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
Image sub-sampling

Why does this look so crufty? Aliasing!

Source: S. Seitz
Why does aliasing happen?

• Consider sampling every P pixels
  
  \[ B_{\frac{N}{P}+k}(Pn) = e^{\frac{i2\pi(N+P)pn}{N}} = e^{i2\pi n} e^{\frac{i2\pi kPn}{N}} = e^{\frac{i2\pi kPn}{N}} = B_k(Pn) \]

• 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:** $2v_{\text{max}} < v_{\text{sample}}$
• To subsample, *remove high frequency components*
• To remove high frequency components, *blur the image with a Gaussian*
Fourier transform

Zeros out high frequencies

Keeps low frequencies

Gaussian filters
Gaussian pre-filtering

- Solution: filter the image, *then* subsample

\[
F_0 \ast H \rightarrow \text{blur} \rightarrow \text{subsample} \rightarrow \text{blur} \rightarrow \text{subsample} \rightarrow \ldots
\]
Gaussian pyramid

\[
\begin{align*}
F_0 & \xrightarrow{\text{blur}} F_1 \\
F_1 & \xrightarrow{\text{subsample}} F_2 \\
\end{align*}
\]

\[
\begin{align*}
F_0 & \xrightarrow{H} F_0^* H \\
F_1 & \xrightarrow{H} F_1^* H \\
\end{align*}
\]
Gaussian pyramids
[Burt and Adelson, 1983]

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

Gaussian Pyramids have all sorts of applications in computer vision

---

Idea: Represent $N \times N$ image as a “pyramid” of $1\times1$, $2\times2$, $4\times4$, ..., $2^k \times 2^k$ images (assuming $N = 2^k$)

![Pyramid Diagram]

- level $k$ (= 1 pixel)
- level $k-1$
- level $k-2$
- ... (omitted levels)
- level 0 (= original image)
Gaussian pyramids - Searching over scales
Gaussian pyramids - Searching over scales
The Gaussian Pyramid

\[ G_0 = \text{Image} \]

\[ G_1 = (G_0 * \text{gaussian}) \downarrow 2 \]

\[ G_2 = (G_1 * \text{gaussian}) \downarrow 2 \]

\[ G_3 = (G_2 * \text{gaussian}) \downarrow 2 \]

\[ G_4 = (G_3 * \text{gaussian}) \downarrow 2 \]
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 \textit{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

\[
\text{Expand (upsample + blur)} - \text{Expand (upsample + blur)} = \text{Expand (upsample + blur)} - \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} = \text{Expand (upsample + blur)} - \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} = \text{Expand (upsample + blur)} - \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} = \text{Expand (upsample + blur)} - \text{Expand (upsample + blur)}
\]
Laplacian pyramid

\[
\begin{align*}
L_4 &= G_4 = \\
L_3 &= G_3 - \text{expand}(G_4) = \\
L_2 &= G_2 - \text{expand}(G_3) = \\
L_1 &= G_1 - \text{expand}(G_2) = \\
L_0 &= G_0 - \text{expand}(G_1) =
\end{align*}
\]
Reconstructing the image from a Laplacian pyramid

\[
\text{Expand (upsample + blur)} + \text{Expand (upsample + blur)} = \text{Expand (upsample + blur)} + \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} + \text{Expand (upsample + blur)} = \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} = \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} = \text{Expand (upsample + blur)}
\]

\[
\text{Expand (upsample + blur)} = \text{Expand (upsample + blur)}
\]
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