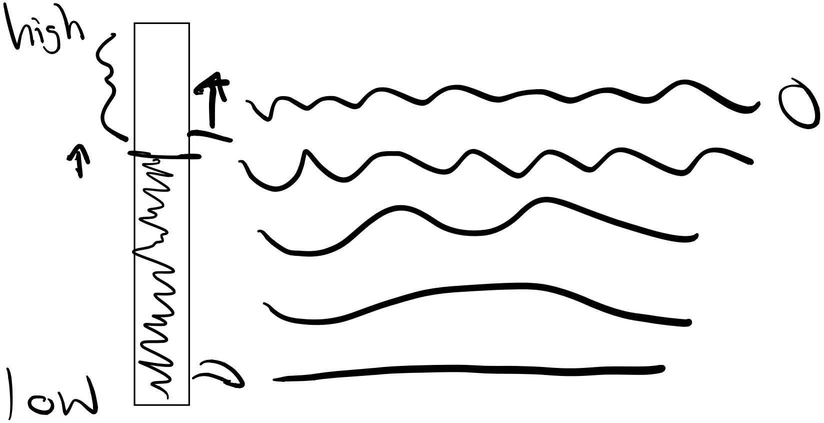
This is (secretly) a Gaussian Pyramid:

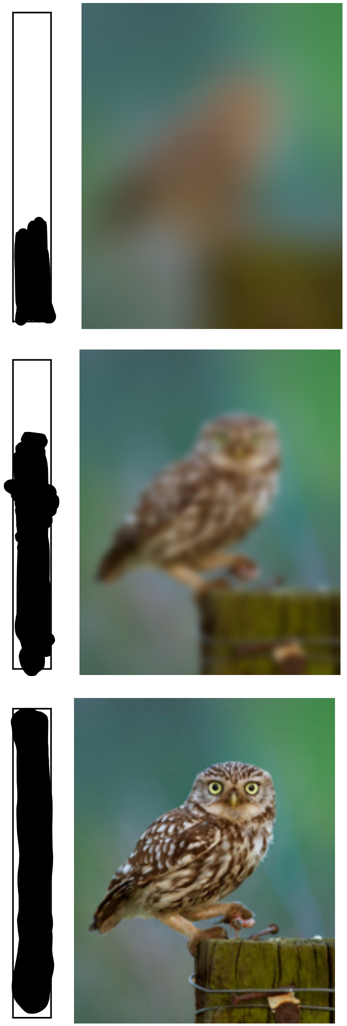


Frequency content in a Gaussian Pyramid

This is (secretly) a Gaussian Pyramid:

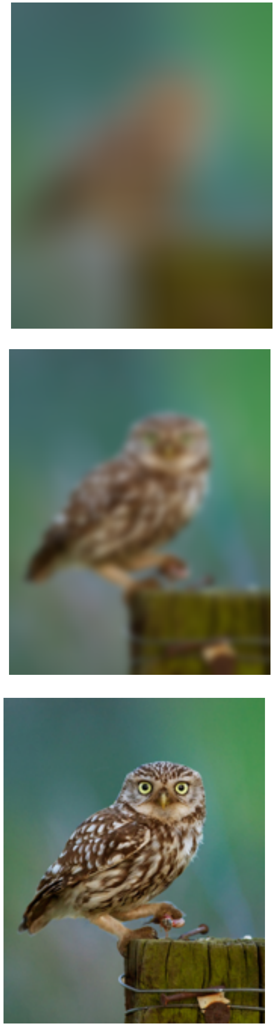
Frequency-ometer:





Frequency content in a Gaussian Pyramid

This is (secretly) a Gaussian Pyramid:

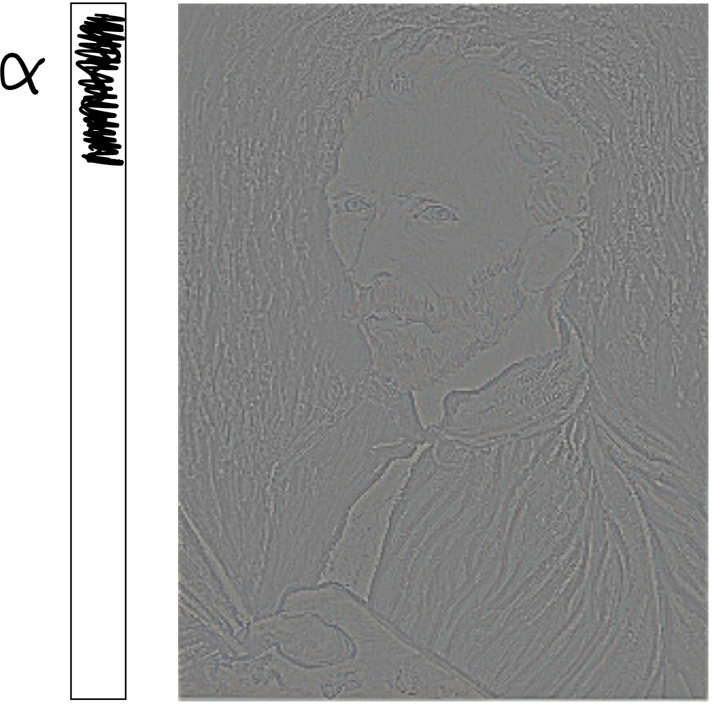


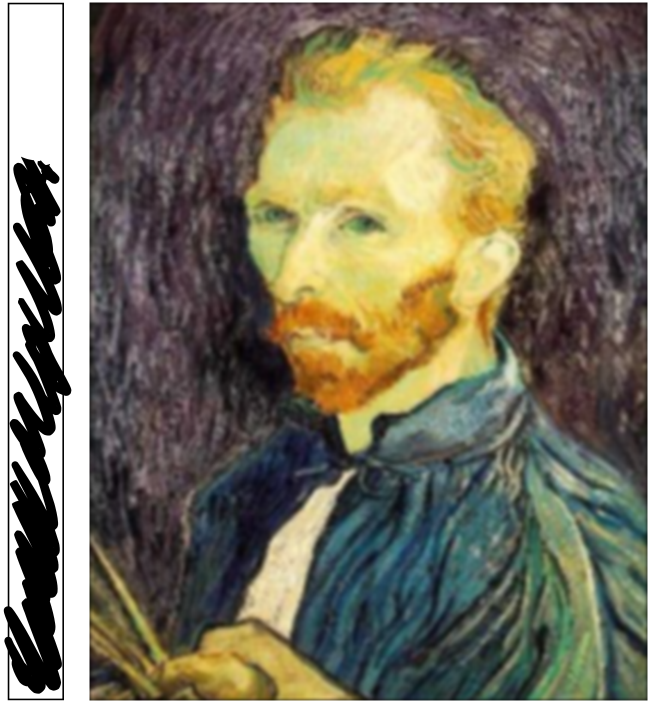
Frequency-ometer:

High-pass filter

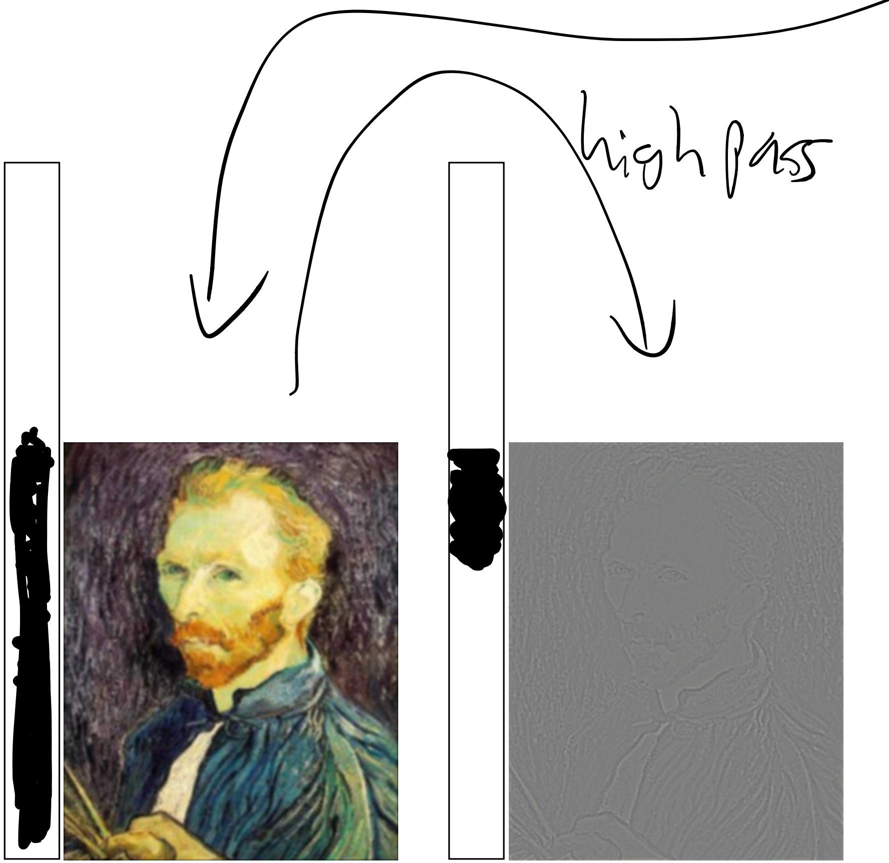
$f - (f * w)$

$f * w$

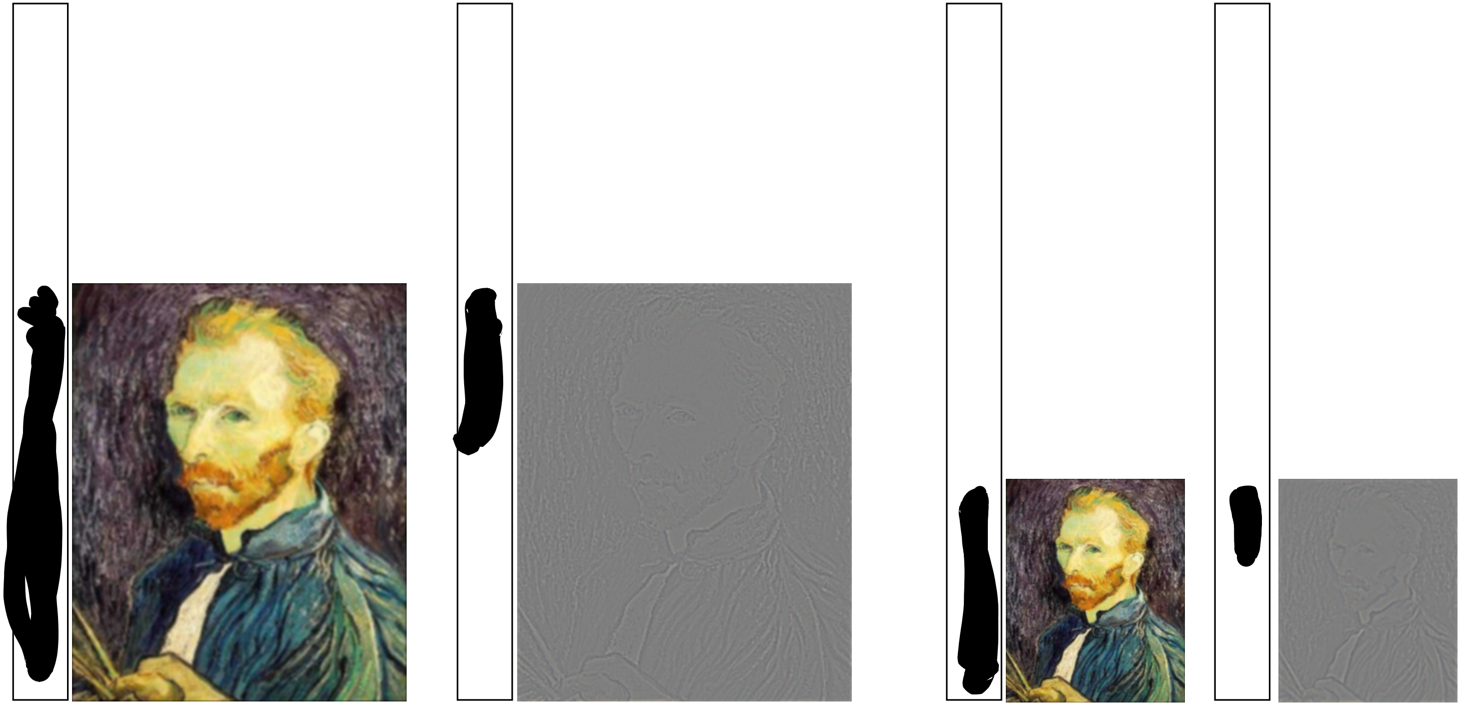




Subsample, then High-pass filter again!

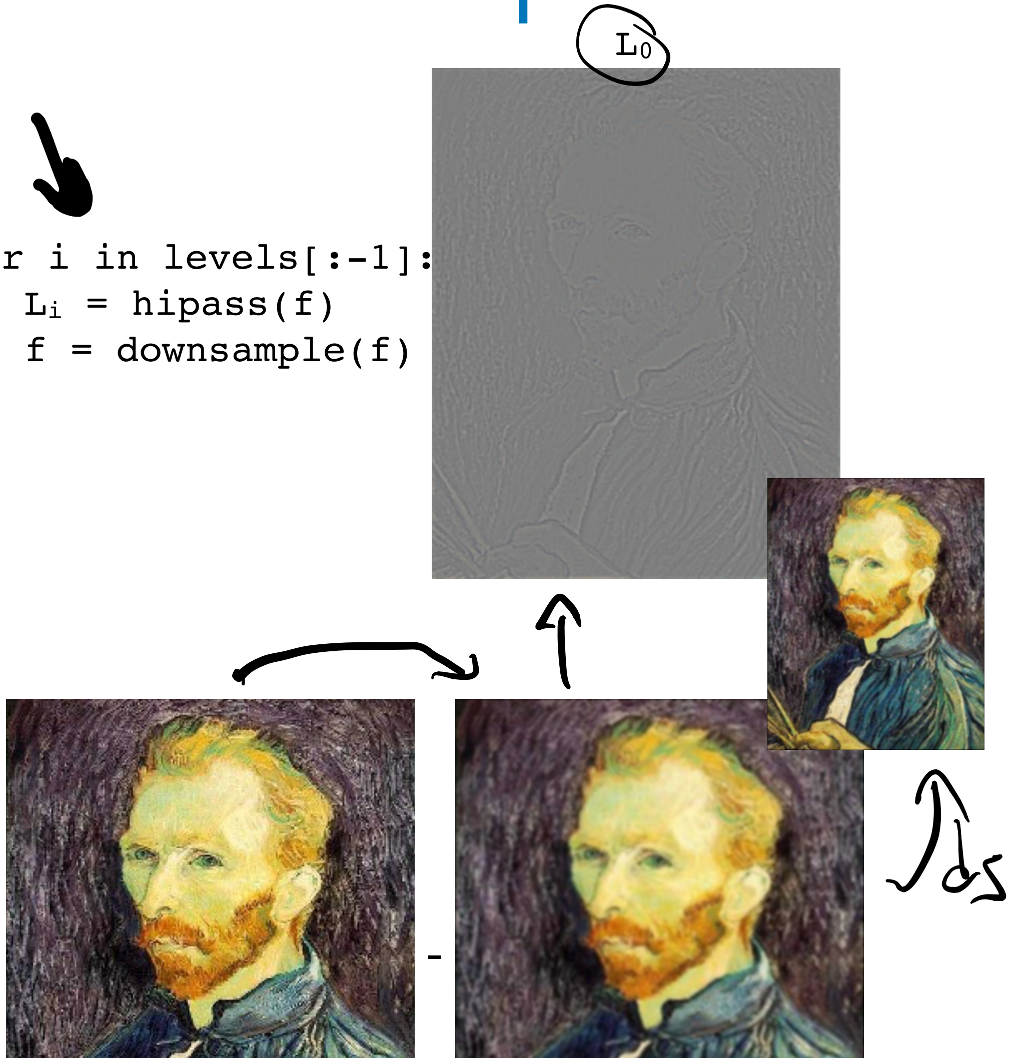


...and again!?



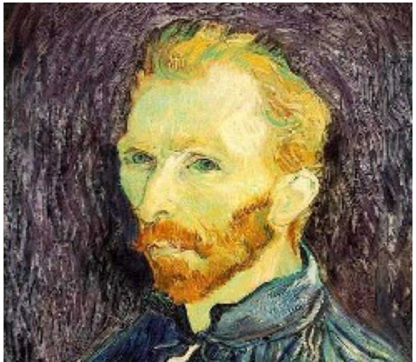
The Laplacian Pyramid

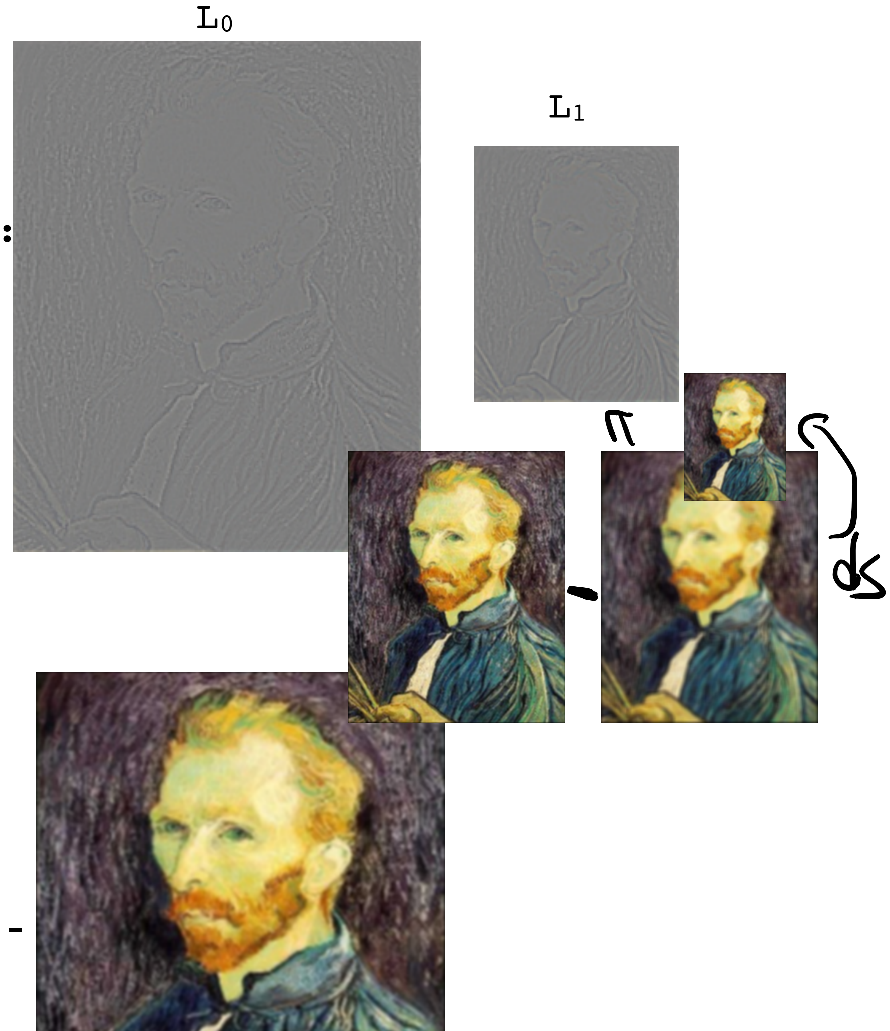
```
for i in levels[:-1]:
    $L_i$ = hipass(f)
    f = downsample(f)
```



The Laplacian Pyramid

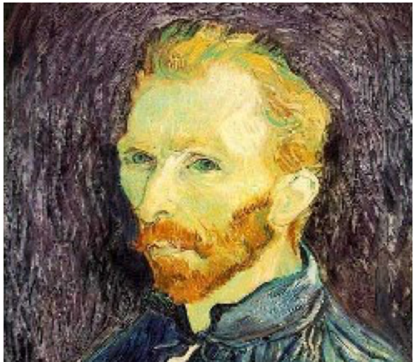
```
for i in levels[:-1]:
    L_i = hipass(f)
    f = downsample(f)
```

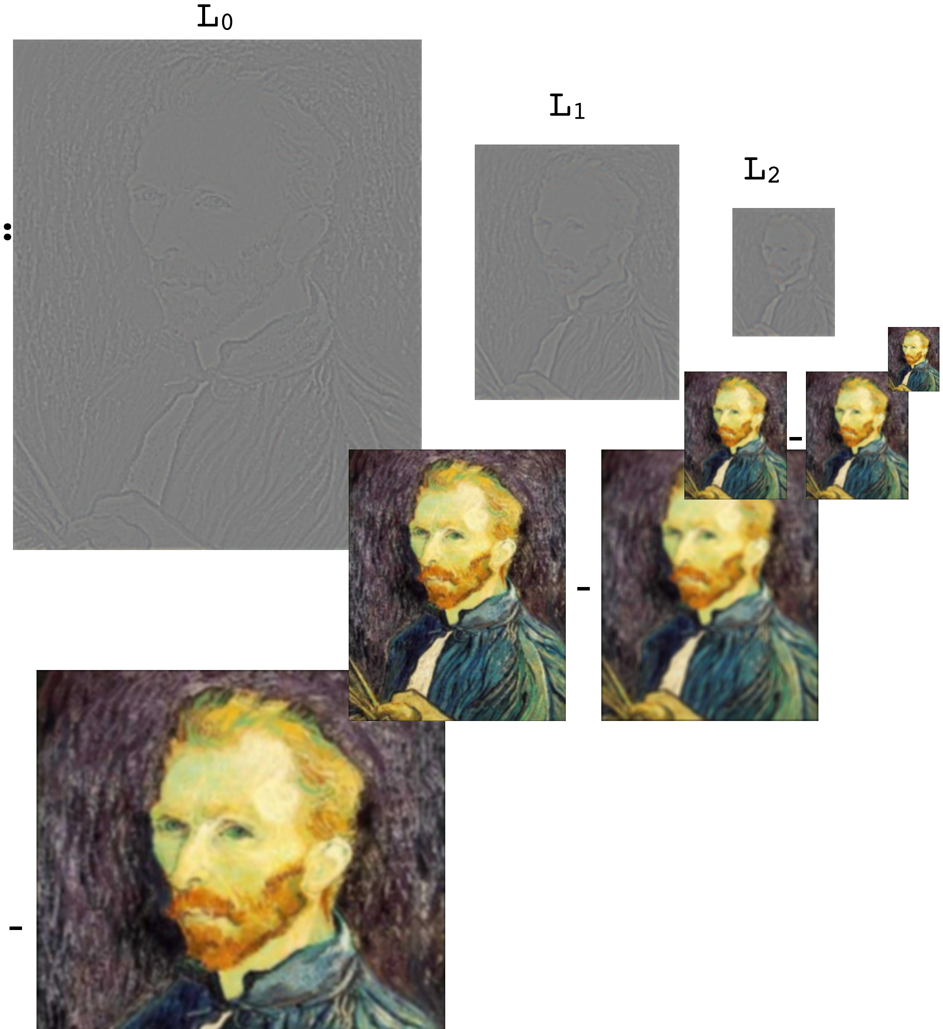




The Laplacian Pyramid

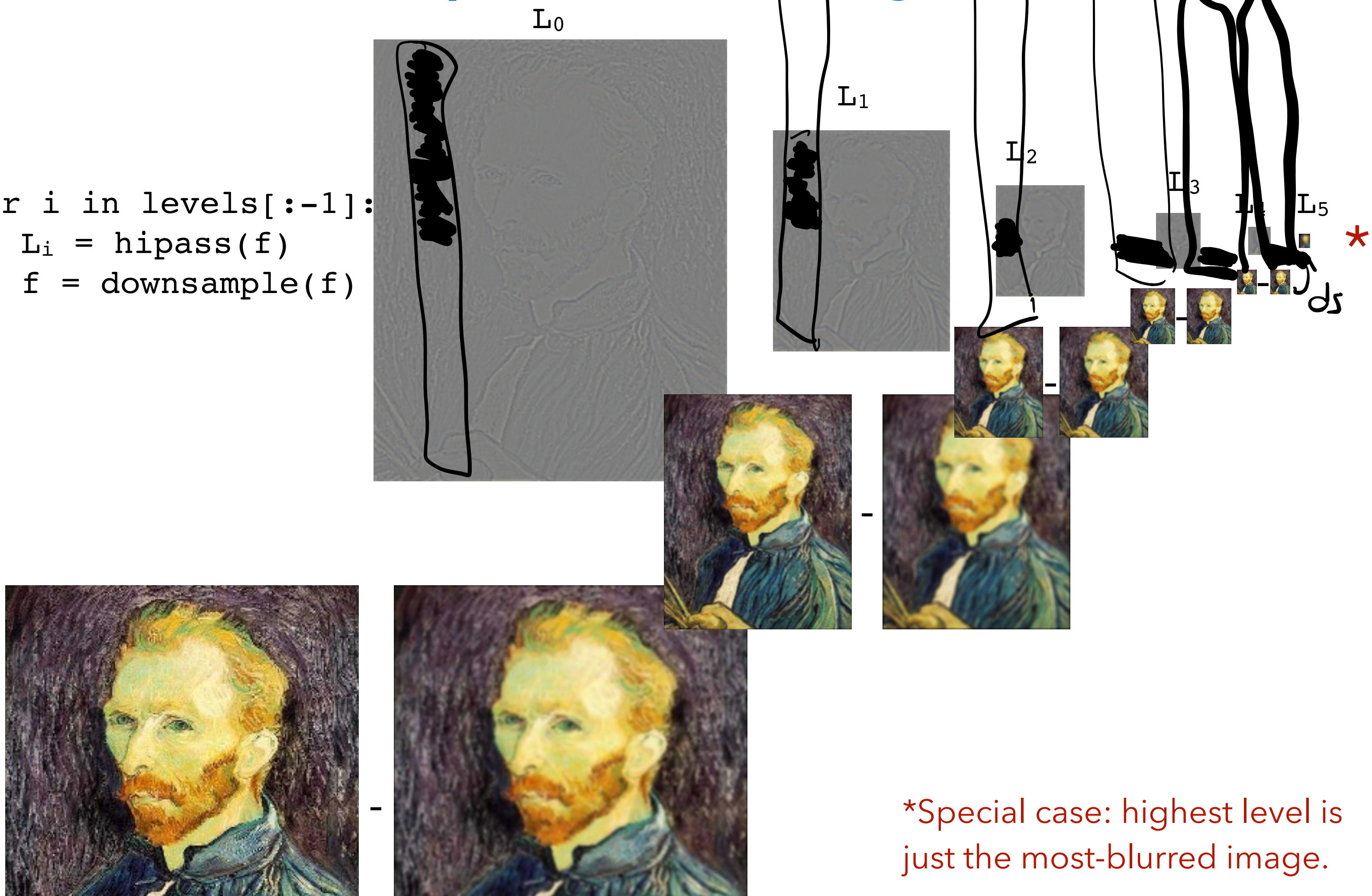
```
for i in levels[:-1]:
    $L_i$ = hipass(f)
    f = downsample(f)
```





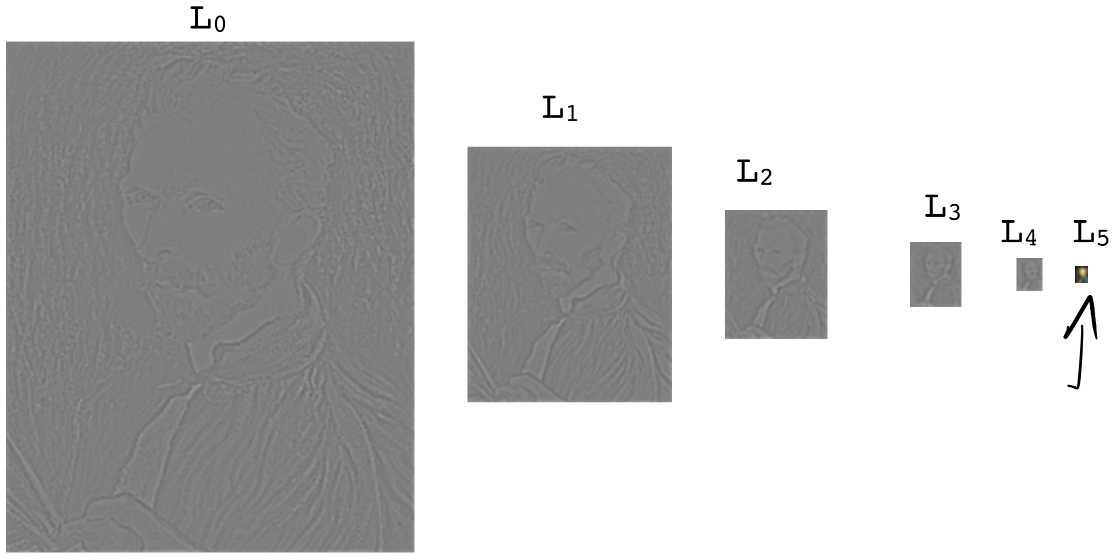
The Laplacian Pyramid

```
for i in levels[:-1]:
    $L_i$ = hipass(f)
    f = downsample(f)
```



*Special case: highest level is just the most-blurred image.

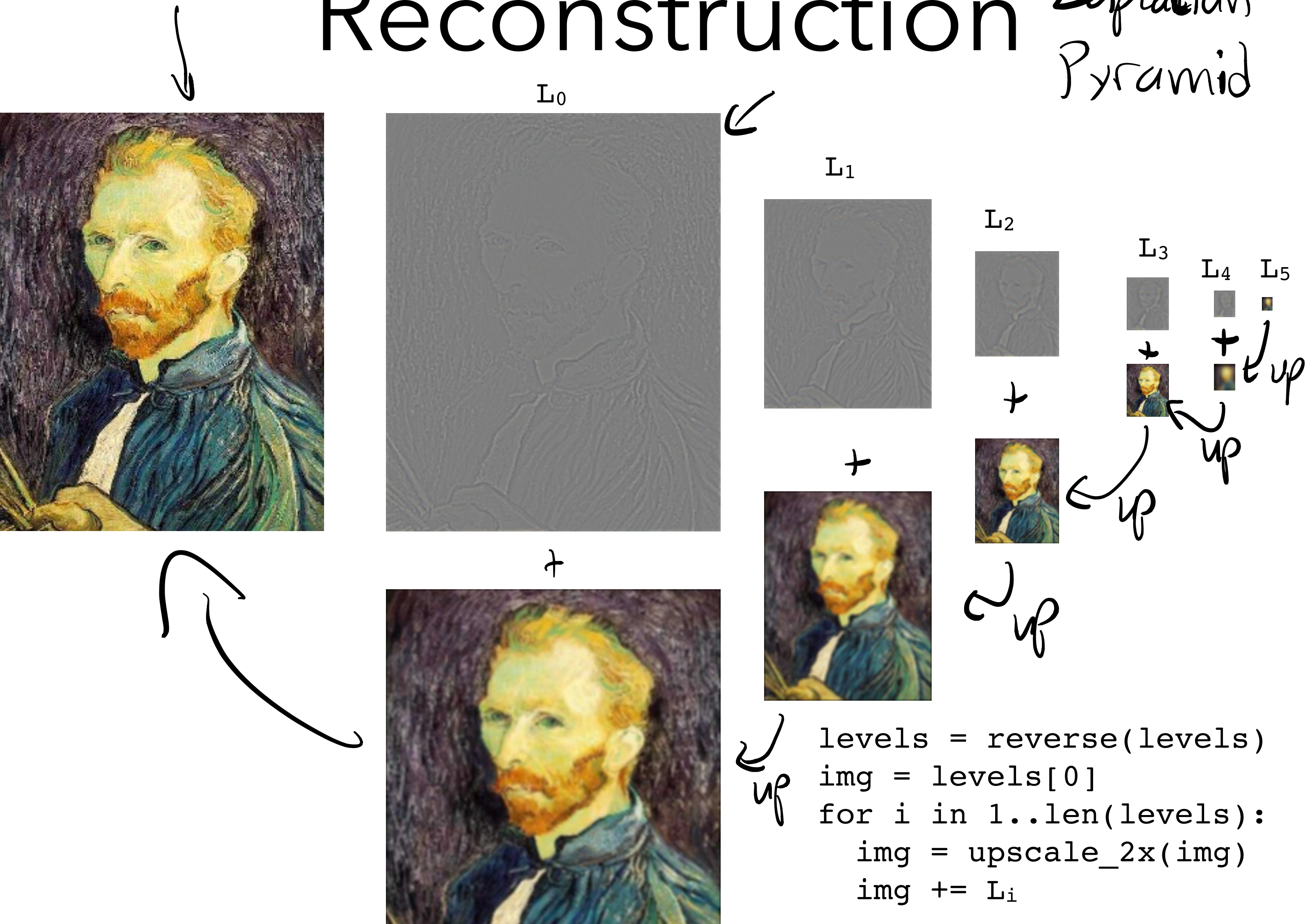
Reconstruction



```
levels = reverse(levels)
img = levels[0]
for i in 1..len(levels):
  img = upscale_2x(img)
  img += L_i
```

Reconstruction

Laplacian Pyramid



```
levels = reverse(levels)
img = levels[0]
for i in 1..len(levels):
  img = upscale_2x(img)
  img += L_i
```

Upsampling

- But how do we make images bigger?
- Again: a naive way and a principled way.

```
levels = reverse(levels)
img = levels[0]
for i in 1..len(levels):
  img = upscale_2x(img)
  img += L_i
```

Upsampling

- This image is too small for my screen. How do I make it 10x bigger?

Upsampling

- This image is too small for my screen. How do I make it 10x bigger?

- Simple approach: repeat each row and column 10 times