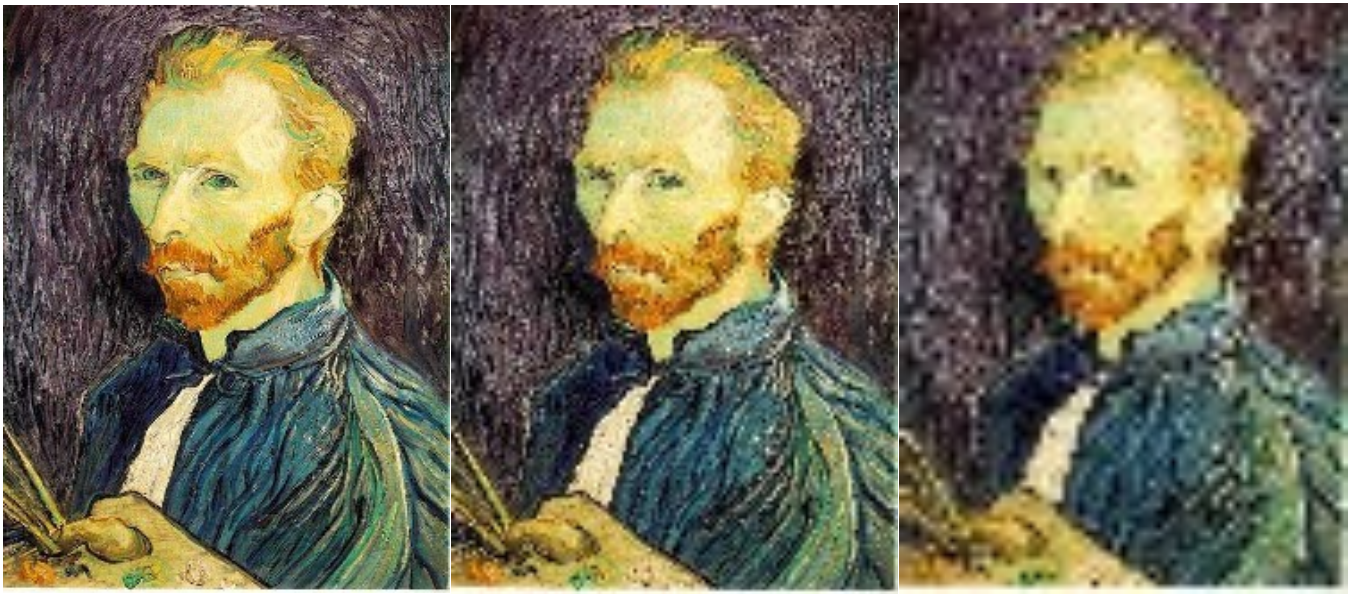
CSCI 497P/597P: Computer Vision



Lecture 6:

Downsampling

Gaussian Pyramids

Laplacian Pyramids

More Numpy

Announcements

- HW1 due tomorrow!
- P1 deadline extended to next Wednesday, but please get started soon. It's not small.
- Feedback Survey

Goals

- Know how to make images smaller:
  - The naive way via subsampling (and why this is bad)
  - The better way by prefiltering (and why this is better)
- Know how and why to construct a Gaussian Pyramid
- Know how and why to construct a Laplacian Pyramid
- Be familiar with the more advanced numpy capabilities demonstrated in the numpy playground.

Image S

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

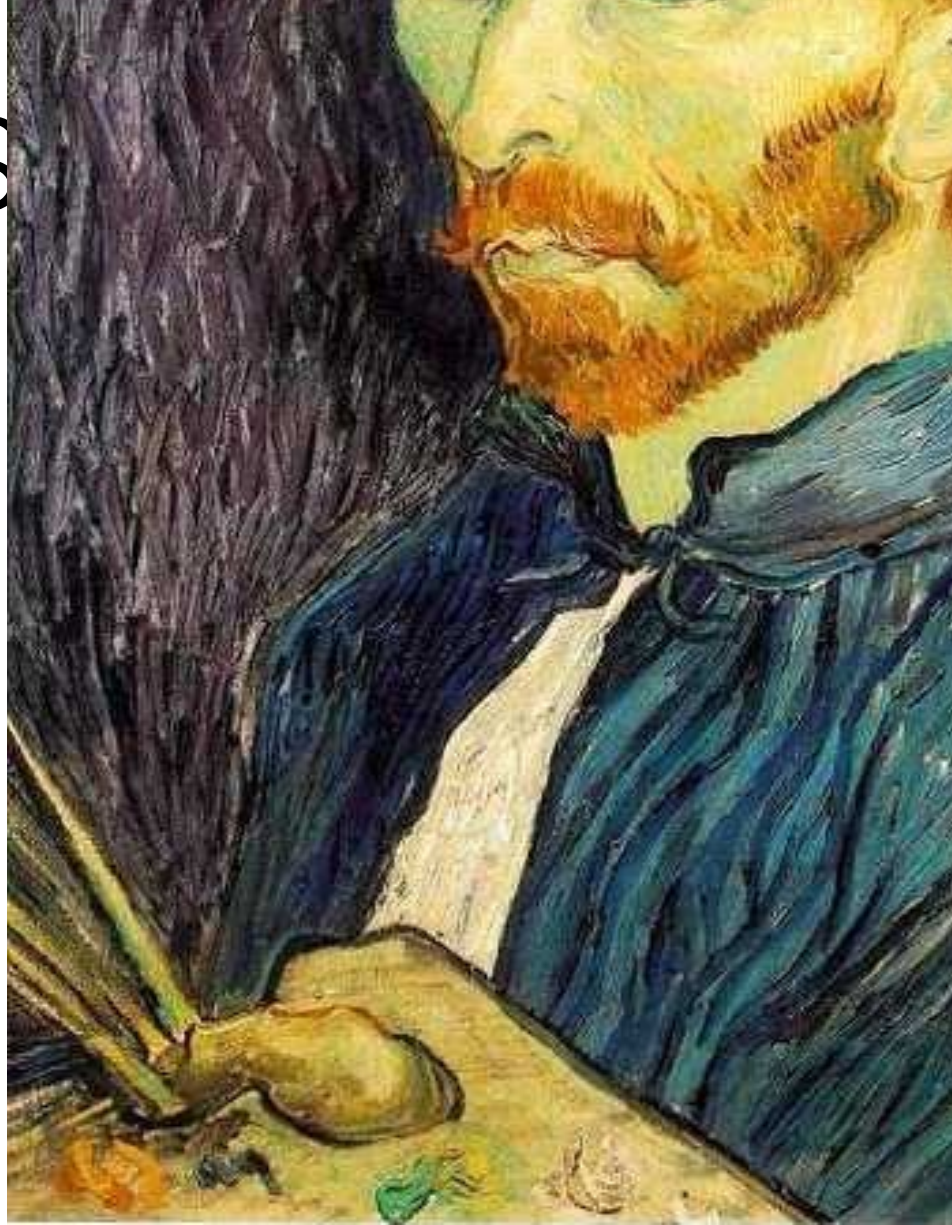
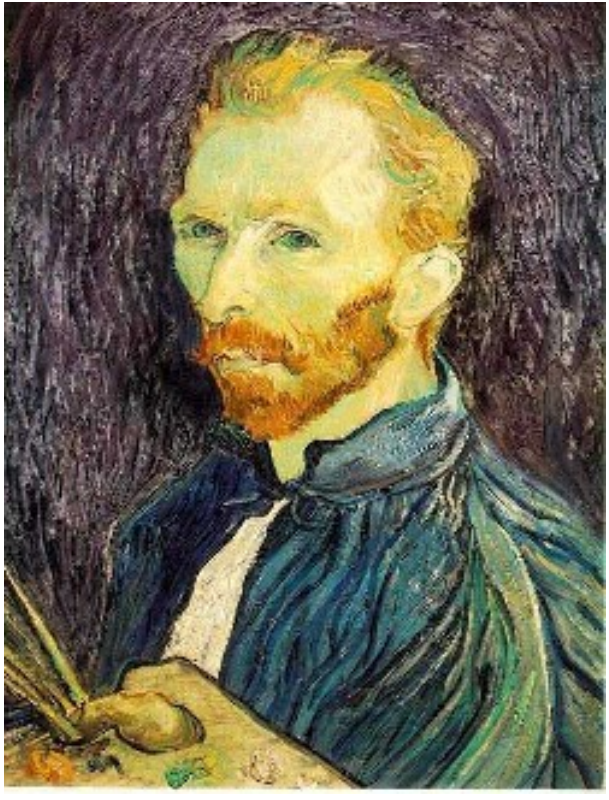


Image Subsampling

Only keep every...

2nd pixel



1/2

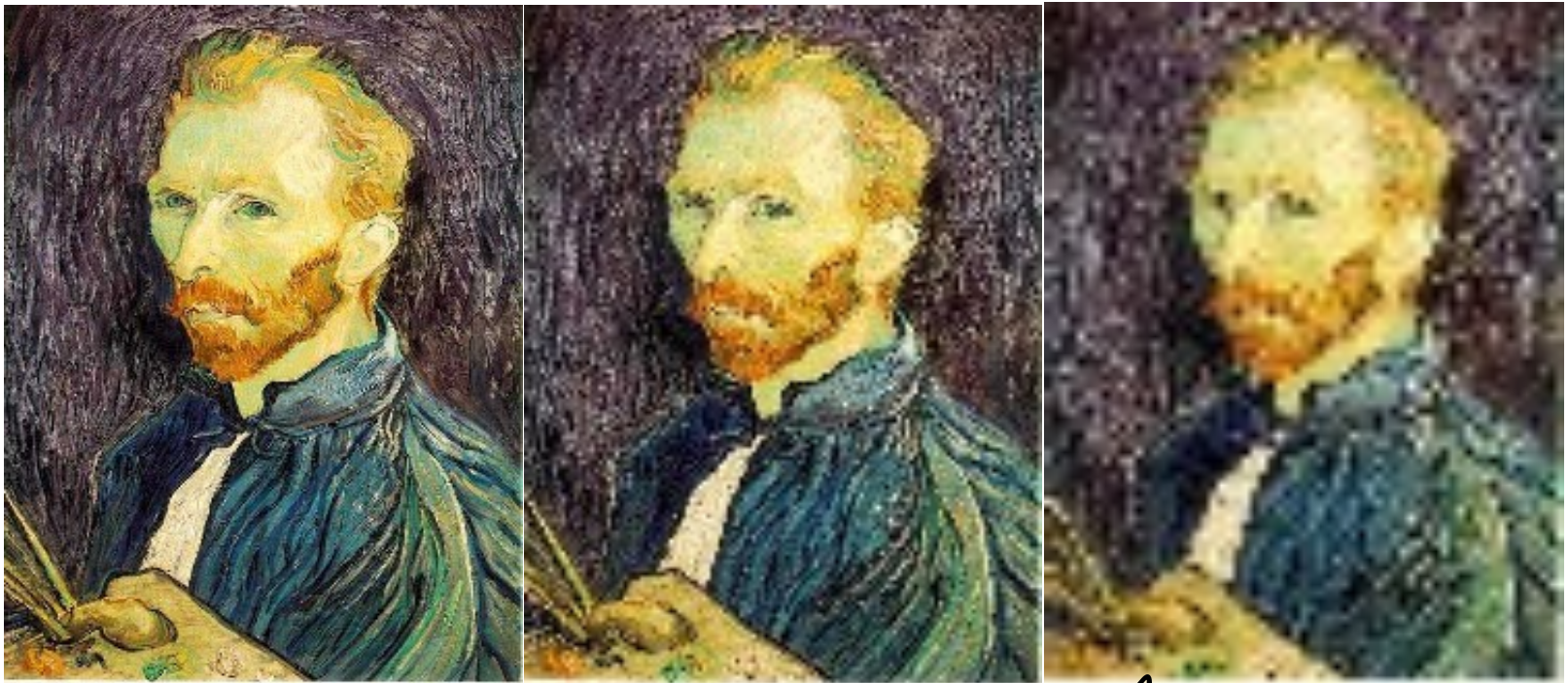
4th pixel

1/4

8th pixel

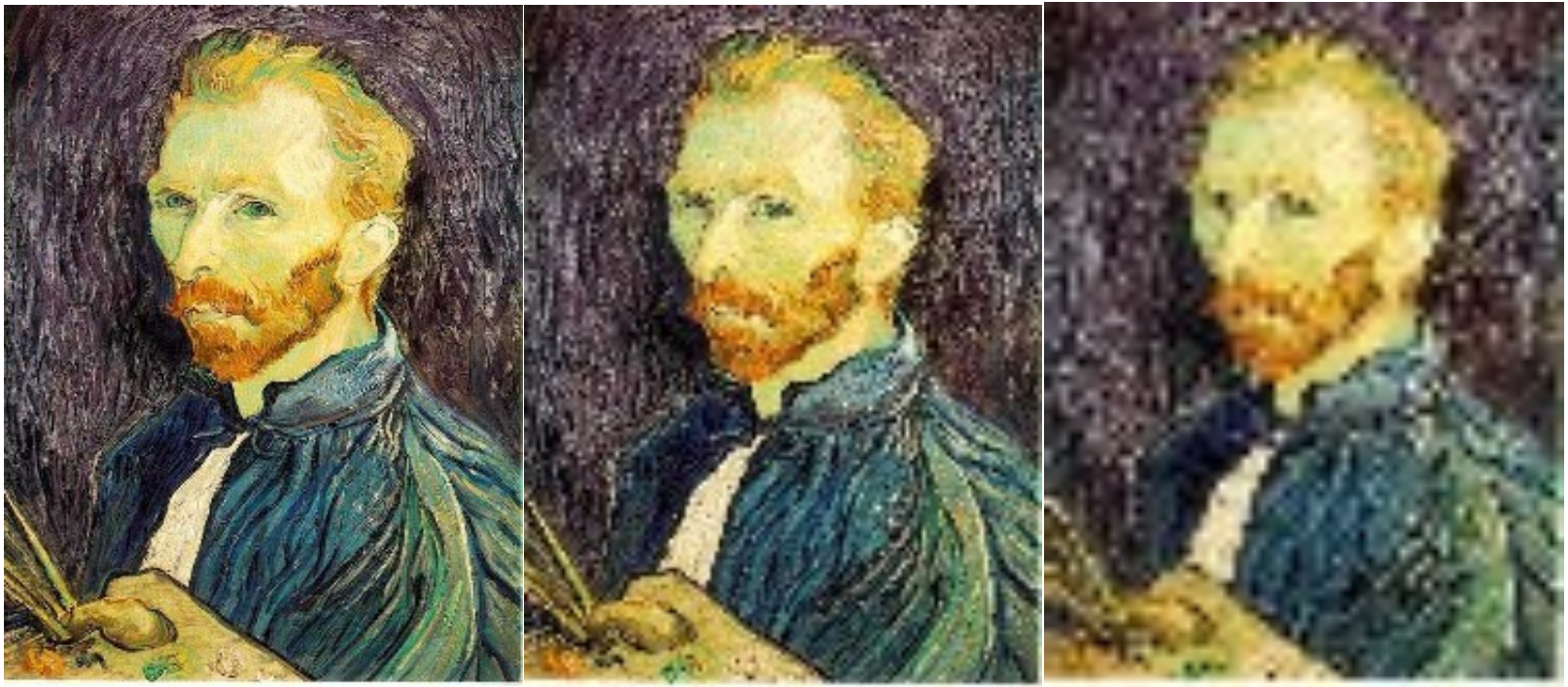
1/8

Image Subsampling



1/2 1/4 1/8

Image Subsampling



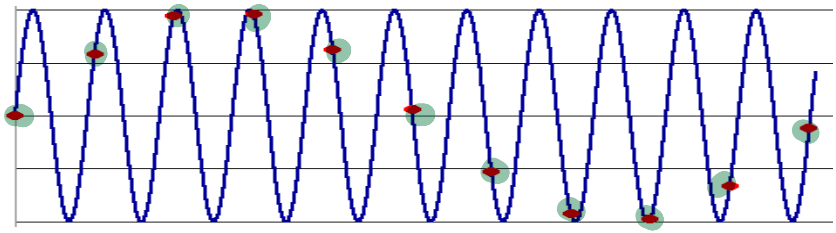
1/2 1/4 1/8

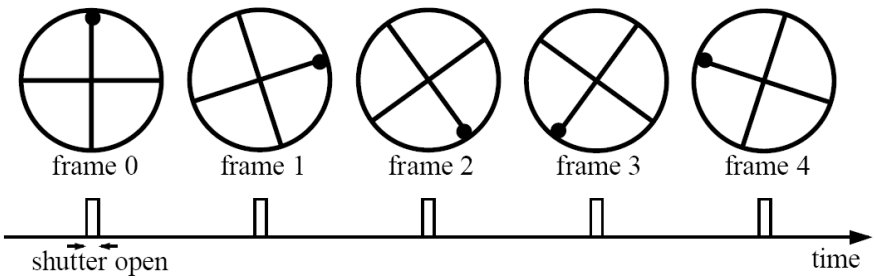
Why does this look so crufty?

Subsampling: Another example

Source: Fredo Durand

Aliasing

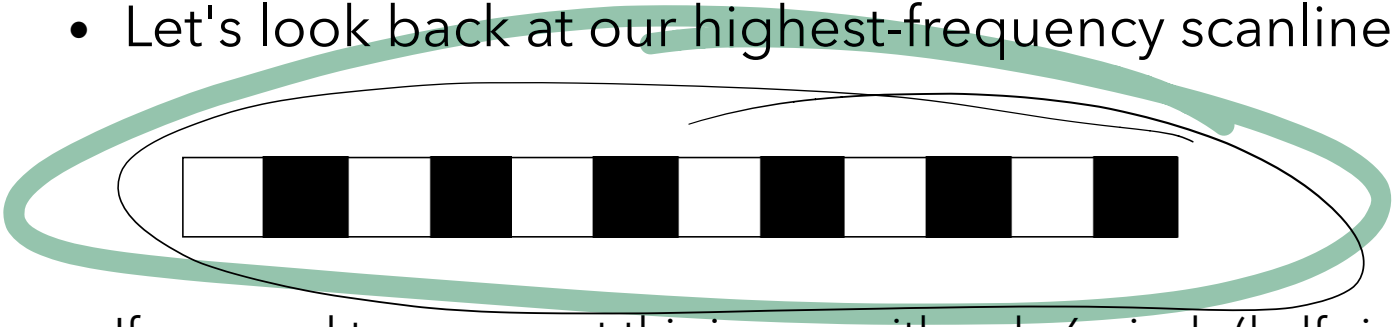




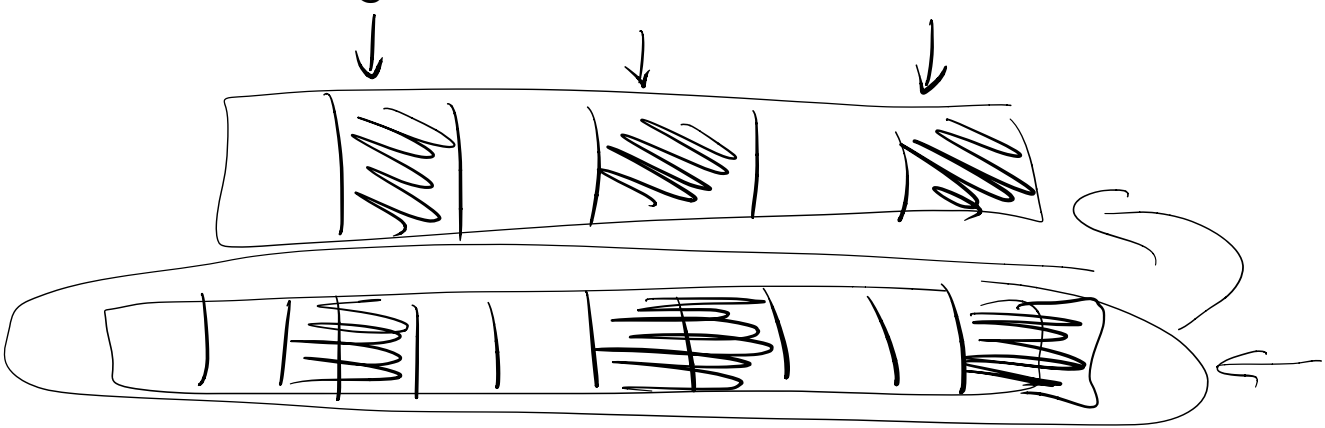
https://en.wikipedia.org/wiki/Wagon-wheel_effect

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:

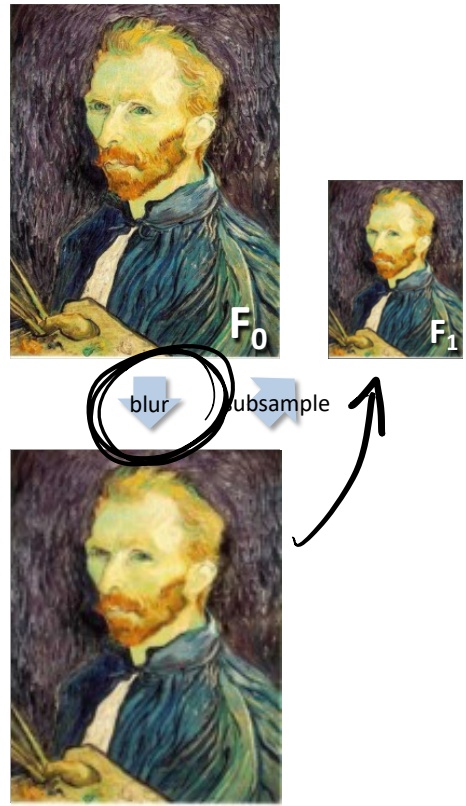
Downsampling

→ blur
→ subsample

Blurring *removes* high frequencies.

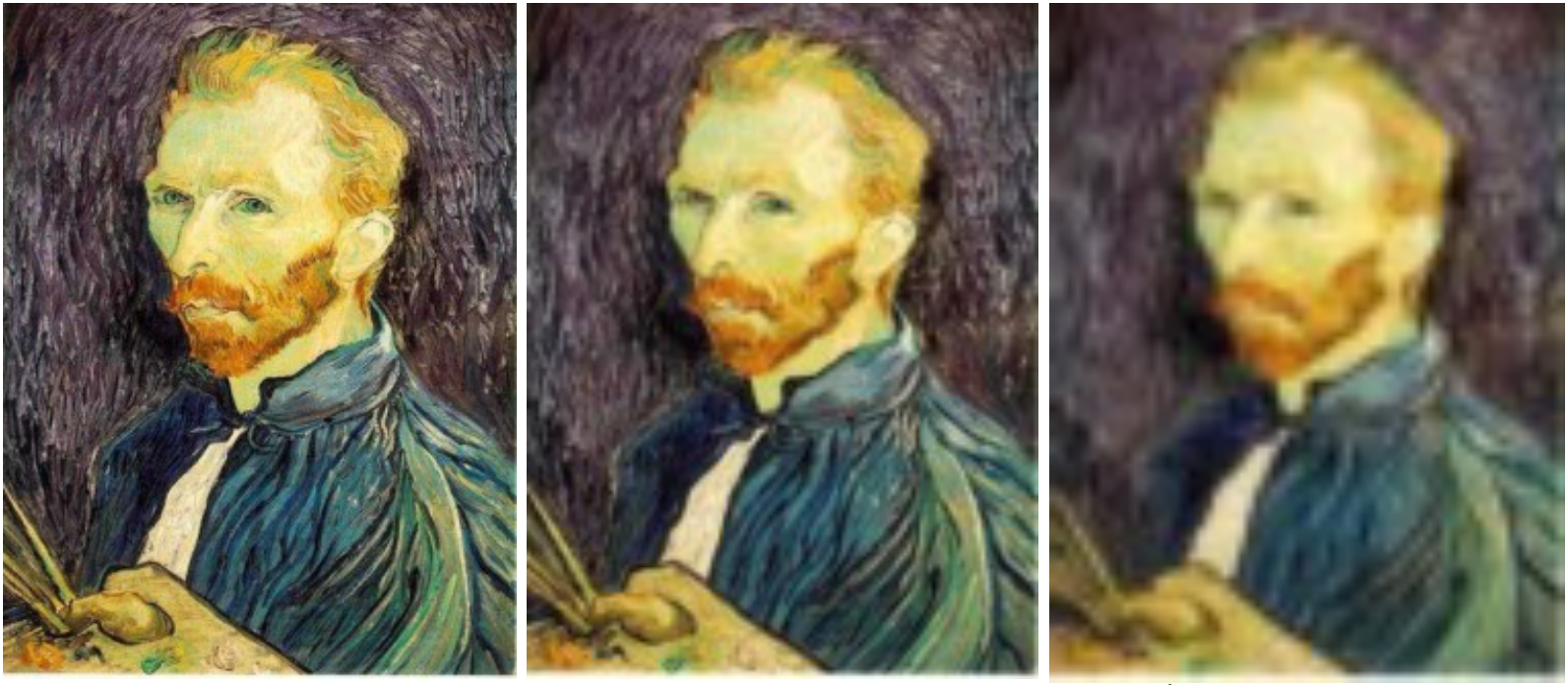
So, to make an image smaller:

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



F_0: Original image
F_1: Downsampled image

Downsampling with Gaussian Pre-filtering

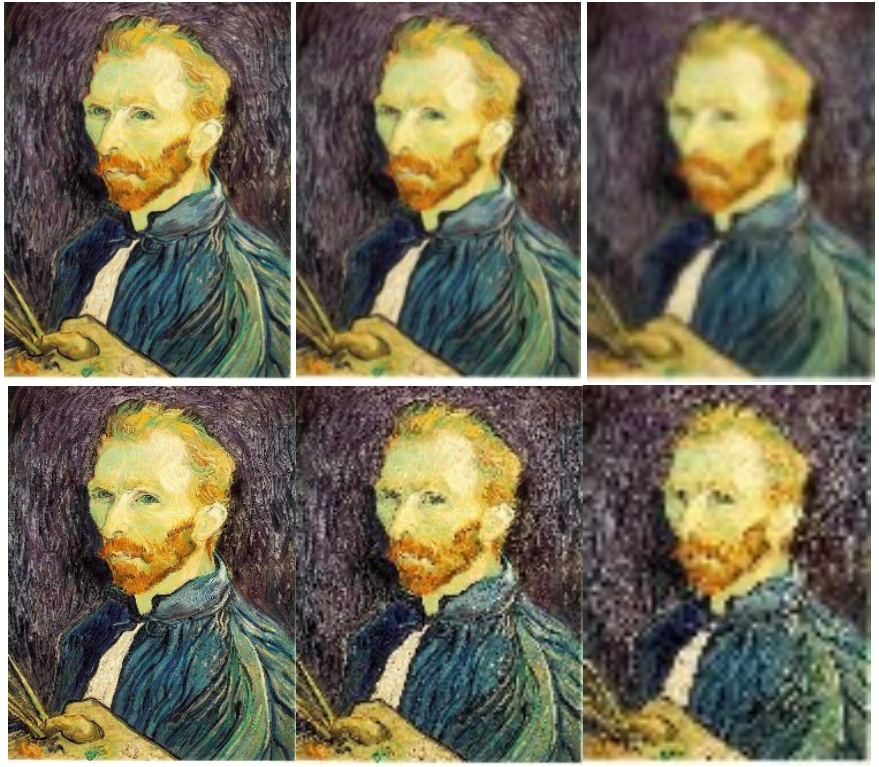


1/2 1/4 1/8

Downsampling with Gaussian Pre-filtering

with

without

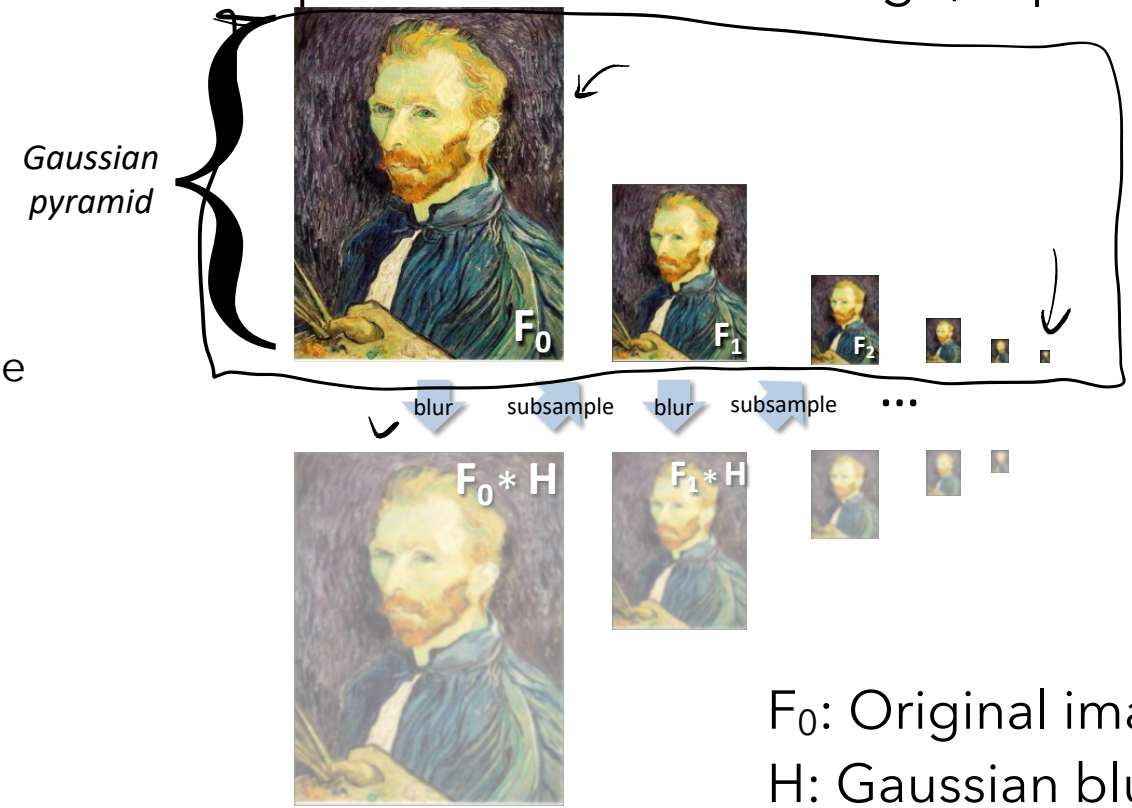


1/2 1/4 1/8

Gaussian Pyramid

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

1. Store
2. Blur
3. Subsample

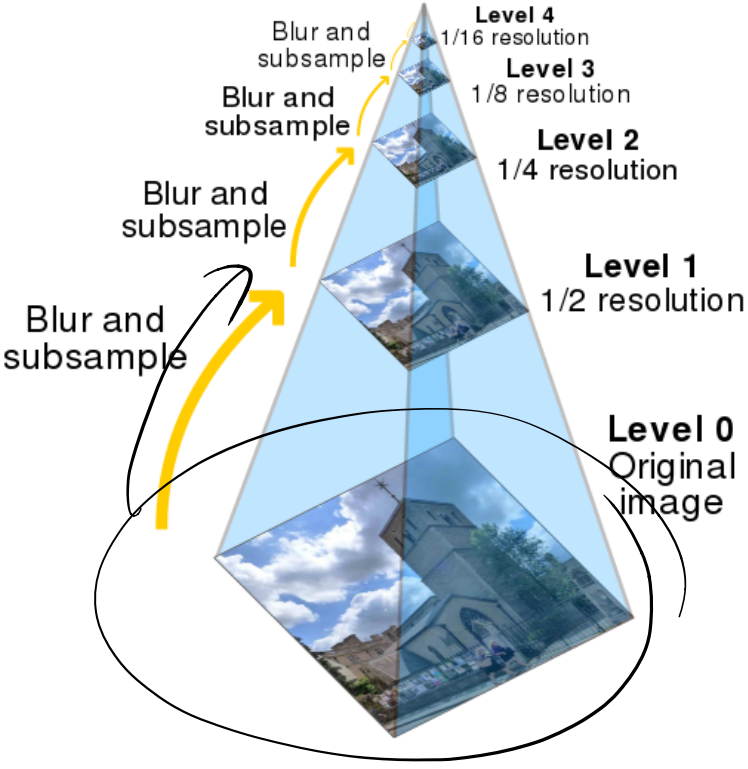


F_0: Original image

H: Gaussian blur

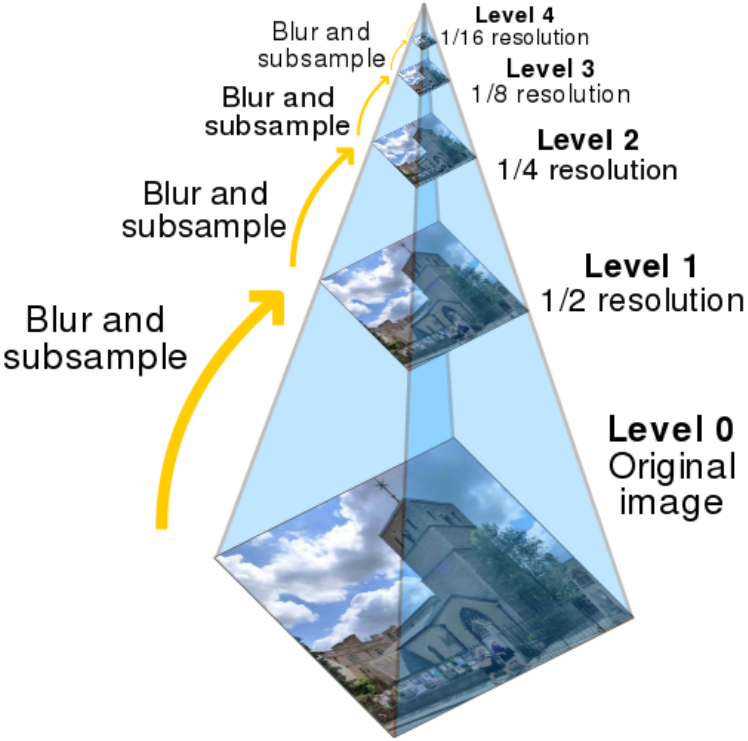
Gaussian Pyramid

A more pyramid-shaped view:

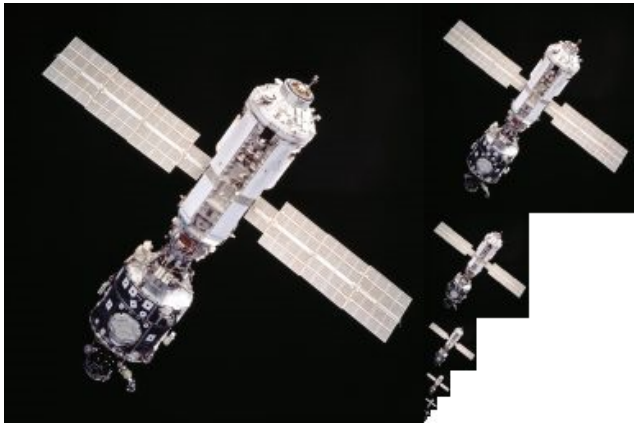


Gaussian Pyramid

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:

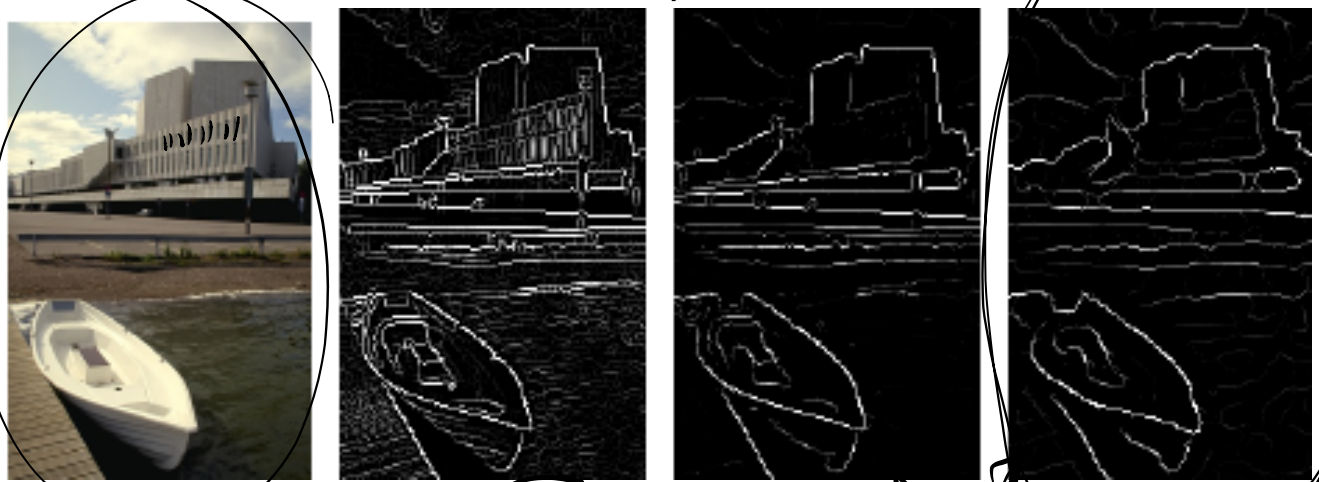
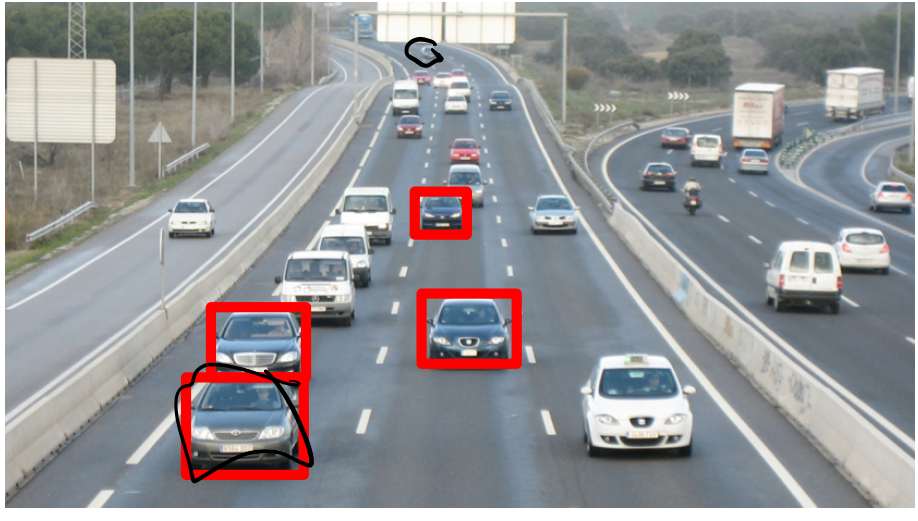


Image source: Baris Sumengen

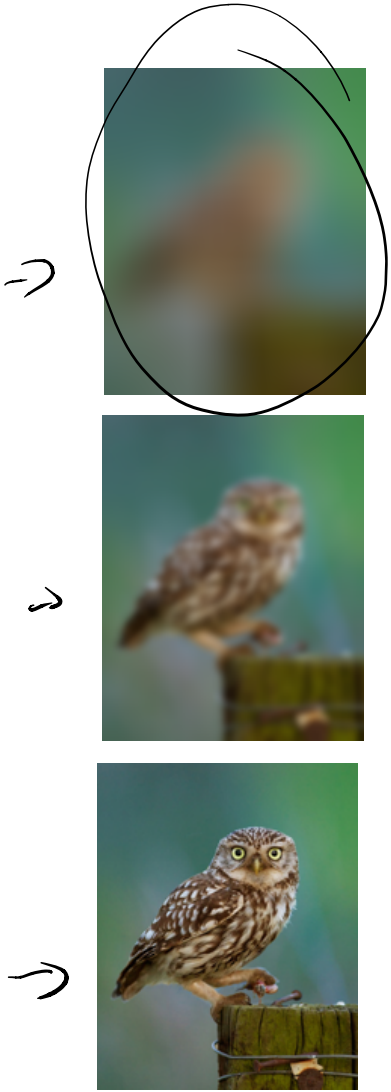
Gaussian Pyramid: But why?

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



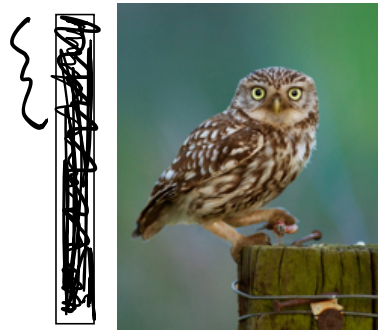
Frequency content in a Gaussian Pyramid

This is (secretly) a Gaussian Pyramid:

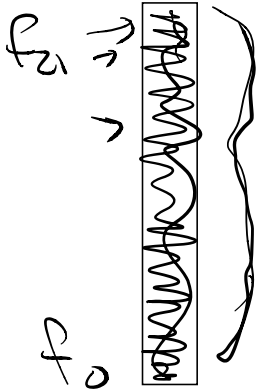


Frequency content in a Gaussian Pyramid

This is (secretly) a Gaussian Pyramid:

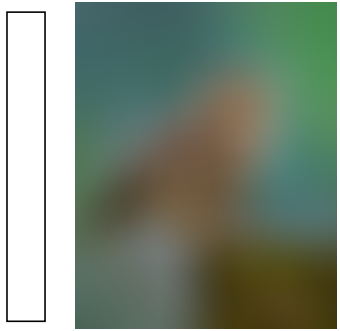


Frequency-ometer:



Frequency content in a Gaussian Pyramid

This is (secretly) a Gaussian Pyramid:



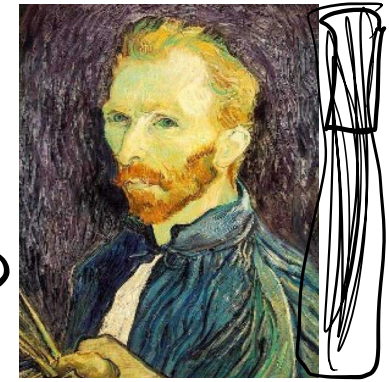
Frequency-ometer:

Some terminology

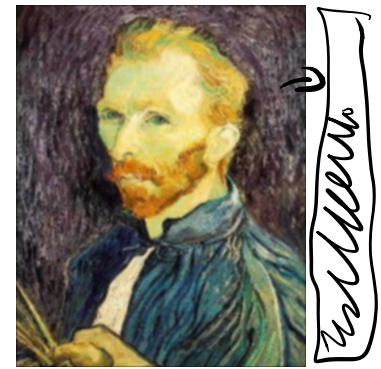
From the signal processing / electrical engineering field:

- 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.

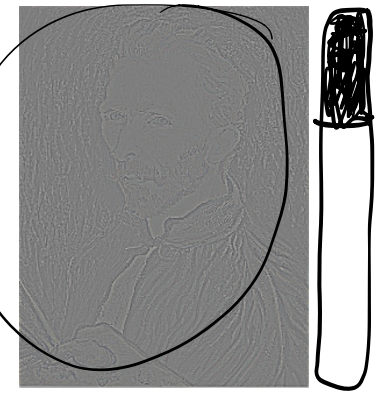
f =



f * w =



f - (f * w) =

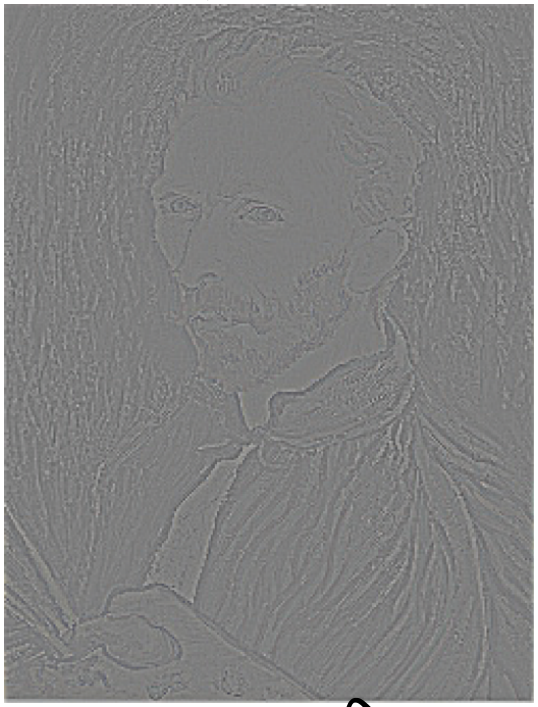


(We could also derive a single filter to accomplish this)

High-pass filter

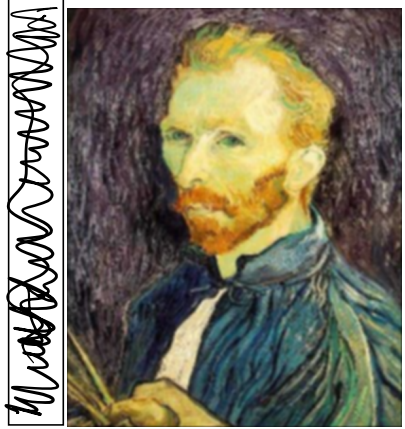
hi-pass

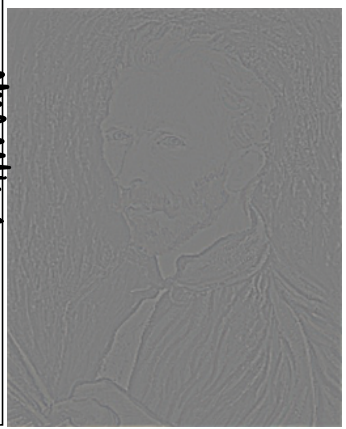
lo-pass



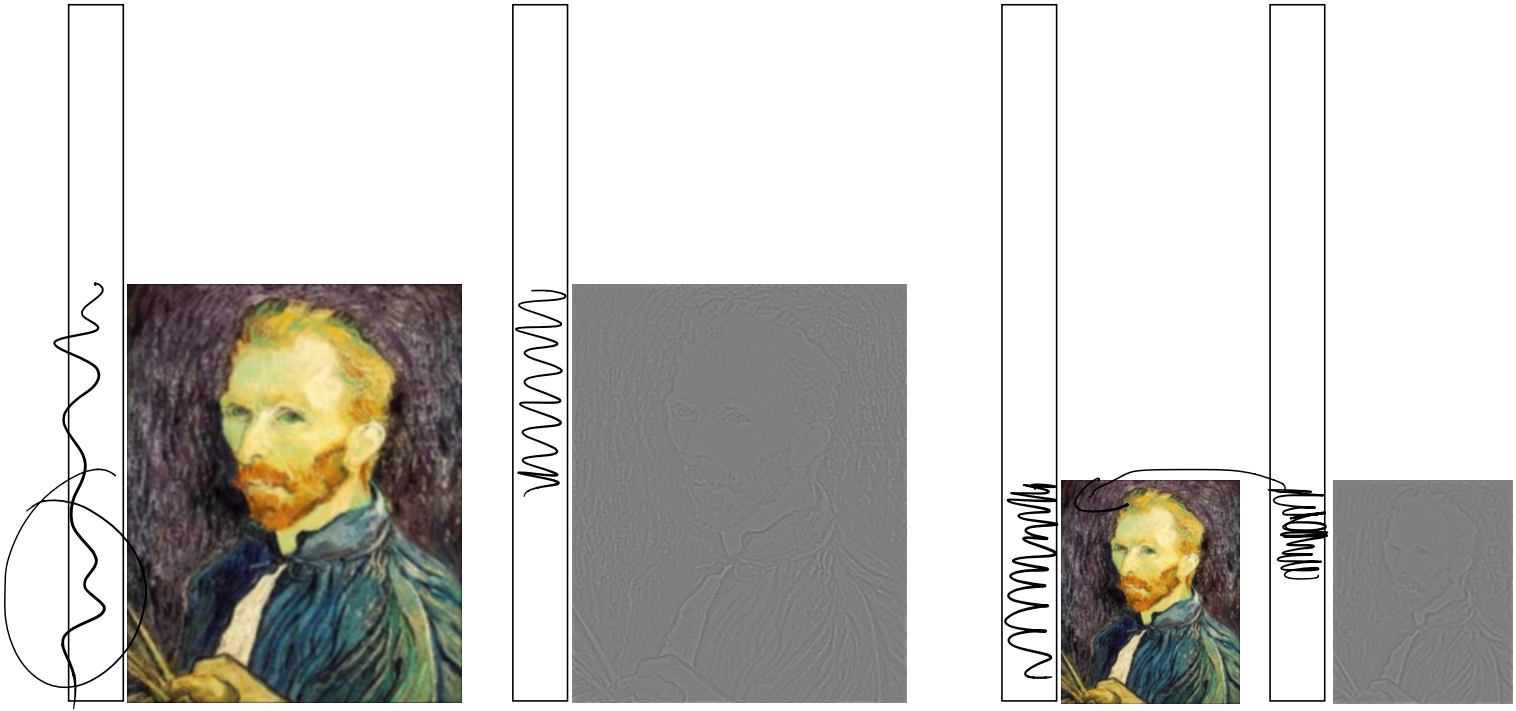
Subsample, then High-pass filter again!

hi pass





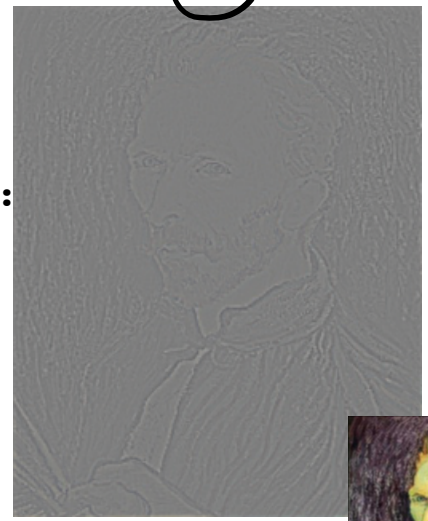
...and again!?

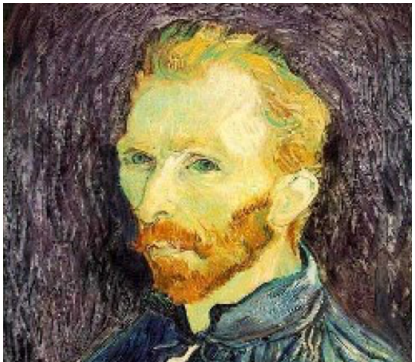


The Laplacian Pyramid

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

L_0



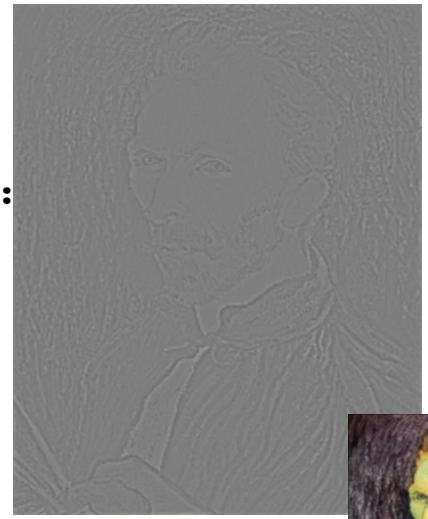


sub

The Laplacian Pyramid

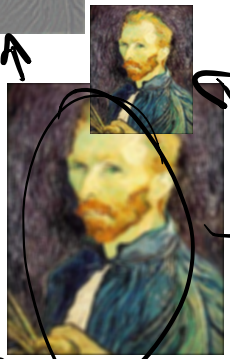
```
for i in levels[:-1]:
    Li = hipass(f)
    f = downsample(f)
```

L_0



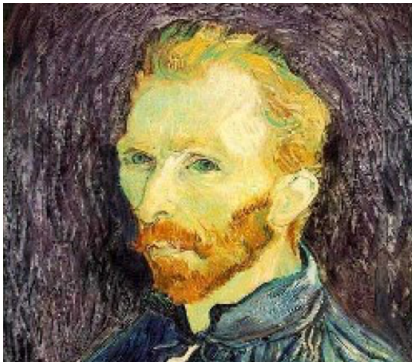
L_1

-



Sub

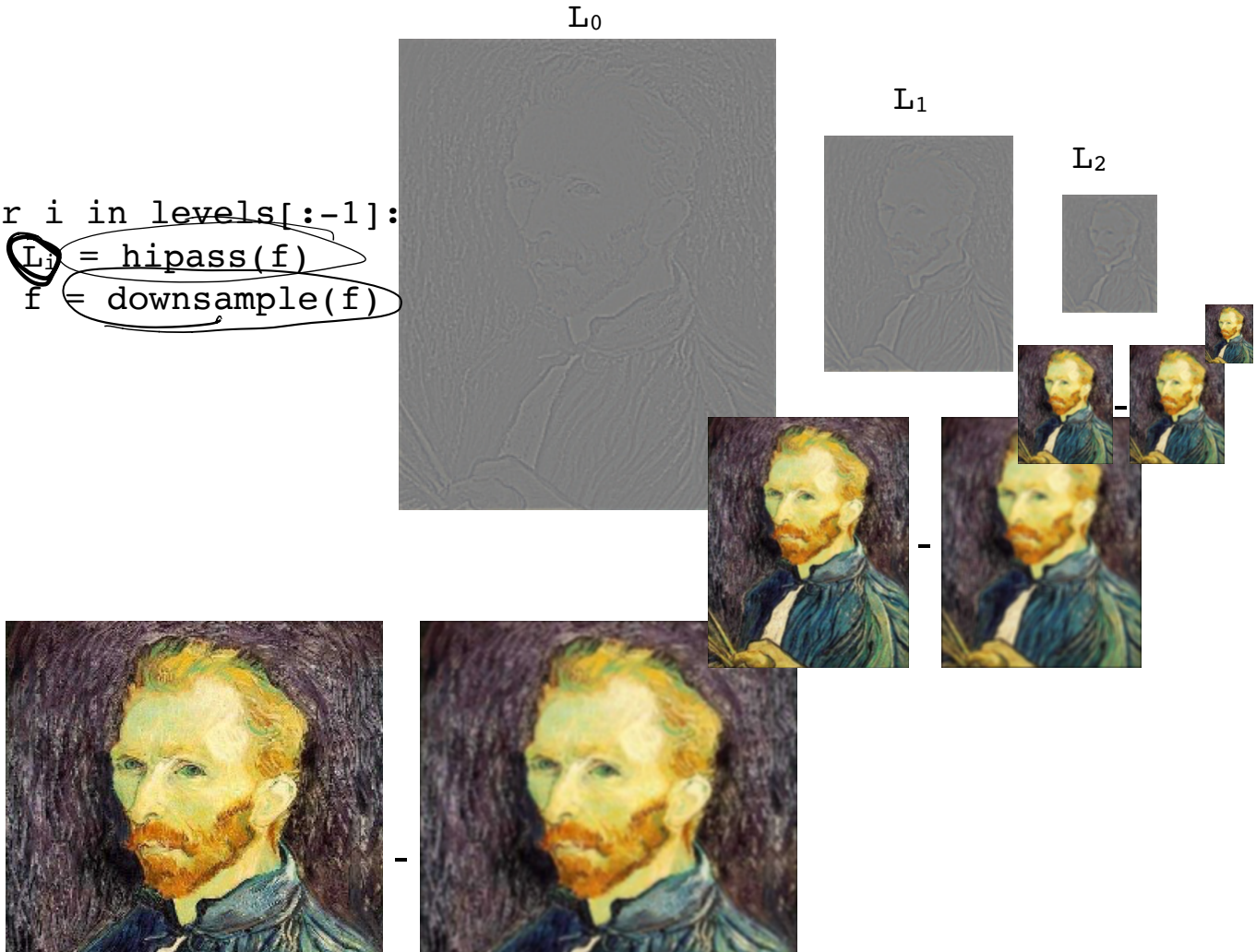
Sub



-

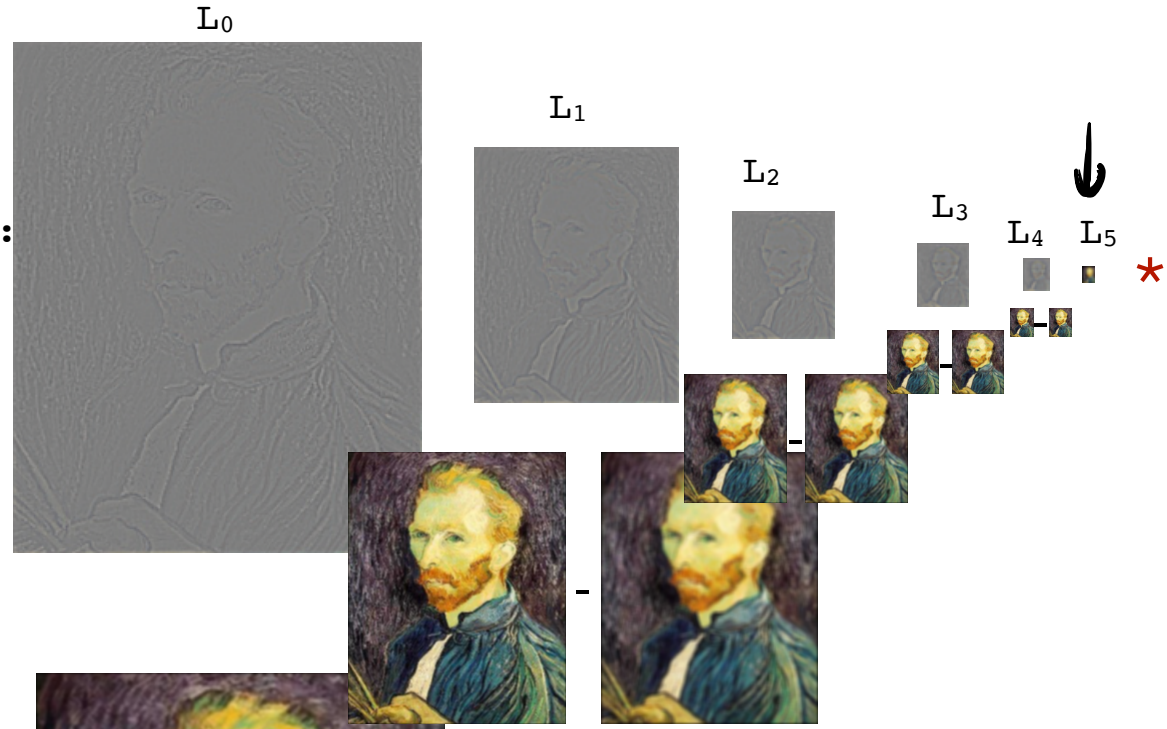
The Laplacian Pyramid

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



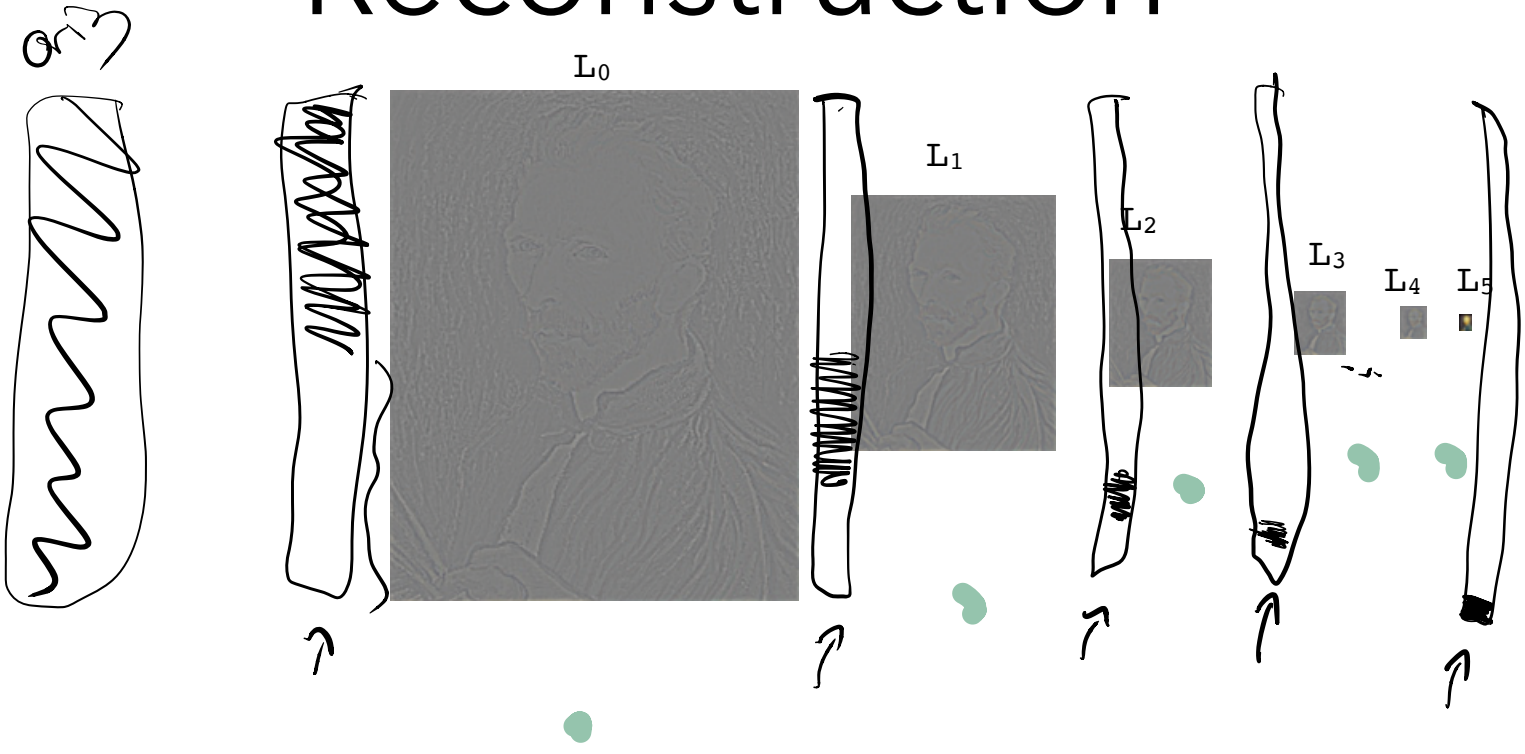
The Laplacian Pyramid

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



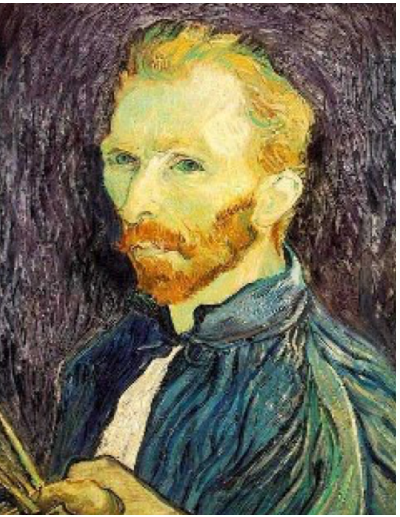
*Special case: last level is just the most-blurred image.

Reconstruction




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

Reconstruction



L_0

+

L_1

+

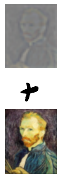
up

L_2

+

up

L_3



+

up

L_4

+

up

L_5


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

More Numpy!

- Tutorials:
  - https://numpy.org/devdocs/user/quickstart.html
  - https://cs231n.github.io/python-numpy-tutorial/#numpy
- Playground:
  - Demo
  - Exercises at the end for your practice.

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 += Li
```

Upsampling

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

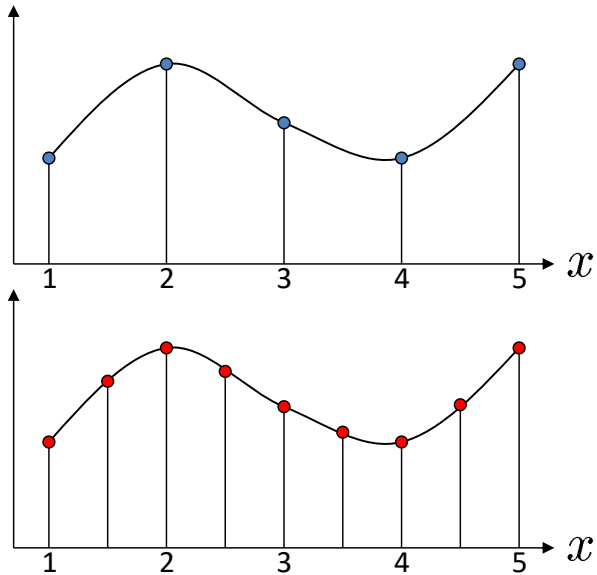
Upsampling

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

- Simple approach: repeat each row and column 10 times

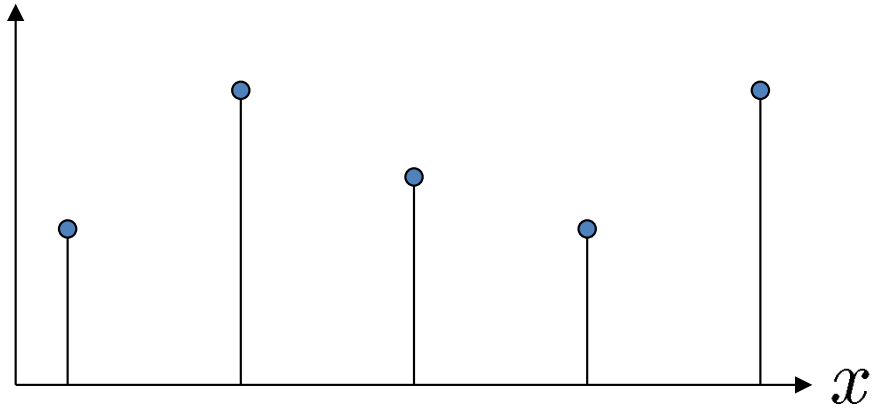
Upsampling: Interpolation

- Another way to look at this: we need to double the *sampling rate*.

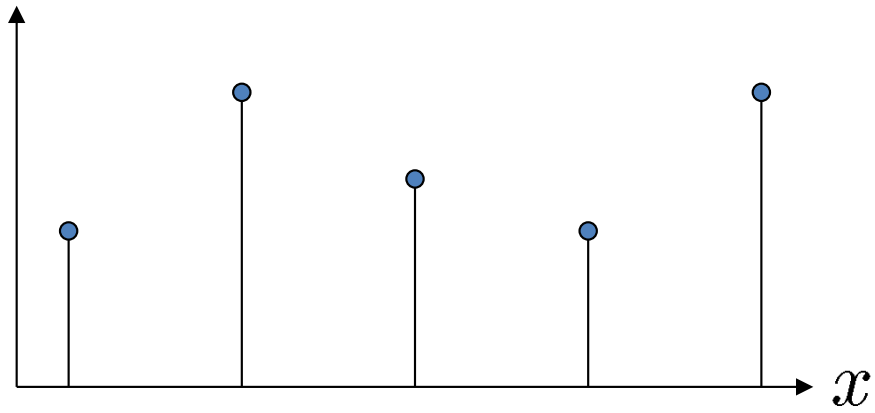


Upsampling: Interpolation

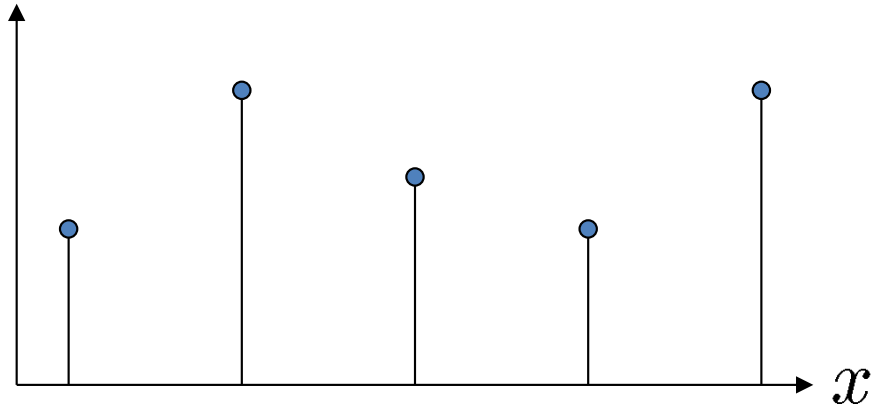
- Another way to look at this: we need to double the *sampling rate*.
- But we don't actually know the continuous function:



Upsampling: Nearest Neighbor

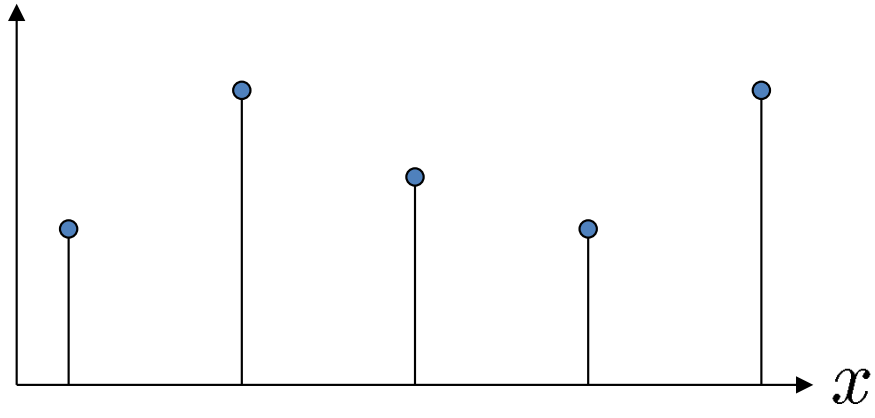


Upsampling: Linear

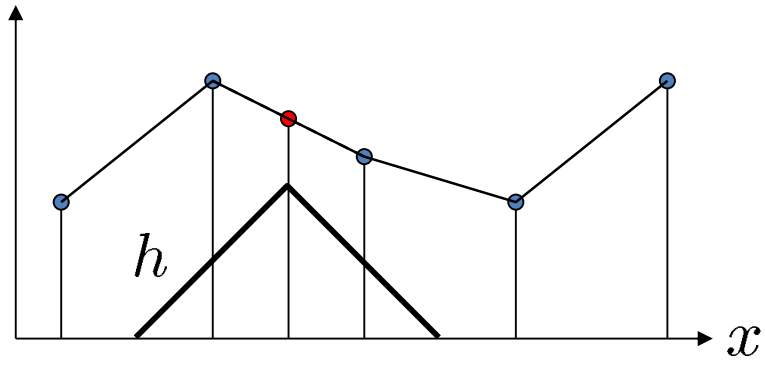


Upsampling: Linear

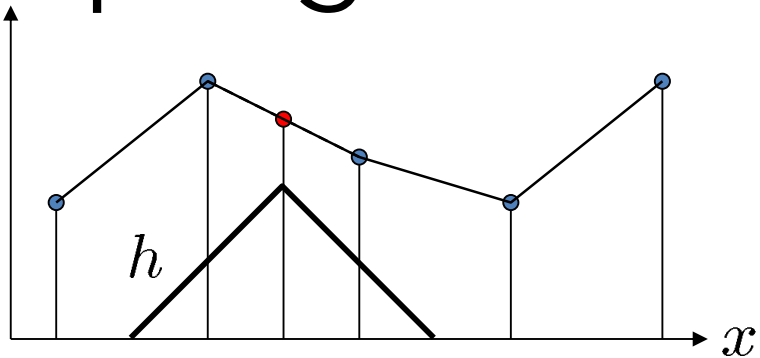
A filtering perspective



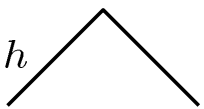
Upsampling: Interpolation

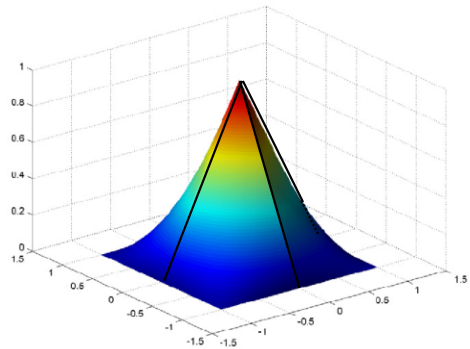


Upsampling Filters in 2D



1	2	1
2	4	2
1	2	1

1D: 

2D: 

"tent filter"

Upsampling in 2D