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 Size

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:

[Image: a scanline pattern]

• 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.

\[ F_0 : \text{Original image} \]
\[ F_1 : \text{Downsampled image} \]
Downsampling with Gaussian Pre-filtering
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

\[
F_0 \Rightarrow F_1 \Rightarrow F_2 \Rightarrow \ldots
\]

Gaussian pyramid

\[
F_0: \text{Original image} \quad H: \text{Gaussian blur}
\]
Gaussian Pyramid

A more pyramid-shaped view:
Gaussian Pyramid

A more pyramid-shaped view:  A storage-oriented view:

Level 4
1/16 resolution

Level 3
1/8 resolution

Level 2
1/4 resolution

Level 1
1/2 resolution

Level 0
Original image
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.

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

hi-pass

lo-pass
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]:
    \(L_i = \text{hipass}(f)\)
    \(f = \text{downsample}(f)\)
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: 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 += L_i
```
Reconstruction

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

1D: $h$

2D: "tent filter"
Upsampling in 2D